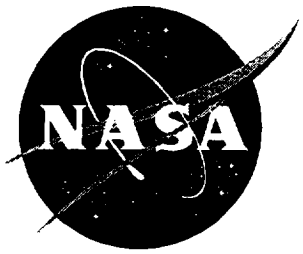
NASA Contractor Report 201732



An Algorithm for Integrated Subsystem Embodiment and System Synthesis

Kemper Lewis
Georgia Institute of Technology, Atlanta, Georgia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Grant NAG1-1564

August 1997

The information in this report was offered as a thesis in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia, September 1996.

Available from the following:

NASA Center for AeroSpace Information (CASI)
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

ACKNOWLEDGMENTS

"Pain and fatigue make cowards of us all. The harder you work, the harder it becomes to surrender." Vince Lombardi

Vince summarized the struggles and triumphs I have experienced during the completion of this dissertation well. I certainly could not have done it without the help of many close friends and colleagues. In the following I have summarized my feelings and appreciation as best I could for as many as I could.

First to God, who provided me with incredible opportunities, perseverance, discipline, and strength to reach this point. He has given me the best friends and family anyone could ask for.

To my advisor, Farrokh Mistree. One paragraph can not hope to describe the gratitude and respect I have for him. He was the most outstanding advisor I could have imagined. However, his influence on me is not temporal; it will live on forever. He has been a great friend, confidant, and person. My heart goes out to Farrokh; he has a special place there.

Very special thanks go out to my committee members:

Bert Bras, whose expertise in optimization and design was incredibly helpful. He also gave invaluable advice concerning my career and life in general.

David Rosen, whose feedback on the mathematical analysis aspects of the dissertation was very influential. He also gave me invaluable advice concerning my career and life in general.

Leon McGinnis, whose expertise in systems engineering and optimization provided invaluable insights and perspectives on my work.

Dan Schrage, whose expertise in aerospace systems design and design processes in general was critical to my work. Dr. Schrage's support for the past 4 years was invaluable to my graduate study and career development.

Janet Allen, whose expertise in decomposition, statistics, and living systems played a large role in the dissertation. Janet was also such a strong source of support in many aspects of graduate school, from career advice, to helping print, copy, and set up events. A very special thanks goes to Janet.

Jarek Sobieski, whose support allowed this research to be accomplished for the last 3 years. His expertise in multidisciplinary design optimization was influential in the fruition of the research. Many parts of this dissertation are based on work performed by Dr. Sobieski. I am eternally grateful to him for his technical feedback and career advice.

Special thanks go out to my family, especially my sister, father, and step-father, Bob Childers. They have supported through every valley and peak, and I am deeply indebted to them.

The next thanks go out to my very close friends in the lab: Jesse Peplinski, Pat Koch, and Stewart Coulter. Without these guys, I would have not made it. We struggled through everything together and I will never forget the great times we had for the past four years. They are the brightest guys I have every worked with. I feel honored to have had the opportunity to work with them.

To other members of the lab: Wei Chen and Tim Simpson. Wei has taught me alot about robust design, optimization, and statistics. She was and continues to be a great influence on me. Tim leads the next generation of academic superstars. He knows when to speak and when to listen. I hope we can continue to work together throughout our careers.

I also thank the other special members of the SRL who have had their own personal effects on me: Matt Bauer, Jack, Srinivas, Matt Marston, Scott, Reid, Keith, Kristie, Zahed, and Brian.

My friends outside of the SRL:
Scott Kollins, who has experienced this wild ride in the front seat with me the entire time. I could not have made it without him. His support in philosophical awakenings, scientific exploration, physical triumph, emotional struggles, and mental innovation has been the most significant influence upon me for the past four years. He is truly a partner in this roller coaster of life we live.

Tom Hawkins and David Griffin have been two pillars of support for eight years. They are the best friends someone could have. I look forward to creating more memories with them for the rest of my life.

Debbie Greco, who has given me more support than she probably realizes. She is truly one of a kind and she will always have a very special place with me.

To my other close friends who have played significant roles in my life: Brittany Griffin, Pete Sproul, Dan Fahley, Paul Euber, Phil Harrell, Dave Magee, Chad, Charlie, Jason, Alan, Melanie, Jan, and Brian.

I would like to thank NASA for their support under the GSRP, NASA NAG-1-1564. I also would like to thank NASA and NSF for their support under the grants, NGT-51102L and DMI-9420405, respectively. The computer costs were underwritten by the Systems Realization Laboratory of the Georgia Institute of Technology.

Special thanks go to the Dallas Cowboys who have given me the needed break on Sundays and the stress relief that brought me close to home.

And lastly, my feelings are aptly summarized by the following passage:

"The desire accomplished is sweet to the soul." Proverbs 13:19

I thank everyone who helped my dream come true. My heart and soul go out to you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

Algorithm
A mathematical rule or procedure for solving a problem.

Central Composite Design (CCD)
Central composite designs are first order fractional factorial designs augmented by additional points which allow the estimation of a quadratic surface model.

Complex Systems
Complex systems consist of a number of subsystems, each embodied by a particular set of components. Each component has its own working principle.

The compromise Decision Support Problem
A multiobjective decision model which is a hybrid formulation, incorporating concepts from Mathematical Programming and Goal Programming.

Concurrent Engineering
A systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of the product life cycle from conception through disposal, including quality, cost, schedule, and user requirements.

Control Variables
Variables which a designer has direct control over. A designer's control vector consists of the design and state variables of a particular subsystem.

Decision-Based Design (DBD)
The fundamental paradigm for designing and creating design methods, rooted in the notions that the principal role of a designer, in the design of an artifact, is to make decisions.

Decision Support Problem (DSP)

A formalization of a type of a decision made by a designer. Two types of decision support problems exist, namely, selection and compromise.

Decision Support Problem Technique

An implementation of Decision-Based Design. It is a technique to support human judgment in designing systems which can be manufactured and maintained through the solution of Decision Support Problems.

Design

A process of converting information that characterizes the needs and requirements for a product into knowledge about a product.

Design of Experiments

The formal techniques of planning an experiment so that appropriate data can be collected and analyzed by statistical methods, resulting in valid and objective conclusions.

Design Methodology

Includes the study of how designers work and think, the establishment of appropriate structures for a design process, the development and application of new design methods, techniques and procedures, and the reflection on the nature and extent of design knowledge and its application to design problems.

Design Variable

Independent variables which a designer must determine values for.

Discipline

A branch of knowledge or teaching. In the context of complex systems, a discipline is a subsystem that is governed by similar physical phenomena.

Efficiency

A measure of the swiftness with which information and design knowledge can be used by a designer.

Effectiveness
Represented by the correctness, completeness, and comprehensiveness of design decisions.

Game
In a general sense, a game is a set of rules completely specifying a competition, including the permissible actions of, and information available to each participant, the criteria for termination of the competition, and the distribution of payoffs. From a systems perspective, a game consists of multiple decision-makers who each control a specified subset of system variables and who each seek to minimize their own cost functions subject to their individual constraints.

Game Protocols
The relationships that exist among a group of players. The protocol dictates the interactions between and information available to each player in a game.

Game Theory
The study of the strategic interactions among players in a game.

Global Sensitivity Equations
A set of equations relating the local partial derivatives and the global full derivatives of the state variables with respect to the design variables using the chain rule.

Lexicon
A stock of terms used in a particular subject, style, or profession.

Mixed Discrete/Continuous Optimization
The modeling and solution of problems which contain both discrete/integer (only a finite number of possible values) and continuous (any real value) design variables.

Multidisciplinary Design Optimization
A methodology for the design of complex engineering systems that are governed by mutually interacting physical phenomena and made up of distinct interacting subsystems.

Players in a game

Classically, players may be people, groups of people or more abstract entities like computer programs or "nature". In design, a player is a disciplinary designer or design team and their associated analysis and synthesis design tools.

Rational Reaction Set (RRS)

Conceptually, the RRS is an embodiment of the decision making strategy of a player as a function of the decisions of another player.

Response Surface Methods (RSM)

A collection of statistical techniques for empirical model building and model exploitation. RSM seeks to relate a response to a number of predictors that affect it.

Satisficing

The idea that a solution is "good enough", but not necessarily the best.

State Variables

Dependent behavior variables which are functions of the design variables. A designer controls the state variables indirectly through the design variables.

Subsystem

A part of the system which may be a system itself, such as the propulsion system of an aircraft. A subsystem is considered to be a group of elements governed by the same physical phenomena. In other words, in this dissertation, a subsystem is considered to be *discipline-defined.*

System

A functionally related group of elements.

Tabu Search

An iterative improvement procedure which starts from an initial solution and attempts to determine a better solution by applying a greatest-descent procedure, subject to short and long term memory criteria.

Taxonomy

The science, laws, or principles of classification.

NOMENCLATURE

ALP	Adaptive Linear Programming
CCD	Central Composite Design
CE	Concurrent Engineering
DBD	Decision-Based Design
DOE	Design of Experiments
DSIDES®	Decision Support in the Design of Engineering Systems (computer software)
DSP	Decision Support Problem
FALP	Foraging-directed Adaptive Linear Programming
GA	Genetic Algorithms
NAND	Nested Analysis and Design
NASA	National Aeronautics and Space Administration
NORMAN®	Simulation experiment sequencing system
RRS	Rational Reaction Set
RSE	Response Surface Equation
RSM	Response Surface Methodology
SAND	Simultaneous Analysis and Design
SA	Simulated Annealing
s	State variable vector
TQM	Total Quality Management
TS	Tabu Search
x	Design variable vector
X	The control vector of a designer, $\mathbf{X} = \{\mathbf{x},\mathbf{s}\}$
Z	Deviation function in a compromise DSP

SUMMARY

Consider the statement, "A system has two coupled subsystems, one of which dominates the design process. Each subsystem consists of discrete and continuous variables, and is solved using sequential analysis and solution." To address this type of statement in the design of complex systems, three steps are required, namely, the embodiment of the statement in terms of entities on a computer, the mathematical formulation of subsystem models, and the resulting solution and system synthesis.

In complex system decomposition, the subsystems are not isolated, self-supporting entities. Information such as constraints, goals, and design variables may be shared between entities. But many times in engineering problems, full communication and cooperation does not exist, information is incomplete, or one subsystem may dominate the design. In addition, these engineering problems give rise to mathematical models involving nonlinear functions of both discrete and continuous design variables.

In this dissertation an algorithm is developed to handle these types of scenarios for the domain-independent integration of subsystem embodiment, coordination, and system synthesis using constructs from Decision-Based Design, Game Theory, and Multidisciplinary Design Optimization. Implementation of the concepts in this dissertation involves testing of the hypotheses using example problems and a motivating case study involving the design of a subsonic passenger aircraft.

CHAPTER 1

FOUNDATIONS FOR INTEGRATED SUBSYSTEM EMBODIMENT AND SYSTEM SYNTHESIS

In this dissertation the principal goal is to:

> Develop a framework for the decision support of formulating a multidisciplinary design problem, decomposing the problem into disciplinary subproblems, modeling the resulting interactions according to realistic assumptions, and solving and coordinating the disciplinary mathematical models.

To establish some context, the following terms are defined:

- ***System*** - a functionally related group of elements or components.
- ***Subsystem*** - a part of the system which may be a system itself, such as the propulsion system of an aircraft.
- ***Complex system*** - a system composed of a number of subsystems where each subsystem is embodied by a particular set of components. Each component has its own working principle. In designing complex systems, it is difficult to make tradeoffs without understanding the complete relationships between all of the components that constitute a subsystem and all of the subsystems that constitute the system.
- ***Design team*** - a group of designers who work on the design of a particular subsystem of a complex system and their associated analysis and synthesis computer tools.
- ***Embodiment*** - to represent in concrete form. Concrete form could be mathematical, geometrical, or prototypical, for instance. Embodiment in this work means to represent *numerically*.

This chapter begins with the motivation and background for the dissertation. In Section 1.1, the overall context of this work is presented which includes discussion of three topics, namely, interactions in design, classification systems, and solution of design models. As a frame of reference, in Section 1.2, background material on Decision-Based Design, the compromise DSP, the ALP Algorithm, and Game Theory is presented. The principal goal of this work is summarized in Section 1.3. Included in Section 1.3 are the fundamental questions to be addressed. Associated with the implementation strategy for achieving the principal goal, the major tasks are identified, the research hypotheses are introduced, and the verification strategy for the dissertation is presented. In Section 1.4, the contributions of this work are justified by summarizing the deliverables and establishing the scientific relevance of this dissertation. The organization of the chapters and appendices of this dissertation are given in Section 1.5.

1.1 MOTIVATION AND BACKGROUND

The fundamental contributions of this dissertation are:

- techniques for implementing game theoretical protocols in the design of complex systems characterized by multiple disciplinary design teams. Developing and integrating game theoretical constructs in the design of complex systems is a primary contribution.
- an effective solution scheme for mixed discrete/continuous design problems. The analogy and constructs that guide the behavior of the scheme are a primary contribution.

Associated with the fundamental contributions, the secondary contributions of this dissertation are:

- a three-level lexicon for the classification of the design of complex systems and their associated design processes. In this contribution, a representation of the product and process is abstracted using linguistic entities.
- a formal proof of the characteristics of a transformation function, as a technical criticism. Nonlinear optimization theory is used in a proof by induction that addresses convex transformations.

To establish the motivation and background for these contributions, consider the statement, "A system has two coupled subsystems, one of which dominates the design process. Each subsystem consists of discrete and continuous variables, and are solved using sequential analysis and solution." In order to address this type of statement in the design of complex systems, three steps are required, namely, (1) the embodiment of the statement in terms of entities on a computer, (2) the mathematical formulation of the coupled subsystems problems, and (3) the resulting solution and coordination at the subsystem and system levels. Developing an *algorithm* to integrate these three domain-independent steps

in complex systems design is the fundamental motivation for this dissertation. In Figure 1.1, the fundamental research areas of this dissertation, which correspond to the three required steps to address the previous statement, are given in the context of the title. In the top, left corner of Figure 1.1, the foundation for the algorithm is developed through the classification of the problem and process. In the top, right corner, the mathematical formulation of the coupled subsystem models is developed (corresponding to the "integrated subsystem" part of the title). On the bottom row, the capabilities to solve the subsystem problems, while coordinating them into an functional system are developed (corresponding to "embodiment and system synthesis" part of the title). The corresponding section numbers, where each area is discussed are given in Figure 1.1. Figure 1.1 is used throughout Section 1.1 as a frame of reference.

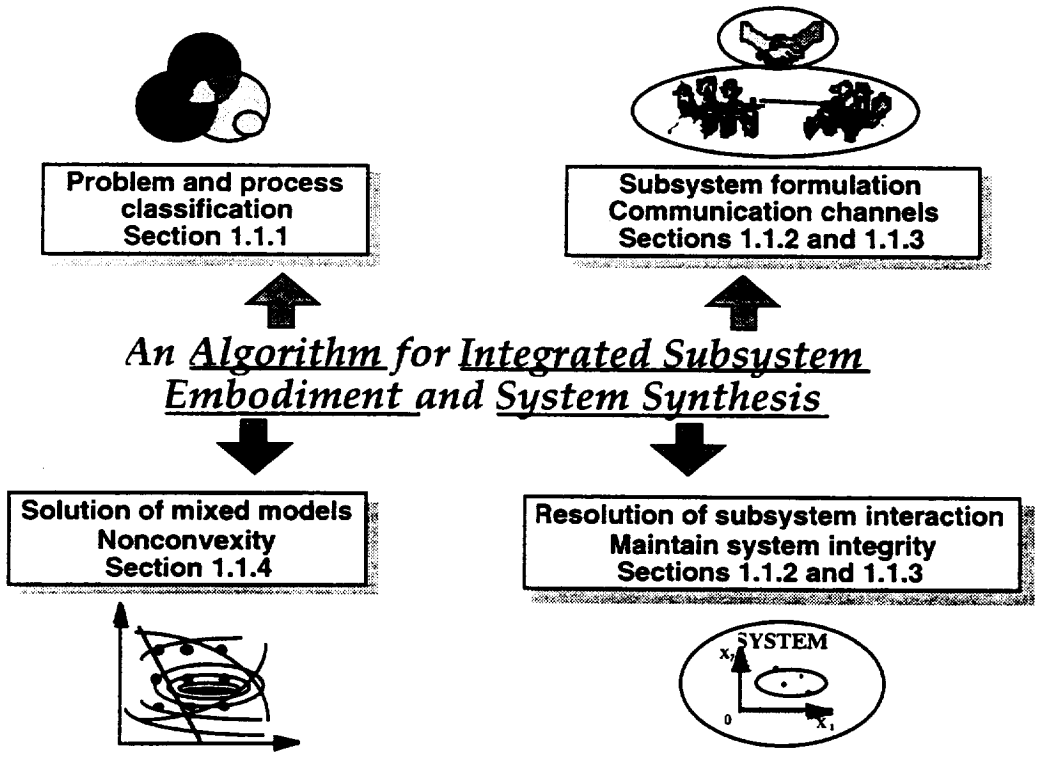


Figure 1.1. Areas of Focus in this Dissertation

The fundamental, big picture umbrella under which this dissertation can be classified is Complex Systems, as shown in Figure 1.2. The focus under complex systems is developing tools and techniques for subsystem embodiment and system synthesis. The research areas of interest are product and process classification, subsystem interactions in design, and the solution of complex design models, precisely the three steps mentioned earlier. These research areas are introduced in the context of design, engineering, and science in Section 1.1. The fourth secondary research area, a generic mathematical decision-making construct is addressed in Section 1.2.1. Literature reviews of these areas are provided in Chapter 2. The various tools relating to each research area are introduced in Figure 1.2 and are presented in detail in Chapter 3.

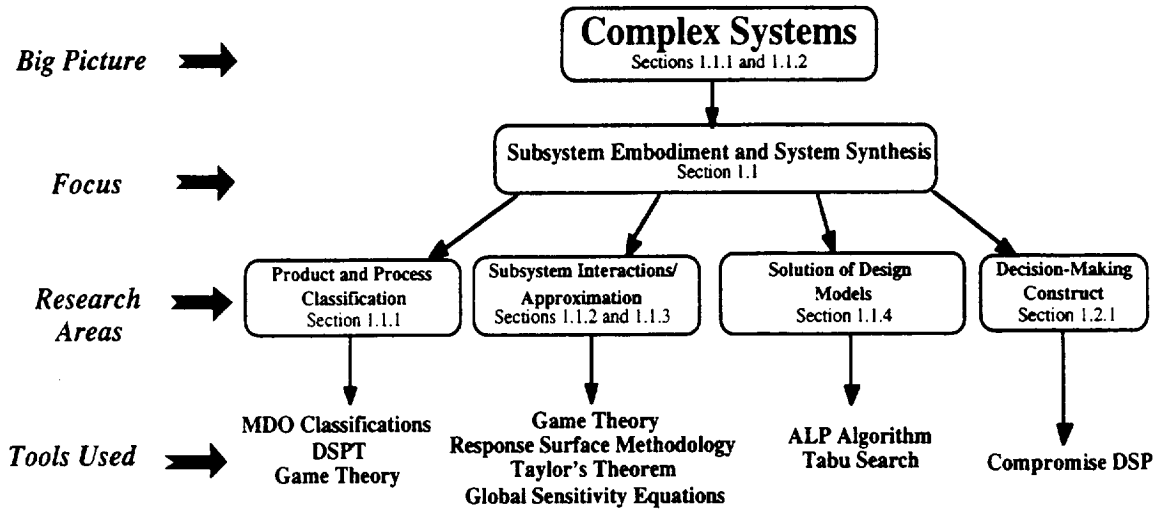


Figure 1.2. The Structure of the Literature Background Review

1.1.1 Engineering Design Processes and the Design of Complex Systems

Design is quite unlike invention in that there is some type of rational methodology to solving problems. Invention involves repeated trial-and-error experimentation whose succes

An Algorithm for Integrated Subsystem Embodiment and System Synthesis

sometimes even depends upon luck. In design, there is a need to plan the process of designing. This planning takes the form of meta-design (Mistree, et al., 1990b, Rogan and Cralley, 1990) where the product and process are partitioned and planned. Meta-design is defined as "the design of the design process" (Mistree, et al., 1990b). Various theories and methodologies have been developed in the engineering design community for describing and improving engineering design processes. Although models of design processes vary significantly for different streams of research, there are some models that are widely acceptable and make intuitive sense to many designers. An example is the four major design phases identified by Pahl and Beitz (Pahl and Beitz, 1984):

- *Clarification of the task* – collection of information about the requirements to be embodied in the solution and also about the constraints.
- *Conceptual design* – establishment of function structures, the search for suitable solution principles and their combination into concept variants.
- *Embodiment design* – starting from the concept, a designer determines the layout and forms, and develops a technical product or system in accordance with technical and economic considerations. Embodiment design is sometimes called preliminary design.
- *Detail design* – all the details of the final design are specified and manufacturing drawings and documentation are produced.

The design of complex systems follows similar phases. Using aircraft design as an example, in Figure 1.3, the aircraft design process is roughly divided into four major phases, i.e., conceptual, preliminary, detail design, and production and support (Schrage, 1992). In Figure 1.3, the flow between different phases and the major tasks implemented in each phase are illustrated. The focus of this dissertation is on the first two stages of Figure 1.3, conceptual and preliminary design. Although, the techniques developed herein can be applied at any point along a design timeline using the appropriate assumptions, the

primary areas of application are when distinct subsystems can be identified and are accounted for in development of a system configuration. More detailed descriptions of the decisions made in each phase and the disciplines involved in aircraft design are provided by Bond and Ricci (Bond and Ricci, 1992) and Raymer (Raymer, 1989).

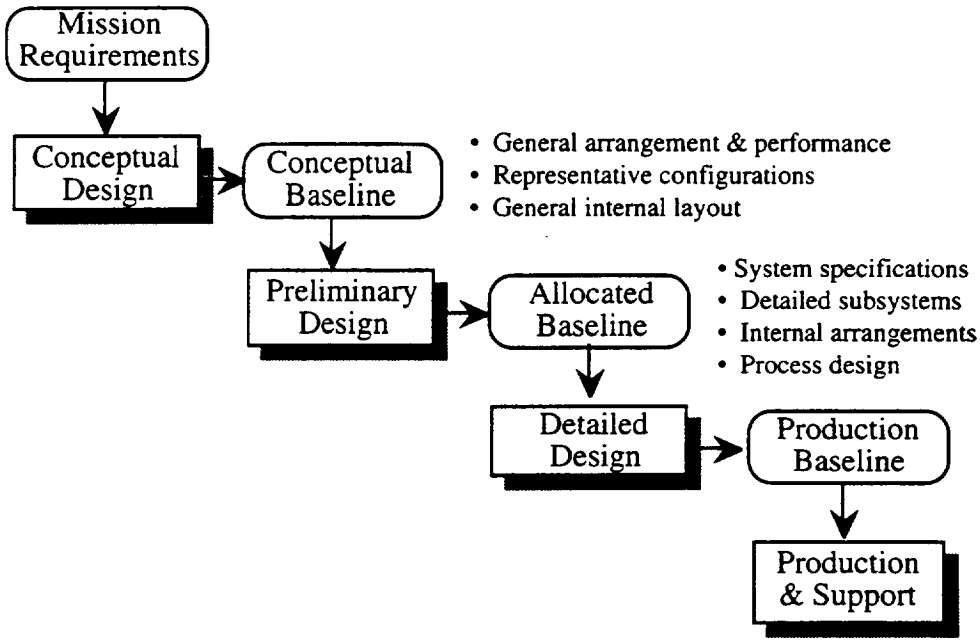


Figure 1.3. Four Phases in Aircraft Design (Schrage, 1992)

In aerospace engineering, FLOPS, the **FL**ight **OP**timization **S**ystem (McCullers, 1993) and ACSYNT (ACSYNT Institute, 1992), the **AirC**raft **SYNT**hesis programs are the two most popular programs for the conceptual design of aircraft. Both of them employ a number of discipline specific modules to perform aircraft analysis and synthesis. However, the modules are all contained *within the same* computer simulation program. In the later stages of design, when domain-dependent codes and tools are used by different design teams who may be separated by geography, by computer platforms, or by organizational structure, the luxury of having *one* encompassing analysis code is not

available. Different design teams may prescribe to using different design methodologies, analysis routines, and synthesis tools. The task of partitioning and planning the processes to design a complex system, being performed by multiple design teams is a difficult task. Each subsystem team may perform its own form of meta-design, but at the system level, meta-design or process design is rarely established. In (Balling and Sobieski, 1994, Cramer, et al., 1994), process classification schemes for complex, multidisciplinary systems are described. This is one of the primary capabilities of Section 1.1 which is necessary in the design of complex systems. Within each subsystem, a form of the linguistic design processes described in Figures 1.2 and 1.3 may certainly be employed, but at the *system level*, a common *linguistic* form of classification, communication, and comparison is needed (Balling and Sobieski, 1994, Cramer, et al., 1994). It is among the interests in this dissertation to expand these forms of classification from a *decision-based perspective*. Foundational to these interests is the motivation to develop *domain-independent* methods and tools that can facilitate the use of domain-dependent analysis/synthesis codes for the design of complex systems.

Broadening the scope to the *product realization processes*, in Figure 1.4 a typical product realization process is shown. The process flows from left to right along the x-axis, from conceptual design to preliminary design to support and beyond. It is recognized that conceptual design is not the origin of this process. Needs recognition, problem definition, etc. must occur before conceptual design. At some point in the process, distinct disciplinary subsystems can be identified (y-axis in Figure 1.4). Design teams for each subsystem must embody their subsystem which requires the solution of a disciplinary design model (z-axis in Figure 1.4). The focus of this dissertation is on the y and z-axes of Figure 1.4. In other words, the focus is on a snapshot in time during a design process. Certainly at different times in a design process, *relationships and model characteristics may*

change, but one of the advantages of the algorithm developed in this dissertation is its lack of dependence on time. Therefore, as long as a system can be represented mathematically on a computer, the algorithm can be utilized, even though the relationships among disciplines/subsystems and the types of decision support tools may change.

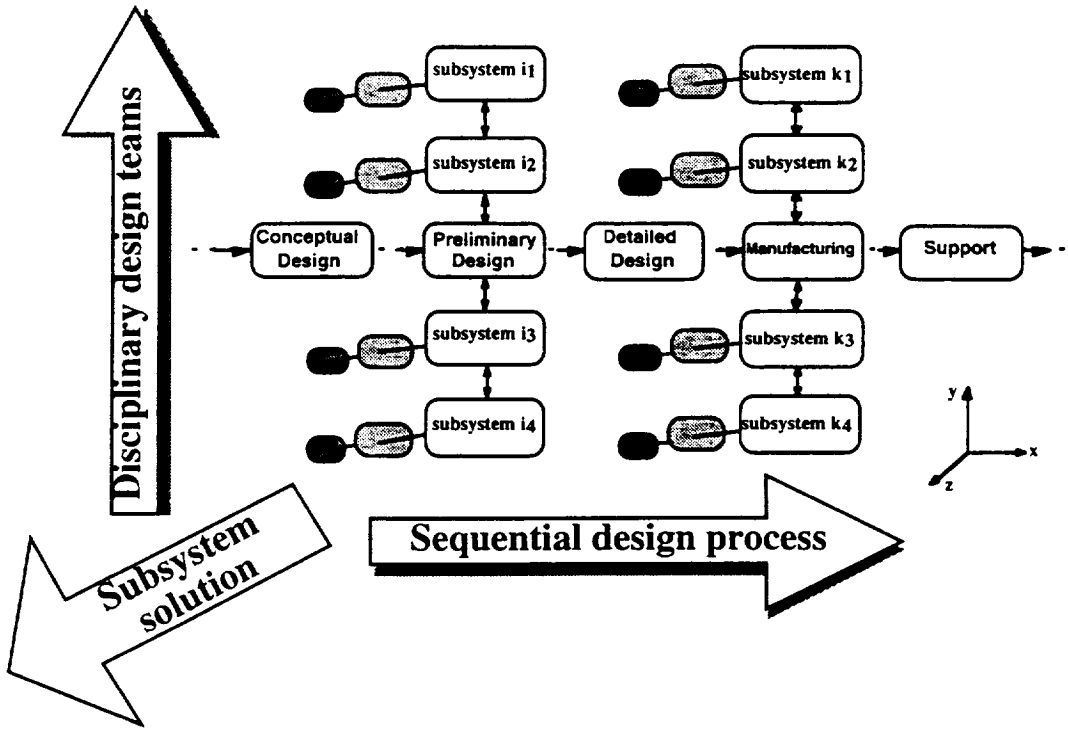


Figure 1.4. A Typical Product Realization Process

Decomposing a problem into smaller problems is a common approach in the sciences. In (Simon, 1982) it is asserted that the design of a complex system can be facilitated through the use of decomposition and coordination techniques.

> *To design such a complex structure, one powerful technique is to discover viable ways of decomposing it into semi-independent components corresponding to its many functional parts. The design of each component can then be carried out with*

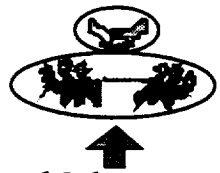
some degree of independence of the design of others, since each will affect the others largely through its function.

Consider the typical approach in chemistry to such a problem:

Often, the only possible course is to decompose a large system into smaller ones and to analyze each subsystem in semi-isolation with simplifying assumptions. Surrounding processes ... are held in a fixed state (Courtois, 1985).

In design, however, there is no such thing as a "fixed state." Design teams are constantly making decisions that affect not only their own subsystems but affect the other subsystems as well. Assuming the other sub-problems are in a "fixed state" while working on one sub-problem is very limiting. Therefore, modeling interactions in design mandates accounting for *changing* states of the other subsystems. However, many times the requirements and objectives of the design teams are in conflict with each other. Concurrent and mutual satisfaction of multiple design teams is difficult and rare. The nature of conflict in design is discussed in the next section.

1.1.2 The Nature of Conflict in Design



An Algorithm for Integrated Subsystem Embodiment and System Synthesis

Specialization and generalization are two concepts diametrically opposed, in theory. In practice, however, the two concepts are struggling to find identities in the same marketplace. Companies preach customization of their product and processes, but also emphasize the broad applications of their practices, processes, and engineers. At an engineering level, specialization must occur. In the design of complex systems, such as aircraft, the system must be decomposed into smaller sub-problems which can be handled by groups of specialists in specific disciplines. On the other hand, the disciplinary

information must be coordinated to produce a functional system. This task becomes one for the company management which sees the broader, more general, big picture issues. However, what happens when disciplinary specialists are each governed by their own "generalist" who prescribes a set of local objectives for the group to meet? Or, what happens when the disciplinary groups are separated geographically, informationally, or organizationally? The general, system level objectives may get lost in the details. Each disciplinary group typically resorts to fulfilling their own requirements while leaving the consideration of nonlocal requirements to other groups. Although, this approach may be advantageous locally, when a general view of the system is taken, the individually motivated decisions of the groups many times are not advantageous for the system as a whole.

From De Bono (De Bono, 1985), who refers to conflicts in design,

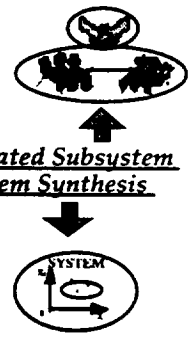
> *We find a genuine clash of interests. The parties want things which are incompatible ... A basic design technique is move away from the obvious clash point and to explore benefits and values in various modifications of the situation.*

The separation between the disciplinary design teams is made wider because of the different approaches to analysis, synthesis, and optimization each group may employ. Uniting each local approach under a global conceptual umbrella is a difficult task. The distinct nature of each subsystem necessitates the capability of a design framework to handle a wide variety of methods and approaches to similar problems. This capability must stretch across a design process, from the meta-design stage to the conceptual design stage, to the detailed and embodiment stages.

In light of the previous discussions, this dissertation is motivated by the need to understand and model the interactions in the design of complex systems in order to develop design methods and tools which can assist decision makers in making decisions throughout a design process. Specifically, the primary interest in this dissertation is to develop an algorithm *capable of handling the domain independent tasks of classifying approaches to a complex design problem, modeling realistic interactions among disciplinary subsystems, and resolving embodiment and coordination problems*. In addition, this dissertation represents an effort to integrate concepts from Concurrent Engineering, Game Theory, and Decision-Based Design with the research issues of Multidisciplinary Design Optimization.

Although cooperation in design is ideally the best scenario, in practice, it is not the most common. The nature of cooperation and communication in design, and the implications and difficulties therein are explored in the next section.

1.1.3 Cooperation and Communication: The Ideal Cases



The principles of give and take pervade our society. In *Descent of Man* (Darwin), Charles Darwin was aware of the role of cooperation in human evolution, writing, "the small strength and speed of man, his want of natural weapons are more than counterbalanced by his ... social qualities, which lead him to give and receive aid from his fellow men." The notions of cooperation and mutual help are further explored in societal and cultural environments in (Nowak, et al., 1995). It is asserted that cooperation has assisted the processes of evolution in everything from humans to small organisms. Isolated factions of noncooperation may exist but it is the innate cooperative drive of society and nature that pervades. These principles can be

mapped to the processes of design. Since engineering design is a human-centered activity usually performed by multiple designers, opportunities for cooperation exist from the personal level to the analysis level. Engineering design, because of its inherent reliance on cooperation among designers and design teams, is certainly not a forum for noncooperation to flourish at any level of detail.

In the early stages of complex system design, system level approaches and tools can be used (e.g., FLOPS and ACSYNT in aircraft design). Cooperation and communication are not a problem since the designers at the system level are each focused on the same problem and are using the same design tools. But at some point, the system level problem becomes too complex and it must be decomposed into smaller problems. With reference to Figure 1.4, this typically occurs in the conceptual or preliminary stages. When subsystems are identified, design teams assigned, and various methods and tools are employed by each group, ensuring cooperation and communication becomes a significant hurdle. Researchers addressing the issues of cooperation have taken a distinctly different approach to the problem than what is observed and practiced in industry. Previous research in modeling the interactions among disciplines has assumed some sort of cooperation and communication, either implicit or explicit (Bloebaum, et al., 1992, Renaud and Gabriele, 1993, Sobieszczanski-Sobieski, 1988), but in industry practice has experienced a somewhat different environment. From (Duffey, et al., 1996) for example, it is found that the leaders of two design teams designing aircraft in the same company for several years had never met face to face until only recently. The cooperation and communication modeled in academic research is usually not applicable in industrial contexts. More times than not, design processes are still largely sequential (Sobieszczanski-Sobieski, et al., 1984), and many times the design groups *do not* even communicate (Duffey, et al., 1996). Furthermore, there exists computational difficulties in ensuring cooperation and

communication in an industrial context. The gap between the research perspective and the industrial perspective of a multidisciplinary design problem is shown in Figure 1.5.

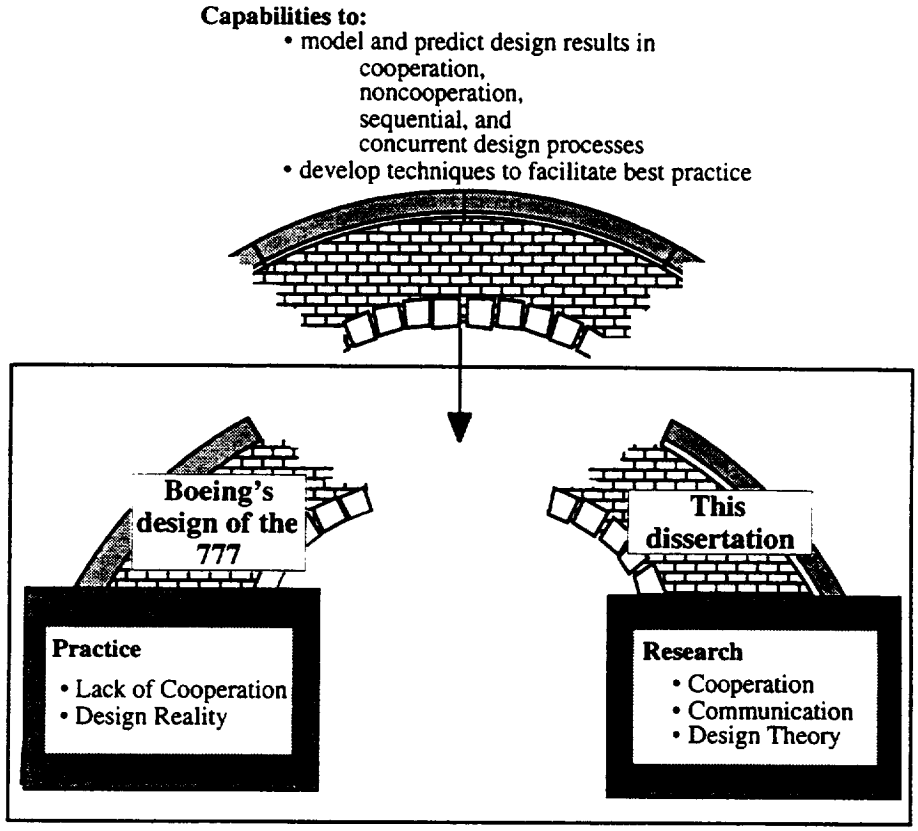
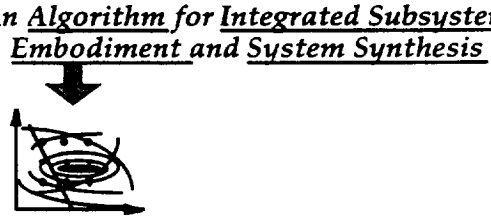


Figure 1.5. Bridge Between Practice and Research in Multidisciplinary Design Problems

To bridge the gap between research and practice from the industrial side, companies such as the Boeing Aircraft Company are making significant strides. In their recently publicized design of the 777 aircraft, Boeing has had great success revamping their design processes to facilitate cooperation and communication among the various disciplinary design teams. Principles from Concurrent Engineering (CE) (Kusiak, 1993) and Total Quality Management (Brassard, 1989) were successfully applied at the personal/organizational level and to a lesser extent at the computational/mathematical level. Extension of CE

principles from the personal level to the mathematical level is not a trivial conceptual jump. It is part of the motivation of this dissertation to provide mathematical techniques and tools for applying principles of CE and Systems Engineering. While these efforts are paying great dividends (Duffey, et al., 1996), there still exists a gap between research and practice in modeling interactions in complex systems design. A contribution of this dissertation is to help bridge this gap from an academic research perspective. A major point of departure in this dissertation is developing methods and techniques to not only model cooperation in complex systems design, but also model sequential processes and processes where the design teams do not communicate or cooperate. Ideally, the keystone of this bridge, as shown in Figure 1.5, would be decision support tools with the capability to model and predict results in cooperation, noncooperation, sequential, and concurrent design processes, and techniques to facilitate and implement best practice strategies. Full construction of such a bridge requires further work both from the industrial and academic perspectives integrated with the concept of *organizational or enterprise design* where the structure of corporations are determined based on design product and process issues. Of course, developing the capability of modeling different interaction protocols means developing the means to solve the models. In the next section, hurdles in the solution of the models are discussed in the context of complex systems.

1.1.4 Mathematical Hurdles to System Embodiment



In the discussion in Sections 1.1.1 and 1.1.2, analysis and synthesis tools are referred to in the third person. That is, they exist, and the effectiveness of them is left up to the disciplinary designers. From the title of this dissertation, the phrase "subsystem embodiment and system synthesis" connotes ideas of

solving multiple design models. In order to complete to the "algorithm" of this dissertation, a solution scheme is developed to help designers solve the disciplinary models. Two characteristics of complex design problems are addressed in the solution scheme, mixed discrete/continuous problems and nonconvex functions.

In complex systems, the design variables, for the most part, can be set to any real value. In other words, the design variables are continuous. There are, however, times when variables exist which can only take on certain values. For instance, the number of engines in an aircraft can only be an integer, and the number of teeth on a gear can only be an integer. Moreover, when existing components are selected "off the shelf," there are only a finite number of possible values. For instance, gears usually come in standard discrete sizes. Bolts and springs usually come in similar standard sizes. Analyzing functions of these types of variables, integer and discrete, presents mathematical challenges, since the derivatives of the functions with respect to the integer or discrete variables *do not exist.* Therefore, any kind of approximation or optimization technique which requires derivatives can not be used. In fact, the solution of these mixed problems is identified in (Papalambros, 1995) as being "one of the most daunting problems in design optimization."

Analytical functions that describe the behavior of a complex system are rarely simple, linear functions. They typically are complex, nonlinear, n^{th}-order equations of multiple variables. Optimization algorithms have difficulty handling functions of this type, which are neither convex or concave, even in small regions. But highly nonlinear equations in complex systems design are a fact of life. Therefore, techniques for handling nonconvexity are necessary for effective and reliable solutions for design models of complex systems. Techniques are well established for finding solutions to convex problems. However, with

nonconvex problems, often heuristic approaches are used, yielding mixed results (Papalambros, 1995).

It is among the contributions of this dissertation to develop a solution scheme to handle mixed discrete/integer/continuous design models characterized by highly nonlinear analysis equations. Foundational to this dissertation, a mindset of *description* is taken as opposed to *prescription*. A primary benefit of using the algorithm developed in this work is the capability to *describe* the results and ramifications of various complex product and process structures through the classification, modeling, and solution of design problems. The descriptive opportunities of the algorithm are presented in the next section.

1.1.5 Opportunities: A Descriptive Approach

In Sections 1.1.1 through 1.1.4 different aspects of the design of complex systems are described as a means to establish the context of the dissertation. The areas of focus are the *classification of design product and process, interactions in design, and the solution of design models*. These are the areas identified in Section 1.1 as being paramount to formulating, modeling, and solving complex design problems. These three areas map one-to-one to the three steps of the algorithm for subsystem embodiment and system synthesis presented in Chapter 3. In Webster's (1984), "algorithm" is defined as

A mathematical rule or procedure for solving a problem.

The term "procedure" connotes a sense of prescription. In other words, an algorithm *prescribes* a set of steps to solve a problem. In a sense, the algorithm in this dissertation is prescriptive, but more importantly, it provides a framework to explore formulating, modeling, and solving a complex design problem. However, the true benefit of the algorithm is its *descriptive* power. The algorithm provides a dynamic framework, and when exercised, can describe the results of different design scenarios when multiple design

teams are involved. Therefore, the mindset for this dissertation is more *descriptive* than *prescriptive*. The results are meant to describe the various resulting product and process implications when a certain prescriptive approach to design is taken. In order to address the issues raised in Section 1.1, several hypotheses are formed in Section 1.3. These hypotheses are constructed based upon a solid technology foundation, rooted in Decision-Based Design and Game Theory. The necessary background for the foundational areas is presented in the next section.

1.2 FRAME OF REFERENCE

In this section, the necessary technology base for the dissertation is given. Detailed presentations of the foundations for the developments of this work are presented in Chapter 3. Three fundamental starting points are offered in this section, the Compromise DSP (and more generally, Decision-Based Design), the Adaptive Linear Programming Algorithm, and Game Theory.

1.2.1 Decision-Based Design and the Compromise DSP

There have been several reviews of design literature (Andreasen, 1987, Cross, 1989, De Boer, 1989, Finger and Dixon, 1989a, Finger and Dixon, 1989b, Hubka and Schregenberger, 1987, Pahl and Beitz, 1984). Although these reviews focus on different aspects of design literature, such as the evolution of design theory, state of the art methods, and research trends, one common characteristic of these reviews is that they all aim at supplying the missing elements for making design more "scientific". It is asserted that, as an emerging science-based discipline, design is still in its pre-theory stage; a lot of experimental studies are still needed. There are various ways to approach design in the

current design research community, for example, prescriptive approaches to design (Hubka, 1982, Pahl and Beitz, 1984), axiomatic approaches (Suh, 1990, Takala, 1987, Tomiyama and Yoshikawa, 1987), decision-based design approaches (De Boer, 1989, Mistree, et al., 1990b, Ostrofsky, 1977), and mathematical-oriented optimization approaches (Hubka, et al., 1988, Vanderplaats, 1984). Moreover, in recent years there has been a growth of artificial intelligence (AI) principles being applied to design (Brown, 1985, Brown and Chandrasekaran, 1986). Independently of the approaches or methods used to plan, establish goals and model systems, designers are and will continue to be involved in two primary activities, namely, *processing symbols* and *making decisions*. Therefore, it is asserted that the process of design, in its most basic sense, is a series of decisions. By focusing upon decisions, a description of the processes can be written in a common "language" for teams from the various disciplines -- a language that can be used in the process of designing. It is this language of decisions that is used to build the lexicon addressed in Section 1.1.1 that can be used to classify processes in complex systems design.

A definition of the term *designing* is as follows (Kamal, et al., 1987, Mistree, et al., 1989):

> ***Designing** is a process of converting information that characterizes the needs and requirements for a product into knowledge about a product.*

In this dissertation, Decision-Based Design (DBD) (Mistree, et al., 1990b, Shupe, 1988), a term coined to emphasize a different perspective from which to develop methods for design, is used as the design paradigm. The paradigm is based on the premise that the principal role of a designer is to make decisions (Mistree, et al., 1990b). This seemingly

limited role is useful in providing a starting point for developing design methods based on paradigms that spring from the perspective of decisions being made by designers (who may use computers) as opposed to design that is assisted by the use of computers, optimization methods (computer-aided design optimization), or methods which evolve from specific analysis tools such as finite element analysis.

It is recognized that the implementation of DBD can take many forms. The implementation form used in this dissertation is the Decision Support Problem (DSP) Technique (Mistree, et al., 1993c, Muster and Mistree, 1988), which is developed as a technique that supports human judgment in designing systems which can be manufactured and maintained. In a computer assisted environment, support for the designer is provided in the form of solutions to Decision Support Problems. Formulation and solution of DSPs provide a means for making the following types of decisions:

> *Selection* - the indication of a preference, based on multiple attributes, for one among several feasible alternatives (Kuppuraju, et al., 1985b, Mistree, et al., 1994, Mistree, et al., 1988).
> *Compromise* - the improvement of a feasible alternative through modification (Bras and Mistree, 1993, Chen, et al., 1994b, Karandikar, et al., 1990, Mistree, et al., 1988).
> *Coupled or hierarchical* - decisions that are linked together - selection/selection, compromise/compromise and selection/compromise decisions may be coupled. (Bascaran, 1990, Bascaran, et al., 1989, Karandikar, 1989).

These types of decisions may also be implemented in an uncertain or conditional environment where decisions account for the risk and uncertainty of the outcome (Allen, et al., 1992, Allen, et al., 1989, Bhattacharya, 1990, Zhou, et al., 1992), or by a rule- based or heuristic approach where reasoning and rules of thumb are used (Kamal, et al., 1992).

Applications of DSPs include the design of ships, damage tolerant structural and mechanical systems, the design of aircraft, mechanisms, thermal energy systems, design using composite materials and data compression. A detailed set of references to these applications is presented in (Mistree, et al., 1990a). The software for implementing the DSP Technique is called DSIDES (**D**ecision **S**upport **i**n the **D**esign of **E**ngineering **S**ystems) (Mistree, et al., 1993a). As a general framework for solving multiobjective, nonlinear optimization problems, a compromise DSP can be used to model each of the aforementioned decisions. The compromise DSP is used in this dissertation as the fundamental mathematical construct for modeling disciplinary-dependent problems and the strategic interactions among them. The general word formulation of a compromise DSP is given as follows.

Given
An alternative to be improved through modification.
Domain analysis information
Find
System *design variables*
Deviation variables associated with the system *goals*
Satisfy
System *constraints*
System *goals*
Bounds on the *system variables*
Minimize
Deviation Function

In a compromise DSP, the *design variables* and *deviation variables* (which measure the deviation between the achievement and target values of the system *goals*) are found subject to satisfying system *constraints*, *goals*, and variable *bounds*. The objective in a compromise DSP is to *minimize the deviation function*, which quantifies the "goodness" of a design. A mathematical overview of the compromise DSP is provided in Section 3.4.4. As part of DSIDES, the solution algorithm for solving compromise DSPs with continuous variables is called the Adaptive Linear Programming Algorithm (Mistree, et al., 1993a). A

brief introduction and discussion of the features of the ALP Algorithm are given in the next section, Section 1.2.2, and detailed discussions are reserved for Sections 3.4.4 and 6.2.2.

1.2.2 The Adaptive Linear Programming Algorithm and the G-function

Solutions to the compromise DSPs can be found using different optimization methods (Mistree, et al., 1993a). The choice of the optimization method depends, to a certain extent, on the problem. Solution algorithms fall into two categories, namely,

- those that solve the exact problem approximately, and
- those that solve an approximation of the problem exactly.

Gradient-based methods, pattern search methods, and penalty function methods fall into the first category whereas methods involving sequential linearization fall into the second category. The ALP Algorithm is based on the sequential linearization of a nonlinear problem. At each stage the solution of the linear programming problem is obtained by a Multiplex algorithm based on (Ignizio, 1985b). Three important features contribute to the success of the ALP algorithm, namely,

- the use of second-order terms in linearization,
- the normalization of the constraints and goals and their transformation into generally well-behaved convex functions in the region of interest, and
- an "intelligent" constraint suppression and accumulation scheme.

These features are described in detail in (Mistree, et al., 1993a) and briefly described in Section 6.2.2. Nonconvex functions in optimization problems are difficult to handle and can cause solutions schemes to find inferior solutions or diverge. However, it is asserted in (Mistree, et al., 1993a) that the ALP algorithm does not have problems dealing with

nonconvex constraints which invariably occur in the real-world engineering design. The effectiveness of the function that is used in the ALP Algorithm to transform nonconvex equations in well-behaved convex equations is investigated in this dissertation. To introduce this transformation, if $C_i(\underline{X})$ and $D_i(\underline{X})$ represent the capability and demand placed on a system in mode i, then, a system constraint is

$$C_i(\underline{X}) \geq D_i(\underline{X}) \quad \text{or} \quad C_i(\underline{X}) - D_i(\underline{X}) \geq 0$$

In the normalized, dimensionless form (Mistree, et al., 1993a) the preceding equation becomes

$$(C_i(\underline{X}) - D_i(\underline{X}))/ (C_i(\underline{X}) + D_i(\underline{X})) \geq 0$$

and hence

$$g_i(\underline{X}) = (C_i(\underline{X}) - D_i(\underline{X}))/(C_i(\underline{X}) + D_i(\underline{X})) .$$

If $r_i(\underline{X}) = C_i(\underline{X})/D_i(\underline{X})$ for a system constraint, then

$$g_i(\underline{X}) = (r_i(\underline{X}) - 1)/(r_i(\underline{X}) + 1) . \tag{1.1}$$

In a compromise DSP, a nonlinear system constraint is represented as

$$(r_i(\underline{X}) - 1)/(r_i(\underline{X}) + 1) \geq 0 \tag{1.2}$$

or,

$$g_i(\underline{X}) \geq 0 .$$

Similar derivations of the g-function are given in (Mistree, et al., 1993a) for system *goals*. The function $g_i(\underline{X})$ is normalized and is therefore nondimensional. This simplifies the task of solving a compromise DSP with constraints expressed in different physical units. The preceding is asserted to be the second important feature of the algorithm. In this dissertation a proof of contradicting this assertion is formalized in Section 3.5.

The ALP Algorithm is used as a fundamental starting point in this dissertation for the development of a solution algorithm for mixed discrete/continuous design problems. The final starting point is game theory, introduced in the next section.

1.2.3 Game Theory as a Decision Support Tool

In a general sense, a "game" is a set of rules completely specifying a competition, including the permissible actions of, and information available to each participant, the criteria for termination of the competition, and the distribution of payoffs (Websters, 1984). From more of a systems perspective, a "game" is defined as follows:

> *Definition 1.1. A game consists of multiple decision-makers who each control a specified subset of system variables and who each seek to minimize their own cost functions subject to their individual constraints (Myerson, 1991).*

In this dissertation, a *designer* is considered a *decision-maker.* Therefore, the definition of a game can be applied directly to a design process characterized by multiple designers who each try to minimize their own cost functions (or maximize performance functions) subject to local constraints. *Game Theory* is the study of the strategic interactions of such games (Von Neumann and Morgenstern, 1944). It is asserted in this dissertation that principles from game theory can be applied to design situations to understand and model the complex relationships among subsystems in the design of complex systems. Game theory has typically been used extensively in economics, business, and military applications. In these applications, the players in the game are large companies, industries, or national military forces. Classically, players may be people, groups of people or more abstract entities like computer programs or "nature". In design a player is defined as follows:

Definition 1.2. A player in design is a disciplinary designer or design team and their associated analysis and synthesis design tools.

In Def. 1.2, again the synergy between a human and computer is found. This concept is introduced in Section 1.2.1, where computer capabilities enhance the designer's abilities to make decisions. In design, since the player is the decision-maker, in order for players to cooperate, cooperation is required at the personal level (designer level) and at the mathematical level (analysis and synthesis levels). In this dissertation, cooperation from a mathematical perspective is explored in the context of game theory.

In game theory, a fundamental assumption is the inherent rationality of the players in the game, but as pointed out in (Bertalanffy, 1968), "human behavior ... falls far short of the principle of rationality." By asserting that the players in design are not only the decision-makers, but their associated analysis and synthesis tools as well, this principle of rationality is satisfied to some extent. The rationality is embodied within the analysis that describes the behavior of the system in terms of physical and mathematical laws. Rationality from a human perspective can be disputed, but from a perspective based on the laws of nature, physics, and mathematics, rationality cannot be disputed, e.g., if the physics of an aircraft problem predicts that an aircraft will not fly, it is safe and rational to assume that it will not fly. Although, analysis is only a support tool for the designer whose responsibility it is to make the final decision, rationality can not be guaranteed, but it is assumed in this dissertation that rationality exists beyond a reasonable doubt.

Application of game theory classically has had two goals. The first goal is a descriptive goal of understanding why the players behave as they do. This includes describing the results of a certain game structure. The second goal is a more practical goal of being able to

advise the players of the game as to the best way to play, or the best strategy to take. For instance, two players could strike a mutually profitable compromise, but each could possibly gain still more by withholding its contributions and information and exploiting the other player. In design, disciplinary design teams, *working for the same company*, should be working towards a common goal: designing a product that meets the customer requirements as closely as possible. Therefore, competitive notions that are present in games such as chess, poker, bridge, baseball, and so on, are not present (or at least should not be present) in design. This is the fundamental difference between classical games studied in game theory and design:

> *The competitive behavior of players in classical games occurs because of the dichotomy of goals. One player wants to win or maximize his profit, and the other player also wants to win or maximize his profit. Simultaneous satisfaction of both players' goals cannot occur. Whereas in design, although disciplines may have their own goals, the disciplines are linked by the same encompassing goal under the umbrella of a company's profit strategy. Therefore, disciplinary design teams theoretically should always cooperate.*

There are many conceptual similarities between design and game theory which provide motivation for this work. The purpose of this dissertation is not to develop methods to ensure reciprocal altruism among designers, but to describe the results when cooperation, exploitation, or noncooperation exists.

Mathematical modeling of strategic behavior, where one decision-maker's action depends on decisions by others, is well-established in wide-ranging applications from economics, to business and military applications (Aubin, 1979, Dresher, 1981, Fudenberg and Tirole, 1991, Mesterton-Gibbons, 1992). If the use of multi-player strategic models in these non-engineering applications is so compelling, it is natural to ask what role such models have in the design of complex engineering systems. After all, design is often a collaborative

activity, with different decision-makers responsible for different subsystems or even design stages (e.g., design, manufacturing, and retirement of a product). Developing the capability to model and understand the role of collaboration, cooperation, and noncooperation in design would benefit anyone who makes decisions in a design process, from management to engineering. Certain protocols lend themselves nicely to modeling interactions in design, namely the cooperative or Pareto formulation when the players cooperate, the Nash or noncooperative formulation when the players act in their own self-interest, and the Stackelberg or leader/follower formulation when one player dominates another. These protocols are described and discussed in Sections 3.3 and 5.5.

1.3 GOALS, FOCUS, AND STRATEGY FOR IMPLEMENTATION

1.3.1 The Principal Goal and Fundamental Questions

Given the inherent complexity at various levels of the design of complex systems, the principal goal of this dissertation is to:

> Develop a framework for the decision support of formulating a multidisciplinary design problem, decomposing the problem into disciplinary subproblems, modeling the resulting interactions according to realistic assumptions, and solving and coordinating the disciplinary mathematical models.

To achieve this goal several fundamental constructs are integrated into a conceptual framework centered around the Decision Support Problem Technique and the compromise DSP. The compromise DSP is used as a generic decision model to incorporate discrete and continuous design variables with notions of cooperation, exploitation, and noncooperation within a multiobjective, nonlinear decision construct. As shown in Figure 1.6, the principles and tools are derived from Game Theory, Response Surface Methodology,

Multidisciplinary Design Optimization, and the Tabu Search. These principles and tools are described in Chapter 3.

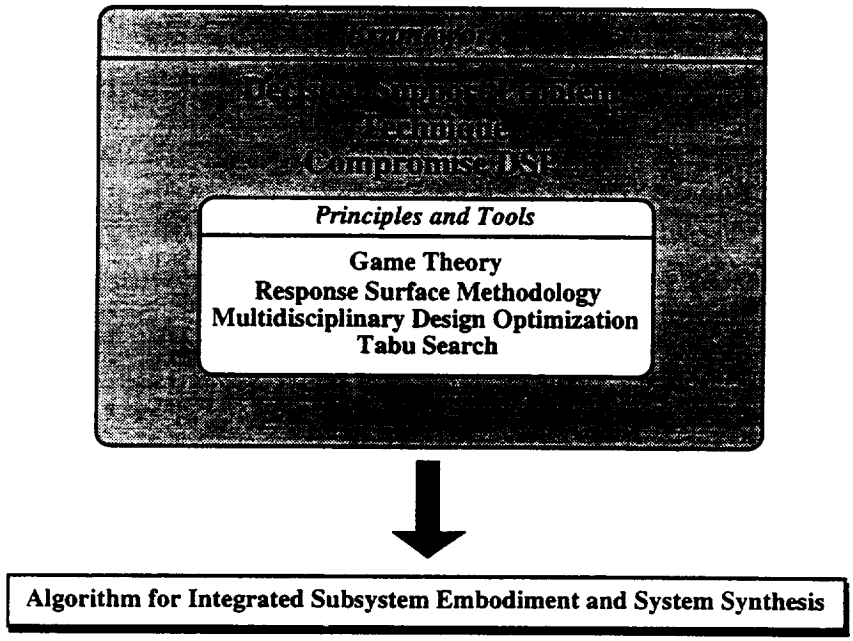


Figure 1.6. Integration of Principles, Tools, and the Framework

During the development of the framework and associated techniques, the following fundamental questions are addressed. After each question the corresponding sections where the motivation and/or technology base for the question is given. These sections correspond to the sections and areas identified in Figures 1.1 and 1.2.

- Question 1: How can complex system design problems and processes be described and classified using an intuitive decision support lexicon? (Sections 1.1.1 and 1.2.1)

- Question 2: How can realistic interactions among design teams and their associated analysis and synthesis tools be modeled and incorporated into a design process? (Sections 1.1.2, 1.1.3, and 1.2.3)
- Question 3: How can mathematical models which consist of continuous, discrete, and integer variables be solved and coordinated? (Sections 1.1.4 and 1.2.2)
- Question 4: Is the g-function of the ALP Algorithm a good transformation of nonconvex functions into well-behaved convex functions? (Section 1.2.2)

The answers to these questions will help bridge some of the gaps found in blind integration of Concurrent Engineering principles at a mathematical level (see Section 1.1.3), and will also help designers to accurately model actual products and processes and solve the resulting models. In the next section, the strategy for answering these questions and verifying the answers is described.

1.3.2 Strategy for Implementation and Verification/Validation

The implementation of this dissertation consists of three phases: 1) identification of the needs and research opportunities in multidisciplinary design, 2) testing the research hypotheses, and 3) further development and verification of the framework using an integrated case study. The following tasks are identified as being necessary to the fulfillment of the three phases.

Phase I: Identification of the needs and research opportunities in multidisciplinary design

Task 1: The first task is identifying the ideal aspects of an algorithm for the integrated design (formulation, decomposition, solution, and coordination) of complex systems that may consist of discrete and/or continuous variables. The design of complex systems first

involves problem and process formulation (meta-design), then embodiment of coupled subsystems, and finally system synthesis. Therefore, the ideal aspects of each stage are identified in Section 1.1, reviewed in Chapter 2, and detailed in Section 3.1.

Task 2: Based on the ideal aspects of such an algorithm, the second task is to identify the research needs and opportunities in the current state-of-the-art technology base. The various research and application areas are covered in Chapter 2.

Phase II: Testing the hypotheses

Task 3: Based on the research opportunities identified, the third task is to identify the research hypotheses for the development of the algorithm for integrated subsystem embodiment and system synthesis. The hypotheses are considered the theoretical foundations for the approach and developments in this dissertation. Ramifications and verification guidelines must be provided for each hypothesis. This is covered in Chapter 3.

Task 4: The fourth task is to test the hypotheses using example problems. These example problems are less complex than the motivating example but are used to illustrate the developments associated with each hypothesis. This work is presented in Chapters 4, 5, and 6.

Phase III: Further development and verification of the framework

Task 5: Having tested and illustrated the hypotheses on less complex examples, the fifth task is to use the motivating study, the design of a passenger aircraft, to further demonstrate and verify the algorithm and its associated techniques. This is presented in Chapter 7.

Task 6: The final task is to summarize the achievements, critically evaluate the work, and discuss future issues and open questions. This is covered in Chapter 8.

Four hypotheses are tested in the development of the algorithm, namely:

Hypothesis I: Classification of problem and process in multidisciplinary design can be facilitated by integrating constructs from Decision-Based Design, Game Theory, and Multidisciplinary Design Optimization. (Answering Question 1)
Hypothesis II: Game theoretic principles can be applied to accurately model and describe the interactions in complex systems design. (Answering Questions 1 and 2)
Hypothesis III: The notion of foraging of wild animals is a natural analogy for optimization and can be used as an effective search technique in the solution of mixed discrete/continuous models. (Answering Questions 2 and 3)
Hypothesis IV: The G-function is a useful transformation of nonconvex functions into well-behaved convex functions. (Answering Question 4)

There are a total of 11 posits which support or help verify these hypotheses. A detailed presentation of the hypotheses and posits is given in Section 3.1.3. Various mechanical examples are used in Chapters 4-6 to verify the hypotheses. Once the hypotheses are tested, the algorithm and its associated techniques are further developed and verified for the design of a complex system using the design of a subsonic transport aircraft as a case study in Chapter 7.

Each question is answered in the following chapters with the following developments.

1) *How can complex system design problems and processes be described and classified using an intuitive decision support lexicon?*
   - Using entities from the Decision Support Problem Technique, a domain-independent classification lexicon is established for multidisciplinary design optimization problems (Sections 3.2.3 and 4.2).

- Game Theory entities can be used to extend problem and process formulation in multidisciplinary design (Section 4.2).

2) *How can realistic interactions among design teams and their associated analysis and synthesis tools be modeled and incorporated into a design process?*
   - Design processes are abstracted as a series of games where the players are the disciplinary design teams and their associated analysis and synthesis tools (Sections 3.3.2 and 5.2).
   - Appropriate definitions are given for the application and development of game theory constructs in Decision-Based Design (Section 5.3).
   - Approximate cooperation is modeled using the Global Sensitivity Equations and first order Taylor series to approximate nonlocal information (Sections 3.3.4, 5.5.1, and 7.4.1).
   - The decision-making strategy of a player in a game is approximated using a second order response surface scheme to construct approximate rational reaction sets (Sections 3.3.4, 5.5.2, 5.6.2, and 7.4.2).
   - Leader/Follower and noncooperative solution strategies are developed for complex design problems characterized by multiple disciplinary models (Sections 3.3.3, 3.3.4, 5.5.3, 5.6.3, and 7.4.3).

3) *How can mathematical models which consist of continuous, discrete, and integer variables be solved and coordinated?*
   - Empirical observations of animals are used to develop a foraging heuristic, which borrows notions from the Tabu Search, Genetic Algorithms, and Simulated Annealing (Sections 3.4.2, 3.4.2, 6.2).
   - An effective solution scheme for mixed discrete/continuous design problems is developed by integrating a foraging heuristic and the ALP Algorithm (Sections 3.4.4, 6.3, and 6.4).

4) *Is the g-function of the ALP Algorithm a good transformation of nonconvex functions into well-behaved convex functions?*

- A formal proof of the g-function and its properties is demonstrated (Section 3.5).

1.4 CONTRIBUTION AND SCIENTIFIC RELEVANCE

The contributions are presented by summarizing the deliverables and establishing the scientific and engineering relevance.

1.4.1 Deliverables - A Summary of the Algorithm

Corresponding to the principal goal of this dissertation, the major deliverable is the development of

- an algorithmic framework for the domain-independent formulation, modeling, solution, and coordination of complex systems characterized by multiple subsystems which each consists of discrete and continuous variables.

Associated with the use of the algorithm, the other deliverables are:

- a three-level lexicon for the classification of the design of complex systems and their associated design processes,
- techniques for implementing game theoretical protocols in the design of complex systems characterized by multiple disciplinary design teams,
- an effective solution algorithm for mixed discrete/continuous design problems.
- a formalized proof by induction of the transformation characteristics of the g-function.

These are considered to be the *fundamental contributions* of this dissertation.

The Algorithm for Integrated Subsystem Embodiment and System Synthesis - An Overview

An algorithm is developed that consists of three primary steps as shown in Figure 1.7. The correspondence of the algorithm steps with the four research hypotheses introduced in Section 1.3.2 and the associated techniques and deliverables introduced in this section is shown in Figure 1.7.

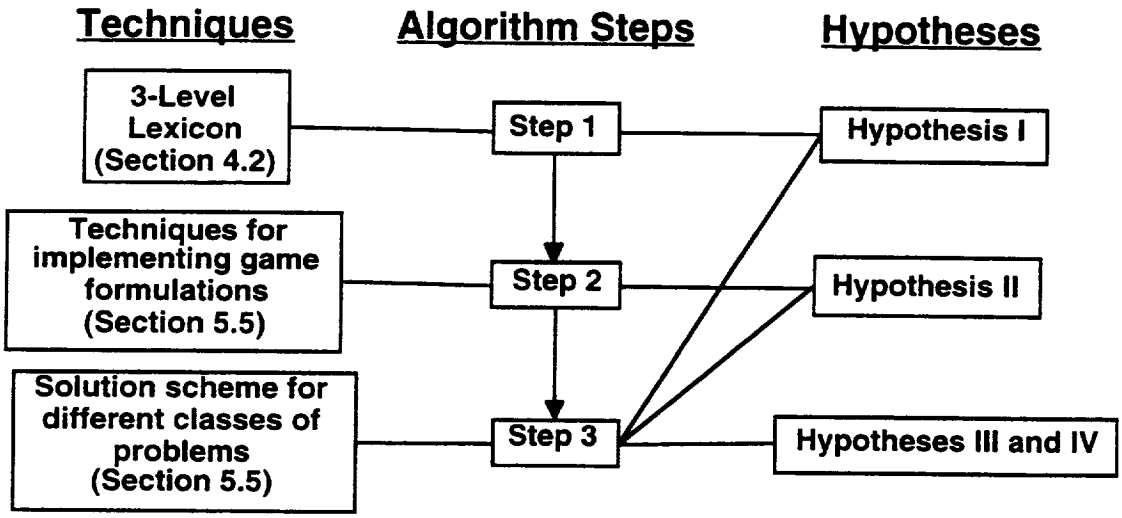


Figure 1.7. Steps, Hypotheses, and Techniques

In general, given a complex system and its associated disciplines, Step 1 of the algorithm is used to classify the process and product using a three-level lexicon. In level 1 of the lexicon, the modeling scheme is classified as being either a single-level or multi-level approach. If a multi-level approach is used, the role of the subsystems in the realization process are identified. In levels 2 and 3 of the lexicon, specific process-based entities are used to further classify the product and process based on the decisions being made and the computer-based tools employed to support these decisions.

In Step 2 of the algorithm, depending upon the classification identified in Step 1, a certain game protocol is used to model the relationship between the disciplines and analysis/synthesis programs (players). Each discipline's local subsystem model is formulated based on their role in the design process. In Step 3 of the algorithm, the models for each player are solved according to the appropriate protocol of the game. Various techniques are developed to facilitate the solution of the different protocols depending upon the information available to each disciplines. *It is inherent, by using game theory constructs, that when the disciplinary models are solved, the problem of coordinating the coupled disciplinary models is also resolved.* This is part of the elegance of game theory (see Chapters 3 and 7) and the motivation behind the phrase "integrated subsystem embodiment and system synthesis" used in the dissertation title.

The Computer Infrastructure for Implementing the Algorithm

Ideally, the use of the algorithm would be automated on a computer system. Certain aspects of the algorithm have been implemented in a computing framework, while others are only conceptual in nature. Integration into a computing framework such as a Design Guidance System (Bras, et al., 1990) or IMAGE (Intelligent Multidisciplinary Aircraft Generation Environment) (Hale, et al., 1996) would be a natural extension of this dissertation. The major components of the *existing* computer infrastructure shown in Figure 1.8 include four processors (a nonlocal approximation processor, module A, a design of experiments/response surface/rational reaction set processor, module B, and a solution processor, module D), each centered about the primary processor, the compromise DSP, module C.

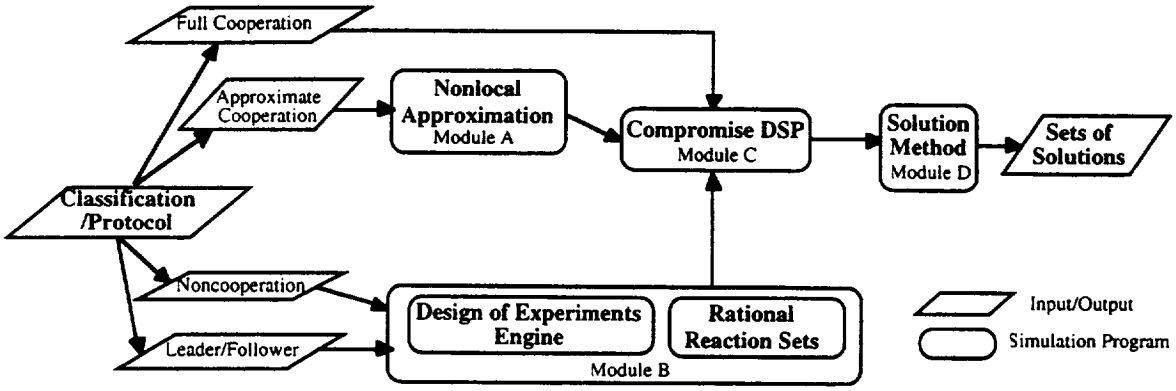


Figure 1.8. Existing Computer Infrastructure of the Algorithm

Each of the other processors is linked to the compromise DSP through a computing interface. Within the compromise DSP lies the domain dependent analyses for the various disciplinary design problems. Given a certain protocol, different processors are used. For the full cooperative protocol, the compromise DSP is the only processor used as shown in Figure 1.8. For the approximate cooperative protocol, the nonlocal approximation processor (module A) is used along with the compromise DSP. For both the leader/follower and noncooperative protocols, the design of experiments processor (module B) is used, coupled with the compromise DSP as shown in Figure 1.8. Each protocol uses some sort of solution method (module D) depending upon the protocol. In this dissertation, NORMAN® (Cartuyvels and Dupas, 1993) is used as the design of experiments processor (module B), and different solution techniques from DSIDES® and Mathematica® are used as the solution processors (module D). A detailed description of the of the computer infrastructure is provided in Section 3.1.2.

What are the Design Applications of the Algorithm?

The primary purpose of the algorithm is to provide decision support in the design of complex systems that are characterized by multiple design teams who each have their own

analysis and synthesis routines. Some of the developments in this dissertation could certainly be used for small design problems. The examples used in Chapters 4, 5, and 6 are simple examples. But as a whole, the algorithm's primary use is for the design of complex systems, some of which are illustrated in Figure 1.9. An aircraft is used in this dissertation as the motivating study, but from the observations in (Duffey, et al., 1996), it is apparent that the automotive and ship building industries could also benefit from the developments of this dissertation. Although, the systems shown in Figure 1.9 are all forms of transportation systems, the developments of this dissertation can be applied to various other forms of complex systems. Succinctly, any system which can be decomposed into interacting subsystems, which are independently analysis-driven, could benefit from the techniques developed as part of this dissertation. Many systems can be broken down even further into sub-subsystems, facilitating the use of the algorithm at multiple levels of detail.

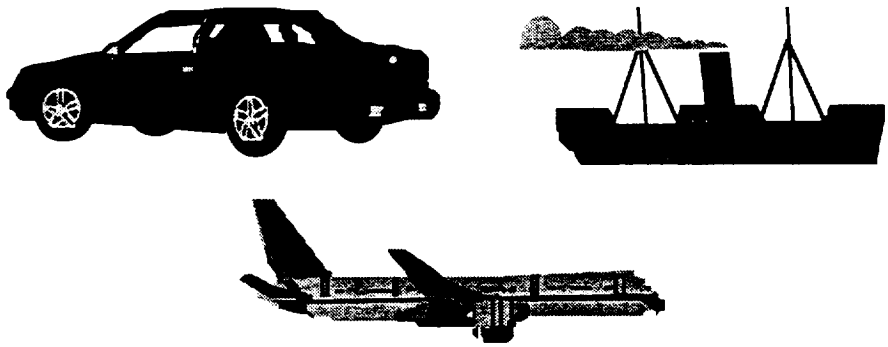


Figure 1.9. Examples of Design Applications

1.4.2 Engineering and Scientific Relevance of This Work

The engineering and scientific relevance of this work is established by establishing three levels of relevance, 1) science and engineering in general, 2) the field of design, and 3) the design of complex, multidisciplinary systems.

Relevance to Science and Engineering

This dissertation is concerned with the study of *systems* in science and engineering. The developments could be applied to a large number of systems in general, but are specified and developed for engineering systems in particular. In Simon (Simon, 1982), the increasing study of such systems is addressed:

> *In science and engineering the study of "systems" is an increasingly popular activity. Its popularity is more a response to a pressing need for synthesizing and analyzing complexity than it is to any large development of a body of knowledge and technique for dealing with complexity. If this popularity is to be more than a fad, necessity will have to mother invention and provide substance to go with the name.*

The motivation for this dissertation is to provide a piece of this "substance" to support the study of complex systems as a body of cohesive knowledge and theory. Although engineering systems are the focus here, the concepts, ideas, and theory could be used in any large-scale system that can be represented mathematically. This could include the design of an organization, an industrial consortium, a city strategic plan, or various examples in biological and behavioral sciences. These situations represent

> *...problems of organized complexity, i.e., interaction of a large but not infinite number of variables, [which] are popping up everywhere and demand new conceptual tools* (Bertalanffy, 1968).

In this dissertation, conceptual, decision-support tools are developed to assist designers in a computer-based design environment. These conceptual tools include the capability to:

- classify and formulate different product and process models of the same system.
- model the interactions in complex systems based on levels of cooperation, information availability, and process structure.
- identify sets of solutions of complex systems, based on product, process, and organizational preferences and structure.

Relevance to Design

Although design's existence as a science or an art is a debate that may never end, design in this dissertation is viewed as a science of the artificial (Simon, 1982). Unlike a scientist, a designer is not simply an *observer* of objects and processes, but is a *controller* of objects and processes. In (Cross, 1993), the science of design refers to

> *that body of work which attempts to improve our understanding of design through "scientific" (i.e., systematic, reliable) methods of investigation.*

In other words, the science of design is the study of design. In many communities, the study of design implies developing design methodologies, based on formal language and theories. According to Cross,

> *DESIGN METHODOLOGY*
> *includes the study of how designers work and think, the establishment of appropriate structures for the design process, the development and application of new design methods, techniques and procedures, and the reflection on the nature and extent of design knowledge and its application to design problems* (Cross, 1993).

Currently two major streams of research activities for developing the science of design through design methodologies exist, namely:

(1) the development of computer-based design tools to aid designers, and
(2) the pursuit of a definitive theory of design.

In this dissertation, contributions in both areas are established. In the first area, the algorithm developed in this dissertation is based on the inherent assumption that designers are using computers to assist in decision-making. Therefore, computer capabilities have been developed to

- use mathematical implementations of game theoretical protocols to model strategic interactions among designers,
- produce effective decision-making information in the form of numerical solutions and geometric representations to describe the design of a system, and
- heuristically search a discrete design space to find areas of promising solutions in complex models.

In the second area, i.e., the pursuit of a definitive theory of design, an attempt is made to formulate, model, and solve complex design problems based on the fundamental paradigm that the principal role of a designer is to make decisions. The compromise DSP is used as a generic decision model to incorporate game theoretical constructs within a multiobjective, nonlinear decision construct. Many design product and process structures, different in concept and implementation, have been studied each using the compromise DSP as the fundamental mathematical construct. The work of this dissertation adds support to the claim in (Bras, 1992, Mistree, et al., 1993a) that the compromise DSP is a domain-independent, generic decision support construct which can be used to support designers in the design of products, processes, and systems.

Relevance to Complex Systems Design

Multidisciplinary Design Optimization (MDO) is described as a methodology for the design of systems where the interaction between several disciplines must be considered, and

where the designer is free to significantly affect system performance in more than one discipline (Sobieszczanski-Sobieski, 1993). Using the definition of *complex systems* introduced at the beginning of Chapter 1, it is obvious that research in MDO focuses on the design of complex systems which are characterized by interacting disciplines. This is precisely the focus of this dissertation as well. Some of the primary developments in this dissertation can aid designers in the design of single disciplined systems, but the true benefit occurs when multi-disciplinary systems are designed. Certainly, the interdisciplinary coupling inherent in MDO tends to present additional challenges beyond those encountered in single-discipline optimization (Sobieszczanski-Sobieski, 1993, Sobieszczanski-Sobieski and Chopra, 1991). This is one of the primary focuses of this dissertation: handling the interdisciplinary coupling in design situations when cooperation may or may not exist among the disciplines and their analysis and synthesis tools. The contributions of this dissertation to MDO include the capability to:

- rapidly explore different product and process structures,
- quantify the disciplinary interactions as functions of design and state variables,
- use effective approximation techniques to increase analysis efficiency, and
- solve mixed discrete/continuous design problems which often occur in MDO.

1.5 ORGANIZATION AND OVERVIEW OF THE DISSERTATION

In Section 1.3.3, the three major phases of establishing, implementing and verifying the developments of this dissertation are presented. In Figure 1.10, a guide to the dissertation and the three phases is given. Chapters 1 and 2 belong to Phase I - identification of the motivation, needs and research opportunities. Chapters 3, 4, 5, and 6 belong to Phase II - development and testing of the research hypotheses. Chapters 7 and 8 belong to Phase III -

further development and verification of the algorithm, along with a summary of the achievements.

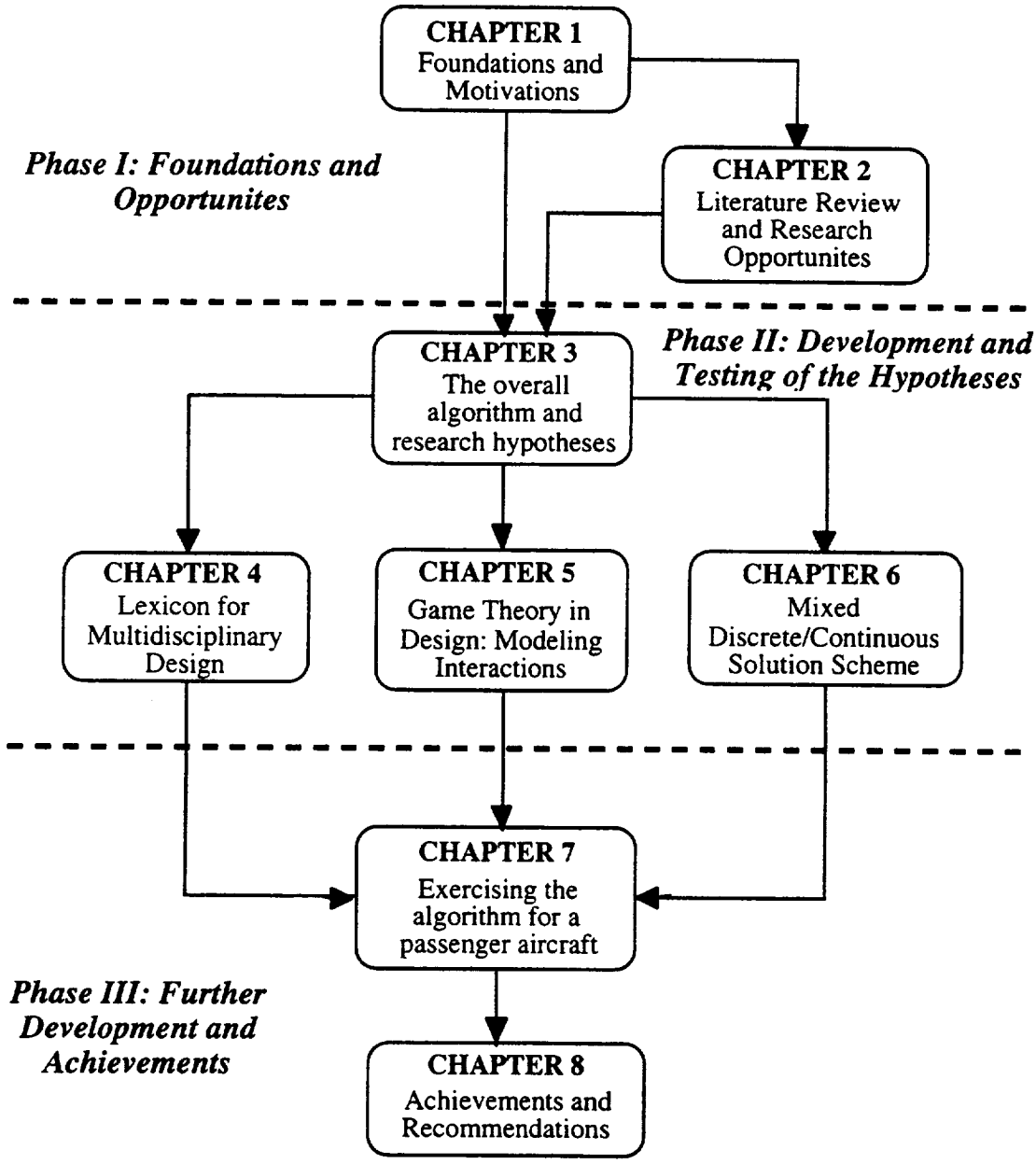


Figure 1.10. Organization of this Dissertation

In Chapter 1, the foundations of the dissertation are laid. The motivation for the work in the context of the design of complex systems is given. The principal goal and fundamental questions are presented. The research hypotheses and strategy for verification are given along with the contributions and scientific relevance of the dissertation.

In Chapter 2, a comprehensive literature review is presented. The review focuses on the research areas established in Chapter 1 which are fundamental to the work, but also covers related research areas in multidisciplinary design optimization and complex systems design. The needs and research opportunities are identified throughout the review.

In Chapter 3, the overall algorithm and its techniques are presented. The posits to support the four hypotheses are presented. For each hypothesis, ramifications and verification guidelines are presented including the approach of applying them in the algorithm. For Hypothesis I, the nature of classifications in design, and the need for multi-levels and domain and time independence is established. For Hypothesis II, design is abstracted as a game, and the game theory protocols applicable to design are presented conceptually and mathematically. For Hypothesis III, the characteristics of design space search in a discrete domain are presented along with the foundation for a mixed discrete/continuous solver. Hypothesis IV is disproven in this chapter using a formal proof from nonlinear optimization theory.

In Chapter 4, the development and testing of Hypothesis I is undertaken. A previous classification system is presented and then expanded to three levels using domain independent terms and entities from the Decision Support Problem Technique and game theory. Various examples are classified and mapped to the previous scheme to illustrate the continuity of the approach to previous work.

In Chapter 5, the foundations of game theory in design (Hypothesis II) are established. A design game is formally defined, and distinctions between discrete, continuous, and mixed games are shown. The protocols introduced in Chapter 3 are expanded and developed in the context of Decision-Based Design. A verification study using the design of a pressure vessel is performed using the developments from the chapter.

In Chapter 6, the mixed discrete/continuous solution scheme is presented and verified (Hypothesis III). The foraging heuristic is developed based on empirical observations of animals foraging for food. Details of the continuous solver, the ALP Algorithm, is presented. Integration of foraging and ALP to produce an effective solution scheme for mixed discrete/continuous design problems is detailed and tested using two examples, the design of a compression spring and a pressure vessel.

In Chapter 7, the motivating case study, the design of a subsonic passenger aircraft, is used to further develop, understand, and verify the contributions of this dissertation. The work in this chapter further verifies Hypotheses I-III. The aircraft study involves two distinct disciplines, the aerodynamics and weights disciplines. Various games are conducted based on different protocols between the players. The resulting designs are explored and implications to modern design processes and products are presented. Although the algorithm is illustrated only for a 2-discipline problem, it is developed with the capability to handle design problems composed of *n-disciplines.*

In Chapter 8, the dissertation is summarized, a critical evaluation is presented and areas of future work are identified.

In Figure 1.11, the running icon of this dissertation is shown. In this icon the roles and relationships of the eight chapters in the development, verification, and implementation of this dissertation are summarized. The similarities between Figure 1.10 and 1.11 are apparent. The three phases of the verification strategy are shown as being phases of construction of Figure 1.11. The chapters in Phase I provide the foundation and motivations. The chapters of Phase II provide the construction and testing of the algorithm and associated hypotheses. Sitting atop the algorithm and foundation is an aircraft, which is the motivating case study of the dissertation, and the primary verification study in Phase III.

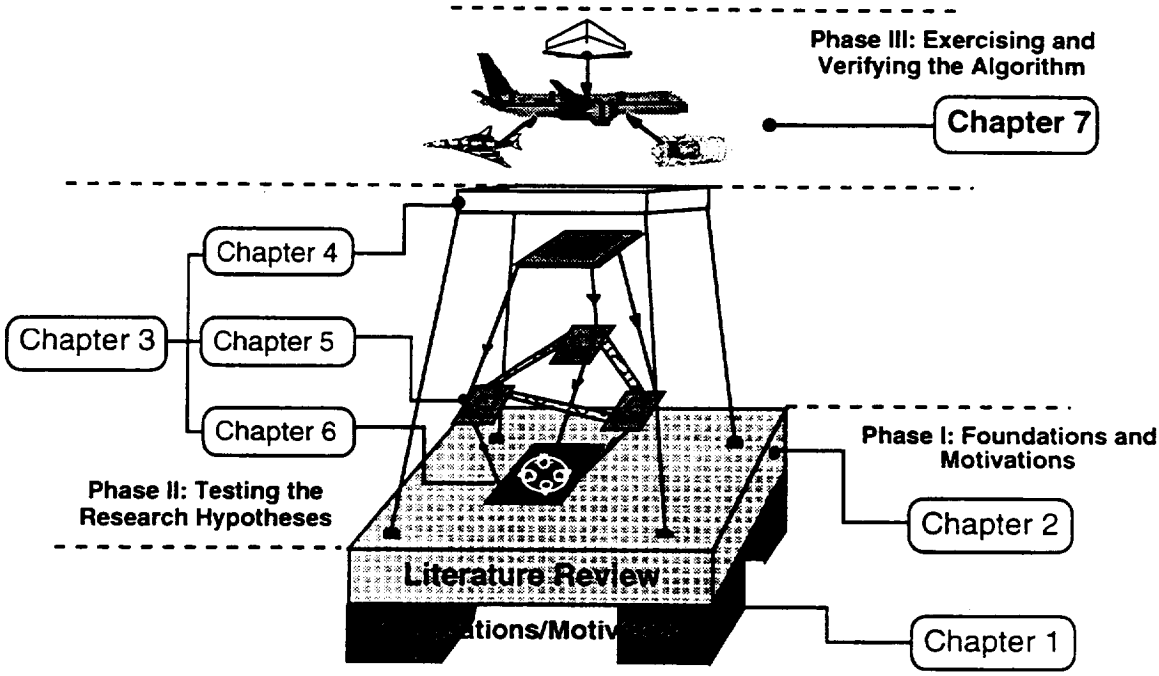


Figure 1.11. Running Icon

In Figure 1.12, the separate pieces, Figure 1.11 is dismantled and the individual pieces are shown. Throughout the dissertation, the pieces of Figure 1.12 are combined one-by-one

and Figure 1.12 is eventually built at the end of Chapter 8. This construction acts as an inventory of the progress and frame of reference of the dissertation.

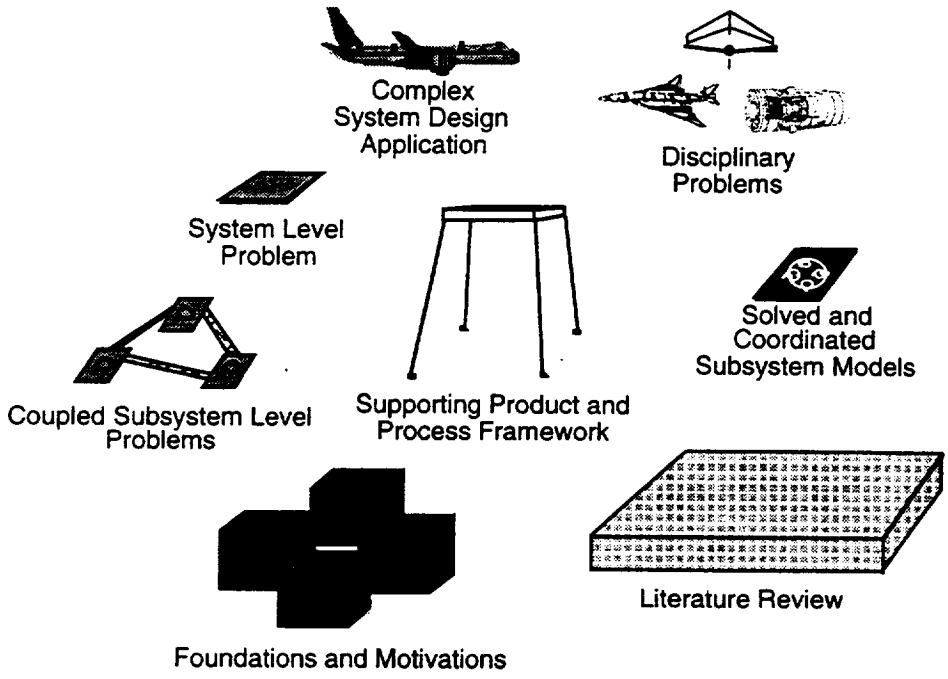


Figure 1.12. Individual Pieces of the Dissertation

The first piece of Figure 1.12, established in this chapter is the foundation blocks. These blocks are constructed and arranged in Figure 1.13 to provide the foundation for the remainder of the dissertation. In the following chapters, the remaining pieces will be combined on top of the foundation and motivations.

Figure 1.13. Frame of Reference: Chapter 1

In Figure 1.14, the structure of the Appendices corresponding to the appropriate chapter are shown. It is assumed throughout this dissertation that the protagonists are female in even-numbered chapters and male in odd-numbered chapters. This is to avoid the continual use of "his or her" and "he or she" in place of epicene pronouns.

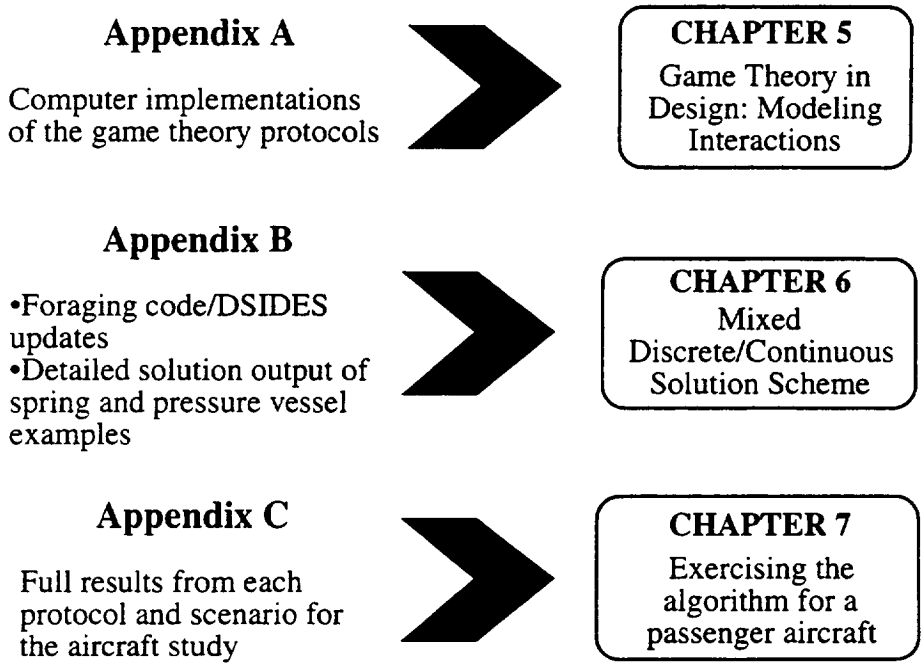


Figure 1.14. A Guide to the Appendices

CHAPTER 2

REALIZING MULTIDISCIPLINARY SYSTEMS - RELEVANT LITERATURE

In this chapter, implementations of Tasks 1 and 2 identified in Section 1.3.2 are discussed. For Task 1, the ideal aspects of an algorithm for the integrated design (formulation, decomposition, solution, and coordination) of complex systems are identified, in a broad sense, as problem and process formulation, embodiment of coupled subsystems, and system synthesis. In Figure 2.1, these three aspects are shown to be the primary stages of system realization, supporting the algorithm. In this chapter the relevant work in these areas is reviewed, establishing the context and foundation of this dissertation. Reviews in secondary areas such as approximation and multiobjective design, supporting the primary areas as shown in Figure 2.1, are also presented. To help provide context and completeness, related areas, namely, the human factor of design, computational costs, robust/quality methods, and decomposition, are reviewed. The corresponding sections for each subject are given in Figure 2.1. For Task 2, the research needs and opportunities in the appropriate areas are identified, throughout this chapter.

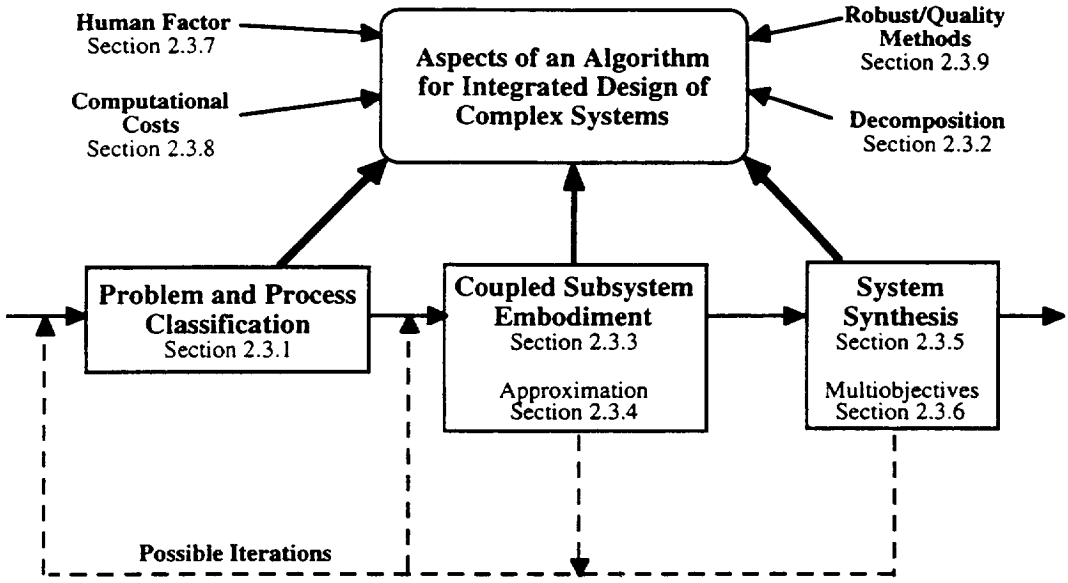


Figure 2.1. Aspects of Algorithm: An Overview of Chapter 2

As defined in Section 1.4.2, the work of this dissertation, in a specific sense, belongs to the field of Multidisciplinary Design Optimization (MDO). The literature review presented in this chapter is discussed from an MDO perspective.

2.1 CHANGE OF PERSPECTIVE IN MULTIDISCIPLINARY DESIGN OPTIMIZATION (MDO)

In the past several years, there has been a shift in the interest and application of the MDO community from mainly aerospace applications, which stemmed from MDO's origins in the field of structures, to fields including mechanical, civil, and electrical engineering, operations research, and materials science. In each of these fields, there are indeed multidisciplinary research issues in the design, development, production, and support of complex systems. Recently, the multidisciplinary research and development in each field are merging into fundamental approaches to complex system design problems. Also, with the advent of systems thinking and doing "more with less," issues usually reserved for the later stages of a design process are brought forward into the initial stages, MDO technology and research is moving from the detailed analysis design stages to the conceptual stages where multidisciplinary system tradeoffs can be rapidly explored effectively and efficiently. This shift parallels a similar shift from traditional calculus based optimization algorithms, where precise mathematical models and couplings are known, to more imprecise techniques where laws of uncertainty guide mathematical models and their interactions. The former certainly has its place in the later stages of design, but in the early stages, the information about a multidisciplinary, complex system may not be fully known and many times is unstructured. This gives rise to the requirement of imprecise techniques for decomposing, analyzing and synthesizing a system model. Thus, the study of complex systems moves into a new age of research and technology, one characterized by both precise and imprecise models, and exact mathematics and fuzzy heuristics.

It is within this "new age" of research and technology that this chapter is motivated. In this chapter, both primary and secondary literature backgrounds are presented. In the primary backgrounds, the foundation for the main research focuses of this work are given. The primary background addresses the issues of problem and process formulation, embodiment of coupled subsystems, and system synthesis. These correspond to the three hypotheses introduced in Section 1.3.2 and are labeled as PRIMARY areas in Table 2.1. In the secondary backgrounds, the foundation for the supporting issues of this work is given in the context of multidisciplinary design optimization of complex systems. These are labeled as SUPPORTING areas in Table 2.1. Also reviewed are related areas in MDO that provide added support and completeness to this work. These are labeled as RELATED areas in Table 2.1.

Table 2.1. Various Areas of Literature Background

PRIMARY: Primary research focuses of this work • Mixed Discrete/Continuous Optimization (Heuristics) • Strategic Interactions • Problem and Process Classification
SUPPORTING: Secondary focuses of this work • Approximation • Multiple objectives
RELATED: Not explicitly addressed in this work, but help define the context and arena of application. • The Human Factor • Decomposition • Quality Design Methods • Information Storage and Transfer

In Figure 2.2, the specific concepts covered in this chapter under the umbrella of MDO are shown. In Section 2.2, the trends in MDO are identified by evaluating the distinct areas of research in MDO, namely its three linguistic components, *multidisciplinary, design, and optimization.* In Section 2.3, the research and application issues under the umbrellas of each research area are surveyed and summarized. As shown in Figure 2.2, these issues overlap beneath the research areas, as they are motivated by questions from more than one area. This is part of the difficulty researchers in MDO face, as the integration of various fields and disciplines poses complex problems. In Figure 2.2, the three primary areas of focus, interactions (under multidisciplinary and design), heuristics (under design and optimization), and classification (under all three), are highlighted. The supporting areas of review in this chapter are given beneath the primary areas in Figure 2.2. The work presented here includes government, industry and academic contributions to this emerging area of research and practical application.

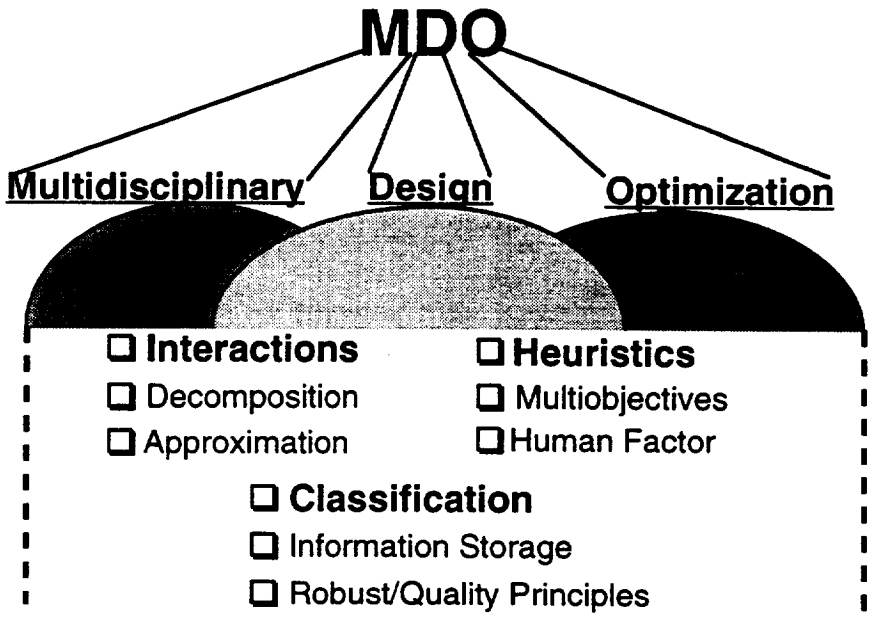


Figure 2.2. Roadmap to Chapter

2.2 MDO: AN INTERNAL DECOMPOSITION

System decomposition is a valuable and many times necessary approach in solving complex systems. The method used to decompose a system, however, is another issue, addressed in Section 2.3.2. Capitalizing on the advantages of decomposition, the field of MDO is investigated in this chapter by applying a linguistic decomposition approach to the term "MDO". This stems from the simple approach used to determine the meaning of "complicated" compound words such as schoolbus, where combining the meanings of the root words "school" and "bus" result in the connotation of the compound word.

Decomposing "MDO", the root words "multidisciplinary", "design", and "optimization" are discovered. Each area calls on certain capabilities from the others to perform its required duties. These calls and demands among the areas are illustrated in Figure 2.3. Each of these terms and the demands each makes on MDO as a whole is investigated in this section.

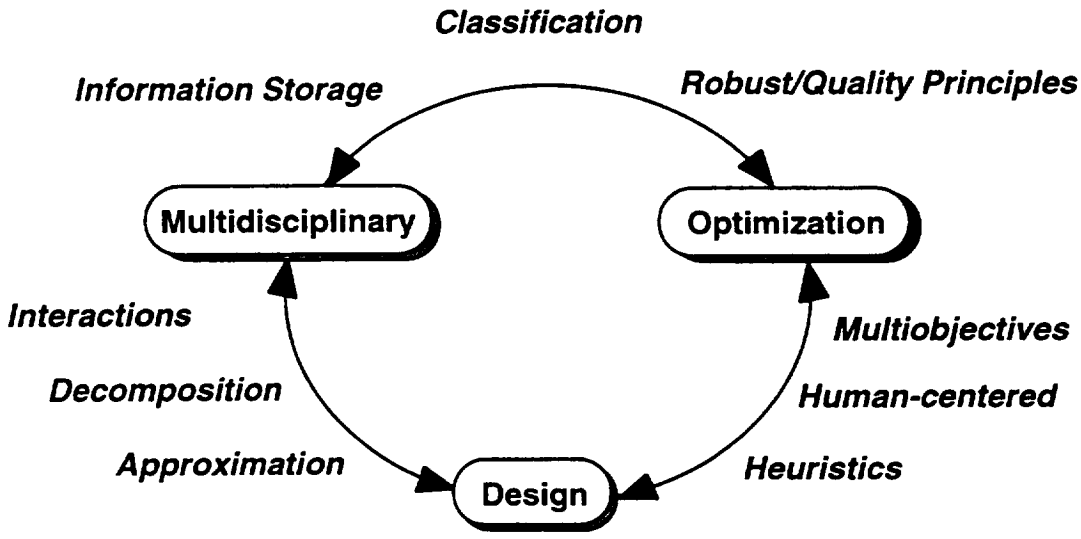


Figure 2.3. Couplings of M, D, and O

Multidisciplinary

The term "multidisciplinary" plays an important role in the complexity of system problems. Individual disciplines have developed mature methods to analyze disciplinary problems. However, when two or more disciplines and their analyses are combined, such as Computational Fluid Dynamics (CFD) in aerodynamics and Finite Element Methodology (FEM) in structures, the problem becomes well beyond what even the most powerful computer of next year can handle. Therefore, some sort of decomposition method is necessary for most multidisciplinary problems to establish less complex, disciplinary-level problems. Many decomposed problems are still too complex to effectively analyze because of the size of the analysis routines. Consequently, a form of approximation may be necessary to replace the exact analysis. It has been shown that the fidelity of an approximate model can decrease and still maintain acceptable analysis accuracy (Chen, 1995a, Malone and Mason, 1991). But, *how approximate can a model be and still maintain accuracy,* is another research issue in complex systems design. Approximation can take many forms, from using approximate models within a discipline to using approximations to represent the effect of one discipline upon another. However, the key notion here is that each discipline plays an important role in the function of the entire system.

In a given discipline, there may exist system variables which are continuous, integer, discrete, or Boolean. Examples of these are wing span, number of engines, gear diameter, or control variables, respectively. Integer, discrete, and Boolean variables will be referred to as discrete variables in the remainder of this dissertation. There are well established methods for solving either purely continuous problems, or purely discrete problems. Continuous methods are largely calculus based, while discrete methods range from integer programming methods to heuristic search methods. Yet, it is the

development of robust methods to support the decision making of designers in problems with mixed continuous/discrete variables that the presence of multiple disciplines demands.

Design

The term "design" mandates the investigation of other issues, including multiobjective system formulations. Practical systems are not single objective in nature. In vessel design, minimizing resistance is similar to minimizing weight in aerospace design, but there are many other design objectives a designer may want to consider. A naval designer may want to minimize the vessel powering and keep seakeeping at acceptable levels for various seastates. Complex systems are frequently multiobjective, but these objectives may have different priorities, according to system requirements and designers' preferences.

Process and human designer issues are also brought in with the term "design". In Decision-Based Design, design is accomplished using the *abilities* of a designer and the *capabilities* of a computer. The principal role of the designer, and *not* the computer, is to make the design decisions. Design, in addition, consists of a series of decisions made by a designer or design team along a timeline. Designers and design teams repeatedly use their decision-making ability together with the computer's decision support capability to make decisions regarding various system and subsystem tradeoffs. Hence, the notion of a designer interface and human-centered design is inherent in any design process. The sense of time, past and present, in a process requires some way of storing and retrieving information to expedite future decisions. Hence, some type of database that links the information from all disciplines for efficient retrieval throughout a design process is necessary.

Optimization

The term "optimization" acts as a dependent *and* independent variable in MDO. Regardless of decomposition, there is a need for the determination of system variables based on system constraints, variable bounds, and with respect to system objectives. This is the independent nature of the term "optimization". Independent of the modeling approach taken or domain of application, optimization techniques are required to solve decision models and support the decision making abilities of a designer.

The link between "optimization" and MDO is where "optimization" plays a dependent role. Previously, researchers in MDO established optimization techniques *depending on* the problem formulation. Moreover, "optimization" has even sometimes assumed the role of a synthesizer, where a number of subsystems are optimized or coordinated into a system level "optimum". However, this process became inefficient as the needs of MDO began to require more than what "optimization" had bargained for. Consequently, "optimization" has now become an integral part of the decision support of MDO, as researchers have embraced the issues inherent in multidisciplinary design *optimization*. "Optimization" now acts as the bearer of good or bad news for the issues and requirements from the "multidisciplinary" and "design" areas. These two areas demand optimization techniques for multiobjectives, mixed continuous/integer systems, designer interfaces, robust global solutions, and post-solution analysis, among others. Researchers have addressed one or more of these various issues, but there does not presently exist a single algorithm to encompass all the needs of MDO in a decision-based environment. This may be a problem too great to handle at present, but certainly is a research issue. There has been extensive work in single objective optimization, but since the term "multidisciplinary" implies multiple objectives, issues in multiple objective modeling, solution, and decision support are the focus of this dissertation.

In this section, the innate issues in MDO are examined by simple decomposition and analysis among the components. The harmonious coordination of these issues is the basic task in MDO. Issues in all three areas are addressed in this dissertation, multidisciplines, design, and optimization. In the next section, the developments and fundamentals of each issue are investigated, including the specific research opportunities that constitute the basis of this dissertation.

2.3 ISSUES IN MDO

Researchers, when addressing the areas of research in Section 2.2, must keep the inherent issues of MDO in mind. Some researchers in MDO have addressed some of these issues by themselves, while others have looked at a combination of a few. Any development in MDO must keep these issues in mind, as it will add to the integrity, broaden the acceptance, and establish the value of MDO.

2.3.1 Lexicon Development

With several individual research directions, a framework for research and application is needed. In most pure science fields, such as chemistry and biology, there is a standard framework and lexicon which are used now and forever. In order for MDO to continue to evolve and establish itself as a distinct field, a framework, including a common lexicon for researchers and industries, is necessary for synergistic communication. This framework must be applicable to the entire MDO process from concept generation to detail design. The most promising attempts at these have come in two forms. First, in (Balling and Sobieski, 1994, Cramer, et al., 1994), lexicons and classifications of

approaches to MDO problem formulation and solution are presented (see Section 4.1.1). These types of classification give the field a form of common communication upon which to base future developments and research. If a common lexicon were established, the various work in academia, industry, and government could be easily classified and compared.

Second, in the Framework for InterDisciplinary Optimization (FIDO) program (Townsend, et al., 1994), a computer framework for MDO is being generated. This type of framework has been shown to be an excellent interface for multidisciplinary design issues among distanced design teams throughout a design process. It is uncertain where the present research in MDO could fit into computer frameworks of this type. Computer frameworks may become simply the housing for MDO research, where developments are integrated into the "guts" of the framework at the system or discipline level. This would allow for future developments, and would permit a design team to design and analyze a system without having to know about the inner workings of the algorithms, schemes, and routines. Also, unclear is how the evolution of the design process from concept design to detailed component design would be accommodated in a computer framework.

Research Opportunities

The primary research opportunity in this area is to further develop these lexicons, establishing common baseline linguistics. The baseline used is the notion of a *decision* and its supporting entities in the Decision Support Problem Technique (Mistree, et al., 1993a, Mistree, et al., 1988, Mistree, et al., 1990b) and Game Theory (Von Neumann and Morgenstern, 1944). The domain-independence and time-independence of such a lexicon are paramount to ensure effective application across disciplines and over an entire,

evolving design process. This is a primary thrust of this work and the focus of Hypothesis I (Sections 1.3.1 and 3.2) and Chapter 4.

2.3.2 Decomposition: Friend or Foe?

The decomposition or partitioning of large systems has long been viewed as being beneficial to the efficient solution of the system. Although breaking a system up into smaller, less complex subsystems may allow for the effective solution at the subsystem level, decomposition makes the system design problem more complicated by requiring the coordination of subsystem solutions into a harmonious system solution. A mirror can be broken apart, the pieces reassembled, and in no way function as a mirror again. This problem in analyzing and synthesizing various subsystems poses a difficult problem in MDO. So why not simply analyze systems at one level, the system level? This creates analysis problems, as complex system models may become too large to handle. When do systems become too large for single-level analysis and require decomposition and multilevel analysis? The answer may lie in the amount and quality of information in a system model at any point in a design process. Both single-level and multilevel approaches are being developed as fundamental approaches to a MDO problem. General decomposition approaches have been developed for generic problems which include information overlap among various tasks, events, or disciplines by Rogers (Rogers, 1989) and Kusiak (Kusiak and Larson, 1995, Kusiak and Wang, 1993). A general decomposition procedure based on the hypergraph representation of known mathematical analysis models is presented in (Michelena, et al., 1995). More specific decomposition and coordination approaches for MDO problems are explored below.

Decomposition schemes initially were hierarchical in nature. An excellent review of hierarchical decomposition is presented in Renaud (Renaud, 1992). On the other hand,

many systems lend themselves to nonhierarchical decompositions instead of hierarchical ones. The development of nonhierarchical decomposition schemes is relatively new compared to hierarchical ones. A review of the early work in nonhierarchical decomposition is also presented in Renaud (Renaud, 1992). Implementations of decomposing larger, more complex problems into smaller, temporarily decoupled disciplinary problems have been studied (Beltracchi, 1990, Korngold, et al., 1992, Olds, 1992, Rohl and Schrage, 1992). Various decomposition and coordination strategies have been developed and implemented based on the Global Sensitivity Equation (GSE) (Sobieszczanski-Sobieski, 1988) approach to couple the nonhierarchical subsystems (Balling and Sobieski, 1994, Bloebaum, et al., 1992, Ford and Bloebaum, 1993, Renaud and Gabriele, 1991, Renaud and Gabriele, 1993, Renaud, et al., 1994). In Bloebaum (Bloebaum and Chi, 1994) the GSE method is extended into handling discrete and continuous variables by decomposing the system according to variable type, resulting in discrete subsystems and continuous subsystems. In (Bloebaum and Chi, 1994), neural networks are included to avoid the expense of continuous re-analysis within the discrete subspace. Typically in MDO, identifiable subsystems exist with both discrete and continuous variables, but this work is a step in this direction. As pointed out, cumulative constraints are difficult to use with discrete variables, adding to the complexity of handling discrete variables.

In related work on the decomposition and resolution of nonhierarchical system models, in Renaud (Renaud and Gabriele, 1991), a first order approximation using the GSE method for system approximation is used. In Renaud (Renaud and Gabriele, 1993, Renaud and Gabriele, 1994) a second order approximation, using second order GSE's, is used to improve the accuracy of cumulative constraint approximation and improve solution convergence. In (Renaud, 1993) the algorithm is further extended to include non-disjoint

decomposition of design variables that allows greater latitude in design subspace optimizations. In (Renaud, et al., 1994), the algorithm is extended to handle mixed discrete/continuous problems. The solution scheme uses neural network to model the design space and a successive simulated annealing algorithm for solution. The case study contains five discrete and one continuous variable. The simulated annealing algorithm includes successive discretizing of the continuous domain until a solution is reached. With more than one continuous variable, this algorithm may not be computationally practical. In addition, "sufficiently" accurate training of the neural network seems to be problem dependent and difficult to quantify.

In Balling (Balling and Sobieski, 1994), an approach to the nonhierarchic decomposition problem is developed whose coordination procedure focuses on the minimization of the norm of the coupling constraint and design constraint violation (called a discrepancy function). A solution scheme that incorporates a cutting-plane algorithm and a move-limit strategy is used to solve the complex discrepancy function. Application of the algorithm to truly multidisciplinary systems is yet to be demonstrated.

In Kroo (Kroo, et al., 1994), compatibility constraints are used at the system and subsystem levels to account for the coupling between levels. At the system level a single objective is used (aircraft range in the case study) and the system constraints ensure that the coupling among the subsystems is maintained. At the subsystem level, the discrepancy between the system variables and their target values is minimized. System variables may overlap among subsystems, creating the coupling among subsystems. Various discrepancy functions are investigated, similar to the investigation of discrepancy function norm formulations in Balling (Balling and Sobieski, 1994). Application to

mixed discrete/continuous systems and handling of multiobjectives is not explicitly addressed.

The decomposition approaches in this section have focused on two primary areas:

- hierarchical modeling where bilevel models are present
- nonhierarchical modeling where some form of cooperation is modeled mathematically

Realistically, these models are not always applicable. First, because of informational or geographical barriers, a model that incorporates noncooperative notions may simulate actual processes better. Second, it is common for certain disciplines to lead or dominate a design process and for others to follow their lead of decision-making (Hazelrigg, 1996). This type of process would demand a model that incorporates sequential relationships among decision makers. The leader/follower formulation is a special case of a bilevel model. In the next section, strategic interactions are addressed.

2.3.3 Strategic Interactions

The design of multidisciplinary systems requires a series of decisions which are made by multiple decision makers, design teams, or organizations. Implementation of Concurrent Engineering principles have made certain strides to facilitating this integrated decision making process at a personal interaction level. However, at the analysis and synthesis levels, a seamless computer infrastructure among the disciplinary software is rare. That is, cooperation at the analysis and synthesis levels does not occur even though the majority of the research in this area has assumed cooperation (see Section 2.3.2). When cooperation is not present, game theory principles of noncooperation and multilevel processes can be beneficial to the modeling of the system and process. In (Rao, et al., 1996, Rao and Mistree, 1995), different game theory formulations are exercised for

simple engineering examples using two simulated designers. The formulations studied are the cooperative (communication exists), noncooperative (no communication exists), leader/follower (one player dominates), and conservative (players assure a minimum gain) protocols. Each designer wants to meet their own objective, but even in the simple examples used in (Rao, et al., 1996, Rao and Mistree, 1995), these objectives are in conflict with each other. Therefore, depending upon the protocol exercised between the players, significantly different solutions are found. Rich insights are gained into the relationships between the players and the process by which the design is performed. The mathematical foundation for the cooperative, noncooperative, and leader/follower models are given in Section 3.3.3.

Research Opportunities

The design of multidisciplinary systems requires a series of decisions that are made by multiple decision makers, design teams, or organizations. Concurrent Engineering principles have made certain strides to facilitating this integrated decision making process at a personal interaction level. In this dissertation, game theory is being used to make similar strides, but at the level of the interactions of mathematical models, analysis packages, and/or synthesis and optimization routines. The use of game theory to model multidisciplinary design processes where cooperation may or may not exist among decision makers in engineering design is of relatively recent origin (Hazelrigg, 1996, Rao, et al., 1996, Rao and Mistree, 1995); its usefulness in many other decision-making sectors such as economics, politics, and strategic warfare is well-established (Axelrod, 1984, Owen, 1995). Game theory was explicitly proposed as a design tool originally in (Vincent, 1983). But the notion of game theory in engineering design and optimization has since been limited (Badhrinath and Rao, 1995, Dhingra and Rao, 1995, Hazelrigg, 1996, Pakala and Rao, 1996, Rao and Chidambaram, 1993, Rao, 1987, Rao and Freiheit,

1991). In these studies, small, simplified problems are used to illustrate the concepts. As a result, there are rich research and implementation opportunities in developing and applying game theoretic principles to complex problems encountered in MDO. Modeling the strategic interactions in MDO as a game is a novel pursuit, and one of the primary contributions of this work. The work of this dissertation in this area support Hypothesis II introduced in Section 1.3.2 and discussed in Section 3.3 and Chapter 5. The development and application of game-theoretic principles to complex systems design are presented in Sections 5.5, Sections 5.6, and 7.4. This is one of the main contributions of this dissertation.

By using system level analysis many issues involved in decomposition strategies can be avoided and Pareto solutions can be found, but system analyses may be too complex to handle computationally. One issue brought on by decomposition strategies is approximation on many levels, from the approximation at the system level to approximation of nonlocal information at the subsystem level. In the next section, the use of approximation in MDO is presented.

2.3.4 Approximation

In a perfect world, approximation would not be needed, as the actual analysis routines across a multidisciplinary system could be used without concern for the computational cost or time constraints. However, until computers become "perfect", approximation is necessary at some level in a MDO process. This approximation may take many forms.

The first and most developed area is the approximation of the derivatives, both global and local, of the state variables of the system with respect to other state variables, fixed parameters, and design variables. Optimization routines have been used approximate

derivatives to determine search directions and magnitudes for decades. Ideally, for an n^{th} order equation, n^{th} order derivatives could be calculated and used in the optimization of the system. Designers however must make a tradeoff between accuracy and efficiency. Much of the derivative approximation work has concentrated on developing schemes to efficiently approximate first order derivatives of a system. A system, however, may consists of many subsystems, each with their own derivatives. In Barthelemy (Barthelemy and Sobieszczanski-Sobieski, 1983), the derivatives of the optimal objective with respect to fixed parameters are calculated directly from the Lagrange multipliers and disciplinary sensitivities. As a method of calculating the global derivatives with respect to the design variables using the local derivatives of the subsystems, the Global Sensitivity Equation (GSE) was proposed in Sobieszczanski-Sobieski (Sobieszczanski-Sobieski, 1988). The GSE method allows for the local derivatives to be calculated using any method. Various studies have been conducted concerning the accuracy of using GSE approximate first-order system derivatives including (Bloebaum, et al., 1992, Korngold, et al., 1992). Further investigations have explored the accuracy vs. efficiency tradeoff by using second-order derivatives in the GSE method (Renaud and Gabriele, 1993, Renaud and Gabriele, 1994). It is an open issue as to when a designer can, with confidence, use the nth-order derivatives and still maintain acceptable efficiency. This may be a function of computer capabilities, but the answer may lie in deeper explanations, such as information theory, which asks 'when is the amount of information enough to make appropriate decision along a design timeline?'

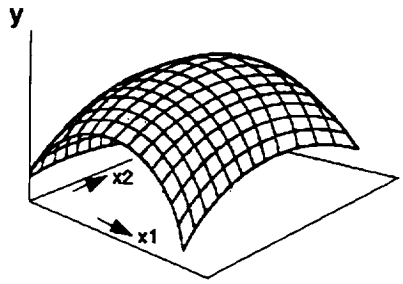
Derivative approximations may become obsolete with the emergence of Automatic Differentiation in FORTRAN (ADIFOR) (Bischof, et al., 1992). In ADIFOR the exact derivatives of a FORTRAN code are calculated analytically by using the numerical entities at a given design point, and employing the chain rule to find total local and global

derivatives. Having the exact derivatives with acceptable efficiency is a major step in numerical analysis and approximation, and may pave the way for further major developments.

The next area of research in approximation is approximation of the design space, locally, nonlocally and globally. From a global or systems perspective, less detailed analytical models have effectively been used to approximate the behavior of an aircraft system (Chen, 1995a, Chen, 1996, Malone and Mason, 1991). The use of low fidelity models is useful in the early stages of design, but it may sacrifice accuracy in more detailed design stages. High fidelity approximation models have effectively been used in more detailed design, but are more computationally expensive. Livne and coworkers (Livne, et al., 1993) accomplished comprehensive wing optimization including structural, aerodynamic, and active control requirements using realistic approximations along with nonlinear programming techniques. So the question becomes, when does a designer "switch" or evolve from less detailed models to more realistic or very detailed models? Or when can approximate models be used in place of full analytical models?

Typically in multidisciplinary systems, the behavior is quite nonlinear and difficult to simulate directly because of the complexity of the local behavior equations. Therefore, methods to approximate the behavior or design space are needed, but again must meet the same accuracy vs. efficiency tradeoffs. In (Chen, 1995a, Engelund, et al., 1993, Unal, et al., 1994) , response surface methodology is shown to be both effective and efficient in design space approximation. Response surface methodology fits a surface to a given set of design points according to a prescribed surface fit equation. To account for nonlinear effects, the fit equation must be at least of order two. In Figure 2.4, a second order response surface and its corresponding equation are shown. Response surfaces allow for

rapid approximation of a design space based on simulated designs at certain settings of the design variables.



$$y = \beta_0 + \sum_i \beta_i x_i + \sum_i \beta_{ii} x_i^2 + \sum_{i \le j} \beta_{ij} x_i x_j$$

Figure 2.4. Second Order Response Surface (Box and Draper, 1978)

Another common approximation method is Neural Nets (NN) which "learn" about the behavior of the system from training data. NN's have been shown to produce effective approximations of the design space (Batill and Swift, 1993). Yet, with any method, poor fidelity may lead to poor approximation. Using the above methods, a surface fit equation of too low an order, or not enough training data may easily lead to erroneous and unacceptable results. On the other hand, with higher order fits and more training data comes more computation time. The balance of these issues has long been resolved through trial-and-error and has been problem dependent. Therefore the development of rigorous approximation strategies based on higher level concepts such as information or even deterministic heuristics that are domain *independent* would be invaluable.

The issues in approximating local design spaces also hold true for approximating nonlocal design spaces in hierarchical or nonhierarchical design optimization. Designers of one subsystem must account for the effects upon other subsystems. Therefore, the

ability for a given subsystem to "see" what how it is affecting and being affected by other subsystems is vital. However, seeing the effects on actual behavior is not realistic. Subsystems only can see effects on approximate behaviors. Various strategies have been implemented to approximate nonlocal states (Balling and Sobieski, 1994, Diaz, Renaud and Gabriele, 1991). Inherent in these approaches is accommodating the approximate coupling between subsystems via objective functions, constraints, or additional design variables.

Many times the level of approximation that is needed or effective is based on heuristic insight or rules. Heuristics play a large role in the design of complex systems from a human decision-making standpoint to a computer-based AI standpoint. There is a need for some sort of heuristics to account for the inevitable uncertainty in any given design process. Heuristics many times take the form of solution algorithm "facelifts" where certain ad-hoc rules, based on the designer's experience or naturally occurring phenomena help solution algorithms become more effective or efficient. In the next section the spectrum of heuristics in MDO is presented.

2.3.5 Heuristics: From Designer to Computer

Heuristics, or rules based on intuition, experience, or natural phenomena, have been used from a designer's point of view in various stages of a design process to "smooth" over rough spots where insufficient or unstructured information is present. In Bloebaum (Bloebaum, 1991), heuristic rules are employed to allocate variables to subsystems, determine the most appropriate move limits, and assign coordination coefficients during system synthesis. In (Kamal, et al., 1992, Peplinski, et al., 1996a, Peplinski, et al., 1996b), heuristic Decision Support Problems are formulated based on sets of evaluation criteria and rules. These evaluation criteria are based on uncertain information in the

concepts. The best concept is selected based on multiple measures of merit. These rules are based on designers' experience with the design of complex systems and are used when the mathematical information to make these decisions is not fully defined or, in other words, uncertain. These types of heuristics are shown originating from the human in Figure 2.5.

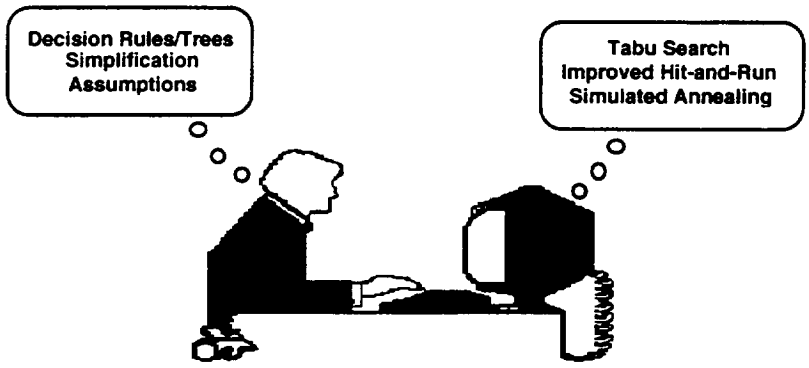


Figure 2.5. Heuristics Across the Design Spectrum: From Human to Computer

Heuristics are also being incorporated from the other end of the human-computer interface, the computer, primarily to aid in the solution of discrete and mixed models. These types of heuristics are inherent parts of the computer in Figure 2.5. In Figure 2.5, the synergy between the human and the computer, and the heuristics employed by each is illustrated. This synergy is also found in Decision-Based Design, a design paradigm that provides the foundation for this dissertation (see Section 1.2.1).

Glover (Glover, 1986) postulates that integer programming methods and artificial intelligence based methods, both stemming from a common origin, are now reuniting and creating a new class of algorithms capable of solving a large class of problems. In MDO,

integer programming or calculus based methods have been used as the optimization techniques to solve design problems. But, multidisciplinary problems inherently consist of both discrete and continuous variables which require other solution methods than calculus based ones. Unlike its continuous counterpart, optimality criteria such as the Karush-Kuhn-Tucker conditions for discrete problems do not exist. Recently, researchers are using a class of algorithms, which Glover calls "artificial intelligence" (Glover, 1986), to solve problems such as mixed discrete/continuous design problems. These artificial intelligence methods are based on various heuristic based searches or pattern moves. A brief review of the most recent advancements with these algorithms follows.

In Renaud (Renaud, et al., 1994), the simulated annealing (SA) algorithm, which is based on the heating and cooling schedule in an annealing process, is used to solve mixed discrete/continuous problems. This algorithm involved sequential discretizing of the continuous domain and then solving the problem using the SA algorithm. They refer to the algorithm as successive simulated annealing (SSA) as the continuous variables are successively discretized, using a finer and finer mesh as a design process progresses. The algorithm was tested with a system with five discrete and one continuous variable with good results. However, the computational expense of discretizing more continuous variables explodes quickly. Therefore, a better method of approximating the continuous domains or using fewer intervals may be improvements to the algorithm. In Sellar (Sellar, et al., 1994) neural networks are used to simulate the continuous and discrete design space. The benefits of neural networks in domain specific applications are evident, but across domains, neural networks must continuously be re-trained according to the problem specific information. For problems that may not change much over long periods of time, neural networks are beneficial, but for systems that are continuously undergoing improvements and changes, neural networks are limited.

In Zhang (Zhang and Wang, 1993) a SA algorithm is developed which modifies the step sizes and neighborhood move strategy based on 1) discrete or continuous variables and 2) optimization process stage. Step sizes are set by the designer based on variable types. Combined moves, where two or more variables can change, are used in the early stages of the process to traverse open spaces quickly, while orthogonal moves, where only one variable changes, are utilized in the later stages of the process. The drawbacks to this approach are the computational expense of the algorithm, the definition of the penalty function in constraint handling, and the sensitivity of the algorithm to cooling schedule parameters which vary with application. Future work included seeking modifications to reduce the computational expense.

In 1982, Glover introduced the tabu search, a heuristic algorithm based on hiding certain moves to prevent cycling and then searching in a given neighborhood for improving designs (Glover, 1989a). The term "tabu" implies that certain moves are not allowed for a certain time frame according to current visit status. In Bland (Bland and Dawson, 1989a, Bland and Dawson, 1989b), the tabu search has been shown to be an effective method to solve discrete problems such as ordering or placement problems. In Ford (Ford and Bloebaum, 1993), the tabu search is used as the discrete solver in the discrete subspace optimizations. One caveat of the tabu search is that it is an unassuming algorithm; that is, it does not know when an optimum or satisfactory solution has been reached. It will continue to search until a maximum iteration is reached. Certainly in a MDO environment where computational efficiency is paramount, stopping criteria must be adequately defined for a given problem. Applications to mixed discrete/continuous problems have not been developed.

Zabinsky (Zabinsky, et al., 1993) has developed Improved-Hit-and-Run, a random search algorithm that at each iteration generates a candidate point for improvement that is uniformly distributed along a randomly chosen direction within the feasible region. This algorithm combines pure adaptive search, which produces an improving point with each iteration, with Hit-and-Run methods, which generate a sequence of random points by providing a random direction and then providing a uniform random point in that direction. The algorithm, largely based on notions in operations research, has been effectively used in composite laminate design (Zabinsky, et al., 1992). It has been demonstrated to be effective in problems with only continuous or only discrete variables, but has the capability to handle both. Its effectiveness in an MDO environment is unknown, but seems promising based on previous results.

Genetic algorithms, which are based on the natural processes of evolution, mutation, and selection based on fitness levels, have shown promise in scheduling and optimization problems in MDO. In Hajela (Hajela and Shih, 1989), genetic algorithms are shown to be an alternative to solving nonconvex optimization problems and in Hajela (Hajela, 1995), genetic algorithms are used in the multidisciplinary design of rotor blades. In McCulley (McCulley and Bloebaum, 1994) genetic algorithms (GA) are used to order the tasks in a multidisciplinary design process. Limited success was found for design problems made up of relatively few tasks. With a large number of tasks or a very large multidisciplinary design problem, the computational expense of GA's is often too high. Other work is being conducted to reduce intelligently the number of evaluations in genetic algorithms without sacrificing solution effectiveness.

Other attempts to solve mixed discrete/continuous problems without the use of heuristic algorithms have had limited success. Cutting plane algorithms in general require a large

number of cuts to produce an integer solution. Branch and bound techniques in nonconvex problems may fathom nodes that are not feasible and also require a large number of function evaluations. In Loh and Papalambros (Loh and Papalambros, 1991), a sequential linearization technique is used. This technique begins by assuming all variables continuous and solving using a continuous solver. Then the solution is rounded to the nearest integer solution and the objective function and constraints are linearized about that point. The problem is then solved using an LP solver and the linearization and LP solution process is repeated until no change in the solution is apparent. Convergence to the global optimum is only guaranteed if the objective is pseudoconvex and the constraints are linear or convex. The authors talk about constraint forms and suggestions to obtain better forms of the constraints for the optimization. Many times in engineering, objective functions and constraints are non-convex. A technique is needed to handle these types of functions.

In Fu (Fu, et al., 1991), a strict penalty function is used to enforce integer values. The continuous problem is solved first, then the penalty function is used to further constrain the integer variables. Different starting vectors are used to ensure the robustness of a solution. An appropriate penalty function may be problem dependent and difficult to identify. Also, the algorithm showed great sensitivity to starting vectors and initial input factors.

These types of algorithms, whether calculus based or heuristic in nature, and the advantages and disadvantages of each are presented as being parallel developments that MDO researchers can utilize. Algorithms to solve mixed discrete/continuous problems are necessary in MDO whether it be at the subsystem level or at the system level. Systems invariably consist of discrete and continuous variables and the development of

robust algorithms to handle the pitfalls involved in continuous, discrete, and non-convex optimization are necessary to the practical evolution of MDO.

Research Opportunities

The ALP Algorithm (see Section 1.2.2) has been used extensively in design problems that consist of continuous or Boolean variables. It is been shown to solve a wide class of problems with great success. The primary opportunity is to expand the ALP Algorithm to handle discrete and integer variables by developing a heuristic search engine for the ALP. This is one of the primary contributions of this work, supporting Hypothesis III in Sections 1.3.2 and 3.4 and Chapter 6.

A fundamental notion in design, many times overlooked, is the presence of multiple objectives in a design problem. Developing the mathematical capabilities to handle multiple objectives to study tradeoff scenarios in complex systems design is necessary to facilitate satisfying the various customer requirements in an effective manner. In the next section, methods to model and handle multiobjectives in design are presented.

2.3.6 Multiobjectives

Multiobjective algorithms and approaches have largely been developed outside the aerospace field, but are now becoming more accepted based on their successful application in fields such as marine design and structures. Many multiple objective or attribute approaches have been developed for application in MDO. A general approach, proposed in Sen (Sen and Yang, 1993) calls for the analysis of design concepts based on multiple criteria (attributes *or* objectives) without clarifying distinct disciplinary boundaries. Attributes are used to make a selection from a set of choices, and objectives are used in the synthesis of a concept. In Sen's approach, both objective and subjective

factors can be used in a design process. Sen uses an analytical hierarchy process (Saaty, 1980) to combine the different criteria from different levels.

In order to analyze a system based on multiobjectives, a solution scheme must be based on a ranking of these objectives. If precise weightings are known (the preference of one objective over another is precisely known), a single objective formulation can be constructed based on relative weights. However, if a designer only knows the preferences (and not by how much one is preferred over another) a priority ranking scheme must be used. In Messac (Messac and Hattis, 1995), "physical programming" is used to capture a designer's preferences in a mathematically consistent manner in order to avoid needless iterations to determine the objective weightings. In Hajela (Hajela, 1990), a branch and bound algorithm is used to incorporate integer and discrete design variables in multiobjective problems. In Matsumoto (Matsumoto, et al., 1993), a fuzzy logic scheme where objectives are ranked as being either "soft" or "hard" is used. Then, once the system is solved using the "hard" objectives, the "soft" objectives are used. If no improvement can be gained from the design based on the "hard" objectives, then a designer may sacrifice some of the "hard" objective in order to improve the "soft" objective. For instance, in Figure 2.6, there are three objectives, one hard and two soft. Based on the designer's objective targets, the shaded region ABC is the area satisfying all three priorities. Within this region, a designer's preferences and allowable deviations can be explored. At the design point x^1, the hard objective F_1 is fully satisfied, but F_2 and F_3 are sub-optimal. If a designer makes a decision to allow some deviation, $F_1 + \Delta 2$ and make F_3 more important than F_2, the design point x^2 could be reached where F_3 is fully satisfied. Likewise, a decision to allow $F_1 + \Delta 1$ and make F_2 more important than F_3, the design point x^3 could be reached.

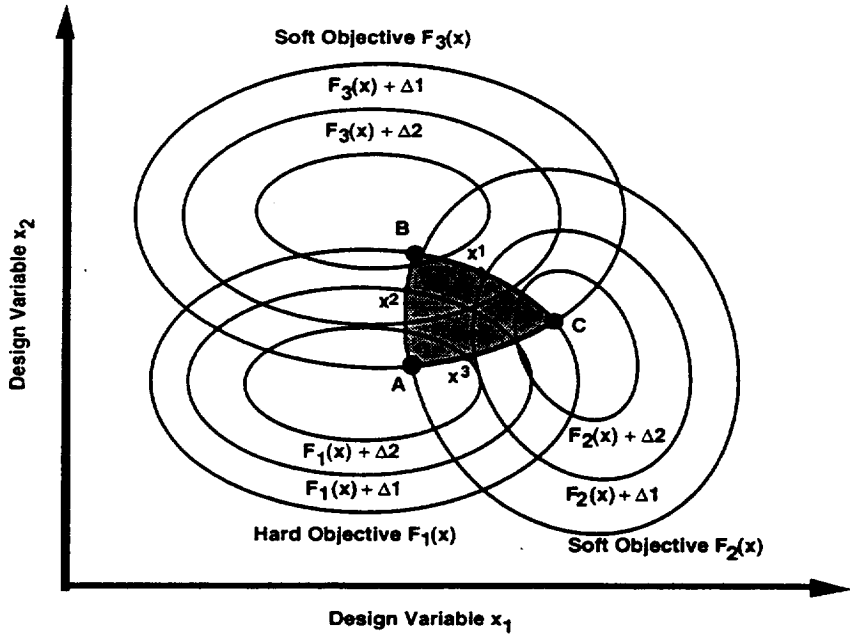


Figure 2.6. Concept of Priority Ranking Strategy

The authors also present objective categories for which the labels "soft" and "hard" apply. For instance, those objectives concerned with the protection of the environment should be "hard" while those concerned with comfort should be "soft". This approach is very similar to the fuzzy priority scheme implemented in (Zhou, 1988).

In (Lewis, et al., 1994), multiobjective designs are analyzed based on the *lexicographic minimum* concept. This concept is defined as follows (Ignizio, 1985a).

LEXICOGRAPHIC MINIMUM Given an ordered array $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ of nonnegative elements f_k's, the solution given by $f^{(1)}$ is preferred to $f^{(2)}$ iff

$$f_k^{(1)} < f_k^{(2)}$$

and $f_i^{(1)} = f_i^{(2)}$ for i = 1, ... ,k-1; that is all higher-order elements are equal. If no other solution is preferred to **f**, then **f** is the lexicographic minimum.

The lexicographic minimum concept is also similar to the approach developed by Stadler (Stadler, 1988) who stresses the history and importance of multiobjective approaches in all types of design. To illustrate the lexicographic minimum concept, consider the design of aircraft that is multiobjective. Say that there are three goals,

- Take-off Weight
- Landing Field Length
- Number of Passengers or Fuselage Volume

that a designer must consider. Designers make the decision that take-off weight is the most important goal, while landing field length is not as important, and number of passengers is least important. Further, say there are three possible designs, shown below with their goal achievement.

$\mathbf{f_1}$ = (210,000 lbs., 5000 ft., 170 passengers)
$\mathbf{f_2}$ = (235,000 lbs., 4500 ft., 190 passengers)
$\mathbf{f_3}$ = (210,000 lbs., 4700 ft., 180 passengers)

To determine the best design using the lexicographic minimum approach, level one is considered first. Designs $\mathbf{f_1}$ and $\mathbf{f_3}$ are equally "good" at this level. Design $\mathbf{f_2}$ is not considered further even though lower priorities may be better than the other designs. At level 2, design $\mathbf{f_3}$ is better satisfied, and therefore, all other levels are ignored, and design $\mathbf{f_3}$ is considered the best design, because of its satisfaction of the top priority goals. This concept has been implemented the Adaptive Linear Programming (ALP) algorithm, introduced in Section 1.2.2 (Mistree, et al., 1993a).

One issue touched on by these approaches is the uncertainty and changing of the information in a design process. In Sen (Sen and Yang, 1993), a group of concepts may be analyzed based on multiple attributes, and the final concept will be analyzed based on multiple objectives. In Matsumoto (Matsumoto, et al., 1993), it is recognized that precise rankings are often unavailable, and identifying broad groups of objectives may be the

only alternative for a designer when there is much uncertainty about the design. Further, preemptive ordering of objectives may precede Archimedean ordering in the earlier stages of design before precise weighting are known. In any case, the interaction of a designer with the computer-based tools, as a means to update system models and/or tools as knowledge is gained, is essential in MDO. In the next section, this issue of human-computer interaction is addressed.

2.3.7 The Human Factor

In Barkan (Barkan and Hinckley, 1993), a very important but often overlooked point is made about design methodologies. Citing studies from U.S. firms, it stresses that following one set of design steps or rules could many times lead to suboptimal designs and highly inefficient design processes. The point Barkan tries to make is for designers in any field to keep their minds open to many theories, methods, and rules concerning what should be done in design. Single structured methodologies such as Functional Analysis, Quality Function Deployment, Robust Design, and Design for Assembly should not be applied blindly across the design process. Using aspects from various methodologies and philosophies throughout a design process is how MDO has been evolving recently.

In Hale (Hale, et al., 1995, Hale, et al., 1996), a design infrastructure is being developed which integrates a decision-based architecture called DREAMS with a computing infrastructure called IMAGE. This work addresses both process and product issues in a design process and establishes the human interface to both the computer-implemented design product and process models. The Framework for Interdisciplinary Design Optimization (FIDO) (Townsend, et al., 1994) program has recognized this need and

developed a "housing" for MDO, but the contents of the various "rooms" are prescribed by the specific residents. In Figure 2.7, the FIDO framework is demonstrated.

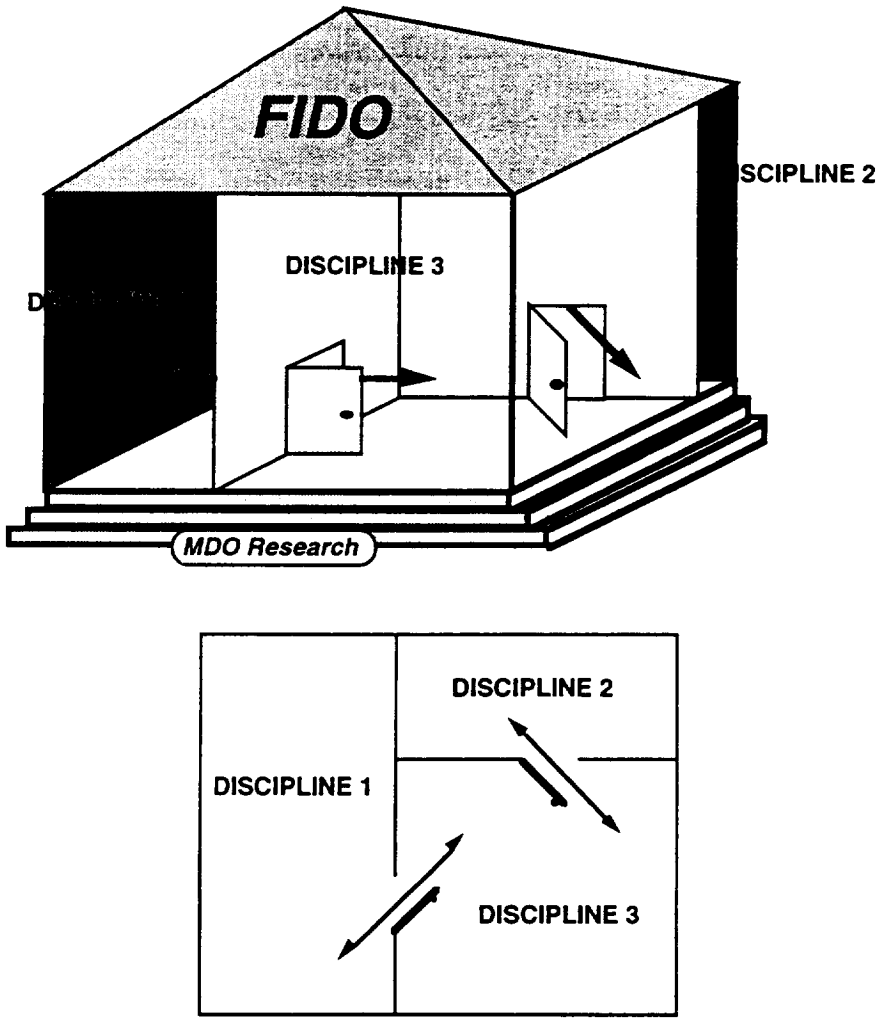


Figure 2.7. Framework of FIDO for MDO Implementation

The residents in the house of a complex problem are the various disciplines in a MDO problem. Each discipline has its own solution software, formulation philosophy, and analysis approach. FIDO allows for the combination of these various methodologies under a single roof, all based on the foundation of research in MDO. But how do the various methodologies of each discipline "see" into the other rooms in the house? The

openings that connect two or more "rooms" or disciplines are resolved using a state-of-the-art, on-line graphics interface where designers can see the progress of the design and can visualize the effects of any changes that are made on the other disciplines and the entire system. This type of framework allows a complex system to reap the benefits from various design methodologies, philosophies, and technologies. In addition, the framework allows for the parallel development and implementation of the disciplines and other sciences that contribute to MDO.

The focus of a designer as an interactive decision maker throughout a design process leads to the need of having appropriate information available for a designer at any given time. In the next section, the issue of information storage, transfer, and availability is discussed.

2.3.8 Information Storage and Transfer

In the design of complex systems, disciplinary design teams working on the same system are many times geographically distanced within the company. Because each discipline is dependent on the others, the information in a usable form from each discipline is necessary for the other ones. Therefore, the use of effective databases is becoming necessary in multidisciplinary design. However, in design, the use of databases to only *store* information is not enough. Information is being instantly utilized by the other design groups. Therefore, the database is being used as a "wipeboard" of sorts where information is stored for a short time, and is replaced by new information, generated by other disciplines. As a result, the role of the database, in complex systems design, is more than storing information, it must *transfer* the information in a usable way. The nature of the information indeed changes as the design process evolves. The database becomes a *dynamic* system rather than a *static* one. These issues and other relevant to

managing information in engineering design are explored in (Fulton, et al., 1989). In (Hazelrigg, 1996), information-based design is addressed and it is asserted that systems engineering is a viable approach to handling the complexities in informationally driven design processes.

The ability to balance the demands of accurate but efficient information is paramount in complex system design and analysis. The availability of information frequently dictates at what level of detail designers can perform experiments or even make decisions. Experimental methods can even be used to lessen the burden of information availability through the use of approximation techniques. Experimental design methods are being talked about in the same context as information availability since there is such a strong relationship between them. In the next section, experimental design methods, design quality, and robust design and their applicability to effective and efficient system design and simulation are discussed.

2.3.9 Experimental Design Methods: Balancing Efficiency and Quality

System simulation is performed at all levels of design from "back of the envelope" calculations in the early stages of design to prototyping in the later stages of design. Making the simulation as efficient as possible while maintaining an acceptable level of effectiveness is an important and difficult issue in system and subsystem simulation. In the following, efficient experimental design methods as well as robust design techniques are presented.

Experimental design methods

In the design of experiments, a finite number of designs in the design space are simulated using prescribed settings of the design variables and system evaluation routines. How

small or large a number the term "finite" implies is the dilemma of full factorial experiments versus fractional factorial experiments. Taguchi utilizes a special class of fractional factorial matrices, called Orthogonal Arrays (OA) to span the design space efficiently while maximizing the effectiveness of the information. OAs also can simulate control factors (design variables) and noise factors (uncontrollable factors, such as environmental effect) in one OA. In Stanley (Stanley, et al., 1992) Taguchi's OAs have been applied to the design of Single Stage To Orbit (SSTO) vehicles. In Lewis (Lewis, et al., 1994), OAs are used to simulate and explore the multidisciplinary behavior of a Boeing 727-200 effectively . Box (Box, et al., 1978) has introduced the Central Composite Design (CCD) experiments as modifications to the OA. These types of experimental methods combined with response surface methodologies produce a powerful simulation tool that can be linked to optimization techniques in complex systems design. This is demonstrated and further explained in (Chen, 1995a, Chen, 1996, Olds, 1994, Unal and Stanley, 1992, October).

Robust systems design

In robust design, the effects of noise factors are reduced without eliminating the causes of the noise. Robust design is an excellent method of designing quality into the design process and product. Taguchi, an early proponent of robust design, builds his philosophy on the notion of not finding optimums, but regions of low variability (Taguchi, 1987). This notion can be traced back to Simon (Simon, 1982), who introduced the notion of "satisficing" as opposed to optimizing. Simon states:

> *"The decision that is optimal in the simplified model will seldom be optimal in the real world. The decision maker has a choice between an optimal decision from an imaginary simplified world, or decision that are 'good enough', that satisfice, for a world approximating the complex real one more closely."* (Simon, 1982)

Another way of putting this is the "betterization" of a design instead of the optimization of a design (Stadler, 1988). Stadler states that the true optimization of a design is close to impossible. A more practical approach is making the design better, or the betterization of a design.

The techniques of Taguchi and the notion of "satisficing" have been applied in various MDO applications. Taguchi's measure of the quality of the design is the signal-to-noise ratio, a ratio of the mean value to its standard deviation. In (Mistree, et al., 1993b, Olds and Walberg, 1993, February, Stanley, et al., 1992) the Taguchi approach to robust design has been incorporated into the design of complex systems such as a Life Satellite Vehicle and SSTO space vehicle. There are drawbacks to Taguchi's approach to robust design. These drawbacks are well documented in (Box, 1988) and include the single objective (signal-to-noise ratio) nature of the approach. Researchers are finding excellent results integrating robust design methods into MDO (e.g., (Chen, 1995a)). However, they must not be applied blindly, but must be intelligently synthesized with other methods and strategies discussed in this chapter. Measuring and maximizing the quality of a product or process along with efficient experimentation is a very important aspect of the design of any system, including multidisciplinary systems.

2.3.10 Applications of MDO

Although the roots of MDO are being attributed to the field of structures in aircraft design, multidisciplinary design optimization has been performed for years in many other disciplines. It is only recently that these areas are being recognized as multidisciplinary design optimization application and research areas. It is the unifying field of MDO which has brought together developments from a variety of applications.

Much of the focus of MDO applications is in the area of flight systems, both orbital and non-orbital. NASA, Boeing, Lockheed, and McDonnell-Douglas are each independently and jointly researching MDO technologies in aircraft design, including the High Speed Civil Transport (HSCT). In space system design, work concentrated at NASA-Langley focuses on applying MDO technologies to the design of advanced, manned transportation system concepts including the new family of space vehicles (Olds, 1992, Stanley, et al., 1994). Also, MDO technology has been applied to trajectory optimization problems in ground to mission vehicles (Beltracchi, 1990). In civil engineering, applications of MDO include the design of steel and concrete systems (Balling, 1993, Fang and Azarm, 1994). In mechanical engineering, applications include the design of damage tolerant structural and mechanical systems, mechanisms (Mistree, et al., 1990a), and thermal energy systems (Bascaran, et al., 1989, Vadde, et al., 1992). Overlapping in each field is the study of materials which forms the foundation of complex systems. Many times the selection of materials is coupled with the determination of physical design variables, further increasing the complexity of the system analysis.

2.4 A LOOK BACK AND A LOOK AHEAD

In this chapter, Phase I of the strategy for implementation and verification of this dissertation, as outlined in Section 1.3.2, is completed. In Chapter 2, the foundation of the dissertation is further solidified, as shown in Figure 2.8. The needs and research opportunities that provide the motivation and background for the dissertation are identified through a comprehensive literature review. The foundation for the dissertation is now complete and in Chapter 3, the algorithm for integrated subsystem embodiment

and system synthesis is presented, building upon the foundation built in Chapters 1 and 2. In Chapter 3, Phase II of the strategy for implementation and verification (see Section 1.3.2) is initiated.

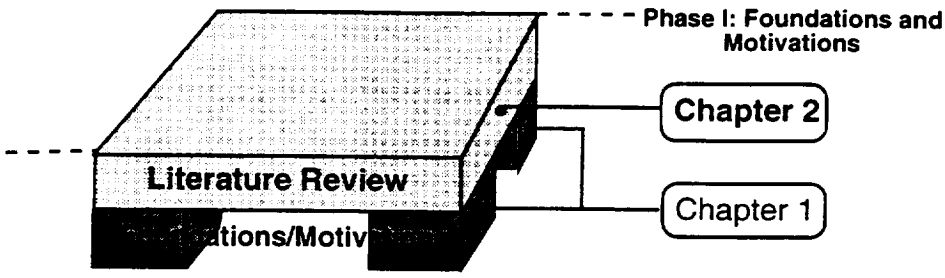


Figure 2.8. Frame of Reference: Chapter 2

CHAPTER 3

THE ALGORITHM, TECHNOLOGY BASE, AND RESEARCH HYPOTHESES - VERIFICATION GUIDELINES

Having addressed the research background and research opportunities in Chapters 1 and 2, the algorithm for integrated subsystem embodiment and system synthesis is first presented in this chapter. The algorithm is presented as a step-by-step approach for integrating the solution and coordination of subsystems. This algorithm is developed based on several research hypotheses. The focus in this chapter is to provide the background, ramifications, and verification guidelines for each hypothesis. Verification studies of the hypotheses are presented in Chapters 4, 5, and 6, and the motivating case study is presented in Chapter 7.

In this chapter, an overview of the algorithm is given. The research hypotheses, and supporting posits follow in Section 3.1.3. Sections 3.2, 3.3, 3.4, and 3.5 are devoted to testing the four hypotheses, respectively. For each hypothesis, ramifications are provided, a literature background is presented, and verification guidelines are discussed. Associated with the respective four hypotheses are a set of characteristics for complex systems design taxonomies (Sections 3.2.1 and 3.2.2), the use of various approximation techniques including the Design of Experiments and Response Surface Methodology (Section 3.3.4), constructs from various discrete optimization algorithms (Sections 3.4.2-3.4.3), and a formal proof of convexity (Sections 3.5.1-3.5.3). For Hypothesis IV, the guidelines for verification are straightforward, and the proof to discount Hypothesis IV is given in Sections 3.5.1-3.5.3. A review of the examples and motivating study used to verify and illustrate the developments of this work is given in Section 3.6. Finally, this chapter is closed with a look back at what has been presented and a look ahead to what is next.

3.1 AN OVERVIEW OF THE ALGORITHM AND RESEARCH HYPOTHESES

3.1.1 An Algorithm for Concurrent Subsystem Embodiment and System Synthesis

The algorithm presented here is *conceptual* in nature. In other words, the algorithm is a conceptual procedure, and not an automated computer system. Parts of the algorithm have supporting computer packages, but an encompassing computer infrastructure does not exist. A schematic of the overall algorithm is shown in Figure 3.1. There are three distinct steps: 1) Classify problem based on system model, 2) Formulate subsystem models and interactions, and 3) Solve and coordinate subsystems.

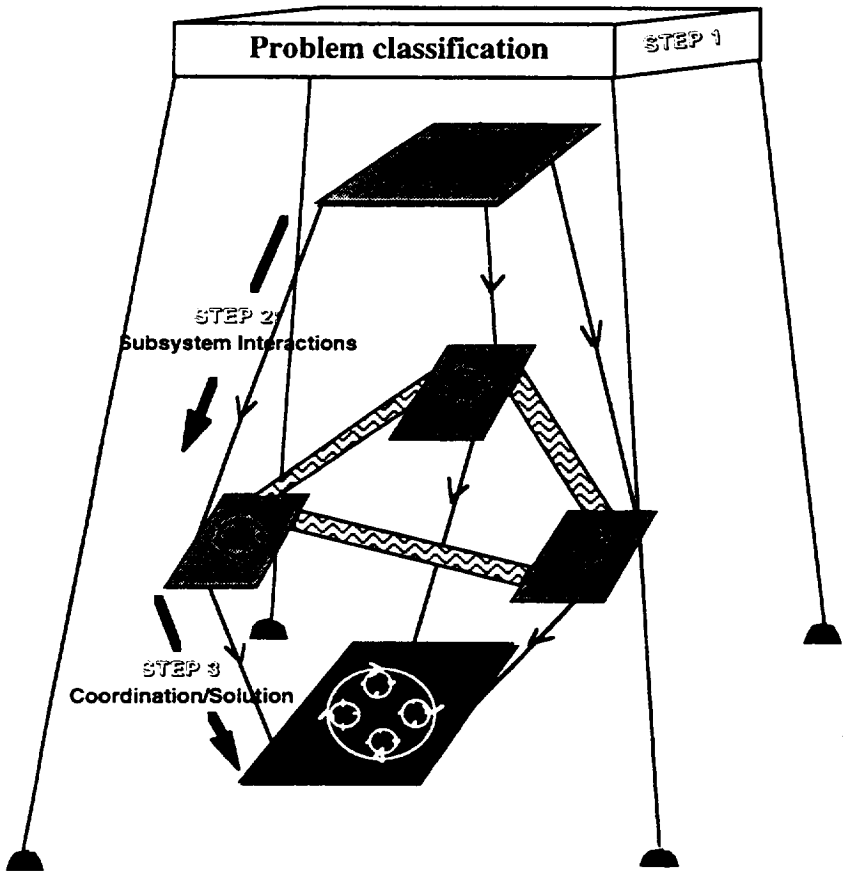


Figure 3.1. Schematic of Overall Algorithm

These steps constitute an *algorithm* as opposed to a *method* simply because of the mathematical rigor behind Steps 2 and 3 (see definition of algorithm in Section 1.1.5). In (1984), method is defined as

A systematic and regular way of accomplishing a given task.

Although the steps introduced are systematic, it is the *mathematics* of the steps that constitute the use of the term "algorithm." These steps are explored with reference to Figure 3.1.

Step 1. Classify problem and process based on structure of the system model.

In Figure 3.1, this step is shown as the supporting structure surrounding the inner parts of the algorithm. This is the function of this step: to provide the foundational support for the remaining steps. In this step, the design problem and process formulation are classified based on the system model and the assigned design teams. A classification system provides a *linguistic basis* for the structuring of a system and its associated *modeling and solution processes*. The classification developed and used is a three-level lexicon that builds upon previous classifications and establishes the decision as the fundamental design construct. The hypothesis and posits supporting this step are provided in Section 3.2.

Step 2. Based on the classification from Step 1, formulate appropriate compromise DSP for each disciplinary subsystem.

In Figure 3.1, this step is shown as the top half of the inner portion, from the top level system to lower level subsystems, which may be interacting. It is asserted that the interactions among the subsystems can be abstracted as games and the relationships modeled using game theory protocols. Based on the protocol (relationship) among the subsystems, established in Step 1, compromise DSPs are

formulated for each subsystem. The four protocols used are full cooperation, approximate cooperation, noncooperative, and leader/follower formulations. These protocols are introduced and defined in Section 3.3.3. Depending upon the protocol, different information is available to the different subsystems. Therefore, the constructs to formulate and process information for each subsystem change with each protocol, but the core compromise DSP of each subsystem remains the same. However, depending on what information is available, the *solution* of the compromise DSPs may change. The hypothesis and posits supporting this step are provided in Section 3.3.

Step 3. Solve the disciplinary models and coordination problem based on the classification and interactions from Steps 1 and 2.

This step again uses one of the four possible protocols: full cooperative, approximate cooperative, noncooperative, and leader/follower. In Figure 3.1, this step is shown in the lower half of the inner portion, from the subsystems to the integrated system at the bottom. The protocol established in Step 1 dictates the solution process that is used to solve and coordinate the subsystem compromise DSPs. Depending upon the protocol and presence of discrete and/or continuous design variables, different solution techniques are used to solve and coordinate the disciplinary models. It is also required to handle nonconvex functions in the solution process. The hypothesis and posits supporting this step are provided in Sections 3.4 and 3.5.

These steps are motivated by ideal aspects of an algorithm for integrated subsystem embodiment and system synthesis, as identified in Section 1.1. In Figure 3.2, an "ideal" algorithm is shown on the left. On the right, the needs and foundation for such an algorithm are shown. Within the body of existing work are various needs or "holes" which

represent open research questions. The holes addressed in this dissertation, reviewed and identified in Chapter 2, are: a basis of linguistic communication, realistic modeling of interactions in multidisciplinary design, the solution of mixed discrete/continuous problems, and the capability of handling nonconvex functions in an optimization context. It is these holes which represent the motivation for this dissertation. These holes are filled through the formulation, verification, and implementation of four hypotheses.

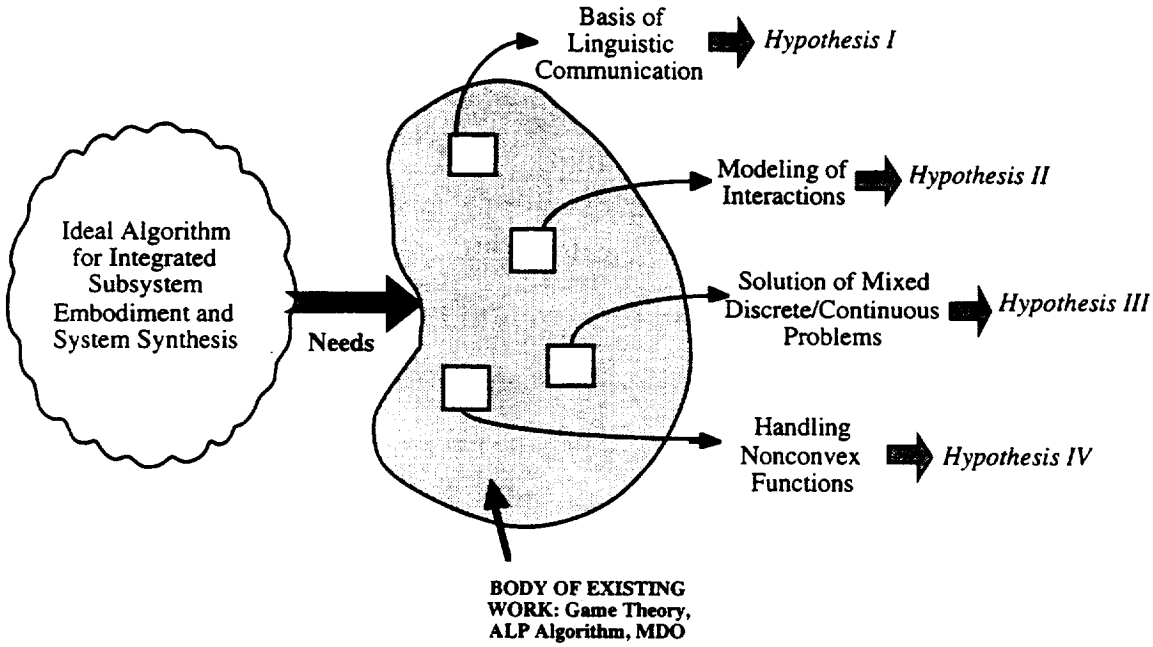


Figure 3.2. Needs, Opportunities, and Hypotheses

The implementation of the hypotheses is realized through three steps which constitute "an algorithm for integrated subsystem embodiment." As a roadmap to this chapter, Figure 3.3 combines the algorithm schematic, research hypotheses, and tools used for each hypothesis. In Sections 3.2 through 3.5, the four hypotheses are presented along with background of the tools used and verification guidelines.

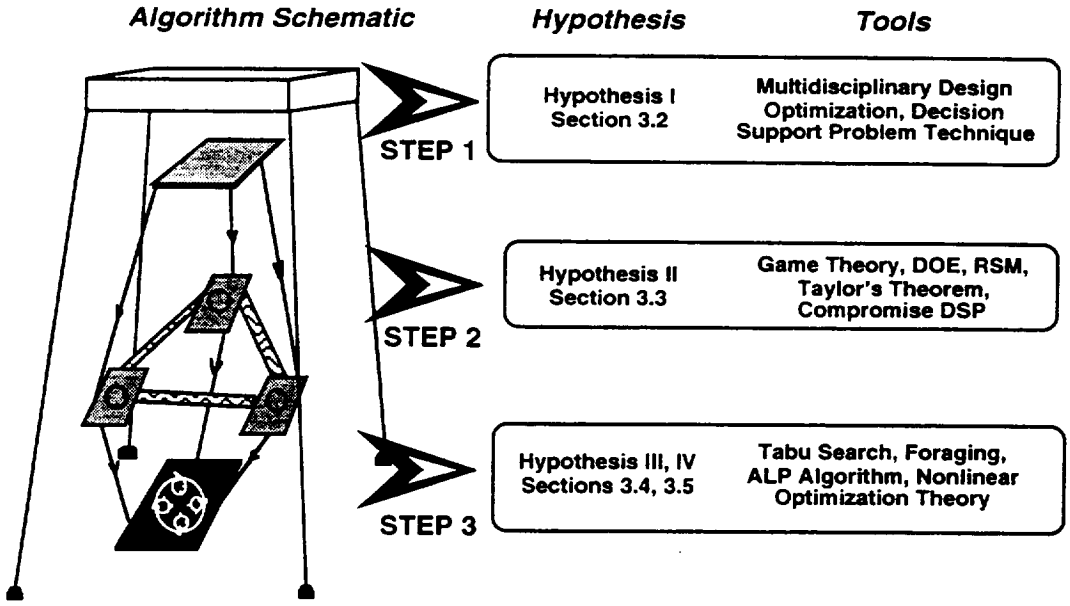


Figure 3.3. The Algorithm, Hypotheses, and Tools: A Roadmap

The tools in Figure 3.3 are shown beside the primary step where they are applied, but there is overlap of the tools in multiple steps. This overlap of the constructs and tools used in each step is shown in Figure 3.4. The constructs and tools of Step 1 are largely conceptual and are not implemented in a formal sense on a computer. The constructs and tools of Steps 2 and 3 have formal structure on a computer. The computer implementations for Steps 2 and 3 are presented in the next section.

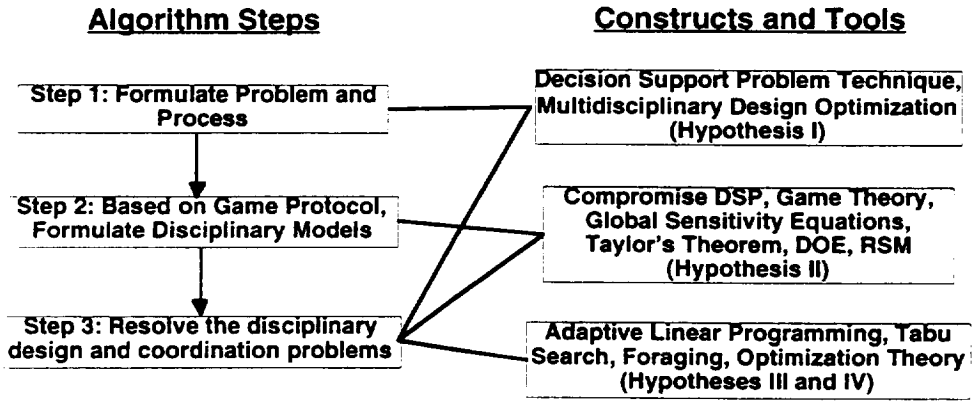


Figure 3.4. Overlap of the Constructs and Tools

3.1.2 Computer Implementation of Algorithm

Depending upon the protocol among the design teams established in Step 1 of the algorithm, different theoretical and computational tools are used. In Step 2, there are four primary game protocols that are used (see Section 1.2.3): full cooperation, approximate cooperation, noncooperation, and the leader/follower protocol. In Figure 3.5, the computer infrastructure for implementing the computer-based portions of the algorithm is shown. The input is the game theoretical protocol that exists among the subsystems (players). This is a result of Step 1 of the algorithm. Then, depending upon the protocol, different tools are used to model the interactions in Step 2 of the algorithm, and to solve the resulting models in Step 3. The major components of the *existing* computer infrastructure shown in Figure 3.5 include four processors (a nonlocal approximation processor, module A, a design of experiments/response surface processor, module B, and a solution processor, module D), each centered about the primary processor, the compromise DSP, module C.

The full cooperative protocol (defined in Section 3.3.3) is the simplest case and uses only the compromise DSP (module C) in Step 2 to formulate the problem. The appropriate solution scheme in DSIDES (module D) is used in Step 3 to solve the formulation depending upon whether discrete variables are present in the model.

The approximate cooperative protocol (defined in Section 3.3.3) utilizes a nonlocal approximation processor (module A) based on the Global Sensitivity Equations (GSE) and Taylor Series in Step 2. The nonlocal approximation processor is embedded within the players' compromise DSPs (module C). The resulting compromise DSPs are solved in Step 3 using the appropriate solution scheme in DSIDES (module D).

For both the noncooperative and leader/follower protocols (defined in Section 3.3.3), a Design of Experiments processor (module B) is used to generate a Response Surface Model of the players' rational reaction sets in Step 2. To generate these RSMs, the DOE processor calls the players' compromise DSPs (module C) as the simulation routine. At each simulation, the appropriate solution scheme in DSIDES (module D) is used to solve the model. In the noncooperative protocol, Mathematica is also used as the solution processor (module D) in Step 3, in order to find the intersection of the rational reaction sets. In the leader/follower formulation, the rational reaction sets are then used by the appropriate players in their compromise DSPs, which are again solved in Step 3 using the appropriate solution scheme in DSIDES (module D). The result is a set of solutions which correspond to the various protocols.

The different tools and techniques used in Steps 2 and 3 for each protocol are shown in Table 3.1. Specific discussion of the computer implementation of these tools and techniques in the various protocols is discussed in Section 5.5.

Table 3.1. Tools used in Each Protocol

	Protocol			
	Full Cooperation	*Approximate Cooperation*	*Noncooperation*	*Leader/Follower*
Tools Used	Sequential Linear Programming (SLP), Heuristic Search	Global Sensitivity Equations (GSE), Matrix Solver, Taylor Series, SLP	Design of Experiments, SLP, Matrix Solver	Design of Experiments, SLP, Heuristic Search

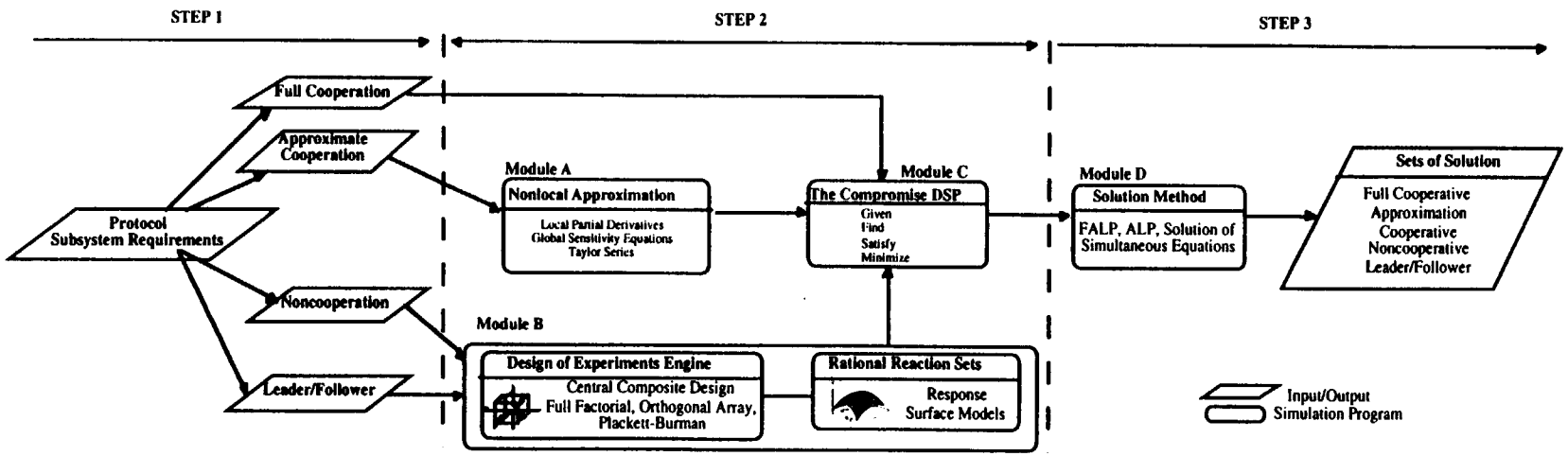


Figure 3.5. Computer Infrastructure for Computer-Based Parts of the Algorithm

Corresponding to these computer-based tools are a set of theoretical approximation concepts which are used in each protocol. Applying game theory principles to complex systems design requires approximation of various game-theoretical constructs because of the complexity of the analyses. The approximation concepts used in each protocol are shown in Table 3.2.

Table 3.2. Approximation Concepts used in Each Protocol

	Protocol			
	Full Cooperation	*Approximate Cooperation*	*Noncooperation*	*Leader/Follower*
Approximation Concepts Used	Sequential Linear Programming	Global Sensitivity Equations (GSE), Taylor Series, Sequential Linear Programming	Response Surfaces, Sequential Linear Programming	Response Surfaces, Sequential Linear Programming

3.1.3 Hypotheses and Posits

In Section 2.3, the areas of research of this work are presented and reviewed. These namely are *problem and process classification, subsystem interaction, mixed discrete/continuous optimization,* and *nonconvexity.* A literature review of these areas is provided in Section 2.3, along with other areas related to these in the design of complex systems. It is in these four areas where the four hypotheses of this work are derived. Associated with each hypothesis is a set of supporting posits. As the hypotheses and posits are unique for this research topic, they are considered the fundamental contribution of this dissertation.

Hypothesis I: Classification of problem and process in multidisciplinary design can be extended by integrating constructs from Decision-Based Design, Game Theory, and Multidisciplinary Design Optimization.

Hypothesis II: Game theoretic principles can be applied to accurately model and describe the interactions in complex systems design.

Hypothesis III: The notion of foraging of wild animals is a natural analogy for optimization and can be used as an effective search technique in the solution of mixed discrete/continuous models.

Hypothesis IV: The g-function is a useful transformation of nonconvex functions into well-behaved convex functions.

Hypothesis I Posits

Posit 1.1: Entities from the Decision Support Problem Technique provide a domain-independent lexicon for multidisciplinary design.

Posit 1.2: Game Theory principles can be used to extend problem formulation in multidisciplinary design.

Hypothesis II Posits

Posit 2.1: Design processes can be abstracted as games where the players are multiple designers or design teams and their associated analysis and synthesis tools.

Posit 2.2: Approximate cooperation can be modeled using the Global Sensitivity Equations and Taylor series to approximate nonlocal equations.

Posit 2.3: First order Taylor series can be used as a good approximation of nonlocal state equations.

Posit 2.4: Second order response surfaces can be used to approximate the Rational Reaction Sets of the disciplinary players in a design game.

Posit 2.5: The compromise DSP can be used as the fundamental construct to develop the game theory protocols and techniques.

Hypothesis III Posits

Posit 3.1: Foraging is a heuristic, under which characteristics from genetic algorithms, Tabu Search, and Simulated Annealing can be grouped.

Posit 3.2: The Tabu Search can be used as the building block for the foraging solution algorithm.

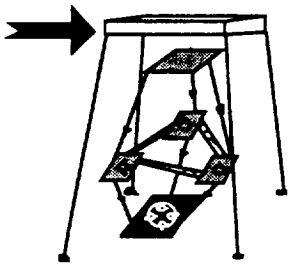
Posit 3.3: The ALP Algorithm along with foraging can be used to effectively solve mixed discrete/continuous problems.

Hypothesis IV Posits

Posit 4.1: Nonlinear optimization theory can be used to prove/disprove the effectiveness of the g-function in transforming nonconvex constraints and goals in the compromise DSP to convex equations.

In the next section, Hypothesis I is discussed, including the necessary technical background, ramifications, and verification guidelines.

3.2 TEST OF HYPOTHESIS I - DEVELOPMENT OF PROBLEM AND PROCESS FORMULATION



Ramifications and verification guidelines are provided for testing Hypothesis I in this Section. The verification guidelines are discussed in Section 3.2.3. It is asserted a classification system for problem and process formulation in design should

- have the capability to classify the roles of multiple designers,
- have the capability to handle multiple levels of detail,
- be domain-independent, and
- be independent of time and technology.

In Section 3.2.2, these four characteristics are discussed in the context of MDO. Design, however, is neither a science nor an art. Therefore, classification systems in design are difficult to implement due to the numerous interpretations of design. In Section 3.2.1, the

amorphous nature of classifications in design is discussed as a precursor to the remainder of Section 3.2.

3.2.1 Nature of Classifications in Design

In accordance with the Collins English Dictionary, *taxonomy* is defined as

- the principles of classification or order (1976).

In the field of science, the term *taxonomy* has evolved into a term, synonymous with classification, based on the principles of order. Taxonomies are used in various areas of science to classify certain parts of the field according to some logical, structured ordering and to facilitate future communication and research among peers in the field. In largely creative areas, such as art, everyone typically has his or her interpretation of the field, and this interpretation is neither wrong nor right because of the large amount of ambiguity. Since design arguably is composed of aspects rooted both in science and in art, an accepted taxonomy is difficult to establish and has not been developed (Muster and Mistree, 1989).

Traditionally, a design process is divided into stages based on the requirements of the project's management. This division is usually some variation of the following process: problem definition, conceptual design, layout design, detail design, and manufacturing design. It has become more apparent that other areas such as designing for assembly, designing for recycleability, and designing for maintenance, or in other words, DFX, must be taken into account in the design process. Unfortunately, this taxonomy provides little information on what is being accomplished in each stage, the information flow among the stages, and the types of decisions required at each stage. Many researchers have attempted to develop a standard classification scheme for design with some of these areas in mind.

Some of these classifications are: taxonomies for mechanical designs and artifacts in the European literature (Hubka, 1982, Pahl and Beitz, 1984, VDI, 1986), classification of the mechanical design research and results based on a scheme of classifying initial and final states of knowledge of an object to be designed Dixon (Dixon, et al., 1988), classification of mechanical design according to design problem, research method, environment, and design process by Ullman (Ullman, 1992), classification of mechanical design by design type and design activity by Snavely (Snavely, et al., 1989), identification of eight approaches to design along with three tasks, design selection, parametric redesign, and design synthesis by Marshek (Marshek and Kannapan, 1987), and an expansion of a taxonomy of the entire Product Realization Process (PRP) by Mills (Mills, 1993). While each of these taxonomies has contributed to the science of design in some form, an accepted classification for communication and comparison does not exist.

It is among the contributions of this dissertation to expand the science of design by expanding classifications such as these to the design of *complex systems*, which may be carried out by *multiple designers*, design teams with their own analyses and syntheses routines at different levels of fidelity. Issues associated with multiple designers and multiple levels are addressed in the next section.

3.2.2 Multi-Player and Multi-Level Formulations

The underlying assumption in many of the general design taxonomies discussed in Section 3.2.1 is that design does not have to be performed by different designers and design teams who may be geographically distanced. However, in an industrial context, this is often the case, and as such, is fundamental motivation for this dissertation as well as for the field of Multidisciplinary Design Optimization (MDO). The term *methodology* is defined in Webster's Dictionary (1984) as

a body of methods, procedures, working concepts, and postulates, etc.

Consistent with that generic definition, MDO can be defined as a methodology for the design of complex engineering systems that are governed by mutually interacting physical phenomena and made up of distinct interacting subsystems (Sobieszczanski-Sobieski and Tulinius, 1991). From this definition, it is obvious that *any* engineering design problem in MDO will be performed by multiple designers and design teams working in multiple disciplines. Several approaches to formulating and solving a multidisciplinary design problem have arisen in a rather ad hoc fashion since the inception of MDO. These approaches include single-level and multi-level formulations, hierarchical and nonhierarchical system decomposition methods, and numerous optimization and analysis processes and approaches at the system and subsystem levels. The fundamental problems in MDO arise in the modeling, solution, and coordination of the system and subsystem models. Therefore MDO is driven by analysis and synthesis, as opposed to general design methods as in Section 3.2.2. As a result the classification of problem and process *in MDO* can be viewed as a subset of the general design classifications. A classification in MDO would be useful at a given snapshot of a general design process where a system model must be formulated and solved. As the field of MDO expands, it becomes increasingly necessary for a consistent classification system that can be used as a form of communication and comparison. It is asserted that the following characteristics are beneficial, if not necessary to an effective classification system in MDO.

- A classification system must address the possibility of *multiple levels*, multiple designers, multiple design teams, and multiple analysis and synthesis levels. But in complex systems design, the primary function of a designer does not change. The function is still *to make decisions*. Therefore, the foundation of the classification, it is asserted, should remain the decision. The focus of a classification system in

MDO should classify the tools, methods, and processes that are used to support the integrated decisions of multiple designers.

- A classification system should address the *nature of the interaction* among the designers and/or design teams from a mathematical, analysis-oriented perspective. Many times design teams who are geographically distanced from the other teams may have to act on their own and make assumptions about the other teams' decisions. Often, some design teams will take the lead in a design process while others will wait to perform their analysis and make their decision until later in the process. Further, some design teams may design according to their own requirements and goals while ignoring the goals of other disciplines. While this may result in an ideal design of their subsystem, it may not translate to an ideal system design. These situations are common in complex systems design and are also representative of typical situations in game theory. Classification systems should account for the nature of the interactions among the design teams at both the personal level and the analysis or synthesis level.

- A classification should be *independent of time-based developments*, such as technology. As technology continues to expand and better and faster tools and methods are developed, classifications should not change. An example is found in the area of chemistry. In chemistry, the framework is in the form of the periodic table. All research and technology, no matter how advanced, can be referred in some sense back to this table, and this will always be true. The pure sciences have set the standard for classifications of some sort. In design, or even multidisciplinary design, this type of framework is difficult due to the inherent lack of structure.

- A classification should also be *independent of the domain of application*. Designers working on aircraft design should be able to use the same classification entities as designers working on ship design. The complex system domain should not affect the classification used.

To address these issues, entities from game theory and the Decision Support Problem Technique (see Section 1.2.1) are integrated with an existing multilevel framework (Balling and Sobieski, 1994), as described in Chapter 4. In this dissertation, value is added to the evolving framework of MDO to stimulate its acceptance as a basis of communication (Lewis and Mistree, 1995). In the next section, guidelines for verifying Hypothesis I are given.

3.2.3 Guidelines for Verifying Hypothesis I: Problem and Process Formulations

Hypothesis I: Classification of problem and process in multidisciplinary design can be extended by integrating constructs from Decision-Based Design, Game Theory, and Multidisciplinary Design Optimization.

Hypothesis I is tested by using two posits. The guidelines and section numbers related to the testing of each posit are given.

Posit 1.1: Entities from the Decision Support Problem Technique provide a domain-independent lexicon for multidisciplinary design.

To verify this posit, a multi-level lexicon is presented in Section 4.2 that includes entities from the Decision Support Problem Technique. Previous attempts at defining a lexicon for multidisciplinary design have been focused in the field of aircraft design. The benefit of having a domain-independent lexicon is that designers from various fields and backgrounds are then able to communicate and compare problem formulations. The DSPT has been used in the design of many systems, from small and simple to large and complex. The work supporting this posit further establishes the DSPT as an effective set of ideas, tools, and entities to support designers in the design of engineering systems.

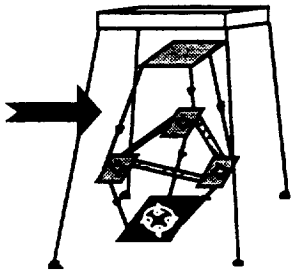
- In Section 4.3 the lexicon is mapped into previous classification systems using *linguistic* entities. The linguistic entities of the lexicon presented are shown to be domain and time independent.
- In Section 4.3 various examples are used to illustrate the effectiveness of the lexicon in complex systems design. These examples include a pressure vessel and a passenger aircraft, both studies used to verify other portions of this work.

Posit 1.2: Game Theory principles can be used to extend problem formulation in multidisciplinary design.

Previous attempts at problem formulation in multidisciplinary design have addressed the existence of multiple levels of designers and disciplines but never addressed anything but cooperative relationships among them. This is point of departure in this work, the extension of problem formulations in multidisciplinary design to design scenarios where full cooperation may not exist. In conversations with industry experts in aircraft design, it is apparent that although researchers in problem formulation and complex systems design assume cooperation in their models, in practice, cooperation is rare. For instance, at the National Aeronautics and Space Administration (NASA) it is common for the aerodynamicists to take the lead in a design process and establish the wing and fuselage profiles first. Then the propulsion, controls, and structural engineers design according to the wing and fuselage design set by the aerodynamics team. In the former Soviet Union, it was a common practice to choose the engines for aircraft first, then design the remaining sections around the engines. These two practices are very different, but would be classified very similarly using previous lexicons in multidisciplinary design. There is a need to account for the fundamental interactions among the subsystems and/or disciplines when complete cooperation is not possible or practical. The work of this posit supports the

capability to account for these types of interactions in a lexicon. In Sections 3.3.3 and 5.5, the linguistic entities of game theory are presented.

3.3 TEST OF HYPOTHESIS II - SUBSYSTEM INTERACTIONS



The fundamental notion of this hypothesis is that a complex design process can be abstracted as a series of games among a set of players (designers and their associated analysis and synthesis tools). The theoretical and intuitive notions to support this Hypothesis are established in this section. Ramifications and verification guidelines are also provided for testing Hypothesis II in this section.

3.3.1 A Typical Complex System Design Model

Typical complex system models are analyzed and solved using one of two strategies: 1) solving the system level problem as a single level problem, or 2) decomposing the problem into smaller problems and solving the smaller problems. In this work, it is assumed that the system level problem is decomposed into smaller, subsystem level problems which must be analyzed and solved accounting for any coupling among the subsystem problems. This is shown in Figure 3.6, where a typical complex system model is decomposed into two subsystem problems. At the system level, there are overall system requirements and system parameters. At the subsystem level, SS1 and SS2 consist of constraints and goals each must *satisfy*, and system and deviation variables each must *find*. Once these variables are found, a system configuration is generated based on the subsystem problem solutions. But this is easier said than done. For one, each subsystem typically needs the values of design variables of the other subsystems. This is illustrated in Figure 3.6. Furthermore, integrating the subsystem solutions into a system level configuration is not a trivial

problem. It must be governed by the coupled variables and a coordination procedure based on the subsystem interactions.

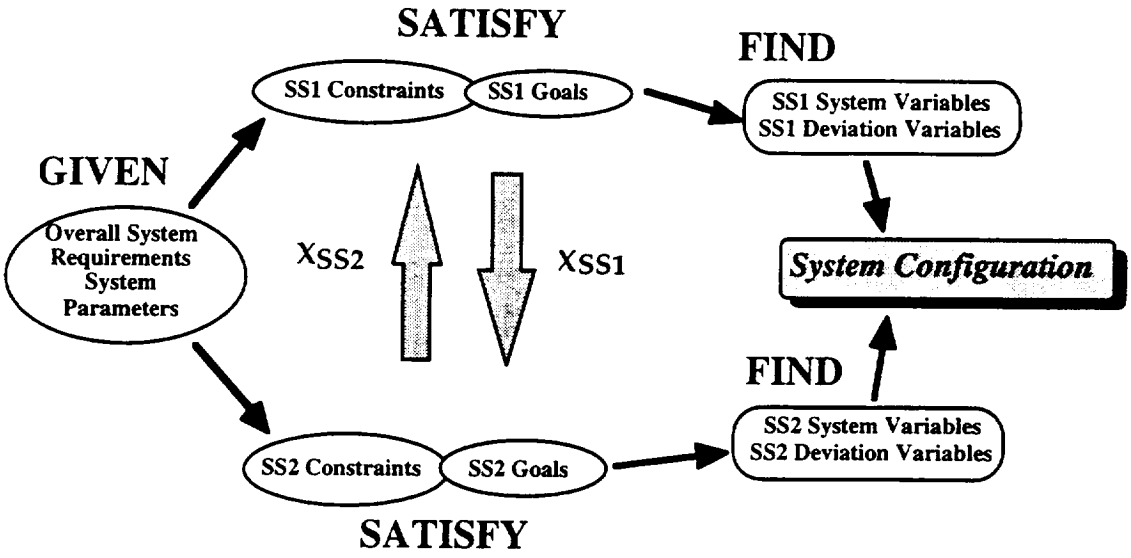


Figure 3.6. A Typical Complex System Model

To illustrate some fundamental integration principles from a mathematical perspective, the following subsystem model (compromise DSP) in Figure 3.7 is presented.

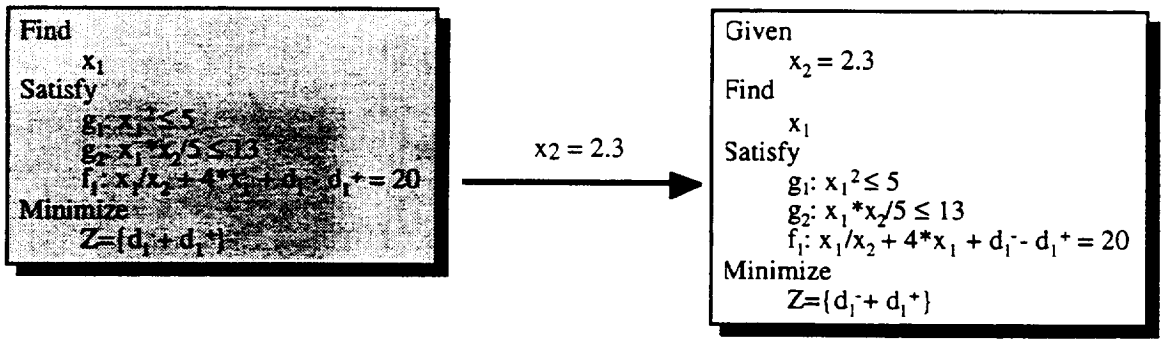


Figure 3.7. A Coupled Compromise DSP

On the left side of Figure 3.7 a very simple compromise DSP representing one subsystem is shown with one local design variable, x_1, two constraints and one goal. The only catch is that x_2, the design variable of another nonlocal subsystem is needed in g_2 and f_1. Without a representation of x_2 the compromise DSP on the left side is *unsolvable*. Typical complex system design practices either require a value of x_2 at each iteration from the other subsystem, or just assume some value. On the right side of Figure 3.7, the value of x_2 = 2.3 is used in the model and is either provided by the other subsystem or taken as an assumption. Either way, the compromise DSP on the right side is now *solvable*. If the actual value is used, it requires extensive information transfer and coordination for the multiple analysis and synthesis iterations in a solution process. However, if assumptions are made, there exists a risk of the assumptions being wrong or infeasible. So the question is posed, *how can the designers of this subsystem know x_2 without making assumptions or requiring large information transfer?* It is asserted that if the designer using the compromise DSP in Figure 3.8 has a representation of x_2 such as $x_2 = f(x_1)$, then the problem is solved. This is illustrated in Figure 3.8.

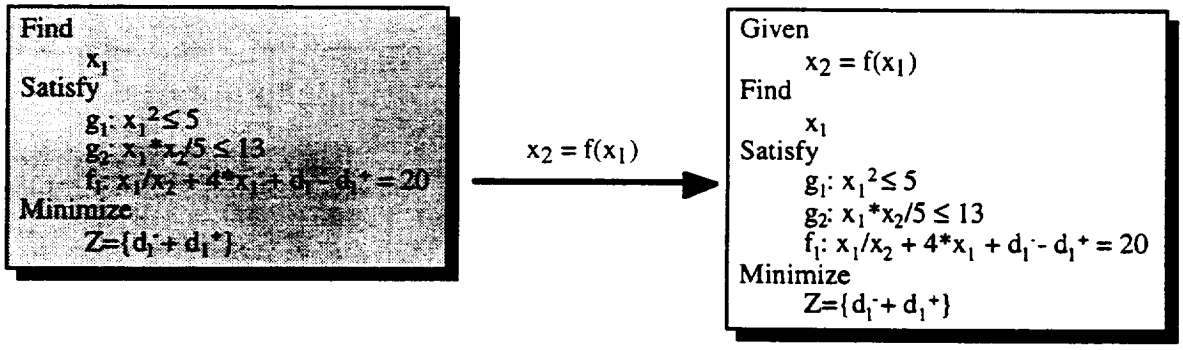


Figure 3.8. A Coupled Compromise DSP: Smarter Coupling

In Figure 3.8, the designer now has a representation of x_2 for *any* value of x_1 which can be used in the compromise DSP to find solutions. This is a major point of contribution of this

dissertation in complex systems design, but it creates a significant mathematical challenge. Constructing a function, $x_2 = f(x_1)$, where the *independent* variable of one subsystem is the *dependent* variable of the other subsystem is not a trivial task when x_2 and x_1 may represent vectors of multiple design variables. Still, this is a simple abstraction of a complex subsystem model. In design problems, there are variables that are not design variables, but describe the behavior of the system. These are called *state* variables, s, or *behavior* variables. Examples may include lift-to-drag ratios, velocities, or stresses. Now consider the compromise DSP of Figure 3.9.

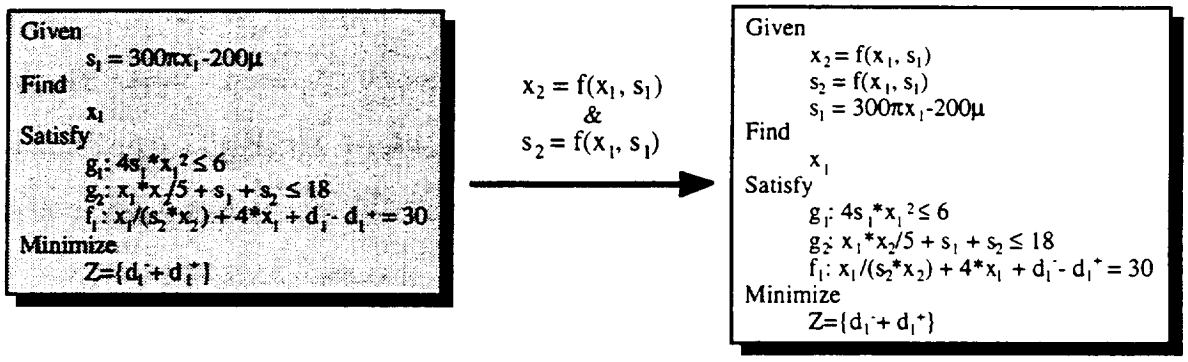


Figure 3.9. A Coupled Compromise DSP: Realistic Coupling

In this case, the designer (player 1) requires both the design variable, x_2, and the state variable, s_2, from the other subsystem (player 2). Therefore, two functions must be constructed,

$$x_2 = f(x_1, s_1) \qquad (3.1)$$

$$s_2 = f(x_1, s_1).$$

This set of equations is in essence the *rational reaction set* of player 2, a fundamental construct in game theory. The rational reaction set is presented and discussed in Sections 3.3.3 and 3.3.4. On the right hand side of Figure 3.9, the designer now has a

representation of x_2 and s_2 for any x_1 and s_1. This is another major point of contribution of this dissertation: the coupling of design and state variables in a game-theoretic context.

In the next section, a typical game is described and the parallel is established between a game among multiple players and a complex design process performed by multiple designers (who use their own analysis and synthesis routines).

3.3.2 A Game as an Abstraction of a Design Process

A "game" consists of multiple decision-makers or players (or designers, in this case) who each control a specified subset of system variables and who each seek to minimize their own cost functions subject to their individual constraints (Myerson, 1991). The game requires these multiple decision makers to select single *decision strategies* to optimize their set of rewards. However, each player's reward depends on the other player's strategies, i.e., a local reward depends on decision variables that are controlled by other players. The fact that players lack control over all decision variables affecting their rewards is what makes a game a game and what distinguishes it from an optimization problem. To illustrate, consider Figure 3.10. When only one decision maker is modeled, the problem is an optimization problem, scalar for single rewards or vector for multiple rewards. However, with more than one decision maker, the problem becomes a game. The focus of this dissertation is on the bottom right-hand corner of Figure 3.10: vector games where there exists more than one decision maker who each have more than one reward. The developments and techniques can be applied, however, to all four quadrants under the appropriate assumptions.

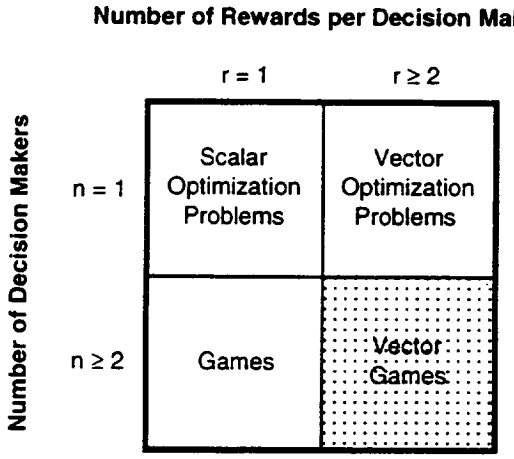


Figure 3.10. Various Formulations in Optimization Theory (Mesterton-Gibbons, 1992)

It is asserted that the processes required to design a complex system can be abstracted as a game. To illustrate, assume that a complex system such as an aircraft has been decomposed into disciplinary subsystems such as propulsion and structures. It is commonly accepted that a model such as

$$\underset{x \in X(p) \subset \Re^n}{\text{minimize}} \; f(x,p) = \{f_1(x,p),\ldots,f_r(x,p)\} \tag{3.2}$$

is the typical starting point for much of the current research and practice in systems modeling and applied optimization. Yet in specific design instances, this assertion should be boldly challenged. For example, when the propulsion designer only controls x and the structures designer controls p, how is p chosen in the propulsion design? Can the propulsion designer assume that the structural designer will always select the vector that is most advantageous to the propulsion design? If not, how should the propulsion designer respond to this conflict? This scenario describes a two-player strategic game where one player controls x and the other player controls p and where p represents all decisions which are outside the scope of the designer controlling x (Aubin, 1979, Dresher, 1981, Von Neumann and Morgenstern, 1944).

Mathematical modeling of such strategic behavior, where one decision-maker's action depends on decisions by others, is well-established in wide-ranging applications from economics to business and military applications (Aubin, 1979, Dresher, 1981, Fudenberg and Tirole, 1991, Mesterton-Gibbons, 1992). If the use of multi-player strategic models in these non-engineering applications is so compelling, it is natural to ask what role do these models have in the design of complex engineering systems? After all, design is often a collaborative activity, with different decision-makers responsible for different subsystems or design stages (e.g., design and manufacturing of a product). But the notion and application of game theory in engineering design is limited (Badhrinath and Rao, 1995, Dhingra and Rao, 1995, Hazelrigg, 1996, Pakala and Rao, 1996, Rao, et al., 1996).

It is asserted that strategic situations in design can be abstracted as *games* among designers or design teams, and depending upon the *protocol* of the game, the resulting designs may be significantly different. There are various game protocols depending on the level of cooperation and behavior of the players. Certain protocols lend themselves nicely to modeling interactions in design, namely the cooperative or Pareto formulation when the players work together and communicate, the Nash or noncooperative formulation when the players act in their own self-interest, and the Stackelberg or leader/follower formulation when one player dominates another. These protocols are illustrated in the next section using simple 2-player terminology from (Vincent and Grantham, 1981).

3.3.3 Protocols Applicable to Design

Let there be two players P1 and P2 who select strategies x^1 and x^2 which belong to strategy sets $X^1(\subset \Re^{n1})$ and $X^2(\subset \Re^{n2})$, respectively. Further, let $f_1(x^1,x^2)$ and $f_2(x^1,x^2)$ be their respective cost or loss functions. The various game-theoretic models between the players

are difficult to solve for two principal reasons: (i) *coupled optimality*, i.e., the cost functions f_1 and f_2 of the two players depend on strategies x^1 and x^2 selected by *both* players, and (ii) *coupled feasibility*, i.e., if $U \subset X^1 \times X^2$ is the set of feasible strategies, then in the presence of constraints, U is not necessarily equal to the product of strategy sets $X^1 \times X^2$. In other words, given $x^1 \in X^1$, x^2 is constrained to the set $S(x^1) = \{x^2 \in X^2 : (x^1,x^2) \in U\}$, and vice-versa. This latter point is subtle; it indicates that one player may affect not just the cost of other player, but that this player may also influence the feasibility of the other player's decisions as indicated in Figure 3.11. In other words, $X^2 = f(X^1)$ in *both feasibility and optimality*.

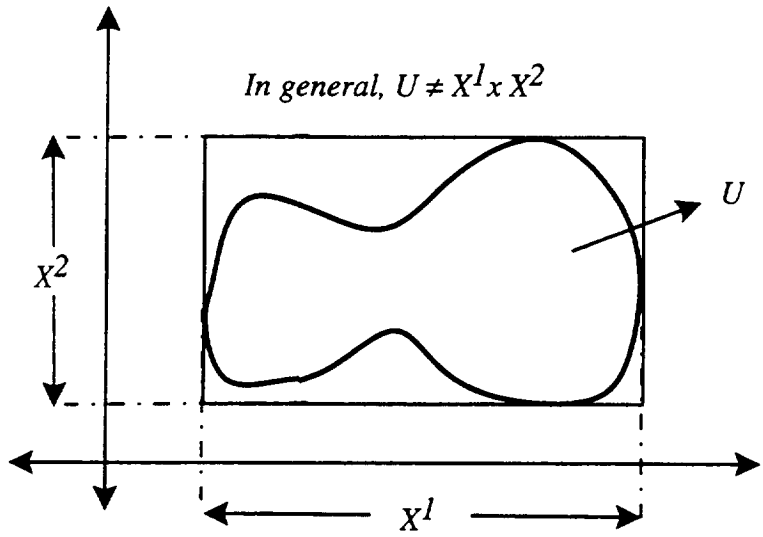


Figure 3.11. Feasible Strategies in the Presence of Constraints

In a given application, once the problem data is established, namely the feasible strategy set $U \subset X^1 \times X^2$ and the respective cost functions, f_1, and f_2, the basic goal is to single out pairs of decisions (i.e., strategies) (x^1,x^2) that correspond to the protocols (relationships) that exist between the two players. For these various protocol models, it is important to examine the stability properties of the solution. A strategy pair (x^{1*},x^{2*}) is individually

stable if neither player has an incentive to unilaterally alter his strategy. Such a pair would be collectively stable or a *Pareto solution* if both the cost functions cannot be simultaneously improved by another strategy pair. The various models and the corresponding solution concepts are classified for the two players according to when the players: (i) cooperate, (ii) act in their own self-interest (noncooperation), or (iii) dominate one another. For simplicity in the presentation, assume that $U \subset X^1 \times X^2$.

Cooperative or Pareto solution: If the players cooperate, they can be expected to obtain better solutions than when they do not. Assuming total cooperation among decision makers, disciplines, or subsystems is the typical optimization approach. Often, it is not unusual to find that by cooperating with one another, both the players can improve on their solution when they do not cooperate. A pair (x^{1p}, x^{2p}) is Pareto optimal if

$$\begin{aligned} &\nexists\, \Delta_1 \ni f_1(x^{1p}+\Delta_1, x^{2p}) < f_1(x^{1p}, x^{2p}) \text{ and } f_2(x^{1p}+\Delta_1, x^{2p}) < f_2(x^{1p}, x^{2p}) \\ &\qquad\qquad \text{and} \\ &\nexists\, \Delta_2 \ni f_1(x^{1p}, x^{2p}+\Delta_2) < f_1(x^{1p}, x^{2p}) \text{ and } f_2(x^{1p}, x^{2p}+\Delta_2) < f_2(x^{1p}, x^{2p}) \end{aligned} \tag{3.3}$$

By definition, the Pareto solutions are collectively stable. However, these solutions need not be individually stable, as each player could do better but at the expense of the other player. The set of Pareto solutions is usually large, thus requiring some additional selection procedures among the Pareto solutions. The Pareto or cooperative solutions occur when players have complete, precise information from the other players. Cooperation is classical game theory. One of the contributions of this dissertation is the notion of *approximate cooperation* where a player does not have complete information from the other players but has approximations of the information needed from the other players.

Nash or noncooperative solution: The Nash or noncooperative formulation occurs when coalition among players is not possible due to organizational, information, or process barriers. This is often the case in designing large systems, where players act independently

and must make assumptions concerning the other players' actions. A strategy pair (x^{1N}, x^{2N}) is a Nash solution if

$$f_1(x^{1N},x^{2N}) = \min_{x^1 \in X^1} f_1(x^1,x^{2N}) \;\&\; f_2(x^{1N},x^{2N}) = \min_{x^2 \in X^2} f_2(x^{1N},x^2) \tag{3.4}$$

This solution has the property of individual stability, but is not necessarily collectively stable. It is also difficult to compute since it is the fixed point of a nonlinear map, namely,

$$(x^{1N},x^{2N}) \in X^{1N}(x^{2N}) \times X^{2N}(x^{1N}) \tag{3.5}$$

where

$$X^{1N}(x^2) := \{x^{1N} \in X^1 : f_1(x^{1N},x^2) = \min_{x^1 \in X^1} f_1(x^1,x^2)\} \tag{3.6}$$

and

$$X^{2N}(x^1) := \{x^{2N} \in X^2 : f_2(x^1,x^{2N}) = \min_{x^2 \in X^2} f_2(x^1,x^2)\} \tag{3.7}$$

are called the *rational reaction sets* of the two players.

Stackelberg solutions: Consider the case when one player dominates another, i.e., the two players have a leader-follower relationship. P1 is a leader if he declares his strategy first by assuming or dictating that the follower P2 behaves rationally. Thus, the model with P1 as leader is

$$\begin{aligned} &\underset{(x1,x2) \in U}{\text{minimize}}\ f_1(x^1,x^2) \\ &\text{satisfying } x^2 \in X^{2N}(x^1) \end{aligned} \tag{3.8}$$

and the model with P2 as leader is

$$\begin{aligned} &\underset{(x1,x2) \in U}{\text{minimize}}\ f_2(x^1,x^2) \\ &\text{satisfying } x^1 \in X^{1N}(x^2) \end{aligned} \tag{3.9}$$

where $X^{1N}(x^2)$ and $X^{2N}(x^1)$ are defined in Eq. (3.6) and Eq. (3.7). For two players, these Stackelberg games are special cases of bilevel models. These models occur in a variety of important applications and have been studied extensively (Azarm and Li, 1987, Azarm and Li, 1995, Falk and Liu, 1993, Loridan and Morgan, 1988, Loridan and Morgan, 1989,

Lucchetti, et al., 1987, Rao and Mistree, 1995, Shimuzu, 1985, Shimuzu and Aiyoshi, 1981, Simaan and Cruz, 1973).

Application of these protocols in the design of complex systems is one of the fundamental motivations of this dissertation. The *mathematics* behind the protocols are borrowed, but implementation of the conceptual constructs in the context of complex system design is part of the novelty of this work (Lewis and Mistree, 1996b). As part of this implementation, various approximation techniques are used to facilitate the application of game theoretical techniques to design. These approximation techniques are discussed in Section 3.3.4.

3.3.4 Approximation Techniques Used in Each Protocol

Approximate Cooperative

The primary approximation technique used in the approximate cooperative formulation is Taylor series. Suppose f(x) is a function of a single variable x and f is differentiable to the nth order on some interval. If x* is a point in that interval, then Taylor's theorem says that the change in f from x* to (x* + ε) is as follows (Reklaitis, et al., 1983):

$$f(x^* + \varepsilon) = f(x^*) + (\varepsilon)\frac{df}{dx}\Big|_{x=x^*} + \frac{(\varepsilon)}{2!}\frac{d^2 f}{dx^2}\Big|_{x=x^*} + \ldots + \frac{(\varepsilon^n)}{n!}\frac{d^n f}{dx^n}\Big|_{x=x^*} + O_{n+1}(\varepsilon) \tag{3.10}$$

where $O_{n+1}(\varepsilon)$ in Eqn. 3.10 indicates terms of (n+1)st order or higher in ε. For multiple variables, the Taylor series takes the form of

$$f(x^* + \varepsilon) = f(x^*) + \nabla f(x^*)(\varepsilon) + (\varepsilon)^T H_f(\varepsilon) + O(\varepsilon). \tag{3.11}$$

For sufficiently small ε, the first order term in Taylor theorem will dominate the others. It is assumed that the first term is sufficient to approximate f. The validity of this assumption is addressed in Section 7.5.1. Taylor series in Eqns. 3.10 and 3.11 is truncated to give

$$f(x^* + \varepsilon) \approx f(x^*) + (\varepsilon)\frac{df}{dx}\Big|_{x=x^*} \text{ or } f(x^* + \varepsilon) \approx f(x^*) + \nabla f(x^*)(\varepsilon) \qquad (3.12)$$

where f is now approximated.

In the context of complex systems design and game theory, different disciplines are coupled through design and state variables. Design variables are the independent variables, so no closed form expression is available. But it is assumed that each discipline has an analysis routine to calculate the local state variables as functions of the design variables. However, nonlocal state variables also affect the local analysis. Therefore, Taylor series are used to approximate the nonlocal state variables, **s**. This is illustrated in Figure 3.12. Each player has his own independent design variables, $\mathbf{x_i}$. There is no formula or equation for these $\mathbf{x_i}$'s. An optimization algorithm, which calls analysis routines to find the values of $\mathbf{s_i}$ (for which equations exist), is used to determine the values of $\mathbf{x_i}$. Approximations of nonlocal state variables (of the other players) are used by each player. It is assumed that actual values of nonlocal $\mathbf{x_i}$'s are used in the approximate cooperative formulation, as no analysis routines exist to simulate and approximate the $\mathbf{x_i}$'s.

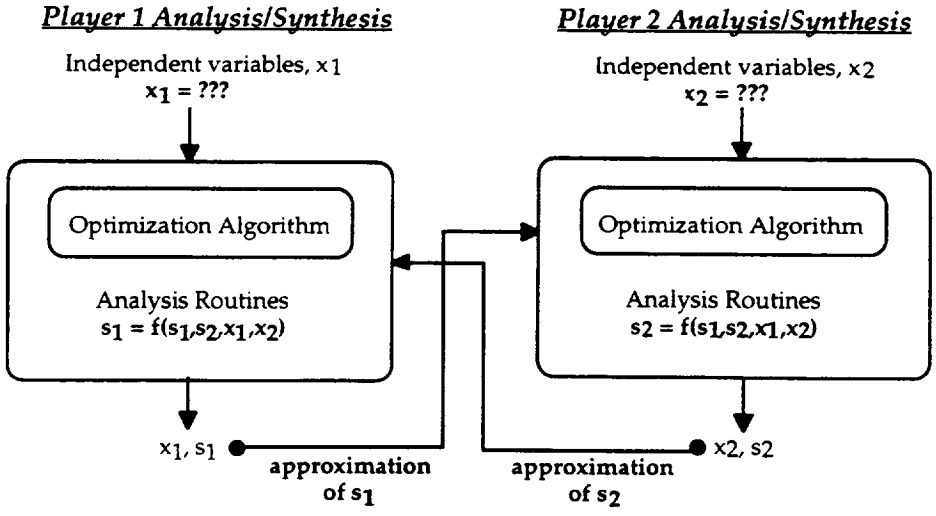


Figure 3.12. Coupling of Behavior Variables

Single and multiple variable representation of the state variables are

$$s(x) \approx s^o + \frac{ds}{dx}\Big|_{x=x^0}(x - x^o), \quad s(x) \approx s^o + \nabla s(x^o)(x - x^o). \tag{3.13}$$

For example, the approximation of a state variable as a function of three design variables in scalar form is

$$s(x_a, x_b, x_c) \approx s^o + \frac{ds}{dx_a^o}(x_a - x_a^o) + \frac{ds}{dx_b^o}(x_b - x_b^o) + \frac{ds}{dx_c^o}(x_c - x_c^o). \tag{3.14}$$

The one major step remaining to complete the Taylor approximation is the determination of the full derivatives of the state variables in Eqn. 3.13, with respect to the design variables, or $\nabla s(x)$. Determining the full derivatives is accomplished using the Global Sensitivity Equations (GSE) method first proposed in (Sobieszczanski-Sobieski, 1988) and successfully used in the design of complex systems (Bloebaum, et al., 1992, Renaud and Gabriele, 1991, Renaud and Gabriele, 1993, Renaud and Gabriele, 1994). Using the GSE method, the total derivatives of the dependent variables can be solved for as functions of the independent variables *from every player.* These derivatives use the local partial derivatives from each player to determine the total derivative. To illustrate, consider a problem with three players, a, b, and c, each with one state variable, s_a, s_b, and s_c, which are functions of the design variables of each player, x_a, x_b, and x_c (Eqn. 3.15). The Global Sensitivity Equations are developed by analyzing the derivative of the functions $\mathbf{s_a}$, $\mathbf{s_b}$, and $\mathbf{s_c}$ (state equations) from the three players with respect to the independent design variables, x_a, x_b, x_c (Eqn. 3.16).

$$\begin{aligned} \mathbf{s_a} &= \mathbf{s_a}\ (\mathbf{x_a}, \mathbf{x_b}, \mathbf{x_c}) \\ \mathbf{s_b} &= \mathbf{s_b}\ (\mathbf{x_a}, \mathbf{x_b}, \mathbf{x_c}) \\ \mathbf{s_c} &= \mathbf{s_c}\ (\mathbf{x_a}, \mathbf{x_b}, \mathbf{x_c}) \end{aligned} \tag{3.15}$$

$$\begin{aligned} \mathbf{ds_a/dx_k} &= \partial s_a/\partial x_k + \partial s_a/\partial s_b(\mathbf{ds_b/dx_k}) + \partial s_a/\partial s_c\,(\mathbf{ds_c/dx_k}) \\ \mathbf{ds_b/dx_k} &= \partial s_b/\partial x_k + \partial s_b/\partial s_a(\mathbf{ds_a/dx_k}) + \partial s_b/\partial s_c\,(\mathbf{ds_c/dx_k}) \\ \mathbf{ds_c/dx_k} &= \partial s_c/\partial x_k + \partial s_c/\partial s_a(\mathbf{ds_a/dx_k}) + \partial s_c/\partial s_b\,(\mathbf{ds_b/dx_k}) \end{aligned} \tag{3.16}$$

Rearranging equation 3.16 into matrix notation produces the following matrix representation of the GSE:

$$\begin{vmatrix} \mathbf{I} & -\partial s_a/\partial s_b & -\partial s_a/\partial s_c \\ -\partial s_b/\partial s_a & \mathbf{I} & -\partial s_b/\partial s_c \\ -\partial s_c/\partial s_b & -\partial s_c/\partial s_b & \mathbf{I} \end{vmatrix} \begin{vmatrix} \mathbf{ds_a/dx_k} \\ \mathbf{ds_b/dx_k} \\ \mathbf{ds_c/dx_k} \end{vmatrix} = \begin{vmatrix} \partial s_a/\partial x_k \\ \partial s_b/\partial x_k \\ \partial s_c/\partial x_k \end{vmatrix} \qquad (3.17)$$

or

$$[M]\,[X] = [B]$$

The partial derivatives in the GSE **[M]** and **[B]** matrices can be found by various methods, and the total derivatives **[X]** can be found using matrix solution techniques. These total derivatives then are used in a Taylor expansion of nonlocal equations, Eqn. 3.13. Therefore, each player uses an approximation of the state equations of the other players. This is the essence of approximate cooperation in design. The implementation of approximate cooperative formulations is illustrated and discussed in Sections 5.5.1 and 5.6.1.

Noncooperative and Leader/Follower

In both the noncooperative and leader/follower formulations, the fundamental construct is the Rational Reaction Set. The Rational Reaction Set, RRS, as defined in Section 3.3.3 and illustrated in Section 3.3.1 quantifies the decision-making strategy of a player in a game. In complex systems design, constructing an exact RRS becomes an insurmountable problem with the existence of multiple variables and multiple nonlinear constraints and goals. Therefore, an approximation of an RRS must be constructed. In this dissertation this approximation is accomplished using the Design of Experiments and Response Surface Methodology.

Design and Analysis of Experiments and Response Surface Methodology

The Design of Experiments is a statistical approach (Box, et al., 1978, Montgomery, 1991) for solving problems that range from engineering to social science. Among various DOE techniques, the Response Surface Methodology is a collection of statistical techniques which support the design of experiments and fitting a response model (Box and Draper, 1987, Khuri and Cornell, 1987). By systematic design and analysis of experiments, a response surface model is used to relate a response (output) variable to the levels of a number of factors or input variables that affect it. In problems using computer simulation tools, performing 'experiments' is equivalent to performing a number of simulations with different input settings. Generally speaking, when fitting the response surface model, the following relationship exists:

$$y = f(\mathbf{x}, \beta) + \text{random error} + \text{bias}, \tag{3.18}$$

where y represents the observed result of the response from the simulation, $\mathbf{x}$ is the vector representing the simulation inputs, and β is the vector representing the coefficients in the regression model. In Eqn. 3.18, the response surface model is represented by $f(\mathbf{x},\beta)$. The *predicted* response is presented by

$$\hat{y} = f(\mathbf{x}, \beta), \tag{3.19}$$

where $\hat{y}$ is the estimated value of the response. It can be noted that the difference between the observed and the estimated values of the response, $y - \hat{y}$, is the random error plus the bias. Random error is defined as

$$\text{random error} = y - E(y), \tag{3.20}$$

where E(y) is the statistical expected value of y. The bias, also called systematic error, is defined as

$$\text{bias} = E(y) - f(\mathbf{x}, \beta). \tag{3.21}$$

In this dissertation, statistical methods are combined with the notions of game theory in the design of complex systems. Simply speaking, what has been proposed is to build approximating functions of responses, i.e., $\hat{y} = f(\mathbf{x}, \beta)$, using the design of experiments (DOE) techniques, specifically the Response Surface Methodology (RSM). But there are primary differences between typical applications of RSM and the implementation proposed in this dissertation (see Section 3.3.4.2). Specifically, since RSM is being used to construct approximations of the decision-making strategy of a player with respect to the control variables of another player, the terms input variables and response must be explicitly defined:

Input variables: For Player I, the input variables are the control variables (design and state) of Player II that are needed (but unknown) by Player I to determine his control variables, and vice versa for Player II.

Response: For Player I, a response is a control variable (design or state) of Player I which is dependent on the input variables of Player II.

In Section 3.3.1, the relationship between the input variables and responses as defined here are illustrated using a simple example. Typical application of RSM will involve a set of input variables (design variables) and a number of responses (state variables, constraints, goals). Therefore, a major point of departure in this dissertation is the application of RSM to non-traditional scenarios encountered in game theory.

There are three main reasons why these statistical methods are used here:

- First, using DOE techniques, it is possible to quantify and study the effect of one player upon another. One player's behavior variables (and design variables, in an optimization context) are functions of not only local variables, but non-local variables under the control of other players. It is important to quantify the non-local effect of other players on the local decisions of each

player. Therefore, the *statistical methods provide an effective way to formalize the interactions among players.*

- Second, the RSM embodies the decision-making strategy of a player regardless of the other players' actions. This embodiment is built by multiple solutions of a player's model (compromise DSP) for different values of the input variables (non-local control variables) in the experimental design. In this way, *the embodiment of each player's decision-making strategy, or the approximations of the Rational Reaction Sets, can be used in solving the game theoretic formulations.*
- Third, in complex systems design, solving a player's model using unknown, symbolic input variables is highly unreasonable due to the existence of multiple design variables, nonlinear constraints, and nonlinear goals. Therefore, using DOE techniques, the unknown input variables can be simulated using numerical values, and the model can be solved for each input setting. By using DOE and RSM techniques to simulate known input variables, approximations of each player as functions of unknown, nonlocal variables can be constructed. Therefore, *constructing a RRS can be facilitated without using symbolic variables,* thus solving a difficult *theoretical and computational* problem of finding an exact RRS using symbolic variables.

It is asserted that the above three considerations address not only the issue of efficiency, but also the issue of effectiveness in designing complex systems using game-theoretical principles. The use of statistical methods makes it possible to approximate a fundamental decision-making construct, the Rational Reaction Set, in game theory.

In Figure 3.13, the general procedure for constructing response surface equations (RSE) is shown. Each step is discussed in the context of its implementation in this dissertation.

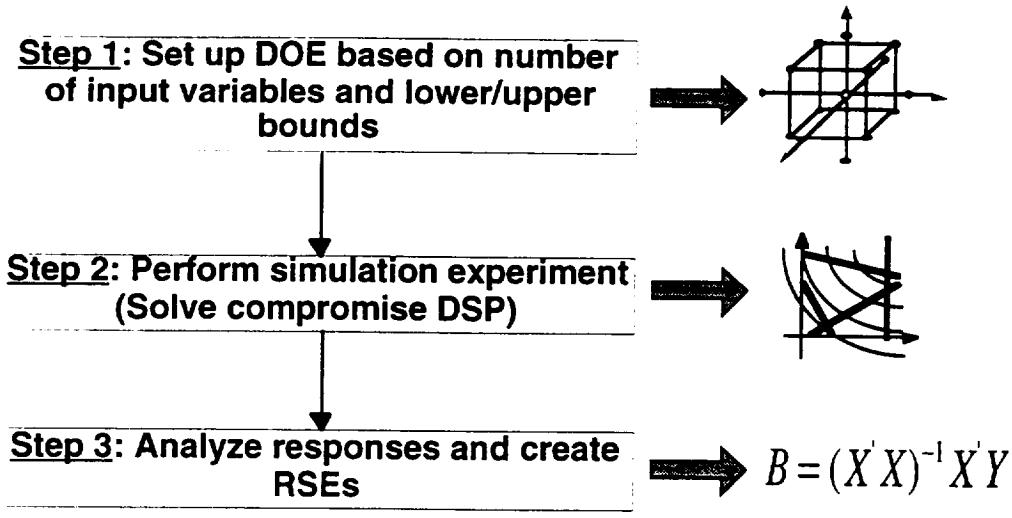


Figure 3.13. Construction of RSE in Game Theory

Step 1: Set up DOE based on number of input variables and their lower/upper bounds

In this dissertation, one specific experimental design is employed. This design is the Central Composite Design (CCD). There are many other experimental designs which can be used such as the full factorial design, fractional factorial design, orthogonal arrays, and Plackett-Burman design. The CCD is used in this work because: 1) it is likely the most effective and widely used experiment for fitting *second-order response surfaces* and studying second-order effects (Montgomery, 1991), and 2) *only second order surfaces* are built in the algorithm of this dissertation. Generally speaking, when picking the experiment, a designer has to consider factors including: 1) number of simulation runs required, 2) ease of implementation, 3) flexibility of the design, 4) recognition of confounding patterns, and 5) ease of analysis. Confounding patterns are dictated by the resolution of the experiment. The definitions of resolution III, IV, and V designs are provided here, as the distinction between experimental designs of different resolution is important.

Resolution III: There are designs in which estimates of main factors are free of confounding with estimates of other main factors, but may be lumped with two-factor interactions. The estimates of two-factor interactions may be lumped with each other.

Resolution IV: There are designs in which estimates of main factors are free of confounding with any other estimates of main factors or two-factor interactions. However, the estimates of two-factor interactions are lumped with each other.

Resolution V: Estimates of main factors and two-factor interactions are free of confounding with any other main factors or two-factor interactions. However, the estimates of two-factor interactions may be lumped with three-factor interactions.

The background for the CCD is given next.

As shown in Figure 3.14, central composite designs are first order fractional factorial designs augmented by additional "star" and center points which allow the estimation of a quadratic surface model of the following form:

$$f(x_1,\ldots x_n) = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n \quad \text{Linear Terms}$$
$$+ \gamma_1 x_1^2 + \ldots + \gamma_n x_n^2 \quad \text{Quadratic Terms} \qquad (3.22)$$
$$+ \beta_{12} x_{1,2} + \ldots + \beta_{n,n-1} x_{n-1,n} \quad \text{Interaction terms}$$

		A	B	C
Factorial portion	1	-1	-1	-1
	2	-1	-1	+1
	3	-1	+1	-1
	4	-1	+1	+1
	5	+1	-1	-1
	6	+1	-1	+1
	7	+1	+1	-1
	8	+1	+1	+1
Star portion	9	$-\alpha$	0	0
	10	$+\alpha$	0	0
	11	0	$-\alpha$	0
	12	0	$+\alpha$	0
	13	0	0	$-\alpha$
Center point	14	0	0	$+\alpha$
	15	0	0	0

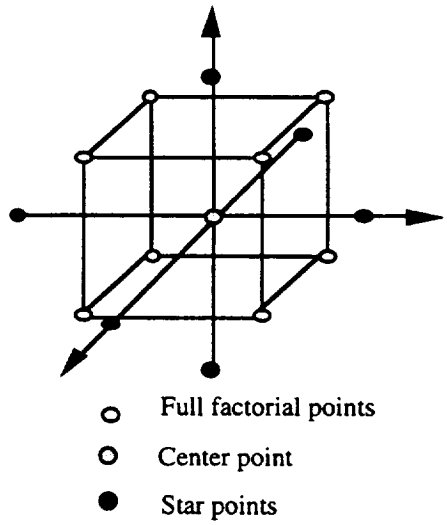


Figure 3.14. Three Variable Central Composite Design

A CCD generally consists of:

1. A complete or fraction of a first-order (2^n) factorial design where parameter levels are coded to the usual -1 and +1 values. This is called *the factorial portion of the design.* For a model as shown in Eqn. 3.22, a resolution IV design is needed to clear the main factors and two-factor interactions with any other main factors or two-factor interactions.
2. Two *"star points"* on the axis of each design variable at a distance α from the center. A central composite design is made rotatable by the choice of α. The value of α for rotatability depends on the number of points in the factorial portion of the design.
3. *Center points.* The number of center points in a physical experiment can be more than one. In this computer experiment, because there is no blocking effect, only one center point is necessary.

The number of experiments needed for fitting a second-order model using a CCD with a Resolution V (estimation of distinct main factor and two-factor interactions are possible) fractional factorial design as the factorial portion is significantly less than that would be required in a three-level full factorial design, as shown in Table 3.3. The benefit of using this technique increases as the number of input factors increases. From the form of Eqn. 3.22, it is noted that a second-order model can be used to study the main effects of a factor (linear terms), non-linear effects (quadratic terms) and the interaction effects (interaction terms).

Table 3.3. A Comparison of Full Factorial 3^n and CCD Design

Factors	Full Factorial 3^n	CCD
3	27	15
4	81	25
5	243	27
6	729	45
7	2,187	79

In this dissertation, an automated statistical software package called NORMAN (Cartuyvels and Dupas, 1993), which is available on the UNIX platform, is applied. The design of experiments supported by this package include Plackett-Burman, 3-level full factorial design, 3 level Box-Behknen design, full factorial 2-level design, fractional factorial 2-level design, central composite design, Latin-Hypercube design, Taguchi OA, user defined experiments, etc.

Step 2: Perform Simulation Experiments

Most of the DOE techniques available in the literature are specifically developed for physical experiments rather than computer simulations. Though most of the technologies for these two types of experiments are similar, the focus and the details are different. Computer experiments differ from physical experiments in that there is no random error. The lack of random (or replication) error leads to important distinctions between computer and physical experiments:

- The adequacy of a response-surface model fitted to the observed data is determined solely by the systematic bias, e.g., the assumed model differs significantly from the exact model.
- Usual measures of uncertainty derived from least-squares residuals have no obvious statistical meaning. Though deterministic measures of uncertainty are available, they may be very difficult to compute.
- Classical notions of experimental unit, blocking, replication, and randomization are irrelevant.

The current methodologies for the design and analysis of physical experiments (Box, et al., 1978, Box and Draper, 1987) are not ideal for complex, deterministic computer models. However, as summarized by Welch and co-authors (Welch, et al., 1990), statistics still plays its role in the following ways:

- The selection of inputs at which to run a computer code is still an experimental design problem.
- Statistical principles and attitudes towards data analysis are helpful however the data are generated.
- There is uncertainty associated with predictions from fitted models, and the quantification of uncertainty is a statistical problem.
- A computer code can be modeled as if it were a realization of a stochastic process.

The concept of computer simulation in this dissertation is different from typical computer simulations (Chen, 1995b, Engelund, et al., 1993, Unal, et al., 1994). Since it is desired to find the value of the responses (say, design and state variables of Player I) as functions of the input variables (say, design and state variables of Player II), the RSE's take the form of

$$x_I = f_1(x_{II}, s_{II})$$
$$s_I = f_2(x_{II}, s_{II}). \tag{3.23}$$

Generating responses for s_I, Eqn. 3.23, is the typical simulation procedure in constructing RSEs. The difficulty occurs as a result of having to find x_I, Eqn. 3.23. Since the x_I's are independent variables, there is **no** explicit function to describe them. They are found by solving a given model. Therefore, a simulation in this dissertation is *the solution of a compromise DSP for one set of input variables.* So, whereas in previous applications of RSM, a simulation consisted of *one* analysis call to a set of equations, in this dissertation a simulation may consist of *multiple* analysis calls during one compromise DSP solution in order to find x_I (and subsequently, s_I as well).

Step 3: Analyze Experiments and Create RSE

In this dissertation, the primary objective of analyzing the results of the designed computer experiments is to: *create a mathematical relationship between the coupled design and state*

variables of multiple players. In other words, the primary objective is to estimate the effect of one player on another in the form of an approximated Rational Reaction Set. Analyzing and creating a response surface model can involve different steps depending on the scope of the study and engineer's preferences (Box and Draper, 1987, Montgomery, 1991). Some common steps are discussed next in the context of constructing a rational reaction set.

Estimate the Significance of Different Factors

In order to be considered as an input in the simulation experiment, a control variable from one player is needed by another player. So, it is already assumed that the control variable is significant. Typically in estimating the significance of different factors, no information is available about the effects of the factors. But with the RRS, it is assumed that all the factors are at least first-order significant. All second order interactions are assumed to be significant as well. That is, no screening experiments are performed to eliminate any meaningless second order interactions (Chen, 1996, Chen, et al., 1994). Screening experiments certainly could be performed, but in this work they are not.

Create a Response Surface Model

To create the relationship between a response and input variable as a response surface model, the most widely used method is the least squares method (Heiberger, 1989). The general least squares problem is to find the coefficients B that minimize the distance between an observed vector Y and linear combination XB of a set of basis vectors X:

$$\min_{B} \; |Y - XB|^2 \tag{3.24}$$

The minimum distance is obtained when XB is the projection of Y into the linear space defined by X:

$$XB = P_X Y = X(X'X)^{-1} X'Y \tag{3.25}$$

Therefore, the solution is

$$B = (X'X)^{-1} X'Y \tag{3.26}$$

where X' is the transpose of the matrix X. From Eqn. 3.26, it is noted that, once the result of Y is available from experiments, the coefficient B in the response surface model can be calculated by matrix operations. In addition to matrix operations, several other algorithms, e.g., Wikinson's Sweep and Beaton's SWP, have been demonstrated to achieve the same purpose (Heiberger, 1989).

Confirmation tests

By using the response surface models to approximate the Rational Reaction Set of a player, it is necessary to confirm the accuracy of surface models. However, confirmation tests can only occur when the model is very simple and an exact RRS can be found. With complex models, finding an exact RRS is virtually impossible. Therefore, in complex systems design there is nothing to compare the approximation with in order to confirm its accuracy. However, in order to confirm the results and validate the approach, a simple example is studied in Section 5.6, where the exact rational reaction sets are known. With complex problems, the accuracy of the approximation is embedded within the solution algorithm, in this case, the FALP Algorithm. In Sections 6.5, the effectiveness of the FALP Algorithm is given, which is a step towards validating the approximated RRS with complex models.

3.3.5 Guidelines for Verifying Hypothesis II: Subsystem Interactions

Hypothesis II: Game theoretic principles can be applied to accurately model and describe the interactions in complex systems design.

Hypothesis II is tested by using five posits. The guidelines and section numbers related to the testing of each posit are given.

Posit 2.1: Design processes can be abstracted as games where the players are multiple designers or design teams and their associated analysis and synthesis tools.

The application of game theory to complex design problems is one of the novel and fundamental contributions of this work. Therefore, testing of this posit requires the mapping of design processes to typical games. Most of the previous research in modeling interactions among designers and their analysis and synthesis tools has assumed cooperation. But, in practice this may not be the case. Consider the two practices of NASA and the Soviet Union's former aircraft design teams described in Section 3.2.3. These practices are quite unlike the concurrent engineering principles that are found in the modern classroom and research annals. These two practices are also drastically different from each other and most likely will result in different aircraft. Which aircraft is "better"? Which design process is "better"? In this posit, insight into possible answers to these types of questions is provided and the capability to descriptively model multidisciplinary design when cooperation may or may not exist is developed. Starting from a Decision-Based Design perspective, the mapping of complex design processes to typical games is accomplished:

- at a general level in Sections 3.3.2 and 3.3.3.
- by presenting specific definitions for the application of game theory in design in Section 5.3.
- and by providing rationale for using the specific game protocols in design situations in Sections 5.4 and 5.5.

Posit 2.2: Approximate cooperation can be modeled using the Global Sensitivity Equations and Taylor series to approximate nonlocal equations.

To verify this posit, two steps are required.

- First the notion of *approximate cooperation* must be defined. In game theory, the majority of game modeling is concerned with full cooperation. However, in complex systems design, analysis and synthesis models are too large to achieve full cooperation. Therefore, the notion of approximate cooperation must be established. This is accomplished in Section 3.3.4.
- Second, the use of the Global Sensitivity Equations and Taylor series and their relation to approximate cooperation is detailed in Section 3.3.4 and Section 5.5.1. This is the typical approach in complex systems modeling -- using an approximation of the required information from the other disciplines. The primary contribution in this posit is the mapping of the GSE approach to approximate cooperation in game theory.

Posit 2.3: First order Taylor series can be used as a good approximation of nonlocal state equations.

To verify this posit, the fidelity of the term "approximate" in approximate cooperation is explored. First order Taylor series are used as the approximation tool, and verification of this approximation is detailed in Section 7.5. The Taylor series approximations must also be differentiable in order to be useful in a gradient based solution scheme. The fundamental contribution of this posit is the integration of previous approaches into the compromise DSP, and mapping of the approach to approximate cooperation. Practically, approximate cooperation is a very inviting concept, as disciplines may not have to cooperate fully with exact representations of nonlocal information, but may be happy

enough with approximations of nonlocal information. In other words, the disciplines are satisficing instead of optimizing.

Posit 2.4: Second order response surfaces can be used to approximate the Rational Reaction Sets of the disciplinary players in a design game.

To verify this posit, the benefits of approximating the RRS of each player as second order equations are illustrated. The benefits of this posit include the capability to quantify the decision-making strategy of each decision maker. Qualitatively, the strategy of a decision maker in the compromise DSP is to "minimize the deviation function." The RRS quantifies this strategy so that the other decision makers can make their decision accordingly.

- In Section 3.3.3, the RRS is defined mathematically.
- In Sections 3.3.4 and 5.5.2, the RRS in the context of design is defined conceptually.
- In complex systems design, finding the exact RRS of a player is virtually impossible because of multiple system variables, and multiple nonlinear constraints and goals. Therefore, in order to quantify the decision-making of each player, the RRS must be approximated. In Section 5.5, the process for constructing these approximations using RSE's is detailed.
- The verification of using these response surface equations as approximations of the rational reaction sets is explored in Section 5.6.

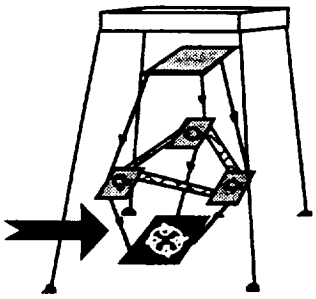
Posit 2.5: The compromise DSP can be used as the fundamental construct to develop the game theory protocols and techniques.

To verify this protocol, the use of the compromise DSP in each game protocol is illustrated. The compromise DSP has been shown to be a fundamental multiobjective mathematical construct (Chen, 1995a, Mistree, et al., 1993a, Mistree, et al., 1994, Vadde, 1995). The

work of this posit further establishes the benefits of using the compromise DSP as the fundamental building block in systems modeling.

- In Sections 5.5.1 and 7.4, the compromise DSP is used as the fundamental construct in modeling full cooperation and approximate cooperation using multiple objectives.
- In Sections 5.5.2 and 7.4, the compromise DSP is used as the mathematical decision-making model to construct the Rational Reaction Sets of each player. Multiple solutions of each player's compromise DSP according to changing input variables allows a Design of Experiments driver to characterize a player's decision-making strategy, which is embodied in a compromise DSP. This is fundamental in both the noncooperative and leader/follower formulations.

3.4 TEST OF HYPOTHESIS III - FORAGING NOTION



In this section, ramifications and verification guidelines are provided for testing Hypothesis III. The notion of design space search in a discrete domain is explored in Section 3.4.1. In Section 3.4.2, the fundamental building block of foraging, the Tabu Search, is introduced. In Section 3.4.3, two other heuristic optimization algorithms, Simulated Annealing (SA) and Genetic Algorithms (GA), are introduced in order to discuss some notions of foraging which are similar to constructs in SA and GA. In Section 3.4.4, the fundamentals of the continuous solver, the ALP Algorithm, are introduced.

3.4.1 Discrete Design Space Search

Methods for the solution of a purely continuous optimization problem are well-established and understood (Reklaitis, et al., 1983). Most are based on calculating a gradient and moving in the appropriate direction in the design space to increase the goodness of the design. Therefore, the design space is searched via a set of directions represented by gradients. If discrete (or integer) variables are present in a design model, gradients of functions *with respect to discrete variables* do not exist. Therefore, the search of a discrete space cannot be accomplished by using derivatives or gradients, and must be based on other heuristic-based methods.

There are many such heuristic methods for solving discrete optimization problems including branch and bound methods, simulated annealing, genetic algorithms, and tabu search. In Section 2.3.4, application of these methods to complex systems design problems are reviewed. However, when discrete *and* continuous variables are present in a design model, purely discrete or purely continuous solvers are not adequate unless some strong simplifications are made in the model. For instance, one could discretize the continuous variables and solve the resulting discrete model using a discrete solver, or one could assume the discrete variables are continuous and then round-off the continuous solution values to the nearest discrete values. These types of methods have been shown to produce sub-optimal solutions in general (Arora and Huang, 1994, Loh and Papalambros, 1991, Papalambros, 1995). Therefore, algorithms must be able to handle and search the discrete *and* continuous design spaces *without* making strong assumptions such as these. In this dissertation, techniques using heuristics and calculus-based methods are combined into one solution algorithm.

The heuristic portion of the algorithm is based on the notion of animals foraging for food in the wild. Certain empirical constructs developed by observing animals foraging for food are similar to characteristics of three previous discrete solution techniques, as shown in Table 3.4. The basic memory structure is built using constructs from the Tabu Search. Identifying portions of solutions that frequently occur in good solutions is similar to identifying schema in Genetic Algorithms. Establishing a dynamic memory is similar to the changing probability distribution used in Simulated Annealing. These three algorithms and their basic solution principles are introduced in the next section.

Table 3.4. Solver Characteristics

Solution Technique	Characteristic
Tabu Search	Memory of visited sites
Genetic Algorithms	Solution schema
Simulated Annealing	Dynamic memory

3.4.2 The Foundation of the Foraging Search: The Tabu Search

In this work, the tabu search is used as the building block for the foraging search. The basis for Tabu Search (TS) is described as follows (Bland and Dawson, 1991, Glover, 1989a, Glover, 1989b). In general terms, TS is an iterative improvement procedure in that it starts from some initial solution and attempts to determine a better solution by applying a greatest-descent procedure. However, TS is characterized by a capability to escape local optima by using short and long term memory of visited solutions. Moreover, TS permits backtracking to previous solutions, which may ultimately lead, via a different direction, to better solutions. The features of a tabu list and aspiration criteria make TS a powerful optimization tool for models characterized by discrete variables (Bland and Dawson, 1991,

Ford and Bloebaum, 1993). Given a set of objectives to be met over a set X, TS proceeds from one point in the design space to another until a chosen termination criterion is satisfied. Since the TS is an unassuming algorithm (it will continue to search without assuming the best solution has been found), the termination criteria usually involve a maximum number of neighborhood searches or time limit. Each $x \in X$ has an associated *neighborhood* $N(x) \subset X$, and each solution $x' \in N(x)$ is reached from x by an operation called a *move*. For discrete variables, the neighborhood is easily defined. TS goes beyond local search by employing a strategy of modifying N(x) as the search progresses, effectively replacing it by another neighborhood N*(x). A key aspect of TS is the use of special memory structures that serve to determine N*(x) and hence to organize the way in which the space is explored. However, researchers using the TS have assumed *constant* memory lists. It is asserted that using constant list lengths limits the search, and by expanding the TS using dynamic memory lists parallels the natural process of foraging more closely and provides a more effective search construct. This assertion is detailed and verified in Sections 6.3 and 6.4, respectively. In addition, the TS does not provide effective decision support information *during* a design process. Therefore, TS is also expanded to provide the designer with effective information concerning the design. This information is based on how animals learn about sites with food during a search. This is also detailed in Section 6.3. Both principles are found in similar forms in two other discrete solvers, Simulated Annealing and Genetic Algorithms. These are the use of dynamic memory in SA and schema identification in GA. In the next section these two algorithms are introduced in order to illustrate these principles.

3.4.3 Simulated Annealing and Genetic Algorithms

Simulated Annealing (SA)

The idea behind SA is to generate random design points and evaluate the goodness of each point (Arora and Huang, 1994). If the trial point is better than the current best value, then the point is accepted. However, SA is also characterized by an ability to escape local minima by accepting points even if they are worse than the best point so far. This acceptance is based on the value of the probability density function:

$$p(\Delta f) = \exp(\frac{-\Delta f}{T_j}) \tag{3.27}$$

where the parameter T_j is the "temperature" at iteration j. This changing temperature is how simulated annealing gets its name, by simulating the process of annealing where the temperature is slowly decreased in order to cool a metal. A high temperature is used initially and slowly decreased as the solution process continues. The new point is accepted if the probability is larger than a random number z, $p(\Delta f) > z$. The acceptance probability steadily decreases to zero as the temperature is reduced. Thus in the initial stages, the method is likely to *accept worse designs*, while in the final stages the worse designs are *almost always rejected.* Eqn. 3.27 is therefore a form of a dynamic memory structure, which parallels the foraging behavior of animals (see Section 6.3). As discussed in Section 2.3.5, SA has been used for mixed discrete/continuous design problems, but it is computationally intensive and not efficient for large problems. Therefore, only one principle of SA is being modeled in this work, the notion of *dynamic memory.*

Genetic Algorithms (GA)

Genetic Algorithms are based on the natural process of genetic reproduction (Arora and Huang, 1994). Their philosophical basis is in Darwin's survival of the fittest theory. A set of design alternatives (represented by binary strings) representing a population in a given generation are allowed to reproduce and cross-pollinate among themselves, with bias

allocated to the most fit members of the population. In a GA, an initial set of designs produces new and better designs using the most fit set members. Three operators are needed to implement GA's: reproduction, crossover, and mutation. Reproduction occurs when an old string is copied into the new population according to the string's fitness. More fit strings, or *schema*, receive higher numbers of offspring, and therefore project their genes (or values of design variables) into more and more populations. Crossover occurs when selected members of the population exchange characteristics among themselves. Mutation occurs when a select few members of the population, determined at random locations on a string, are switched from 0 to 1, or 1 to 0. As discussed in Section 2.3.5, GA's are computationally intensive for large problems. Therefore, only one principle of GA is being modeled in this work, the notion of *schema identification*.

These three algorithms, TS, SA, and GA, are all very useful because they do not require the calculation of gradients, and therefore differentiability requirements of the models can be relaxed. In Section 6.3, a heuristic discrete algorithm is developed based on notions of foraging, integrating constructs from these algorithms. The foraging search is coupled with a continuous, gradient-based solver to solve mixed discrete/continuous problems (Lewis and Mistree, 1996a). In the next section, the background for the continuous solver, the ALP Algorithm is given.

3.4.4 The ALP Algorithm and the Compromise DSP

The ALP Algorithm as introduced in Section 1.2.2, is the solution algorithm for continuous compromise DSPs. In Section 1.2.1, a conceptual overview of the compromise DSP is given. In this section, a mathematical overview is given. The compromise DSP is a multiobjective decision model which is a hybrid formulation (Mistree, et al., 1993a), incorporating concepts from both traditional Mathematical Programming (Winston, 1995)

and Goal Programming (Ignizio, 1983). The compromise DSP is used to determine the values of design variables to satisfy a set of constraints and to achieve as closely as possible a set of conflicting goals. The compromise DSP is used to model such decisions since it is *capable of handling constraints, goals, and multiple objectives* (Mistree, et al., 1994). In particular, the compromise DSP offers the following capabilities:

- accurately represent single-objective or multi-objectives,
- use either preemptive or Archimedean formulation to prioritize objectives,
- have hard constraints or soft constraints (goals),
- quickly generate results for several different weighting schemes,
- handle discrete or continuous variables.

The system descriptors, namely, system and deviation variables, system constraints, system goals, bounds and the deviation function are described in detail elsewhere (Mistree, et al., 1993a) and are therefore is not repeated here.

The mathematical form of the compromise DSP is summarized in Figure 3.15. In the compromise DSP, each goal, A_i, has two associated deviation variables d_i^- and d_i^+, which indicate the extent of the deviation from the target. The deviation variables, d_i^+ and d_i^-, are both positive, and the product, $d_i^+ \cdot d_i^- = 0$, ensures that at least one of the deviation variables for a particular goal is always zero. If the problem is solved using a vertex solution scheme (as in the ALP algorithm (Mistree, et al., 1993a)), then this condition is automatically satisfied.

Given

An alternative to be improved through modification.
Assumptions used to model the domain of interest.
The system parameters:

n	number of continuous system variables
l	number of discrete/integer system variables
p+q	number of system constraints
p	equality constraints
q	inequality constraints
m	number of system goals

$g_i(\underline{X})$ system constraint functions

$$g_i(\underline{X}) = C_i(\underline{X}) - D_i(\underline{X})$$

$f_k(d_i)$ function of deviation variables to be minimized at priority level k for the preemptive case.

Find

$X_i \quad i = 1, \ldots, n+l$

$d_i^-, d_i^+ \quad i = 1, \ldots, m$

Satisfy

System constraints (linear, nonlinear)

$g_i(\underline{X}) = 0; \quad i = 1, \ldots, p$

$g_i(\underline{X}) \geq 0; \quad i = p+1, \ldots, p+q$

System goals (linear, nonlinear)

$A_i(\underline{X}) + d_i^- - d_i^+ = G_i; \quad i = 1, \ldots, m$

Bounds

$X_i^{min} \leq X_i \leq X_i^{max}; \quad i = 1, \ldots, n$

$d_i^-, d_i^+ \geq 0; \quad i = 1, \ldots, m$

$(d_i^- \cdot d_i^+ = 0; \quad i = 1, \ldots, m)$

Minimize

Archimedean or preemptive deviation function (lexicographic minimum)

$$Z = [f_1(d_i^-, d_i^+), \ldots, f_k(d_i^-, d_i^+)]$$

Figure 3.15. Mathematical Form of a Compromise DSP

Three important features contribute to the success of the ALP algorithm, namely,

- the use of second-order terms in linearization,
- the normalization of the constraints and goals and their transformation into generally well-behaved convex functions in the region of interest,
- an "intelligent" constraint suppression and accumulation scheme.

These features are described in detail in (Mistree, et al., 1993a) and briefly described in the following paragraphs.

First and second order algorithms need the derivatives (with respect to the design variables) of the constraints and goals in addition to the values of these quantities. The ALP algorithm is a modified second order algorithm (only the diagonal second order terms are used). This is one of the principal deviations from other SLP algorithms that were developed based on the well-known work of Stewart and Griffith (Stewart and Griffith, 1961). This is the first principal feature of the algorithm. The derivatives are determined numerically using the central difference formula. After solving the linear problem, this solution can be used to improve the second order approximation using the ALP algorithm. A block diagram of the implementation of the ALP algorithm is shown in Figure 3.16.

A user specifies the input to the software implementation of the algorithm in the form of a DSP template. This template consists of data and user provided FORTRAN routines. The data is used to define the problem size, the names of the variables and constraints, the bounds on the variables, the linear constraints, and the convergence criteria. The FORTRAN routines are used to evaluate the nonlinear constraints and goals, to input data required for the constraint evaluation routines and the design-analysis routines, and to output results in a format desired by the user. Access is provided to a design-analysis program library from the analysis/synthesis cycle and also within the synthesis cycle. In the design of major systems it is desirable to use the design-analysis interface associated with the analysis/synthesis cycles (e.g., structural design requiring the use of a finite element program).

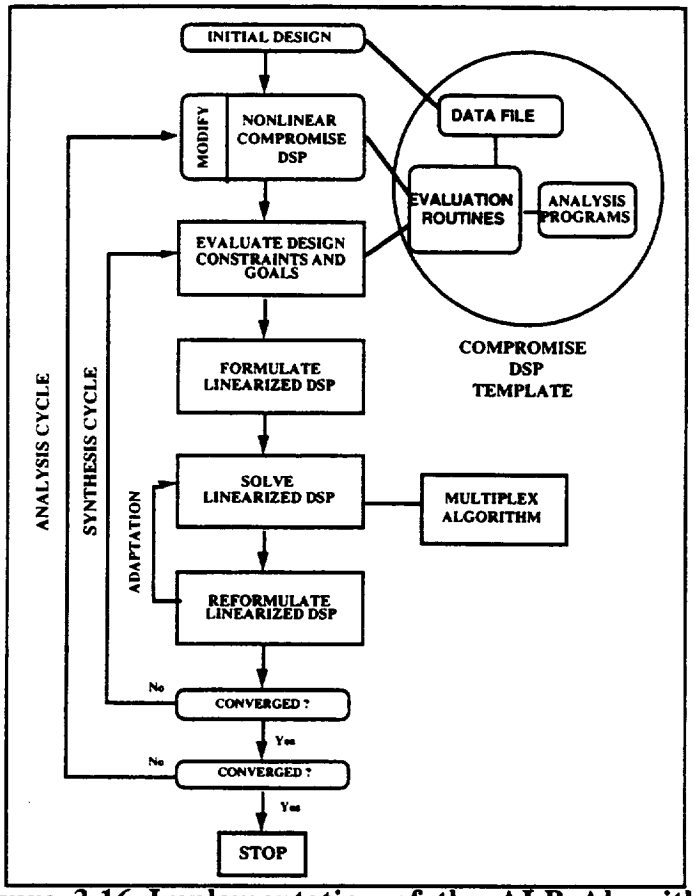


Figure 3.16 Implementation of the ALP Algorithm For Solving Compromise DSPs (Mistree, et al., 1993a)

Once the nonlinear compromise DSP is formulated, it is approximated by linearization. At each stage the solution of the linear programming problem is obtained by a Multiplex algorithm based on (Ignizio, 1985b). The choice among these algorithms depends on the form of the deviation function, which is the measure of how well the system goals are met. The deviation function that is given in the mathematical form of the template can be implemented in two ways:

1. In the Preemptive form the deviation function is given as a lexicographic minimum of the goal deviation variables (Ignizio, 1985a). Usually, goals are not equally important. To determine a solution on the basis of

preference, the goals may be rank-ordered into priority levels. For example, customers rate certain product qualities higher than other qualities.

2. In an Archimedean form the deviation function is given as a weighted function of the goal deviation variables. This reduces the formulation of the template to a traditional single objective problem. This formulation is used when exact quantitative relationships among the goals are known.

The integration of the ALP Algorithm with the foraging heuristic is detailed in Section 6.3. In the next section, guidelines for verifying Hypothesis III are discussed.

3.4.5 Guidelines for Verifying Hypothesis III

Hypothesis III: The notion of foraging of wild animals is a natural analogy for optimization and can be used as an effective search technique in the solution of mixed discrete/continuous models.

Hypothesis III is tested by using three posits. The guidelines and section numbers related to the testing of each posit are given.

Posit 3.1: Notions of foraging can be modeled to create a heuristic, under which characteristics from Genetic Algorithms, Tabu Search, and Simulated Annealing can be grouped.

In Section 6.2, foraging as a natural process and an analogy to optimization is introduced. Certain empirical observations of various animal species foraging for food are quantified in Section 6.3. It is shown that these observations are similar to processes in other heuristic algorithms. Subsequently, it is asserted that foraging, or the notion of animals looking for

the most food in the smallest amount of time, is the *natural* equivalent of *artificial* optimization processes.

The foraging heuristic is a computer-based tool that designers can use to make decisions in a design process. Furthermore, foraging itself is based on an everyday decision-making process in nature. This creates an interesting closed loop of an abstraction of a natural decision making process which provides support to designers who are making decisions.

Posit 3.2: The Tabu Search can be used as the building block for the foraging solution algorithm.

It is well-known that animals use their memory to recall sites which have already been visited. This is precisely the premise under which the Tabu Search was developed. Therefore, in Section 6.3, it is shown how the Tabu Search is used as the fundamental building block of the foraging search.

Posit 3.3: The ALP Algorithm along with foraging can be used to effectively solve mixed discrete/continuous problems.

To verify this posit, the integration of ALP and foraging is presented in Section 6.3 and the effectiveness of the resulting algorithm to solve mixed discrete/continuous design problems is presented. The effectiveness of this posit is verified in Section 6.4. The developments to support this posit allow designers to solve mixed discrete/continuous design problems using constructs from discrete solvers and continuous solvers. Therefore, designers (or computer tools) do not have to approximate discrete variables as being continuous or continuous variables as being discrete. Instead, the *actual* design problem can be modeled and solved.

- In Section 6.3, the integration of the two constructs is detailed. The resulting algorithm is called Foraging-directed Adaptive Linear Programming (FALP).
- In Sections 6.4, verification of FALP is presented for two well-studied examples: the design of a coil compression spring and the design of a cylindrical pressure vessel. It is shown that FALP produces significantly better solutions than previous published algorithms.
- In Section 7.5, FALP is used to solve the disciplinary players' compromise DSPs in certain protocols in the study of a passenger aircraft. Whereas the studies in Sections 6.5 are single level problems with a single objective, the aircraft study consists of two players each with discrete and continuous variables, and each with multiple nonlinear goals.

3.5 TEST OF HYPOTHESIS IV - CONVEXITY

In this section, some ramifications and verification guidelines are provided for testing Hypothesis IV. It is asserted in (Mistree, et al., 1993a), that a transformation function, the g-function, is an effective way to transform nonconvex functions into well-behaved convex functions in the ALP Algorithm. In this section, the basic definitions of convexity in optimization theory are given, as well as a proof of the transformation of functions using this g-function.

3.5.1 Handling Convexity

One of the most difficult issues in nonlinear optimization is handling nonconvex constraints. Previous attempts to handle nonconvex functions have developed special algorithms to handle nonconvex functions or have established conditions under which

certain classes of functions can be transformed into convex functions (Feng, et al., 1990, Floudas and Visweswaran, 1990, Styblinski and Tang, 1990, Thach and Konno, 1993, Vaidyanathan and El-Halwagi, 1994, Ye, 1992). Most of classic optimization theory is based on the assumption that the constraints are each convex. However, in complex systems design, constraints are usually highly nonlinear and are neither convex or concave. Many derivative-based solution algorithms have difficulty solving problems when constraints are not convex. The continuous portion of FALP, the Adaptive Linear Programming Algorithm is derivative-based (Mistree, et al., 1993a). To overcome the mathematical hurdles present when nonconvex constraints are present, a "g-function" (see Section 1.2.2) transformation is proposed in (Mistree, et al., 1993a). In this section, this g-function is investigated. It is proven that the g-function does not retain the convexity of a constraint in regions where the constraint is convex, and *theoretically does not transform* nonconvex constraints into well-behaved convex functions.

The g-function transformation is a one-to-one mapping of the original constraint into a new function over the same domain. Consider a general constraint of the form:

$$g(x): C(x) \geq D. \tag{3.28}$$

where C(x) is the constraint (capability) equation, x is the design variable vector, and D is the constraint limit (demand). In a standard compromise DSP, this constraint is then normalized as,

$$g(x): \frac{C(x)}{D} - 1 \geq 0 \tag{3.29}$$

$$r = \frac{C(x)}{D} \tag{3.30}$$

$$g(x) = r - 1 \geq 0. \tag{3.31}$$

The g-function is then divided by the term (r+1), which will always be positive. The g-function representation of a constraint becomes:

$$g(x) = \frac{r-1}{r+1} \geq 0. \tag{3.32}$$

The convexity of this g-function (Eqn. 3.32) is investigated in Sections 3.5.2 and 3.5.3.

In general, a function f is convex if the Hessian matrix of f is positive definite or positive semidefinite for all values of x_1, ...,x_n (Reklaitis, et al., 1983). On the other hand, a function f is concave if the Hessian matrix of f is negative definite or negative semidefinite for all values of x_1, ...,x_n. The Hessian matrix of a function f(x_1, ...,x_n) is a nxn symmetric matrix given by

$$H_f(x_1,...,x_n) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right] = \nabla^2 f. \tag{3.33}$$

For H to be positive definite or positive semidefinite, the following conditions must be met:

1. All diagonal elements are positive.
2. The leading principal determinants are positive.

In the ALP Algorithm, a heuristic simplification is used (Mistree, et al., 1993a) to determine the convexity of a function. *Only the principal diagonals* are used in determining the convexity of a given function. This simplification is a relaxation of the convexity restriction, as a function may be considered convex from the main diagonal terms, but in reality may not be convex due to the principal determinants. In the following proof, the g-function, *as implemented in the ALP Algorithm* is investigated.

In the following sections, the convexity of the g-function (Eqn. 3.32) with respect to the design variables (Eqn. 3.33) is investigated. It is proven by induction that the g-function in the ALP Algorithm is not a good transformation of nonconvex functions into well-behaved convex functions. The proof begins in Section 3.5.2 with the case when the g-function is a

function of only one system variable (n=1). In Section 3.5.3, the case when the g-function is a function of k-1 system variables (n=k-1) is considered. The proof is concluded by induction for the case when n=k in Section 3.5.3.

3.5.2 The Single Variable Case (n=1)

Assume that the vector **x** contains only one design variable. The convexity of this function is investigated by taking first and second derivatives with respect to a representative design variable, x.

$$\frac{dg(x)}{dx} = \frac{(r+1)\frac{dr}{dx} - (r-1)\frac{dr}{dx}}{(r+1)^2} = \frac{2\frac{dr}{dx}}{(r+1)^2} \tag{3.34}$$

$$\frac{d^2g(x)}{dx^2} = \frac{2\frac{d^2r}{dx^2}(r+1)^2 - 4(\frac{dr}{dx})^2(r+1)}{(r+1)^4} \tag{3.35}$$

In order for g to be convex, the second derivative (Hessian with only one element) must be non-negative,

$$\frac{d^2g(x)}{dx^2} \geq 0. \tag{3.36}$$

This occurs when,

$$\frac{2\frac{d^2r}{dx^2}(r+1)^2 - 4(\frac{dr}{dx})^2(r+1)}{(r+1)^4} \geq 0$$

$$2\frac{d^2r}{dx^2}(r+1)^2 \geq 4(\frac{dr}{dx})^2(r+1)$$

$$\frac{d^2r}{dx^2}(r+1) \geq 2(\frac{dr}{dx})^2. \tag{3.37}$$

Investigating Eqn. 3.37, it is obvious that the right hand side will always be non-negative. The term (r+1) on the left hand side will always be positive since the constraint involves positive quantities and limits. Therefore, the left hand side must be a larger non-negative number for the g-function to be convex. Consider two cases: 1) the original constraint is convex, and 2) the original constraint is concave.

Case 1

If the constraint is convex, then $\frac{d^2r}{dx^2} \geq 0$, and the g-function becomes convex when

$$\frac{d^2r}{dx^2}(r+1) \geq 2\left(\frac{dr}{dx}\right)^2 \tag{3.38}$$

and concave when

$$0 \leq \frac{d^2r}{dx^2}(r+1) \leq 2\left(\frac{dr}{dx}\right)^2. \tag{3.39}$$

This is a strong result, as the g-function still may be convex, but may not be *everywhere* convex, as the original function is.

Case 2

If the constraint is concave then

$$\frac{d^2r}{dx^2} \leq 0 \tag{3.40}$$

and the g-function is also everywhere concave, as the left hand side of Eqn. 3.37 is ≤ 0, and the right hand side of Eqn. 3.37 is ≥ 0. Therefore, the g-function does not transform the original constraint into a convex function.

3.5.3 The Multiple Variable Case

<u>n=2</u>

The vector **x** consists of two design variables. With multiple variables, the convexity of a constraint is determined by the Hessian, ∇r^2. The Hessian of the original constraint is

$$\nabla r^2 = \begin{bmatrix} \frac{\partial^2 r}{\partial x_1^2} & \frac{\partial^2 r}{\partial x_1 \partial x_2} \\ \frac{\partial^2 r}{\partial x_2 \partial x_1} & \frac{\partial^2 r}{\partial x_2^2} \end{bmatrix}. \tag{3.41}$$

Again, two cases are investigated: case 1) the original constraint is convex, and 2) the original constraint is concave.

Case 1

If the original constraint is convex, then ∇r^2 is positive definite or positive semidefinite for all x, and

$$\frac{\partial^2 r}{\partial x_1^2} \geq 0, \text{ and } \frac{\partial^2 r}{\partial x_1^2}\frac{\partial^2 r}{\partial x_2^2} - \left(\frac{\partial^2 r}{\partial x_2 \partial x_1}\right)^2 \geq 0. \tag{3.42}$$

The Hessian of the g-function, ∇g^2 is,

$$\nabla g^2 = \begin{bmatrix} \frac{2\frac{\partial^2 r}{\partial x_1^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_1})^2(r+1)}{(r+1)^4} & \frac{2\frac{\partial^2 r}{\partial x_1 \partial x_2}(r+1)^2 - 4(\frac{\partial r}{\partial x_2})(\frac{\partial r}{\partial x_1})(r+1)}{(r+1)^4} \\ \frac{2\frac{\partial^2 r}{\partial x_2 \partial x_1}(r+1)^2 - 4(\frac{\partial r}{\partial x_1})(\frac{\partial r}{\partial x_2})(r+1)}{(r+1)^4} & \frac{2\frac{\partial^2 r}{\partial x_2^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_2})^2(r+1)}{(r+1)^4} \end{bmatrix} \quad (3.43)$$

For the g-function to be convex, ∇g^2 (Eqn. 3.43) must be positive definite or positive semidefinite, so the two sub-determinants must be

$$\frac{2\frac{\partial^2 r}{\partial x_1^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_1})^2(r+1)}{(r+1)^4} \geq 0, \text{ and}$$

$$\frac{4\frac{\partial^2 r}{\partial x_1^2}\frac{\partial^2 r}{\partial x_2^2}(r+1) - 8(\frac{\partial r}{\partial x_1})^2\frac{\partial^2 r}{\partial x_2^2} - 8(\frac{\partial r}{\partial x_2})^2\frac{\partial^2 r}{\partial x_1^2} - 4(\frac{\partial^2 r}{\partial x_1 \partial x_2})^2(r+1) - 16(\frac{\partial r}{\partial x_2})(\frac{\partial r}{\partial x_1})\frac{\partial^2 r}{\partial x_1 \partial x_2}}{(r+1)^5} \geq 0 \quad (3.44)$$

In the ALP Algorithm, only the main diagonal terms are used to determine the convexity of the function, therefore, only

$$\frac{2\frac{\partial^2 r}{\partial x_1^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_1})^2(r+1)}{(r+1)^4} \geq 0, \text{ and } \frac{2\frac{\partial^2 r}{\partial x_2^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_2})^2(r+1)}{(r+1)^4} \geq 0 \quad (3.45)$$

are necessary. As in the single variable case, for Eqn. 3.45 to be satisfied,

$$\frac{\partial^2 r}{\partial x_1^2}(r+1) \geq 2(\frac{\partial r}{\partial x_1})^2, \text{ and } \frac{\partial^2 r}{\partial x_2^2}(r+1) \geq 2(\frac{\partial r}{\partial x_2})^2. \quad (3.46)$$

Unlike the original constraint which is everywhere convex, these conditions may not be satisfied everywhere. Therefore, the g-function transformation is not useful for the multi-variable case as well.

Case 2

If the original constraint is concave, then ∇r^2 is negative definite or negative semidefinite, and

$$\frac{\partial^2 r}{\partial x_1^2} \leq 0, \text{ and } \frac{\partial^2 r}{\partial x_1^2}\frac{\partial^2 r}{\partial x_2^2} - (\frac{\partial^2 r}{\partial x_2 \partial x_1})^2 \leq 0. \quad (3.47)$$

For the g-function to be concave, ∇g^2 must be negative definite or negative semidefinite, so the two sub-determinants must be

$$\frac{2\frac{\partial^2 r}{\partial x_1^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_1})^2(r+1)}{(r+1)^4} \leq 0, \text{ and}$$

$$\frac{4\frac{\partial^2 r}{\partial x_1^2}\frac{\partial^2 r}{\partial x_2^2}(r+1)-8(\frac{\partial r}{\partial x_1})^2\frac{\partial^2 r}{\partial x_2^2}-8(\frac{\partial r}{\partial x_2})^2\frac{\partial^2 r}{\partial x_1^2}-4(\frac{\partial^2 r}{\partial x_1 \partial x_2})^2(r+1)-16(\frac{\partial r}{\partial x_2})(\frac{\partial r}{\partial x_1})\frac{\partial^2 r}{\partial x_1 \partial x_2}}{(r+1)^5} \le 0 \quad (3.48)$$

In the ALP Algorithm, only the main diagonal terms are used to determine the convexity of the function, therefore, only

$$\frac{2\frac{\partial^2 r}{\partial x_1^2}(r+1)^2-4(\frac{\partial r}{\partial x_1})^2(r+1)}{(r+1)^4} \le 0, \text{ and } \frac{2\frac{\partial^2 r}{\partial x_2^2}(r+1)^2-4(\frac{\partial r}{\partial x_2})^2(r+1)}{(r+1)^4} \le 0. \quad (3.49)$$

As in the single variable case, for Eqn. 3.49 to be satisfied,

$$\frac{\partial^2 r}{\partial x_1^2}(r+1) \le 2(\frac{\partial r}{\partial x_1})^2, \text{ and } \frac{\partial^2 r}{\partial x_2^2}(r+1) \le 2(\frac{\partial r}{\partial x_2})^2. \quad (3.50)$$

If the original constraint is concave then $\frac{\partial^2 r}{\partial x_1^2} \le 0$ and $\frac{\partial^2 r}{\partial x_2^2} \le 0$ (again, only the diagonal terms of the Hessian are used), and the g-function is also everywhere concave, as the left hand side is ≤ 0, and the right hand side is ≥ 0 for both equations in Eqn. 3.50. Therefore, the g-function does not transform the original constraint into a convex function.

n=(k-1)

The vector **x** consists of multiple design variables. With multiple variables, the convexity of a constraint is determined by the Hessian, ∇r^2. The Hessian of the original constraint is

$$\nabla r^2 = \begin{bmatrix} \frac{\partial^2 r}{\partial x_1^2} & \cdots & \frac{\partial^2 r}{\partial x_1 \partial x_{k-1}} \\ \vdots & & \vdots \\ \frac{\partial^2 r}{\partial x_{k-1} \partial x_1} & \cdots & \frac{\partial^2 r}{\partial x_{k-1}^2} \end{bmatrix}. \quad (3.51)$$

Again, two cases are investigated: case 1) the original constraint is convex, and 2) the original constraint is concave.

Case 1

If the original constraint is convex, then ∇r^2 is positive definite or positive semidefinite for all x, and

$\frac{\partial^2 r}{\partial x_1^2} \ge 0$, $\quad \frac{\partial^2 r}{\partial x_1^2}\frac{\partial^2 r}{\partial x_2^2}-(\frac{\partial^2 r}{\partial x_2 \partial x_1})^2 \ge 0$, ..., each sub-determinant up to (k-1) ≥ 0.

The Hessian of the g-function, ∇g^2 is,

$$\nabla g^2 = \begin{bmatrix} \dfrac{2\frac{\partial^2 r}{\partial x_1^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_1})^2(r+1)}{(r+1)^4} & \cdots & \dfrac{2\frac{\partial^2 r}{\partial x_1 \partial x_{k-1}}(r+1)^2 - 4(\frac{\partial r}{\partial x_{k-1}})(\frac{\partial r}{\partial x_1})(r+1)}{(r+1)^4} \\ \vdots & & \vdots \\ \dfrac{2\frac{\partial^2 r}{\partial x_{k-1} \partial x_1}(r+1)^2 - 4(\frac{\partial r}{\partial x_1})(\frac{\partial r}{\partial x_{k-1}})(r+1)}{(r+1)^4} & \cdots & \dfrac{2\frac{\partial^2 r}{\partial x_{k-1}^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_{k-1}})^2(r+1)}{(r+1)^4} \end{bmatrix} \tag{3.52}$$

For the g-function to be convex, ∇g^2 must be positive definite or positive semidefinite, and taking the main diagonals,

$$\frac{2\frac{\partial^2 r}{\partial x_1^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_1})^2(r+1)}{(r+1)^4} \geq 0 \text{ , ..., } \frac{2\frac{\partial^2 r}{\partial x_{k-1}^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_{k-1}})^2(r+1)}{(r+1)^4} \geq 0. \tag{3.53}$$

each must be greater than or equal to zero. As in the single variable case, for Eqn 3.53 to be satisfied,

$$\frac{\partial^2 r}{\partial x_1^2}(r+1) \geq 2(\frac{\partial r}{\partial x_1})^2, \text{ ..., } \frac{\partial^2 r}{\partial x_{k-1}^2}(r+1) \geq 2(\frac{\partial r}{\partial x_{k-1}})^2. \tag{3.54}$$

Unlike the original constraint which is everywhere convex, these conditions may not be satisfied everywhere. In essence, it becomes increasingly difficult for the convexity conditions to be satisfied with more design variables.

Case 2

If the original constraint is concave, then ∇r^2 is negative definite or negative semidefinite, and

$$\frac{\partial^2 r}{\partial x_1^2} \leq 0, \qquad \frac{\partial^2 r}{\partial x_1^2}\frac{\partial^2 r}{\partial x_2^2} - (\frac{\partial^2 r}{\partial x_2 \partial x_1})^2 \leq 0, \text{...., each sub-determinant up to (k-1)} \leq 0$$

For the g-function to be concave, ∇g^2 must be negative definite or negative semidefinite, and

$$\frac{2\frac{\partial^2 r}{\partial x_1^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_1})^2(r+1)}{(r+1)^4} \leq 0, \text{ ..., } \frac{2\frac{\partial^2 r}{\partial x_{k-1}^2}(r+1)^2 - 4(\frac{\partial r}{\partial x_{k-1}})^2(r+1)}{(r+1)^4} \leq 0. \tag{3.55}$$

As in the single variable case, for Eqn. 3.55 to be satisfied,

$$\frac{\partial^2 r}{\partial x_1^2}(r+1) \le 2(\frac{\partial r}{\partial x_1})^2, \ \ldots, \ \frac{\partial^2 r}{\partial x_{k-1}^2}(r+1) \le 2(\frac{\partial r}{\partial x_{k-1}})^2. \tag{3.56}$$

If the original constraint is concave then $\frac{\partial^2 r}{\partial x_1^2}$, ..., $\frac{\partial^2 r}{\partial x_{k-1}^2} \le 0$ (again, only the diagonal terms of the Hessian are used), and the g-function is also everywhere concave, as the left hand side is ≤ 0, and the right hand side is ≥ 0 for both equations in Eqn. 3.56. Therefore, the g-function does not transform the original constraint into a convex function.

In conclusion, it is shown that the g-function does not transform concave functions into well-behaved convex functions in the ALP Algorithm. In addition, the g-function does not guarantee full retaining of the convexity of convex functions. These proofs are shown for the cases of number of design variables = 1, 2, and k-1. Therefore, by induction, it can be shown that it is true for the case of number of design variables = k. As previously mentioned, in the preceding proof, it is assumed that only the principal diagonals are used to determine the convexity of the function. This relaxation results in a *stronger proof*, as even with the relaxed conditions, the g-function does not transform nonconvex functions into well-behaved convex functions in the ALP Algorithm.

The g-function is currently a *recommended option* in the ALP Algorithm. Therefore, in order to implement the g-function, the user would have to hard-code the transformation of the constraints and goals *themselves*. That is, the g-function is not part of the source code of the ALP Algorithm. Therefore, when the ALP Algorithm is used in Chapters 5, 6, and 7 to solve design models, the g-function is not used. It is noted that although it has been shown for general functions across $x \in \Re$, this is not to say that the g-function may work well in certain cases numerically in a small neighborhood, $x \in \aleph$. Since the ALP Algorithm operates on small intervals, it is left to future work to investigate the merit of the

g-function from an empirical and numerical standpoint. In addition, functions which are neither concave or convex have not been investigated.

3.5.4 Guidelines for Verifying Hypothesis IV: Convexity

Hypothesis IV: The g-function is a useful transformation of nonconvex functions into well-behaved convex functions.

Hypothesis IV is tested and rejected using one posit. The guidelines and section numbers related to the testing of this posit are given.

Posit 4.1: Nonlinear optimization theory can be used to prove/disprove the effectiveness of the g-function in transforming nonconvex constraints and goals in the compromise DSP to convex equations.

Hypothesis IV is verified by testing Posit 4.1 which supports the use of Hypothesis IV. This is a very straightforward posit, as a formal proof is constructed. In Sections 3.5.2 and 3.5.3, it is proven that the g-function theoretically does not transform nonconvex functions into well-behaved convex equations. This proof is constructed across the entire analytical independent variable range. The proof presented in Section 3.5.3 does not discount that in certain circumstances, in a small neighborhood around a given design point, the g-function *numerically* may be used to construct well-behaved convex functions based on the step size. This is precisely how the ALP Algorithm operates, therefore, further *numerical* investigation of the g-function and improvement of the transformation function is warranted. However, the theoretical foundation of the g-function has been investigated.

3.6 A SUMMARY OF THE VERIFICATION AND MOTIVATING STUDIES

In Section 1.3, the strategy for implementing and verifying the method and approach in this dissertation is provided. Two primary verification problems, the design of a pressure vessel, and the design of a compression spring are first used to explain and verify the research hypotheses associated with the developments of this work in Chapters 4, 5, and 6. Two pressure vessel problems are studied, one with multiple players to verify Hypothesis II in Chapter 5, and one with discrete and continuous variables to verify Hypothesis III in Chapter 6. In Chapter 4, variations on the pressure vessel and aircraft design problems are included to verify Hypothesis I. Having tested the hypotheses, the second part of the verification strategy is the further development and verification using a motivating study, the design of a Boeing passenger aircraft. The aircraft design problem is presented in Chapter 7.

These studies have been chosen based on the motivating research issues identified in Chapter 2. A summary of the representative features of each study is given in Table 3.5, including the type of analysis, nonlinearity, number of goals, type of decision variables, overall complexity, type of confirmation tests, and relevant hypotheses.

The complexity increases in Table 3.5 from the first pressure vessel problem to the motivating Boeing case study. Each problem is marked by nonlinearity. In the first pressure vessel problem the level nonlinearity is low (quadratic or cubic equations), while the aircraft problem is highly nonlinear. The compression spring and second pressure vessel problem are single-objective for comparison and illustration purposes, while the other studies are multi-objective. The compression spring and second pressure vessel

problem consist of both discrete and continuous decision variables to verify Hypothesis III, while the first pressure vessel problem is purely continuous to verify Hypotheses I and II. The motivating case study is mixed discrete/continuous. Different types of confirmation tests are used depending upon the availability of previous studies, analytical equations, and numbers of variables

Table 3.5. Features of Example Problems and Motivating Study

	Pressure Vessel I	Compression Spring	Pressure Vessel II	Boeing Aircraft
Type of Analysis	structural, economic	structural	structural, economic	aerodynamics, weights, propulsion, economic
Nonlinearity of Function	nonlinear	nonlinear	nonlinear	highly nonlinear
Goals	multiple	single	single	multiple
Decision Variables	continuous	discrete and continuous	discrete and continuous	discrete and continuous
Overall Complexity	very low	low	low	high
Type of Confirmation Tests	3-D plots, analytical comparison	exhaustive search	comparison with previous studies	analytical
Hypothesis Used to Test	I, II	III	III	I, II, III

3.7 A LOOK BACK AND A LOOK AHEAD

In this chapter, the algorithm for integrated subsystem embodiment and system synthesis and the research hypotheses associated with the development of the algorithm are presented. For each hypothesis, ramifications, relevant theoretical information, and verification guidelines are provided as a means to establish the foundations for the research approach. The three steps of the algorithm, corresponding the first three hypotheses are

explored in Sections 3.2, 3.3, and 3.4, respectively. The final hypothesis is disproved in Section 3.5. In Figure 3.17, the progress of the dissertation is shown. Building upon the theoretical foundation and literature review of Chapters 1 and 2, in Chapter 3 the overall algorithm is introduced in Chapter 3. Chapter 3 begins phase II of the verification strategy of this dissertation. In Chapters 4, 5, and 6, phase II is continued as the three specific steps and hypotheses associated with the algorithm are explored in more detailed and verified using the case studies introduced in Section 3.6.

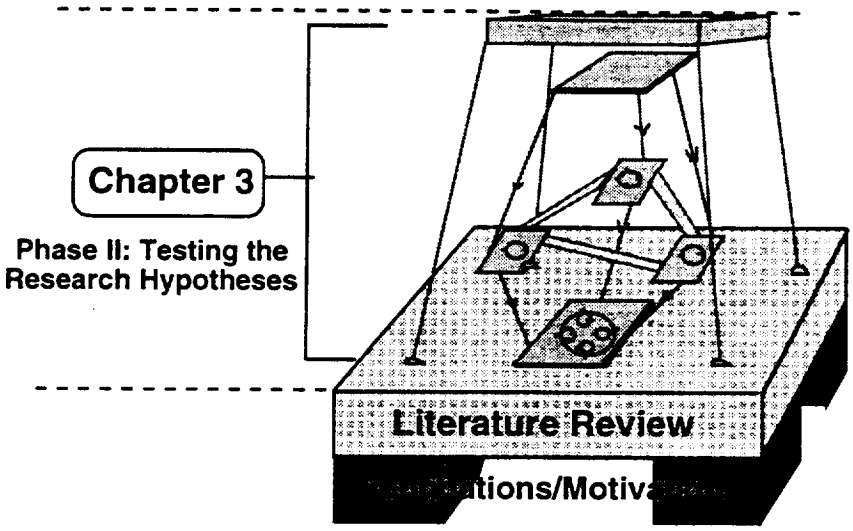
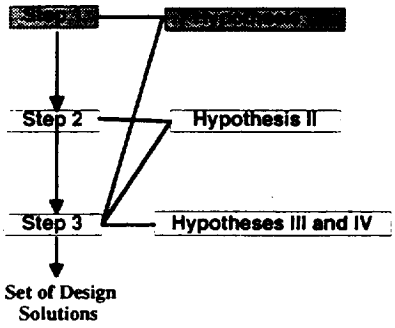


Figure 3.17. Frame of Reference: Chapter 3

CHAPTER 4

CLASSIFICATION AND FORMULATION OF MULTIDISCIPLINARY DESIGN PROBLEMS: A DECISION-BASED PERSPECTIVE

A multidisciplinary design process consists of multiple designers or design teams, each with a specified domain of expertise. This expertise is domain-dependent but design is an interactive, integrated transformation of information. Therefore, the designers and design teams must be able to communicate with the other teams at various levels of detail, from integrated cross-disciplinary teams to integrated computer infrastructures of "talking" software. Many times, however, complete communication and cooperation are not possible at either level. Because of the complexity of multidisciplinary design problems, there is a need to "step back" and design the design *process*. In other words, a meta-design phase is needed to define and structure linguistically the design process and product (Mistree, et al., 1990b, Mistree, et al., 1993c). There are many approaches to formulating and solving complex design problems including various single and multi-level approaches. There is, however, no common set of linguistic entities to compare and map these approaches to each other. In this chapter, multidisciplinary design optimization (MDO) is approached from a game-theoretic, Decision-Based Design (DBD) perspective and classification schemes for multidisciplinary design problems are explored. The exploration and developments in this chapter provide support for the first step of the algorithm (Figure 1.7), Hypothesis I, and Posits 1.1 and 1.2, presented in Section 3.1.



NOMENCLATURE

From (Balling and Sobieski, 1994):

SAND - Simultaneous Analysis and Design

NAND - Nested Analysis and Design

i: denotes discipline number i

s_i: disciplinary *state variables* which comprise the *state equations*

r_i: *residuals* in the state equations

y_{ij}: *coupling functions*, contains those function computed in discipline i which are needed in discipline j

y_{ij}^*: *coupling variables*

x: *system design variables* needed by more than one discipline

x_i: *disciplinary design variables*

g_i: *design constraint functions*

f_i: *design objective functions*

c_i: *cumulative design function* determined by a system analyzer

d_i: *discrepancy functions*

disciplines: subsystems described by a common underlying physical principle

Disciplinary *analyzers* seek values for the state variables that reduce the residuals in the state equations to zero. That is, analyzers try establish to equilibrium conditions by changing the state variables.

Disciplinary *evaluators* find the residuals in the state equations for given values of the state variables. That is, evaluators are usually sets of equations that only evaluate the value of the equations for a set of constant state variables.

From (Mistree, et al., 1990b):

selection is the process of making a choice between a number of possibilities taking into account a number of measures of merit or attributes.

compromise is the process of determining the "right" values (or combination) of design variables, such that, the system being designed is feasible with respect to constraints and system performance is maximized with respect to multiple, possibly conflicting goals.

heuristic decision (Kamal, 1990) is, roughly speaking, a combination of a preliminary selection and compromise decision. The solution process for a heuristic decision differs from the compromise and selection DSPs and involves reasoning.

LI: *Lateral interactions* between subsystems

FVI: *Forward vertical interactions* among parent system and subsystems

RVI: *Reverse vertical interactions* among parent system and subsystems

d_i^+, d_i^-: *deviation variables*, measures difference between goal i target values and actual achievement.

Z_i: deviation function of model i

state variables: dependent variables which describe the behavior of a system.

state equations: equations which are functions of the state and design variables and describe the behavior of a system.

4.1 TECHNOLOGY BASE: CONCEPTUAL CONSTRUCTS

In this chapter, Hypothesis I, which corresponds to the first step of the algorithm presented in Section 3.1, is explored. This hypothesis is:

> *Hypothesis I:* Classification of problem and process in multidisciplinary design can be facilitated by integrating constructs from Decision-Based Design, Game Theory, and Multidiscplinary Design Optimization.

Posits 1.1 and 1.2, which support this hypothesis are as follows.

> ***Hypothesis I Posits***
>
> *Posit 1.1:* Entities from the Decision Support Problem Technique provide a domain-independent lexicon for multidisciplinary design.
>
> *Posit 1.2:* Game Theory principles can be used to extend problem formulation in multidisciplinary design.

Explanation and description of each posit is provided in the context of the formulation and solution of complex design problems characterized by multiple disciplines. Several approaches to formulating and solving a multidisciplinary design problem have arisen in a rather ad hoc fashion over the years. These approaches include single-level and multi-level formulations, hierarchical and nonhierarchical system decomposition methods, and numerous optimization and analysis processes and approaches at the system and subsystem levels. In Balling (Balling and Sobieski, 1994, Cramer, et al., 1994), a classification system for formulation of MDO problems is presented. In this chapter,

- this classification system is explored and extended from a game-theoretical, decision-based perspective,
- a framework is provided within which research activities and open questions in the field of MDO can be articulated in the future,

- the *linguistic* entities most often used by designers and researchers in MDO to describe both the system *and* the process are identified,
- the first step of the algorithm of this dissertation presented in Chapter 3 is embodied with the classification framework.

Specifically, the focus of the classification presented in this chapter is on the types of decisions made by designers and how they affect the decisions of the other designers. Entities from the DSP Technique are integrated with the Balling-Sobieski (B-S) framework (Balling and Sobieski, 1994, Cramer, et al., 1994) and domain-independent linguistic terms to build the taxonomy (Lewis and Mistree, 1995).

In this chapter, it is shown that the Balling-Sobieski framework is consistent with that of the Decision Support Problem Technique through the use of *linguistic* entities describing the same type of formulations. It is shown that the underlying linguistics of the solution approaches are the same and can be coalesced into a homogeneous framework with which to base the research, application, and technology of MDO upon. Identifying linguistic entities is only the first step in designing complex systems. These terms must be embodied on a computer according to a parsing and translation scheme. Identification of these terms facilitates the development of a complete system and process taxonomy for MDO.

In Sections 4.1.1 and 4.1.2, the foundational principles of this chapter are presented. These include the background of the B-S scheme and the Decision Support Problem Technique (DSP Technique). The game theory foundation is presented in Sections 1.2.3 and 3.3.3. In Section 4.3, these concepts are integrated and their continuity illustrated in complex systems design. The DSP Technique approach is mapped into the B-S scheme and the synergy between the two approaches is illustrated. This chapter is closed with some assertions concerning the implementation and application of the taxonomy in complex

system design. The mindset of this chapter is one of description as opposed to prescription. The lexicon is presented as a means to describe the necessary product and process issues that designers, researchers, and engineers must handle in complex systems design as opposed to prescribing a specific method that designers must follow.

4.1.1 The Balling-Sobieski Scheme

A brief overview of the classification to MDO problems presented in (Balling and Sobieski, 1994) is given in this section. The classification scheme is rooted in the following assumptions:

- that complex systems consists of distinct disciplinary subsystems which may or may not overlap,
- and that these systems can be represented by a mathematical model.

The validity of these assumptions is returned to in Section 4.3. As a frame of reference, in Figure 4.1 a generic representation of a coupled, three-discipline system is shown.

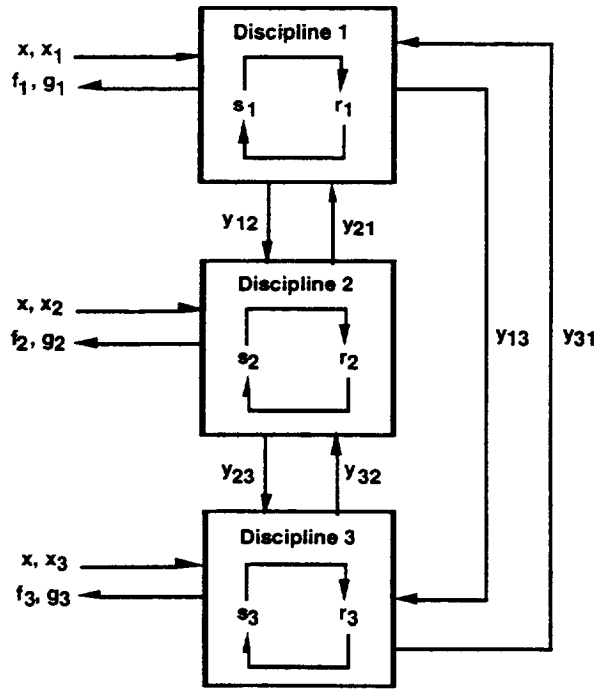


Figure 4.1. A Three-Discipline Coupled System (Balling and Sobieski, 1994)

Depending on the level of analysis, the modules in Figure 4.1 may refer to disciplines, components, or processes. This figure is representative of a typical three-discipline coupled system. It is the decomposition of the system, subsystem coupling and solution, and system synthesis that pose major research and application problems in MDO. The terms used in the figure, as well as other common terms are defined below (see also Nomenclature) and taken from (Balling and Sobieski, 1994).

s_1, s_2, s_3: disciplinary *state variables* which comprise the *state equations*

r_1, r_2, r_3: *residuals* in the state equations

y_{12}, y_{13}, y_{21}, y_{23}, y_{31}, y_{32}: *coupling functions*, y_{ij} contains those function computed in discipline i which are needed in discipline j.

y_{12}^*, y_{13}^*, y_{21}^*, y_{23}^*, y_{31}^*, y_{32}^*: *coupling variables*

x: *system design variables* needed by more than one discipline

x_1, x_2, x_3: *disciplinary design variables*

g_1, g_2, g_3: *design constraint functions*

f_1, f_2, f_3: *design objective functions*

Disciplinary *analyzers* seek values for the state variables that reduce the residuals in the state equations to zero. That is, analyzers try to establish equilibrium conditions by changing the state variables.

Disciplinary *evaluators* find the residuals in the state equations for given values of the state variables. That is, evaluators are usually sets of equations that only evaluate the values of the equations for a set of constant state variables.

The primary task at hand is summarized as follows:

Determine the values of the design, state, and coupling variables that satisfy the state equations, the coupling equalities, the design constraints, and the design objective functions.

Based on this, six classifications for fundamental approaches to MDO problem formulation and solution are presented by Balling and Sobieski, which depend on three criteria:

1) System vs. Multilevel decomposition
2) Simultaneous (SAND) vs. Nested Analysis and Design (NAND) at the *system* level
3) Simultaneous (SAND) vs. Nested Analysis and Design (NAND) at the *subsystem or discipline* level

At the discipline level, SAND implies that the disciplinary design and state variables are determined simultaneously by the optimizer, while NAND implies that the optimizer determines only the disciplinary design variables and requires determination of the state variables at each iteration. Thus, at each iteration of the optimizer, disciplinary evaluators are called for SAND while disciplinary analyzers are called for NAND. At the system level, SAND implies that the system design variables and coupling variables are determined simultaneously by the system optimizer, while NAND implies that the system optimizer determines only the system design variables and requires calls to a system analyzer to determine the coupling variables at each iteration. The "optimizers" at the system level or discipline level could be gradient based or heuristic in nature, depending on the problem formulation. Further classifications can be generated if these approaches are combined or linked sequentially within one design problem.

Each approach has a three-part name consisting of the overall decomposition descriptor, the solution approach at the system level, and the solution approach at the subsystem level. The first part indicates whether the approach is a single-level or multi-level approach. The middle and last parts of the name indicate whether the SAND or NAND approach is used at the system and discipline levels, respectively. The B-S scheme has inherently assumed cooperation is the only possible form of communication among design teams. In reality this is not the case (See Sections 1.1.2 and 3.3.2). Therefore, the B-S scheme is extended using game-theoretic entities. In (Rao and Mistree, 1995), SAND and NAND bilevel

models are explored using game theory formulations and constructs. This work extends the integration of game theory into MDO. In the B-S scheme, if a single-level approach is used, the game is cooperative, as the disciplinary problems are combined into one single-level problem. Therefore, a set of Pareto solutions are ideally available. If a multi-level approach is used, then the disciplines must be designated a status in the design game. The possible designations are:

- cooperative: each disciplinary design team cooperates and has a representation of the other teams' information. If the representation is exact, then it is full cooperation. If it is approximate, then it is an approximate cooperation scenario,
- noncooperative: each disciplinary design team has to make assumptions about the other teams,
- leader: a discipline either decides first or dominates a process, assuming the followers behave rationally,
- follower: a discipline either waits on another discipline or is dominated by another one.

The theoretical and mathematical descriptions of each designation in the context of game theory are given in Section 3.3.3.

4.1.2 A Decision-Based Perspective

Decision-**B**ased **D**esign (DBD) is offered as a starting point for the creation of design methods that are based on the notion that the principal role of an engineer, in the design of a product or process, is to make decisions. An introduction to DBD is presented in Section 1.2.1. Independently of the approaches or methods used to plan, establish goals and model systems, designers are, and will continue to be involved in two primary activities, namely, *processing symbols* and *making decisions*. Therefore, it is asserted that the process of design, in its most basic sense, is a series of decisions. By focusing upon

decisions, a description of the processes is available which is written in a common "language" for teams from the various disciplines -- a language that can be used in the process of designing.

It is recognized that the implementation of DBD can take many forms; the implementation used is the **D**ecision **S**upport **P**roblem (DSP) Technique. It is being developed and implemented to provide support for human judgment in designing systems that can be manufactured and maintained. It was indicated that this approach to engineering design is embodied in the DSP Technique and the principal support for human designers is provided through the formulation and solution of **D**ecision **S**upport **P**roblems (DSPs). The software to solve DSPs on the computer is called DSIDES (Decision Support in the Design of Engineering Systems) (Mistree, et al., 1993a). Details about the mathematical structure of the DSPs are presented in (Mistree, et al., 1993a, Mistree, et al., 1993c). Entities from the computer implementation of the DSP Technique are used to model processes (Bras and Mistree, 1991, Mistree, et al., 1990b). These entities, called Support Problems, are shown in Figure 4.2. It is these entities that are integrated with the linguistic entities of the B-S scheme from Section 4.1.1 in the classification system of this dissertation.

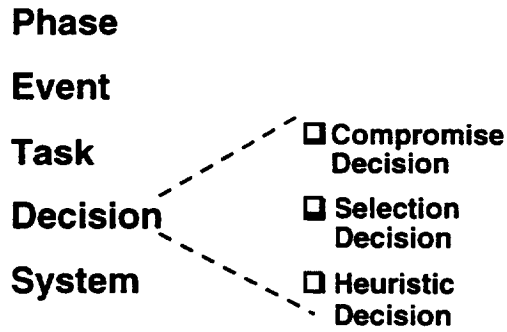


Figure 4.2. Potential Support Problem Entities

The *phase* entity is used to represent pieces of a partitioned process. *Events* occur within a phase. *Tasks* and *decisions* are used to model phases and events. Tasks and decisions require direct involvement of human designers and/or systems. Phases and events are accomplished by performing tasks and making decisions. A task is an activity to be accomplished. The design process itself is a task for the design team, namely, "design a suitable product". A task itself may contain other tasks and decisions, even phases and events, as in the design task. However, simple tasks like "run computer program A" do not involve decisions.

In this chapter, the focus is on *decisions*, which are only a small portion of the DSP Technique, but the primary notion in DBD. More specifically, the focus is on *coupled* DSPs. By focusing on coupled DSPs, they can be mapped into the B-S scheme as coupled approaches to MDO problems. Examples of coupled DSPs include coupled selection-compromise and compromise-compromise formulations. The solution algorithm for continuous Decision Support Problems is the Adaptive Linear Programming (ALP) Algorithm (Mistree, et al., 1993a) (see Sections 1.2.2 and 3.4.4). Decision Support Problems and the ALP Algorithm are based on the notion of *satisficing* solutions, or solutions that are "good enough", as opposed to *optimizing* solutions (Simon, 1982). In the B-S scheme, *optimizers* are used extensively in the classification. However, throughout this chapter, the ALP Algorithm is referred to as a *solver* instead of an *optimizer*. In Section 4.2, a classification scheme is presented which consists of terms from both the B-S scheme, game theory, and the DSP Technique. Domain-independent terms inherent in complex systems design which embody certain open research areas are also integrated.

4.2 A DECISION-BASED CLASSIFICATION

Scientific lexicons, or classification structures generally consist of a number of levels of identification. For example, consider the field of biology. Any living entity can be classified according to the accepted framework in biology. This framework begins at the *kingdom* level and continues to the *species* level, getting more specific with the lower levels. The levels of the taxonomy presented in this section also correspond to a given level of detail, but in addition to the system, classification of the process is included as well. Each level classifies a portion of the design process and product. The taxonomy is rooted in the notion of Integrated Product and Process Design (IPPD) where issues concerning the design product *and* the process to required to reach the final product are simultaneously addressed. The taxonomy proposed has three levels as shown in Figure 4.3. Each level is explained in the following sections.

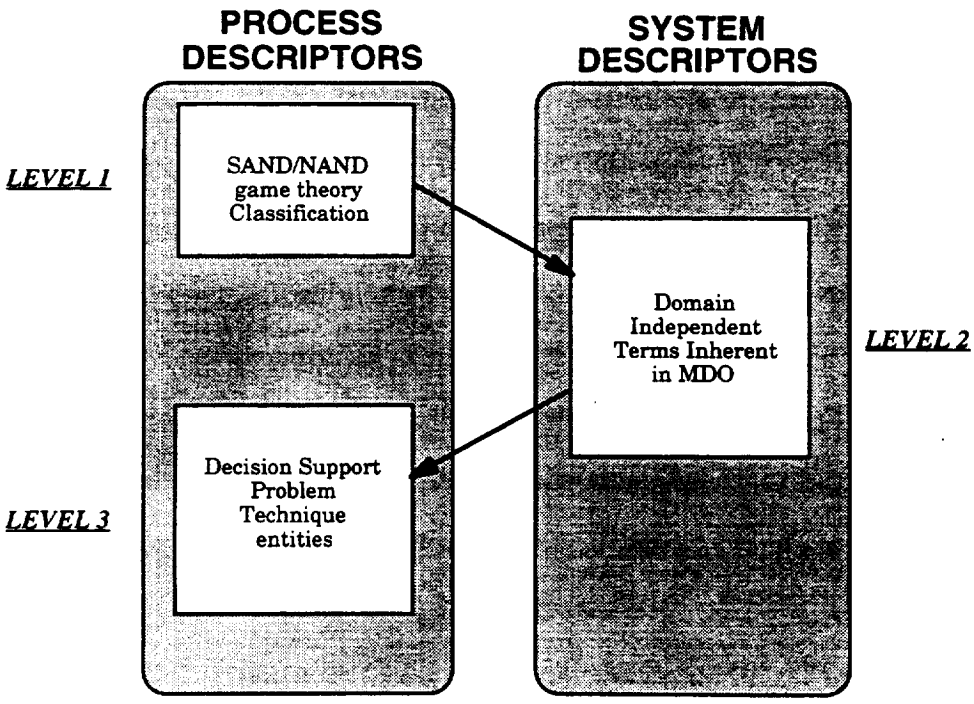


Figure 4.3. General Taxonomy

4.2.1 Level 1: Overall System and Process Formulation

Level 1 consists of the scheme proposed by Balling and Sobieski with game theoretical extensions. In this scheme, the overall analysis and solution scheme for the problem at hand is identified (Balling and Sobieski, 1994) and the interaction structure is identified using terms from game theory to classify the roles of the disciplines. By using the B-S scheme, insight into the structure of the system is apparent, but classification of the solution process is the primary focus. Once the first level terms have been determined, the second level of the classification is used to classify the problem and process further.

4.2.2 Level 2: System Definition

Level 2 contains domain independent linguistic terms that are used to define the system and the structure of the disciplines. These terms are inherent in complex systems design and MDO. A sample of these terms is shown at the top level in Figure 4.4. The process of identifying the domain independent terms involves surveying the relevant work, both research and application, in the field of MDO. Of course, there are countless terms used by different contributors, but the aim is to identify fundamental terms which are intrinsic to and define MDO as an emerging field of research and application. The domain independent terms do not connote any type of technological information concerning specific optimization algorithms, analysis packages, approximation techniques, etc. These terms are independent of time-based developments such as technology. The domain independent terms should act as an umbrella to the specific system developments in academia, government, and industry, while the other taxonomy levels encompass the process developments. Within this framework it is espoused that there are various "open" linguistic statements, such as "solution method", or "level of approximation". The "solution method" used varies according to problem requirements, system characteristics, researcher background, and so on. It is within these types of open statements that the

individual research and applications evolve in academia, industry, and government. In Figure 4.4 it is illustrated where many of the research topics and practical applications of MDO fit into some of the domain independent entities. For instance, under "solution", research areas include discrete methods, continuous methods, and multiple objective techniques.

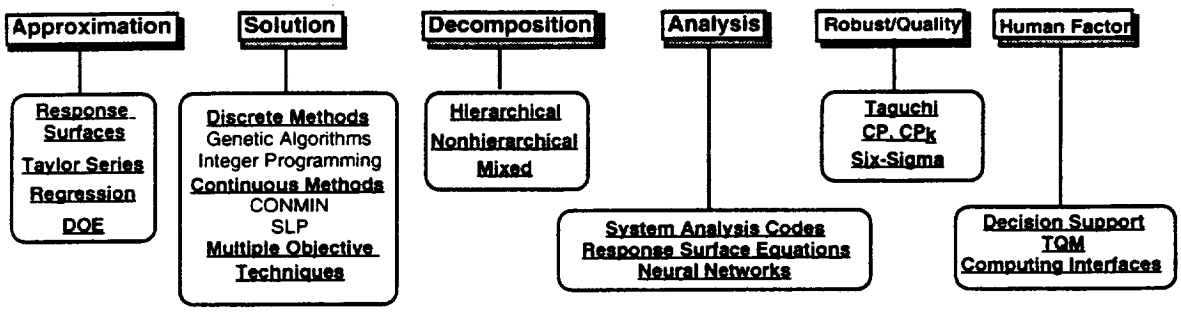


Figure 4.4. Examples of Linguistic Research Terms in MDO

4.2.3 Level 3: Process Definition

Level 3 contains the process independent base entities of the DSP Technique (Bras and Mistree, 1991) which are introduced in Section 4.1.2. These are the basic entities for a designer that are independent of the system or process at hand. These entities classify the type of action that must be made in order to perform the terms of level 2. Then, based on the action that must be made, the appropriate support tool, whether it be computer-based, experiment-based, or rule-based, can be used to help designers make decisions.

As technology continues to expand and better and faster approaches are developed, taxonomies should not change. If they do change, then they do not represent a robust, time-independent description of a set of entities. It is asserted that any lexicon in technical fields must be independent of technology. For instance, the Balling-Sobieski framework is independent of technology. In the B-S framework the simple classification of "evaluators"

could include crude Simpson's integration of strain energy to detailed finite element analysis and simulation. Therefore, the framework is independent of time-based developments, such as technology. A similar analogy is found in the area of chemistry. In chemistry, the framework is in the form of the periodic table. All research and technology, no matter how advanced, can be referred in some sense back to this table, and this will always be true. The pure sciences have set the standard for classifications of some sort. Granted, in design, or even multidisciplinary design, this type of framework is difficult due to the inherent lack of structure. In this chapter, value is added to this evolving framework of MDO to stimulate its acceptance as a basis for communication. In Section 4.3, the verification for Hypothesis I is provided by linguistically mapping the classification of three example problems using the B-S scheme and entities from the DSP Technique. Further verification of Hypothesis I is provided in Section 4.3 by classifying these example problems using the complete classification system of Section 4.2.

4.3 MAPPING OF APPROACHES: AN INTEGRATION OF IDEAS

In this section, it is illustrated how various applications of the DSPT to designing complex systems can be mapped into the B-S scheme. In particular, the design of a passenger aircraft, a thermal energy system, and a pressure vessel subject to design and manufacturing requirements are used to illustrate the mapping. The classification of these examples are also given, as further support for Hypothesis I. In Figure 4.5, a roadmap of the general mapping is given for this section. Problem formulations using coupled DSPs are mapped into the Balling-Sobieski formulations. In Figure 4.5, examples are given of both formulations including all coupling functions. This mapping includes comparing the *linguistic* entities of the formulations and illustrating the consistency among the entities.

The objective function in coupled DSPs is in the form of a *deviation function*, which characterizes the deviation from the goal achievements and the goal aspirations. The objective function in the B-S framework is either a standard objective function or a *discrepancy function* which is used to characterize the discrepancy in the goals, constraints, and coupling functions. In both frameworks, the form of the objective function is minimized. Using coupled DSPs, interactions between subsystems are modeled using *lateral interaction constraints* (LI), and interactions between the system and subsystems are modeled using *reverse* and *forward interaction constraints* (RVI, FVI). In the B-S framework, coupling is modeled using *coupling functions* (y_{ij}) and *cumulative design functions* (c_i). This mapping also represents the highest and lowest level of the taxonomy presented in Section 4.2 and establishes the continuity between the levels of classification in the taxonomy. Throughout this section, after the linguistic terms from the classification presented, the equivalent linguistic terms from the B-S scheme are given in parentheses.

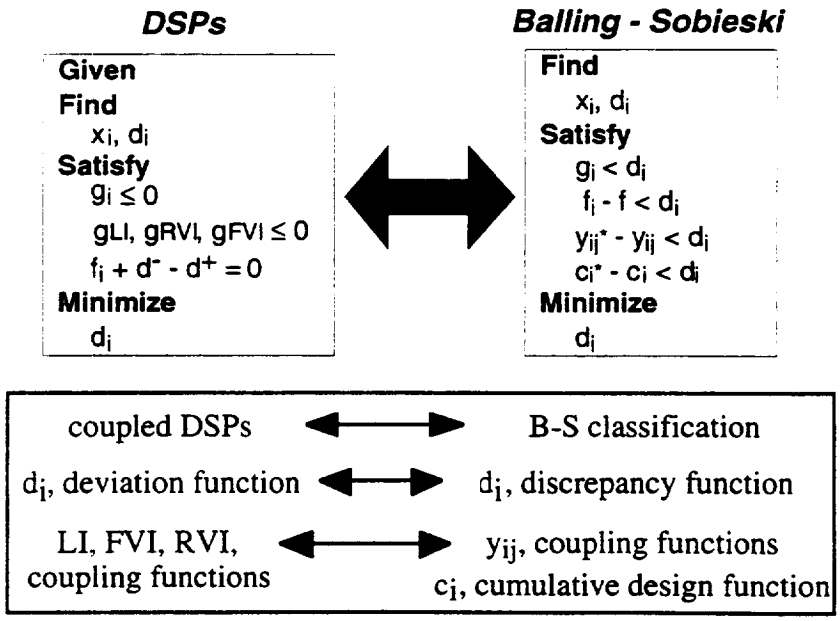


Figure 4.5. Overall Mapping of DSPs into B-S Framework

DSPs have been used to design many complex systems, including aircraft, ships, damage tolerant structural and mechanical systems, and thermal energy systems. Three of these examples are used in this section to support Hypothesis I.

Aircraft: Single-SAND-SAND-cooperative

In (Lewis, et al., 1994), technical, economic, and quality issues are addressed in the design of a passenger aircraft. The problem statement for the study is as follows:

> *A three engined subsonic jet transport is to be acquired. To ensure that the aircraft is operational from many airports the take-off field length should be less than 6,500 ft and the landing field length should be as close to 4,500 ft as possible. It is required that the range of the aircraft exceed 2,000 nmi.*
>
> *It is desirable that the airplane carry about 190 passengers, have a useful load fraction of 0.5, an endurance of 0.03 hours, and a range of 2,400 nmi. It is also desirable that the missed approach climb gradient be as large as possible.*
>
> *At this early stage, the variables to be determined are the wing span and area, fuselage diameter and length, installed thrust, take-off weight, airfoil thickness location parameter, wetted area to planform area ratio, useful load fraction, airfoil form factor, fuselage form factor, airfoil thickness ratio. The solution should provide information on the size of the aircraft based on geometrical parameters, aerodynamic considerations, the Federal Air Regulations, quality considerations, and economic issues.*

In Figure 4.6, the framework of this single level compromise DSP approach is given along with the systems descriptors of the compromise DSP and the technical, economic, and quality evaluation routines.

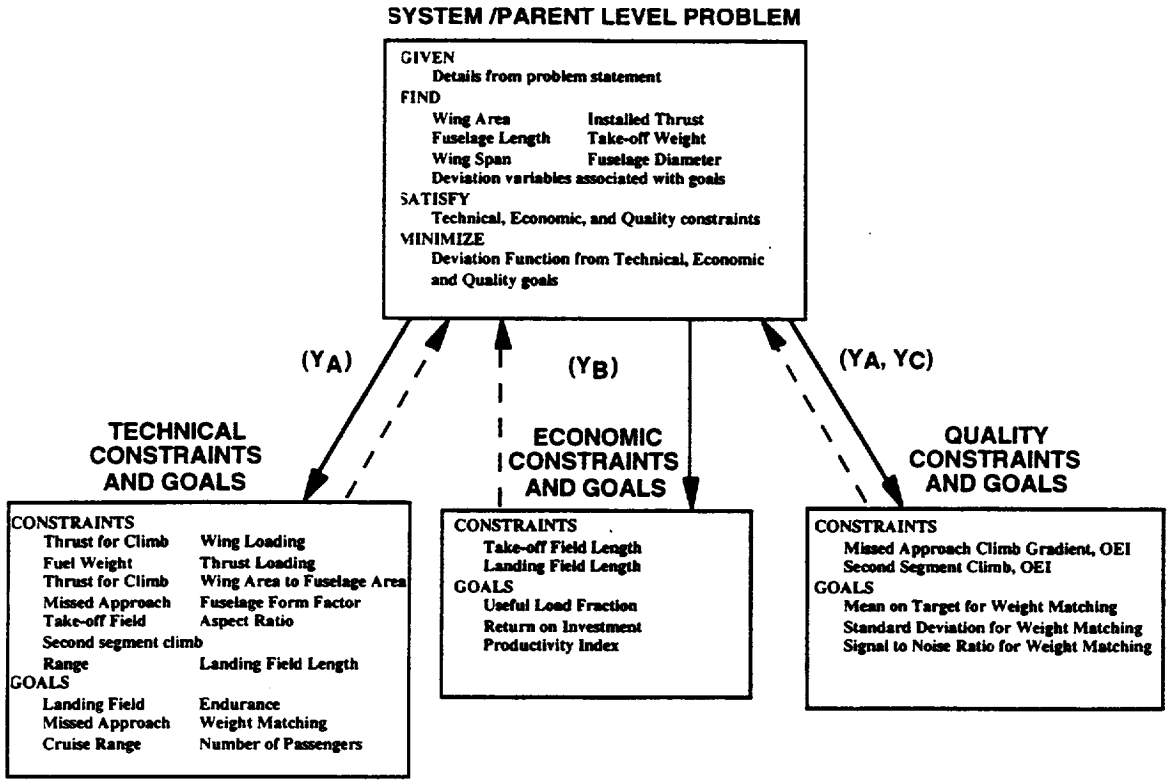


Figure 4.6. Multiobjective Aircraft Compromise DSP

In Figure 4.7, this single level solution approach is presented in the B-S framework. The first level of classification, single or multilevel approach, is considered. Since the system problem is formulated at a single system level, the first classification level is "Single." Since a single level is used, the game designation is *cooperative*. The second level of classification, SAND or NAND at the system level is considered next. In the current compromise DSP formulation, the computation of the design variables and the variables describing aircraft technical, economic, and quality performance (*state variables*) by the *system solver* is simultaneous. As indicated earlier, the *system solver* is the ALP Algorithm (Mistree, et al., 1993a), which is the solution algorithm for compromise DSPs. In the ALP Algorithm (*system solver*), the design and coupling variables are found and the constraint violations (*residuals*) and deviation function (*residuals*) are minimized based on

the constraint, goal, and coupling function information from the various disciplinary evaluators. Therefore, at the system level, the approach is classified as SAND.

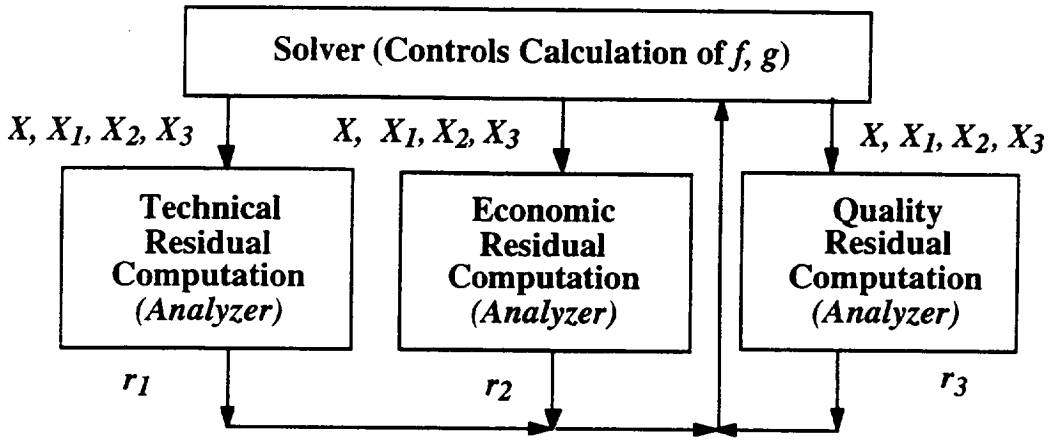


Figure 4.7. Single-SAND-SAND Formulation

The third level classification, SAND or NAND at the subsystem or discipline level is then considered. The compromise DSP formulation in this case is decomposed into three disciplines, technical, economic, and quality performance. Evaluator subroutines are called from the ALP Algorithm (*system solver*) for the technical, economic, and quality constraint and goal calculations. These subroutines may be part of the compromise DSP or may be separate subroutines depending on their sizes. At the discipline level, only evaluation is performed, and the deviation variables and constraint violations (*residuals*) are returned to the ALP Algorithm (*system solver*) along with values for the constraints, goals, and coupling functions. Therefore, at the discipline level, the classification is SAND. The classification of this approach at level 1 of the classification is given as Single-SAND-SAND-cooperative.

A representative of the full classification of the aircraft example is shown in Figure 4.8. At level 2, different linguistic terms can be used, but it is assumed in this example that the representative term at the second level is *solution*, as the model must be solved by performing a type of decision. As shown in Figure 4.8, multiple terms must be used at level 2 to classify the nature of the product being designed, but only *solution* is demonstrated here. The third level classification is given as *decision*. In this example, a *compromise DSP* is formulated and solved. Using the classification, the aircraft problem and process to solve it have been structured according to linguistic entities. A single-level cooperative formulation is used and solved using simultaneous analysis and design. The solution of the single-level formulation is found by solving a compromise DSP.

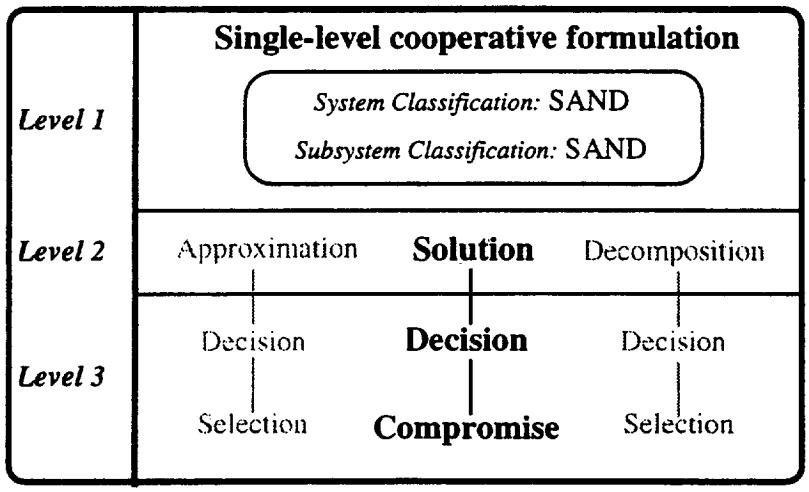


Figure 4.8 Representative Classification: Aircraft Example

Thermal Energy System: Multi-SAND-SAND-cooperative

In Kuppuraju (Kuppuraju, et al., 1985a), an approach at multilevel system decomposition and solution is presented using hierarchical compromise DSPs. The problem statements for the parent and subsystem problems are given.

Parent level: It is necessary to determine the quantity of raw materials (Ash A and Ash B) that have to be purchased each day to run a power plant. Information on the amount of Ash A, Ash B, and acid, and the yield of each Ash is given. The funds available to buy raw materials are limited to $17,000/day. The Ash is subject to storage space restrictions in the factory.

Subsystem level-Coal Problem: Based on the amount of raw materials purchased and hence the daily production quota, the exact amounts of Coal A and Coal B needed to fire the furnaces is to be determined. It is desirable that the cost of coal not exceed $600/day. The fuels have to satisfy the heat requirements and in addition conform to the pollution regulations and the handling constraints in the factory.

Subsystem level-Beam Problem: Given the amount of finished product produced in a day, it is desired to design supports for the centrifuge that can withstand this load. It is important to increase the surface area of the beam by as much as possible in order to enhance heat dissipation. The materials available for the beam are malleable cast iron, gray cast iron, and steel. Parameterized dimension of the beam along with information on the operation of the motor are given. The dimensions of the beam are limited by the volume of material available, a surface area limit, and a safety factor of 2.

In Figure 4.9, the hierarchical framework of this multilevel compromise DSP approach is given along with the systems descriptors of each compromise DSP. In Figure 4.10, this multilevel solution approach is presented in the B-S framework. The approach in Figure 4.9 is certainly *multi-level* as disciplinary design problems exist at the subsystem level. At the system level, there is a system level compromise DSP that is solved using the ALP Algorithm (*system solver*).

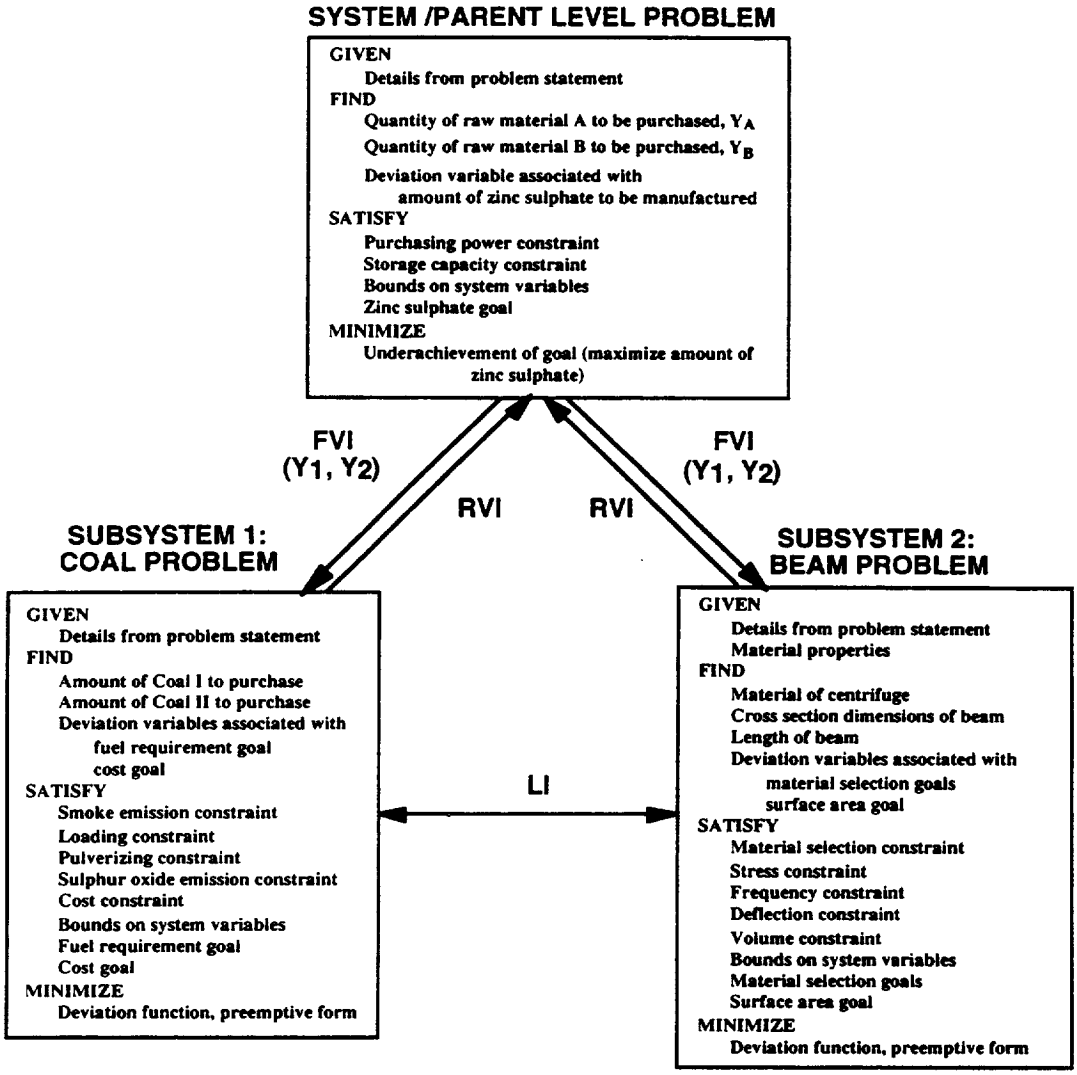


Figure 4.9. Multi-Level Thermal System: Coupled Compromise DSPs and Interactions

Then at the discipline, or subsystem, levels there are disciplinary compromise DSPs. Included in this approach are *lateral, forward,* and *reverse interaction functions* that dictate the coupling between the system and subsystems and among the subsystems. These interaction functions allow the formulation to be a *cooperative* one, because even though the disciplinary problems are distinct, the interactions are modeled and accounted for using interaction functions. That is, the subsystems are not acting on their own, and neither is

dominating the process. The linguistic entities of this approach are mapped to their equivalents in the B-S scheme are given.

Lateral interactions, LI → y_{ij}, coupling functions

Forward and reverse interactions, FVI and RVI → c_i, cumulative design function; f, objective function; x, design variables

d_i, deviation function → d_i, discrepancy function

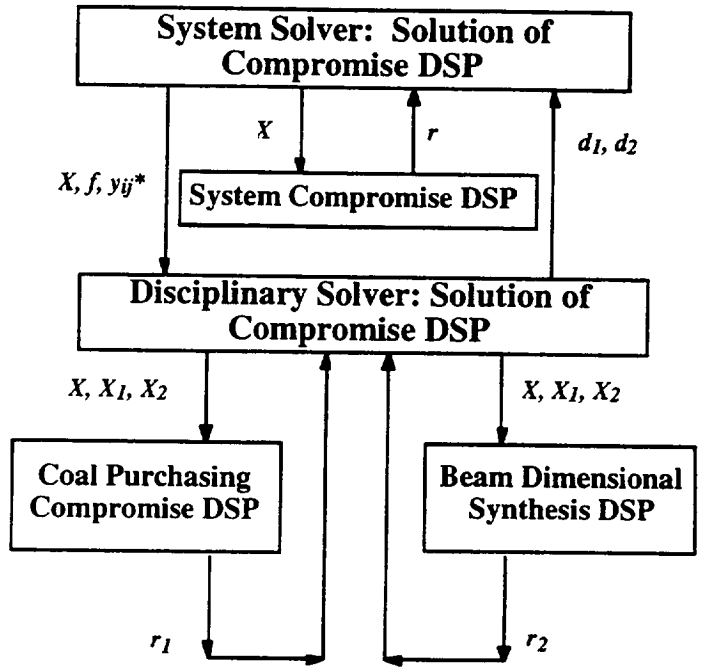


Figure 4.10. Multi-SAND-SAND Formulation

In Figure 4.10, the lateral interaction coupling functions, LI, are represented by the disciplinary design variables $\mathbf{X_1}$ and $\mathbf{X_2}$. Additional coupling considerations in this approach are the reverse and forward interactions, RVI and FVI, between the system and subsystem levels. The reverse and forward coupling functions between the system level and subsystems are denoted by *objective functions*, f, or *design variables*, x. These denote information passed from the system to the subsystems. Each subsystem compromise DSP

(which may depend on the solution of the system compromise DSP) is solved using the ALP Algorithm. The ALP Algorithm (*disciplinary solver*) of each subsystem is used to minimize the deviation function (*discrepancy function*) and finds the subsystem design variables. At the system level, the ALP Algorithm (*system solver*) is used to solve for the system design variables and system deviation function (*residuals*). The system solution depends on the deviation functions (*discrepancy functions,* d_i) from the subsystem problems, which may be in the form of reverse coupling functions (RVI), as in Figure 4.9. This multilevel approach is classified as Multi-SAND-SAND-cooperative at level 1 of the classification. A representative of the full classification of the thermal energy example is shown in Figure 4.11. Since a multi-level formulation is used, Figure 4.11 only shows one subsystem classification at levels 2 and 3. The other subsystems would have their own classifications at level 2 and 3 describing their subsystems. At level 2, different linguistic terms can be used, but it is assumed in this example that the representative term at the second level is *solution*, as the model must be solved by performing a type of decision. As shown in Figure 4.11, multiple terms must be used at level 2 to classify the nature of the product being designed, but only *solution* is demonstrated here. The third level classification is given as *decision*. In this example, a *compromise DSP* is formulated and solved. Using the classification, the thermal energy problem and process to solve it have been structured according to linguistic entities. A multi-level cooperative formulation is used and solved using simultaneous analysis and design. The solution of the problems in the multi-level formulation are found by solving compromise DSPs.

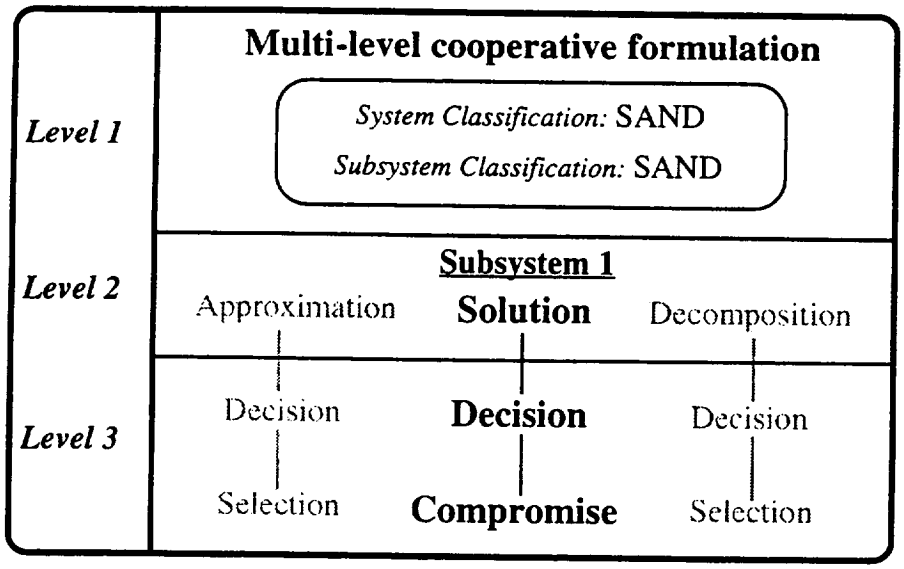


Figure 4.11. Representative Classification: Thermal Example

Pressure Vessel: Multi-SAND-SAND-cooperative

In (Karandikar and Mistree, 1992a, Karandikar and Mistree, 1992b, Karandikar and Mistree, 1992c, Karandikar and Mistree, 1993) the design of a pressure vessel, considering both design and manufacturing issues is presented. The problem statement for the pressure vessel is given below.

> *Design a cylindrical composite material pressure vessel with hemispherical and closures and having a volume of 10 mm^3. Two materials (carbon/epoxy composites) are available for fabricating the pressure vessel. The pressure vessel is to be manufactured by filament winding and the specifications for the filament winding operation need to be determined. The pressure vessel is subjected to internal pressure loads and a constant temperature difference across its thickness.*
>
> *The pressure vessel should not fail under the given loading conditions and should be manufacturable using the available filament winder. The performance factor of the pressure vessel is to be maximized. There is experimental evidence to suggest that is advantageous to keep the ratios of the boss opening diameter to the chamber diameter between 1/10 and 1/5. On manufacturing, the volume fraction of fibers and the degree of cure across the body of the pressure vessel should be uniform and the residual stresses in the vessel should be minimized. It is desirable that the fabrication time and the material cost be kept to a minimum.*

The solution process involved simulation of the technical performance and manufacturing considerations. The technical (design) variables and the manufacturing variables are found using separate compromise DSPs, but the compromise DSPs are coupled through the system variables, both design and manufacturing. This approach is similar to the previous approach to the thermal energy system, but at the system level, the compromise DSPs feed into a selection DSP where the best concept is selected based on the information from the compromise DSP solutions. The model of the system using entities from the DSP Technique is shown in Figure 4.12.

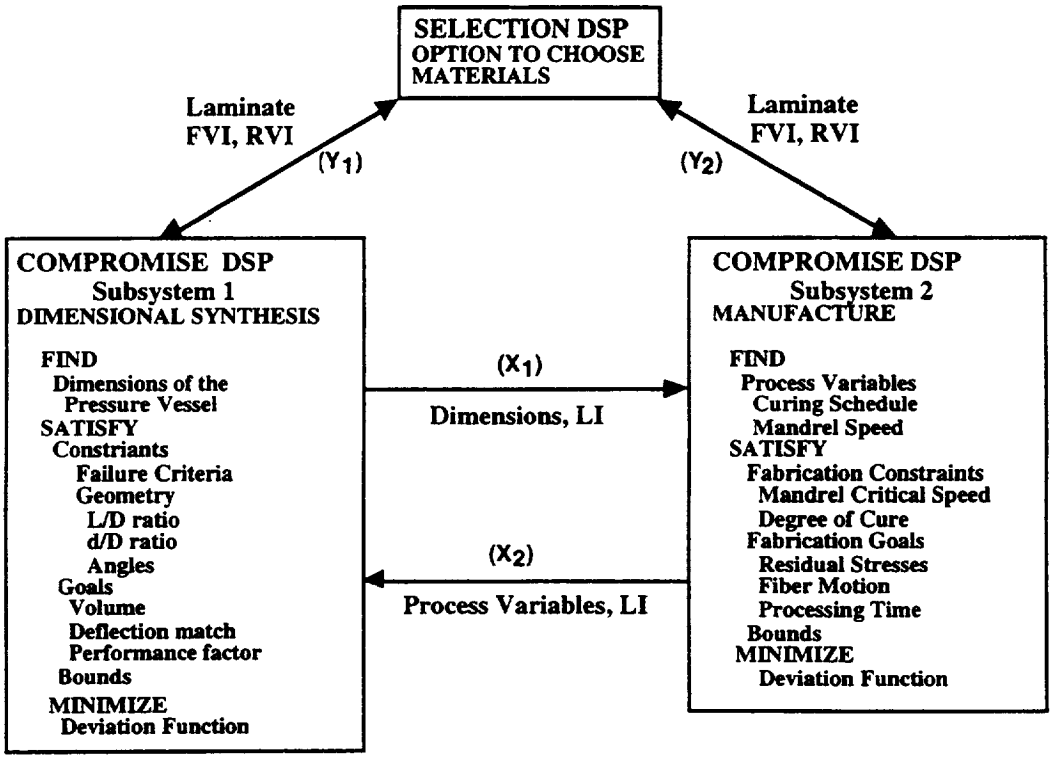


Figure 4.12. Multi-Level Pressure Vessel: Coupled Selection-Compromise DSPs and Interactions

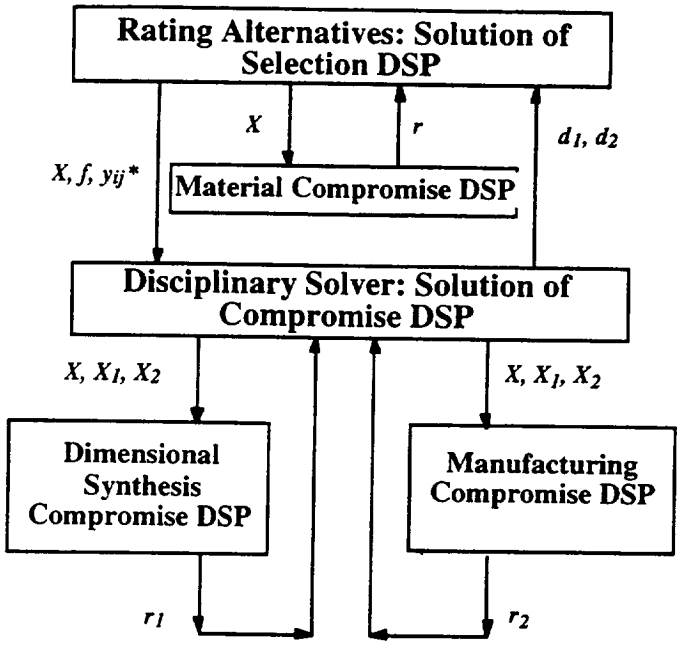


Figure 4.13. Multi-SAND-SAND Formulation

The corresponding schematic in the B-S scheme is shown in Figure 4.13. The solution approach is *multi-level*, since the model includes subsystem compromise DSPs. At the system level, the decision is a *selection Decision Support Problem*. This type of decision is distinctly different from a compromise DSP, as it involves choosing the best alternative from a pool of candidates. The solution of a selection DSP involves determining the alternative with the highest *merit function*. A selection DSP can be formulated as a linear goal programming problem and solved using the ALP Algorithm (*system solver*). The merit function in a selection DSP consists of evaluation criteria that includes information about the system goals, **f**, design variables, **x**, and coupling variables, $\mathbf{y}_{ij}^*$. This information is passed to the subsystems (FVI in Figure 4.12). At each iteration, information from the subsystems is passed to the system level (RVI in Figure 4.12) which characterizes how well the subsystem goals are met (*discrepancy functions*, d_i in Figure 4.13). The information in the selection DSP may be imprecise and objective in many

cases, but nonetheless the information is valuable to designers in complex system design (Mistree, et al., 1994, Mistree, et al., 1988). Due to the formulation and solution of selection DSPs, the system level classification is SAND. At the subsystem levels, a separate compromise DSP is formulated for each subsystem. The compromise DSPs are solved using the ALP Algorithm (*disciplinary solver*), similar to the thermal energy example. Therefore, at the subsystem level, the classification is SAND. Since the coupling variables are used and are being passed between the subsystem models, this problem is a *cooperative* one. That is, the subsystems are not acting on their own, and neither is dominating the process. The complete level 1 classification is Multi-SAND-SAND-cooperative. A representative of the full classification of the pressure vessel example is shown in Figure 4.14. Since a multi-level formulation is used, Figure 4.14 only shows the system level classification at levels 2 and 3. The subsystems would have their own classifications at level 2 and 3 describing their own subsystems. At level 2, different linguistic terms can be used, but it is assumed in this example that the representative term at the second level is *solution*, as the model must be solved by performing a type of decision. As shown in Figure 4.14, multiple terms must be used at level 2 to classify the nature of the product being designed, but only *solution* is demonstrated here. The third level classification is given as *decision*. In this example, a *selection DSP* is formulated and solved. Using the classification, the pressure vessel problem and process to solve it have been structured according to linguistic entities. A multi-level cooperative formulation is used and solved using simultaneous analysis and design. The solution of the system problem in the multi-level formulation is found by solving a selection DSP.

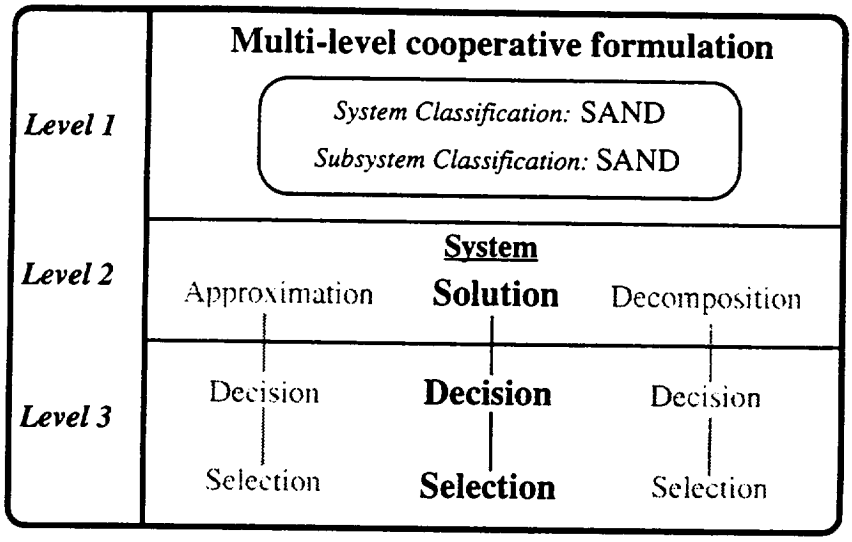


Figure 4.14. Representative Classification: Pressure Vessel

According to the assumptions of Balling and Sobieski (see Section 1.1), the B-S classification is useful when a mathematical model is available. In the approach to designing a pressure vessel, imprecise information (information not based on mathematical models, for instance) is used in selection DSPs. This type of information can and must be used in complex system design. In the later stages of a design process, the information may be completely precise, but in the earlier stages, designers must have the capability to classify imprecise approaches. To facilitate this, designers can move to the second and third levels of the classification presented where domain independent MDO terms can be used to describe the system and decision types can be classified. By only using the B-S scheme, designers are not able to discern between selection and compromise decisions, although they are completely different in philosophy and application. In the pressure vessel example, by using the extended taxonomy, the selection and compromise DSPs can be identified which would allow designers to apply the appropriate formulation and solution tools for each.

In this section, the synergy between Support Problem entities of the DSP Technique and the B-S classification is presented by mapping the approaches into the each other in order to add value to the classification system. They are both domain-independent, and by integrating game theoretic principles and the DSP entities, the classification is extended to being independent of time as well. Representative classifications for various examples are presented as further verification of Hypothesis I. The work supporting Hypothesis I is not meant to prescribe a new way of designing complex systems, but is used as a way to linguistically describe the common entities among different approaches to complex design processes. This mindset is illustrated in the next section.

4.4 THE MINDSET TAKEN IN THIS CHAPTER

The mindset of this chapter is one of *description* as opposed to *prescription*. The lexicon is presented as a means to describe the necessary product and process issues that designers, researchers, and engineers must handle in complex systems design. Ideally, the lexicon presented could be integrated into a computer-based design guidance system to guide a design teams through a design from problem formulation to final product design. This would require a parser to 1) identify the *linguistic* entities of a problem or process statement, and 2) embody the entities on a computer. The embodiment of these terms on a computer involves embedding a set of information characteristic to each term in the lexicon. For instance, in Figure 4.15, a prototypical interface for such a system is given. The discipline shown, the structures discipline, is identified as the *leader* in a *multilevel* formulation. This identification is often determined by a project manager, or may be prescribed based on organizational design or information barriers. Many times, disciplines

are not given a choice of being a leader or a follower or even cooperating. The relationships among disciplines *is typically dictated* by existing organizational constructs.

The system classification for the aircraft problem in Figure 4.15 is *SAND*, and the subsystem classification is *NAND*. The level 1 classification is shown at the top of Figure 4.15. At the second level, the *solution* (MDO term) of this disciplinary problem requires the solution of *heuristic DSP* (third level classification using DSP entities). A set of algorithms could be linked to the *heuristic* classifier to solve the structures problem using a leader/follower game theory protocol. Also, at the second level, the term *approximation* is shown in Figure 4.15. This term is classified at the third level as a *selection* decision, as a selection must be made for the level of approximation and approximation technique to use.

The designer ideally would be able to "click on" a given box, and be given information about the box, as illustrated in Figure 4.15. A parser could interpret the entries and then be linked to support tools which embody the entries in terms of computer entities. So, embedded within each box could be a set of computer-based tools, such as mathematical models, approximation techniques, or solution algorithms which could be invoked interactively or automatically by a designer.

The information in the lexicon is independent of domain and time, and would feed into the domain-specific methods, algorithms, and techniques. The developments of this chapter have been presented and implemented in a descriptive mindset as a means to lay the foundation for future prescriptive implementations.

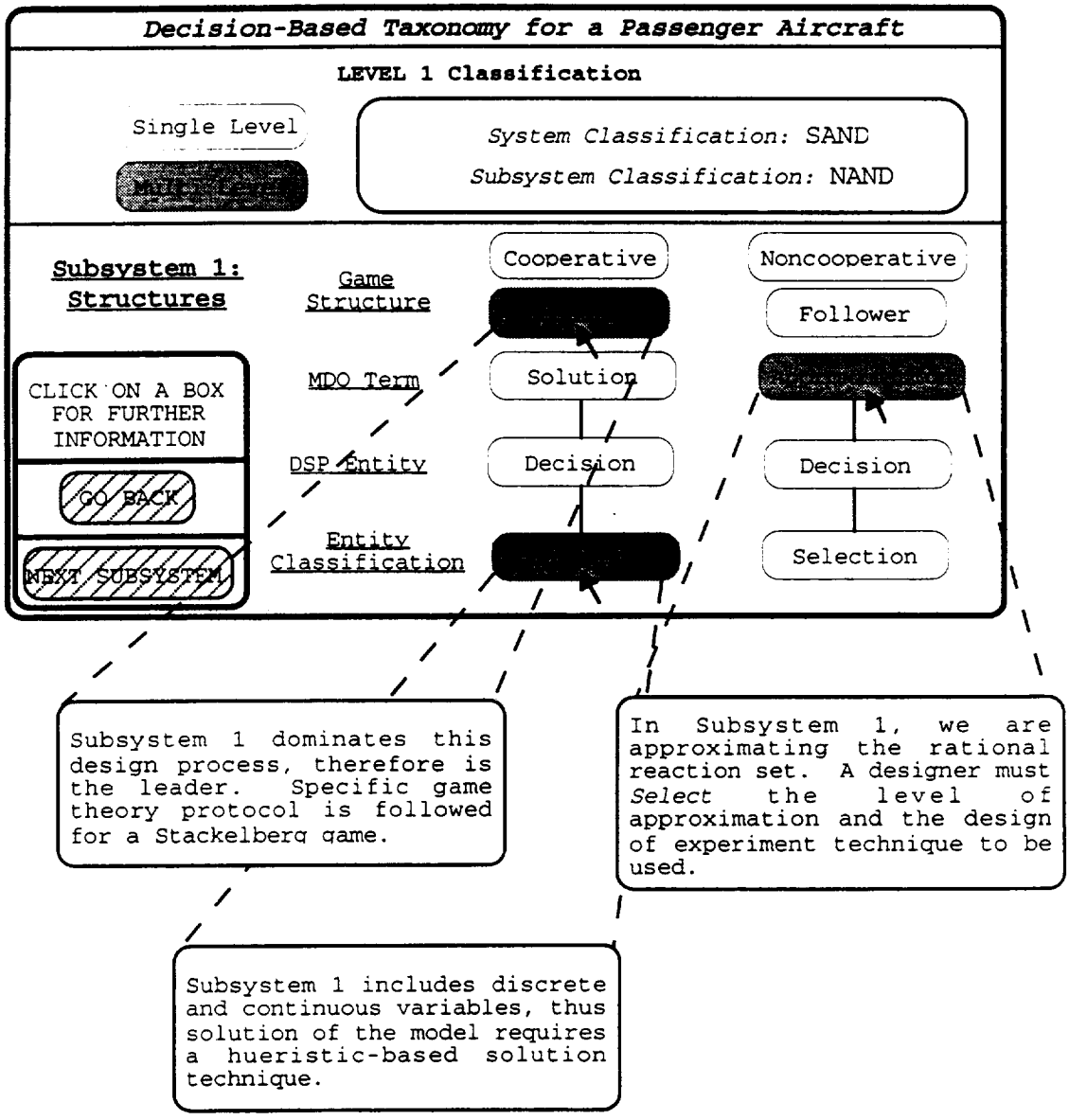


Figure 4.15. Typical User Interface of the Classification System

4.5 A LOOK BACK AND A LOOK AHEAD

In this chapter, MDO is approached from a game-theoretic, decision-based perspective and classification schemes are explored for designing complex systems and processes. A

game-theoretic, decision-based approach to design is mapped into the previous classification schemes proposed in (Balling and Sobieski, 1994, Cramer, et al., 1994). Each is independent of technology and together they classify the processes necessary in MDO. By integrating different levels of process and system descriptors into a lexicon, value is added to the B-S classification scheme by enhancing its breadth and depth in system and process classification. It is asserted in order to facilitate future communication in the field, computer implementation of this lexicon is needed. Implementation of this lexicon on a computer and the models (words or mathematical) supporting the system and process would aid application of this taxonomy to complex systems. It is acknowledged that the work of this chapter is only a precursor for a bigger goal. To be fully functional, a parser is needed to 1) identify the *linguistic* entities of a problem or process statement, and techniques are needed to 2) embody the entities on a computer to aid designers in the design of complex systems in MDO. The embodiment of these terms on a computer involves embedding a set of information characteristic to each term in the lexicon. This information is independent of domain and time, and feeds into the domain-specific methods, algorithms, and techniques.

Ideally, a designer would use the lexicon to examine and identify the key activities and characteristics of the system and processes at hand. Identification of these terms would help create models of the system and process in terms of domain independent terms. Then these models of the system and process can be solved, analyzed, synthesized, etc., in the context specific to the application. In this chapter, it is attempted to lay the foundation for further developments in this area. Establishing a common lexicon among researchers and developers in the area would facilitate communication and aid designers in establishing the structure of both a system and a process. Establishing a lexicon would allow designers either to rapidly change the classification of the approach or effectively introduce new

technology within an entity. Establishing this structure based on a common lexicon could increase the efficiency of the process and effectiveness of design decisions.

In this chapter, the first step of the algorithm presented in Section 3.1.1, is explored and developed. In the first step, the overall structure of the problem and solution process is classified. The work of this chapter is provided to support posits 1.1 and 1.2 which relate to Hypothesis I presented in Section 3.1.1. The observations relating to each posit are discussed.

Posit 1.1: Entities from the Decision Support Problem Technique provide a domain-independent lexicon for multidisciplinary design.

The lexicon presented in this chapter has been applied to various design problems from various domains. The entities in the DSPT are not dependent on any time-based developments such as technological improvements. Designers will continue to make decisions, and design will continue to be a sequence of phases and events. The DSPT entities have been integrated with linguistic terms from multidisciplinary design optimization and game theory to provide an encompassing framework for problem and process classification. The DSPT entities are discussed in Section 4.1.2 and are illustrated in various examples in Section 4.3. The linguistic entities of the DSP Technique are shown to be equivalent to those in the B-S scheme, establishing the linguistic synergy of the classification.

Posit 1.2: Game Theory principles can be used to extend problem formulation in multidisciplinary design.

The linguistic entities of game theory are used in Section 4.2 to expand the previous work in classifying product formulation into domains where the disciplinary models and their design teams may or may not cooperate. Noncooperation in theory is not advantageous in design, but in practice it is common. Also, true concurrency is rare; many times subsystems are designed sequentially. Game theory entities are used to describe these scenarios. The precise role of a discipline in a complex design process can be identified, which helps structure the process and allocate

resources. Therefore, game theory is used to extend problem classification for realistic complex design product and process formulations.

The classification of Chapter 4, as shown in Figure 4.16, provides the framework for the overall algorithm, presented in Chapter 3. This framework rests upon the foundation established in Chapters 1 and 2. The developments of Chapter 4 support Phase II of the strategy for verification and testing of the hypotheses. In Chapters 5 and 6 the two remaining steps of the algorithm and associated hypotheses are presented to fill out the complete algorithm.

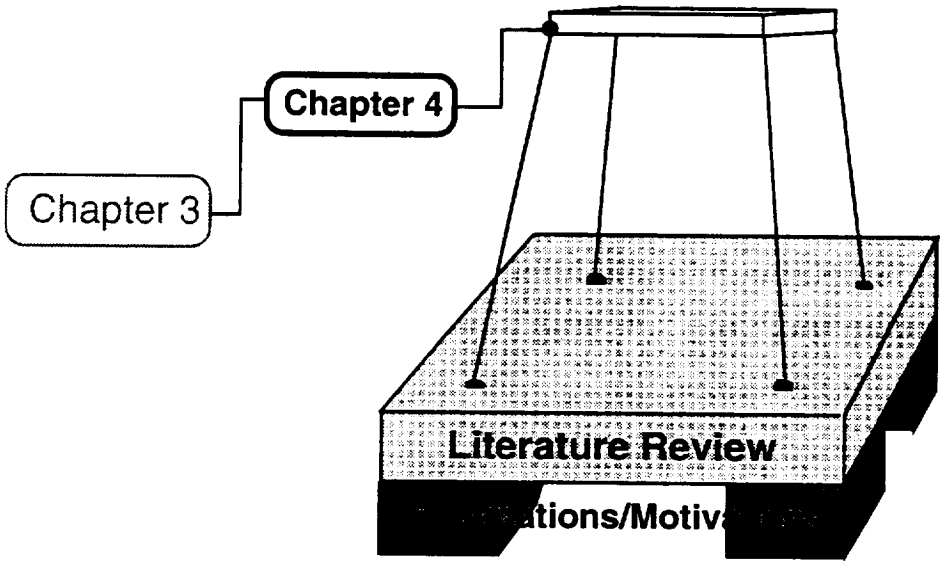


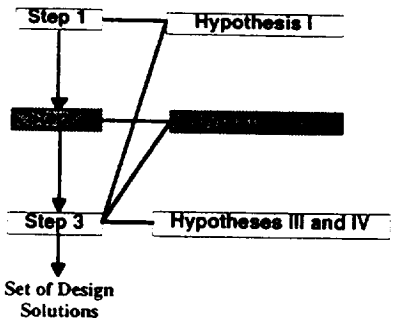
Figure 4.16. Frame of Reference: Chapter 4

CHAPTER 5

GAME THEORY IN COMPLEX SYSTEMS DESIGN: A CONCEPTUAL BASIS

Design is a process of decisions which are made by multiple decision makers, design teams, or organizations. In complex systems such as aircraft, the decisions are typically made by design groups organized by discipline. Ideally, a seamless Concurrent Engineering philosophy could be applied to a company's design process among disciplines. In reality, however, the simultaneous nature of information flow and cooperation, inherent in CE, among design teams makes concurrency difficult, if not impossible. M.L. Dertouzos and the Massachusetts Institute of Technology (MIT) Commission on Industrial Productivity, in their report *Made in America* (1989), found that six recurring weaknesses were hampering American manufacturing industries. The two weaknesses most relevant to product development were 1) technological weakness in development and production, and 2) failures in cooperation. The remedies to these weaknesses are considered the essential twin pillars of CE: 1) improved development process, and 2) closer cooperation (Schrage and Gordon, 1992). In the MIT report, it was recognized that total cooperation among teams in a CE environment is rare in American industry, while the majority of the research in mathematically modeling CE has assumed total cooperation.

Therefore, the focus of this work is on this notion of *cooperation*. Much has been written about the design of complex systems based on the implicit assumption that the design teams *cooperate*. There is a paucity of work dealing with strategic interactions in which the teams do not, or more directly, cannot cooperate. In this chapter, it is asserted that a complex design process with multiple designers or design teams can be abstracted as a series of *games* among design teams and that applying game theoretic principles to these processes can generate rich insights into design process and product structure. The use of game theory in engineering design is of relatively recent origin; therefore, the use of game theory within the context of Decision-Based Design requires further definition. This chapter provides the foundation for Step 2 of the overall algorithm introduced in Chapter 3. It provides support and verification for Hypothesis II (Figure 1.7) and Posits 2.1, 2.4, and 2.5 presented in Section 3.1.3 and shown below.



Hypothesis II: Game theoretic principles can be applied to accurately model and describe the interactions in complex systems design.

Posit 2.1: Design processes can be abstracted as games where the players are multiple designers or design teams and their associated analysis and synthesis tools.

Posit 2.4: Second order response surfaces can be used to approximate the Rational Reaction Sets of the disciplinary players in a design game.

Posit 2.5: The compromise DSP can be used as the fundamental construct to develop the game theory protocols and techniques.

5.1 FOUNDATIONS OF GAME THEORY IN DESIGN

Designing complex systems includes the difficult task of integrating disciplinary design teams each with their own analyses, syntheses, and decision processes. Optimizing such a system on a global scale is realistically impossible, but finding a solution which is "good enough" and robust is achievable. With only one decision maker (or design team), the problem becomes a scalar or vector optimization problem. However, in Multidisciplinary Design Optimization (MDO), many decision makers (design teams) may exist, and each decision maker's strategy to optimize his reward(s) often depend on the strategies and decisions of other decision makers. Therefore, the focus in this chapter is on problems characterized by:

- multiple decision makers who each have single rewards, and
- multiple decision makers who each have multiple rewards.

This focus in the context of optimization theory is shown as the shaded region in Figure 5.1. The modeling of strategic and optimal behavior based on the actions of other individuals is known as a *game* and the study of the strategic behavior is *game theory*.

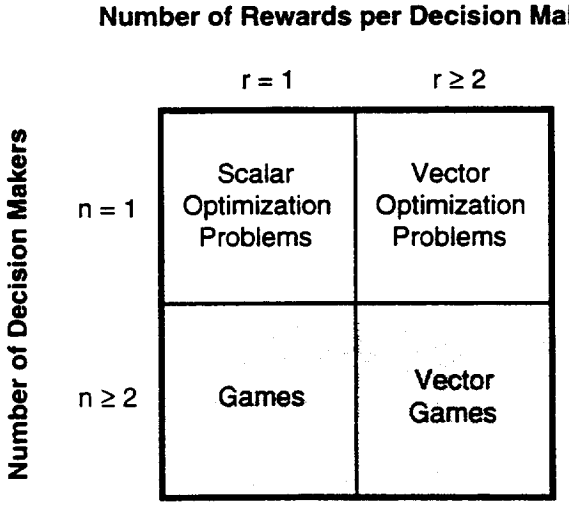


Figure 5.1. Various Formulations in Optimization Theory (Mesterton-Gibbons, 1992)

Typical courses in optimization focus on the upper-left quadrant, namely scalar optimization problems with one objective and one decision maker. In rare instances, problems in the upper-right quadrant, namely vector optimization problems with one decision maker are covered in advanced courses. In this chapter, the focus is on the lower two quadrants as a means to expand the application of optimization theory to problems that frequently occur in complex system design.

As mentioned before, the use of game theory in engineering design is of relatively recent origin; its usefulness in many other decision-making sectors is well-established. For instance, the 1994 Nobel Prize in Economics was awarded to two economists and a mathematician for their work in game theory. In awarding the prize to John F. Nash, John C. Harsanyi and Reinhard Selten, the Swedish Academy said the following in its citation: "Everyone knows that in games (such as chess and poker), players have to think ahead and devise a strategy based on countermoves from other players. Such strategic interaction also characterizes many economic situations, and game theory has therefore proved to be very useful in economic analysis." Although in games such as chess, there is a winner and a loser, this type of strategic interaction also occurs in complex system design, but the primary, overriding goal is the same for each player in the game: to meet the requirements and objectives as well as possible. The interactions, conflicts, and resolution processes in design parallel those in economics, board games, or any other strategic environment. In the next section then, it is asserted that a complex design process with multiple designers or design teams is simply a series of *games*.

5.2 DESIGN AS A GAME

In a general sense, a "game" is a set of rules completely specifying a competition, including the permissible actions of and information available to each participant, the criteria for termination of the competition, and the distribution of payoffs (1984). From a systems perspective, a "game" consists of multiple decision-makers or players who each control a specified subset of system variables and who each seek to minimize their own cost functions subject to their individual constraints (Myerson, 1991). This definition can also be applied to a design process; the design of a complex system is performed by multiple designers, who make decisions, and who each control their own design variables and are trying to minimize their objective functions subject to some technical and economic constraints. It is clear at least at a conceptual level, a design process and a typical game are similar in formulation.

To illustrate further, assume that a complex system such as an aircraft has been decomposed into disciplinary subsystems such as propulsion and structures. It is commonly accepted that a model such as

$$\underset{x \in X(p) \subset \Re^n}{\text{minimize}} \; f(x,p) = \{f_1(x,p),\ldots,f_r(x,p)\} \tag{5.1}$$

is the typical starting point for much of the current research and practice in systems modeling and applied optimization. And yet in specific design instances, this assertion should be boldly challenged. For example, since the propulsion designer only controls x and the structures designer controls p, how is p chosen in the propulsion design? Can the propulsion designer assume that the structural designer will always select the vector that is most advantageous to the propulsion design? If not, how should the propulsion designer respond to this conflict? This scenario describes a two-player strategic game where one

player controls x and the other player controls p, and where p represents all decisions which are outside the scope of the designer controlling x (Aubin, 1979, Dresher, 1981, Von Neumann and Morgenstern, 1944).

In processes of designing complex systems, the situation described is a very common practice. That is, complex design processes are performed many designers, each of which only controls a subset of the entire system variables. However, each designer certainly is not in isolation. The design of complex systems necessitates the coordination of multiple disciplines and designers, each with their own interests, goals, requirements, constraints, and analysis routines. At best, their state is one of semi-isolation; their decisions affect the outcome of the other disciplines through subsystem interfaces, which may be geometric, functional, behavioral, or logistical. There is extensive overlap and interaction of variables, constraints, and goals (hierarchically and nonhierarchically) which requires coordination and/or heuristic ordering. Since each designer has multiple objectives, and these objectives may conflict with the objectives of the other designers, there results a continual strategic interaction among designers or design teams. Ideally, complete cooperation occurs and each designer is aware of all the others and the decisions made by each. In well-controlled design problems, the typical research assumption of perfect or approximate communication is extremely beneficial (Sobieszczanski-Sobieski, 1988). Realistically, this is not always the case. In some cases, a Nash noncooperative formulation models a system and the lack of interaction among design teams more accurately (Nash, 1951). Although design teams may not explicitly choose to "not cooperate", due to the *lack of information* available to them, the scenario can be modeled as a noncooperative formulation. Each design team will have to make worst case assumptions concerning the other teams. Further, in many cases, a Stackelberg leader/follower formulation more accurately models the sequential interactions among design teams throughout a design process. Stackelberg formulations

are also effective in modeling the presence of a dominating design team which often makes their decisions first while assuming the other design teams will behave rationally.

So in essence, complex design processes can be abstracted as forms of a *game* among various players. These forms of games in design are quite unique applications of game theory, however. This uniqueness of applying game theory to design processes for complex systems is described and defined in the remainder of this chapter.

5.3 A DESIGN GAME DEFINED

Before full definitions are given, the assumptions under which this work operates are given.

Assumption 1: Models of players are mathematically explicit. That is, there exist full mathematical relationships in the form of equations. This does include use of fuzzy set theory and stochastic variables, as they can be represented by equations.

Assumption 2: The common link among each designer is the *hypothetical* single company under which they all work. Therefore, the notion of noncooperation is not intuitive. They each strive to act in the company's best interests, but information availability prevents full compliance.

Assumption 3: Disciplinary analysis and synthesis packages are not shareware. That is, each discipline does not have access to the other disciplinary software, even though each discipline may depend on the design information from other packages.

Assumption 4: The *design* variables of various designers or design teams do not overlap. The local control of each designer is exclusive. That is, if one designer controls $x^1 \in E^m$ and the other controls $x^2 \in E^s$, then

$$E^m \cup E^s = E \text{ and } E^m \cap E^s = \emptyset.$$

Given these assumptions, the definitions of game theory in the context of engineering design are now presented.

Definition 5.1. In design, a *player* in a game is the decision maker, which is embodied by a designer or design team and his associated analysis and synthesis packages.

Classically, players in a game may be people, groups of people or more abstract entities like computer programs or "nature". A principal tenet in game theory is the inherent or allocated decision-making ability or capability of each player in a game. A motivating tenet of this work is the notion of Decision-Based Design (DBD), where the principal role of a designer is to make decisions. In DBD, a computer may support a game player in making a decision, but the final decision is that of the player. Therefore, it is asserted that a decision maker in a design process (embodied by a designer or design team) is equivalent to a player in a game. The associated analysis routines, computer software and hardware in this game do not make decisions. They do not play the game. They support the decision-making strategy of a designer from a mathematical perspective. It is asserted that only the human decision makers play the game in design.

Definition 5.2. A *strategy* of a designer is the motivating principle of a decision formulation.

In classical game theory, players chose their strategies based on the information available to them. These strategies may change as more information becomes available. In design, however, the strategy is usually explicitly dictated in the decision formulation as the motivating principle. In a design process, a designer's motivation is to design a product that meets all requirements, technical, economic, safety, quality, etc. At some point in a design process, a formulation of a decision is typically given in terms of the system or

subsystem variables, constraints, and objectives. At this level of detail, a designer's motivation typically becomes embodied in the system objectives, such as "minimize cost" or "maximize quality." It is asserted that at a fundamental level, designers make one of two decisions, selection or compromise (Mistree, et al., 1993c). The *strategy* implied in the selection DSP is to maximize the merit function (Mistree, et al., 1994). The *strategy* implied in the compromise DSP is to *minimize* the deviation function (see Section 1.2.1) (Mistree, et al., 1993a). The deviation function is a measure of the difference between what can be achieved and what is desirable. It is asserted in (Mistree, et al., 1994) that this form of a strategy is generic and domain independent. It also encompasses single objective models such as "minimize weight."

Definition 5.3. A *payoff* is the value of the motivating function at a given move.

In the context of the compromise DSP, a payoff value is the value of the deviation function for a given set of values of the system variables. While, in most applications of game theory the players strive to *maximize* the payoff, in the context of the compromise DSP, the payers strive to *minimize* the payoff to each of them. Therefore, in the compromise DSP, the payoff can be viewed as the *cost incurred.* The deviation function of a designer can be viewed as the cost incurred by the designer.

Definition 5.4. The *state* of a player is described completely by the system variables and state variables.

A player's model is defined by the system variables, state variables, constraints, goals, and deviation function. In this chapter, one of the goals is to illustrate the equivalency of a player in game theory and a designer or design team in systems design. It is asserted that the following terms are equivalent in this work.

Game Theory	**Design**
Iterated Game ---------->	Design Process
Player ---------->	Designer/design team and their associated analysis/synthesis routines
Player's problem ---------->	Disciplinary design model
Cost Incurred ---------->	Deviation Function

In the context of complex systems design, a player's model is defined by the disciplinary problem. For instance, a structural design team's model is defined by the physics of a structural problem and the finite element codes, weight approximations, and any other associated analysis and synthesis codes. But the decisions of a structural design team are dependent on the decisions of other disciplinary design teams (and more generally, assumption number three at the start of this section). Certainly, the structural design problem depends on the size of the wing, the amount of thrust available, etc., and conversely, the structural design affects the other disciplinary design problems. However, the complexity of the overall system design problem warrants additional considerations when applying game theoretical principles. Typically, in game theory, the only information that is transferred is the values of the local design variables to another player. Yet in complex systems design, there is a need for transferring more than just design variables. Each designer or design team ideally would like not only the information concerning the design variables, but also the state variables describing the *state* of the other designers.

Consider again the structural design problem. The structural designer ideally would like to know the values of the systems variables of the aerodynamic designer such as the wing area and wing span. This is where classical game theory would stop. However, the structural designer would also need the values of the state variables such as the lift-to-drag

ratio on take-off, landing, and cruising. These are state variables which are *functions* of the design variables. So, someone schooled in functional analysis could ask:

If the state variables are functions of the design variables, why not just transfer the design variables?

This question in effect asks, why not just transfer the design variables, and allow the other disciplines access to the analysis codes where the state variables are calculated? Doing this may make sense in small problems, but in complex design problems where the analysis codes are large, expertise and judgment may be required to use the codes, and designers may be geographically separated, this is not practical, if even feasible. Therefore, the control vectors of each designer are defined in Def. 5.6.

Def 5.5. The *control* vector of a designer consists of the *design* variables and the *state* variables.

Mathematically, this is equivalent to

$$\mathbf{X} := \{\mathbf{x},\mathbf{s}\} \tag{5.2}$$

where $\mathbf{X}$ is the control vector, $\mathbf{x}$ is the design variable vector, and $\mathbf{s}$ is the state variable vector. These terms are formally defined in Section 4.1. Throughout the remainder of this chapter, the vector notation $\mathbf{X}$ will infer $\{\mathbf{x},\mathbf{s}\}$ unless specified.

The discussion of design variables thus far has not addressed the *type* of design variables that may exist in the control variable vector of each player. Often in complex systems design, design variables are continuous, discrete, integer, *and* Boolean in the same problem. Depending upon the type of design variable, the type of game and solution technique may change dramatically. Therefore, in order to use constructs from game theory, the basis of discrete, continuous and mixed games must be established. In the next section, the distinction between discrete and continuous games is presented including the

types of analysis and solution techniques required by each. Section 5.4 is concluded by addressing the notion of a mixed game in the context of the compromise DSP.

5.4 DISCRETE, CONTINUOUS, AND MIXED GAMES

The application of game theory has taken two primary paths. The problems in each path are distinctly different, but the applied theory is the same. These two paths are *discrete* and *continuous* games. Discrete games occur when the players' decision variables are found by making a selection among a discrete number of alternatives, such as the choice of materials. The Prisoner's Dilemma (Axelrod, 1984, Gleick, 1986, Hofstadter, 1985, Luce and Raiffa, 1957) is an example of classical discrete game and has been studied extensively using game theoretical techniques. Continuous games occur when the players' decision variables can take any real value, such as size of a beam. Application of game theory principles to these two types of problems is similar, but the method of solution can be completely different. The work in this thesis focuses on a third type of game, a *mixed* game where the control variables are both discrete and continuous. The method of solution of this type of game can vary greatly from the solution of purely discrete or continuous games.

5.4.1 Discrete Games

To illustrate a discrete game, consider the following problem.

Assume there are two designers (or design teams) working on the conceptual design of a passenger aircraft. They each control one discrete (configuration) variable that can have 2 values. For instance, the propulsion player could choose either 2 or 4 engines, and the aerodynamics player could choose single or double delta wing formations. The propulsion design player's objective is to bring the range of the aircraft as close to 5000 nmi as

possible, while the aerodynamics player's objective is to bring the lift-to-drag ratio of the aircraft as close to 20 as possible. Each player's objective is a function of the decision made by *both* players. The compromise DSPs of the two players are as follows:

Player 1: Propulsion

Given

Range = $f(X_1, X_2)$

Find

$X_1 \in [X_{1A}, X_{1B}]$, d_P^-

Satisfy

Range + d_P^- = $\text{Range}_{\text{Target}} = 5000$

Minimize

Z = Deviation from Range Target

$Z = d_P^- = f(X_1, X_2)$

Player 2: Aerodynamics

Given

Lift-to-Drag = $f(X_1, X_2)$

Find

$X_2 \in [X_{2A}, X_{2B}]$, d_A^-

Satisfy

Lift-to-Drag + d_A^- = $\text{L/D}_{\text{Target}} = 20$

Minimize

Z = Deviation from L/D Target

$Z = d_A^- = f(X_1, X_2)$

The payoff matrix of this game is shown in Table 5.1 and the four possible solutions are plotted in Figure 5.2. In the payoff matrix, the deviation functions of each player are given for each possible configuration. As formally defined in Section 5.3, a player's deviation function can be viewed as the cost incurred to a player. For instance, if Player Propulsion chooses X_{1A} and Player Aerodynamics chooses X_{2A}, the deviation function of Player Propulsion is 3000 (a Range of 2000) and Player Aerodynamics is 10 (an L/D of 10).

Table 5.1. Deviation Functions of 2 Players

		Player Aerodynamics	
		X_{2A}	X_{2B}
Player Propulsion	X_{1A}	(3,000, 10)	(2,000, 8)
	X_{1B}	(4,000, 4)	(1,000, 6)

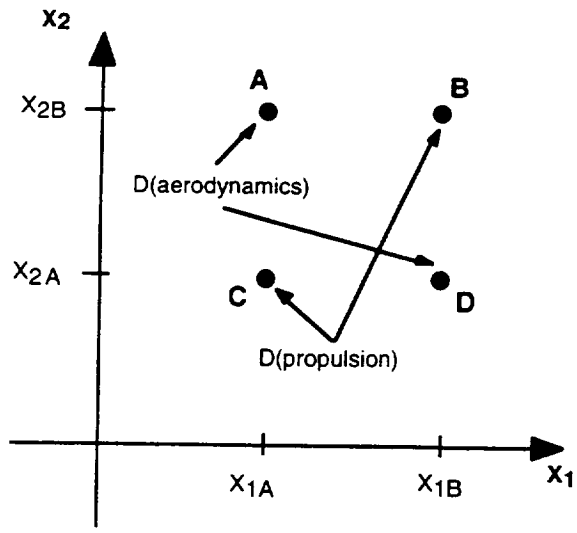


Figure 5.2. Possible Discrete Solutions

In this simple game, the solution depends on the protocol of the game (i.e., if cooperation exists, or if a sequential order exists). Each player wants to *minimize* his entry in the matrix. The different solutions are explored to illustrate the effects the different protocols have on the resulting solution of the problem.

Cooperative Solution

The cooperative solution for this game depends on the importance placed on the two objectives. If the range is considered to be the more important objective, then the solution is (1,000, 6) (point B in Figure 5.2), as this maximizes the range. If the lift-to-drag ratio is considered more important, then the solution is (4,000, 4) (point D in Figure 5.2), as this maximizes the lift-to-drag ratio. At these two solutions, both players cannot simultaneously improve upon their solutions. As is defined in Section 3.3.3, this is the definition for a cooperative or Pareto solution.

Noncooperative Solution

The noncooperative solution of this game is constructed by formulating each player's *rational reaction sets*. Construction of the rational reaction set assumes that information about the other player's strategy is not known and mathematically answers the following question: "No matter what decision player B makes, what decision can player A make to ensure that he does as well as possible?" In this problem, the rational reaction set of Player Aerodynamics is

$$D(Aerodynamics) = \begin{cases} X_{2A} \text{ if } X_1 = X_{1B} & (\text{point } D \text{ in Figure 5.2}) \\ X_{2B} \text{ if } X_1 = X_{1A} & (\text{point } A \text{ in Figure 5.2}) \end{cases} \tag{5.3}$$

and the rational reaction set of Player Propulsion is

$$D(Propulsion) = \begin{cases} X_{1A} \text{ if } X_2 = X_{2A} & (\text{point } C \text{ in Figure 5.2}) \\ X_{1B} \text{ if } X_2 = X_{2B} & (\text{point } B \text{ in Figure 5.2}) \end{cases} \tag{5.4}$$

Both of these RRS's are shown in Figure 5.2. The noncooperative solution, if it exists, is the intersection of the two rational reaction sets. In this game the intersection is

$$D(Aerodynamics) \cap D(Propulsion) = \varnothing. \tag{5.5}$$

In other words, no solution exists for the noncooperative protocol. This is evident from Figure 5.2; the two rational reaction sets do not intersect.

Stackelberg Leader/Follower with Player Propulsion as the Leader

In the Stackelberg formulation, the leader has the advantage of knowing how the follower will react to his decision. In other words, the leader knows the rational reaction set of the follower. Therefore, Player Aerodynamics' (follower) strategy is to choose X_{2B} if Player Propulsion (leader) chooses X_{1B}, and to choose X_{2B} if Player Propulsion chooses X_{1A}. Since the leader in this game knows this information about the follower, the leader chooses X_{1A} to minimize his deviation function, so the payoff for the players is (2,000, 8) at the point (X_{1A}, X_{2B}) (point A in Figure 5.2).

Stackelberg Leader/Follower with Player Aerodynamics as the Leader

Player Propulsion's (follower) strategy is to choose X_{1B} if Player Aerodynamics (leader) chooses X_{2B}, and to choose X_{1A} if Player Aerodynamics chooses X_{2A}. Since the leader in this game knows this information about the follower, the leader chooses X_{2B}, so the payoff for the players is (1,000, 6) at the point (X_{1B}, X_{2B}) (point B in Figure 5.2). It is not necessarily an advantage to be a leader or follower. In this game, player Aerodynamics would prefer to be the leader, but player Propulsion would prefer to be the follower.

Obviously, this is an exaggerated simplification of a design decision. In design, many variables are continuous. That is, they can take on any positive real value. In the next section, the solution of various protocols is illustrated for a continuous game.

5.4.2 Continuous Games

Continuous games occur when the decision variables are continuous. Payoff tables such as Table 5.1 cannot be constructed with an infinite set of variable values. In addition, the solution of a particular protocol of a continuous game requires more than a simple exhaustive search or inspection, which may be adequate in the discrete domain. Therefore, methods to solve continuous games borrow from the field of nonlinear optimization. To illustrate, consider the following problem.

Assume the same two designers (or design teams) are working on the conceptual design of a passenger aircraft. They each control one variable which can take on any real value. For example, the propulsion player could control the installed thrust, and the aerodynamics player could control the wing area. The propulsion design player wants to maximize the range of the aircraft, and the aerodynamics design player, wants to maximize the lift-to-drag ratio of the aircraft. It is *assumed*, for illustration purposes that the deviation functions of each player can be *approximated by quadratic functions*. Each player's objective is a function of the decision made by *both* players. The two players compromise DSPs are as follows:

Player Propulsion

Given

Range = $f(X_1, X_2)$

Find

X_1, d_P^-

Satisfy

Range + d_P^- = $\text{Range}_{\text{Target}}$

Minimize

Z = Deviation from Range Target

$Z = d_P^- \approx (X_1 - 2)^2 + X_2^2$

Player Aerodynamics

Given

Lift-to-Drag = $f(X_1, X_2)$

Find

X_2, d_A^-

Satisfy

Lift-to-Drag + d_A^- = L/D_{Target}

Minimize

Z = Deviation from L-to-D Target

$Z = d_A^- \approx (X_1 - X_2)^2$

This problem is the same as the discrete problem in Section 5.4.1, *except* the control variables are now continuous. That is, they can take on any real value, whereas in Section 5.4.1, they could only take on one of two values. In addition, each player has an explicit mathematical form of the deviation function. Solving this type of game is quite different from simply analyzing a payoff table. Knowledge of optimization theory or more frequently, nonlinear programming techniques now become a necessity in order to find a solution. In Figure 5.3, the level sets of the deviation functions of each player, d_A and d_P, are plotted as functions of the design variables of each player, X_1 and X_2. Each player wants to bring his deviation function to zero. In Figure 5.3, this occurs along the line $X_1 = X_2$ for player aerodynamics (d_A), and at the point $X_1 = 2$, $X_2 = 0$ for player propulsion (d_W). Obviously, both conditions cannot be simultaneously met. Therefore, solution of this problem again depends upon the protocol and interactions between the players.

Cooperative Solution

The cooperative solution for this game depends on the importance placed on the two objectives. The set of Pareto solutions can be constructed by using a composite deviation function,

$$d = \alpha_1 d_P + \alpha_2 d_A = \alpha_1[(X_1 - 2)^2 + X_2^2] + \alpha_2[(X_1 - X_2)^2]. \tag{5.6}$$

To find the set of solutions to this problem, the partial derivatives are taken

$$\frac{\partial d}{\partial X_1} = 2X_1 - 2\alpha_1 X_2 - 4\alpha_2 = 0 \tag{5.7}$$

$$\frac{\partial d}{\partial X_2} = 2X_2 - 2\alpha_1 X_1 = 0 \tag{5.8}$$

where

$$0 \le \alpha_1 \le 1, \quad 0 \le \alpha_2 \le 1, \quad \alpha_1 + \alpha_2 = 1.$$

Solving these equations, the set of solutions are

$$X_1 = \frac{2\alpha_2}{(1 - \alpha_1^2)}$$

$$X_2 = \frac{2\alpha_1\alpha_2}{(1 - \alpha_1^2)}. \tag{5.9}$$

This set of solutions (which depend on the weights assigned to the two objectives) is shown in Figure 5.3 as the line between points A and B. Along this line, both players cannot simultaneously improve upon their solutions. As presented in Section 3.3.3, this is the definition for a cooperative or Pareto solution.

Noncooperative Solution

The noncooperative solution of this game is constructed by formulating each player's *rational reaction sets*. To formulate the RRS of the aerodynamics player, it is necessary to determine what value of X_2 would be advantageous for the aerodynamics player for any value of X_1. For any value of X_1, the aerodynamics player can minimize his deviation (to a value of zero) by setting $X_2 = X_1$. Therefore, the rational reaction set for the aerodynamics player (shown in Figure 5.3) is

$$D(Aerodynamics) = \{(X_1, X_2) \in E^2 | X_2 = X_1\}. \tag{5.10}$$

To formulate the RRS of the propulsion player, it is necessary to determine what value of X_1 would be advantageous for the propulsion player for any value of X_2. For any value of

X_2, player propulsion can minimize his deviation function by setting $X_1 = 2$. Therefore, player propulsion's rational reaction set (shown in Figure 5.3) is

$$D(\text{Pr}opulsion) = \{(X_1, X_2) \in E^2 | X_1 = 2\}. \tag{5.11}$$

From Figure 5.3, it is clear that the intersection of the two sets occurs at (2, 2) with a payoff of (d_P, d_A) = (4,0). This corresponds to the solution with the Aerodynamics player as the leader, and is shown as point C in Figure 5.3.

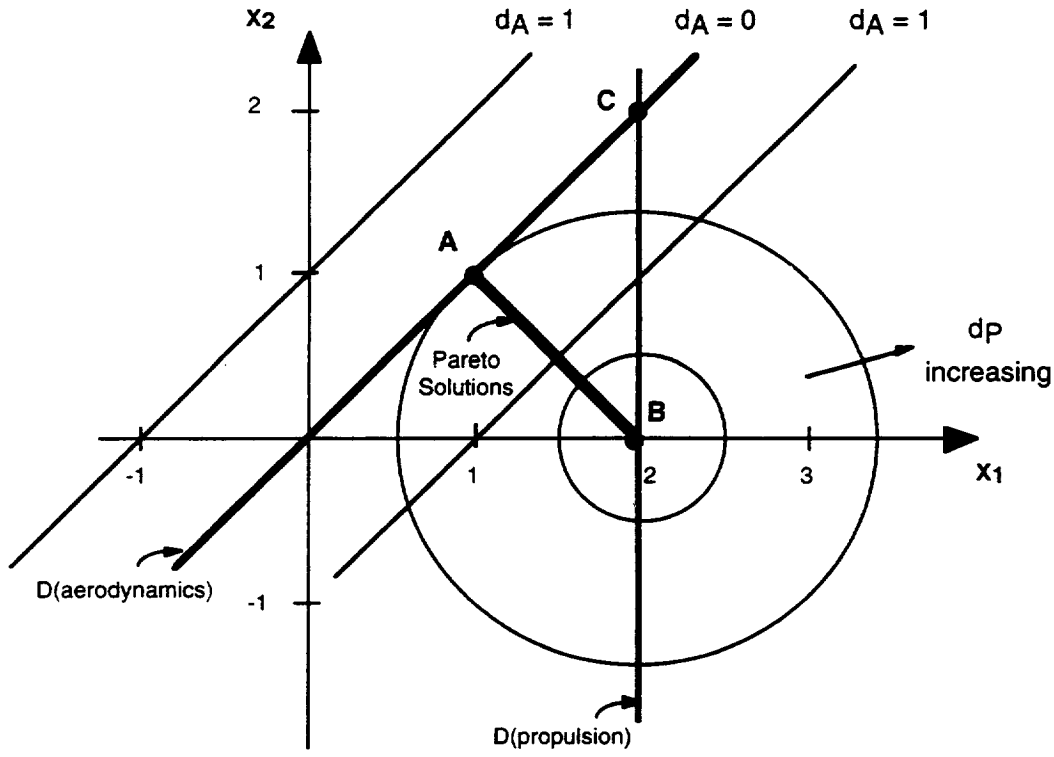


Figure 5.3. Solutions for Various Protocols

Stackelberg with Player Propulsion as the Leader

Since the leader in this game knows the strategy of the follower in the form of player aerodynamics' rational reaction set, the leader (propulsion) chooses $X_1 = 1$ to minimize

$$Range[X_1, X_2(X_1)] = (X_1 - 2)^2 + X_1^2 \tag{5.12}$$

and subsequently, player aerodynamics chooses $X_2 = X_1 = 1$, and the solution is (1, 1) and the payoff for the players is $(d_P, d_A) = (2, 0)$. This is shown as point A in Figure 5.3.

Stackelberg with Player Aerodynamics as the Leader

Since the leader in this game knows the strategy of the follower in the form of player propulsion's rational reaction set, the leader (aerodynamics) chooses $X_2 = 2$ to minimize

$$\text{Lift} - \text{to} - \text{Drag}[X_2, X_1(X_2)] = (2 - X_2)^2 \tag{5.13}$$

and subsequently, player propulsion chooses $X_1 = 2$, and the solution is (2, 2) and the payoff for the players is $(d_P, d_A) = (4, 0)$. This is shown as point C in Figure 5.3. With these simple examples, it is obvious that the solution to the problem differs depending upon the protocol between the players. In order to ensure the best overall solution in design, it is paramount to explore and understand the results and implications of each protocol. This exploration is presented in Chapter 7.

In complex systems design, and engineering design in general, many times the design variables are not all discrete or continuous, but are a mixture of continuous and discrete variables. In this case, the game becomes a mixed game.

5.4.3 Mixed Games: Application to Design

In complex systems design, designers or design teams usually control multiple system variables that are not all discrete, but are continuous, integer, and discrete. They also have state variables, equality and inequality constraints on the design, and multiple objectives to meet as closely as possible. A typical game in the design of a complex system combines aspects of discrete and continuous games. The focus in this work is not on discrete or continuous games, but on mixed discrete/continuous games. Although the developments in this work can certainly be used for discrete or continuous games, in this thesis, they are

illustrated primarily using mixed games. The examples presented in Sections 5.4.1 and 5.4.2 are simple illustrations of the principles of game theory. The foundation for game theoretic principles in complex system design are developed and applied to a representative system in this thesis. The departure from the previous game theory examples can be summarized by:

- presence of discrete (including integers) and continuous variables,
- presence and coupling of state variables,
- multiple objectives within each problem,
- multiple disciplinary nonlinear constraints, and
- the use of extensive disciplinary, platform-dependent analysis routines.

Formulating models for each player and finding the various protocol solutions, as in Sections 5.4.1 and 5.4.2, is a much more difficult problem with the introduction of these aspects commonly found in the design of complex systems. The general form of a mixed discrete/continuous compromise DSP is given in Section 3.4.4, Figure 3.15. When two or more compromise DSPs are coupled, certain aspects of the general compromsie DSP may be augmented. For the case of a 2-player game, the compromise DSP of player 1includes the following changes:

- Possible *given* information from the other player, $\mathbf{X_2}$
- Constraints and Goals are functions of the control variables of the other player, $\mathbf{g(X_1, X_2)}$, $\mathbf{f(X_1, X_2)}$
- Deviation Function is also a function of the control variables of the other player, $\mathbf{Z(X_1, X_2)}$.

The different protocols, formulations, and solutions of a mixed discrete/continuous game, involving multiple mixed compromise DSPs, in complex systems design are illustrated and explored in Chapter 7.

Although the definitions in Section 5.3 and examples in this section have been specified for abstracting design as a form of a game, the fundamental principles of game theory remain

the same. There are many protocols in game theory that are used to model various situations among the game players. It is asserted that three of these protocols have relevance to the situations often found in design processes among the designers or design teams. The mathematical basis for these various protocols is presented in Section 3.3.3, and the implementation strategies for each protocol are presented next.

5.5 GAME PROTOCOLS IN DESIGN

As introduced in Section 3.3, the focus in this dissertation is on three primary protocols applicable to design processes: cooperative, noncooperative, and leader/follower. In all three protocols, some form of approximation is used to generate a useful solution. The approximation tools used for each protocol are shown in Table 5.2.

Table 5.2. Protocol Approximation

Protocol	What is Approximated?	Approximation Tool
Cooperative	Nonlocal State Variables	GSE and Taylor's Theorem
Noncooperative	Rational Reactions Sets	Design of Experiments and Response Surfaces
Stackelberg Leader/Follower	Rational Reactions Sets	Design of Experiments and Response Surfaces

The implementation of each approximation strategy is discussed in this section. The cooperative formulation is constructed at two distinct levels, full cooperation and approximate cooperation, and their application to complex systems design is presented in the Section 5.5.1.

5.5.1 The Cooperative Formulations

If the players cooperate, they can be expected to obtain better solutions than when they do not. This is the typical optimization approach: to assume total cooperation among decision makers, disciplines, or subsystems. Previous work in multidisciplinary design has assumed cooperation exists among the players (Bloebaum, et al., 1992, Renaud and Gabriele, 1994).

Full Cooperation

The steps to construct and solve the full cooperative protocol are as follows:

❶ Combine each players' model into one encompassing model.
❷ Solve the model using appropriate continuous, discrete, or mixed solution technique.

The full cooperation protocol is illustrated in Figure 5.4. The disciplinary compromise DSP of Player 1 and Player 2 are combined into one compromise DSP in Figure 5.4 and solved using the *cumulative* design variables, constraints, goals, and deviation variables of both players. It must be stressed that determining priorities on the goals when the encompassing compromise DSP is formulated is not a trivial matter unless a simple Archemedean scheme with equal weighting is used. Insight into the customer and problem requirements must be used when establishing weights or priorities.

Although conceptually the full cooperative protocol is simple and theoretically sound, Pareto solutions are very difficult to compute in complex systems designs since the models of the different disciplinary players (designers, or design teams) utilize different analysis packages and many times are solved at different points in a process using approximate or incomplete information. In other words, a single objective which combines the objectives from players A, B, and C using a weighted sum such as

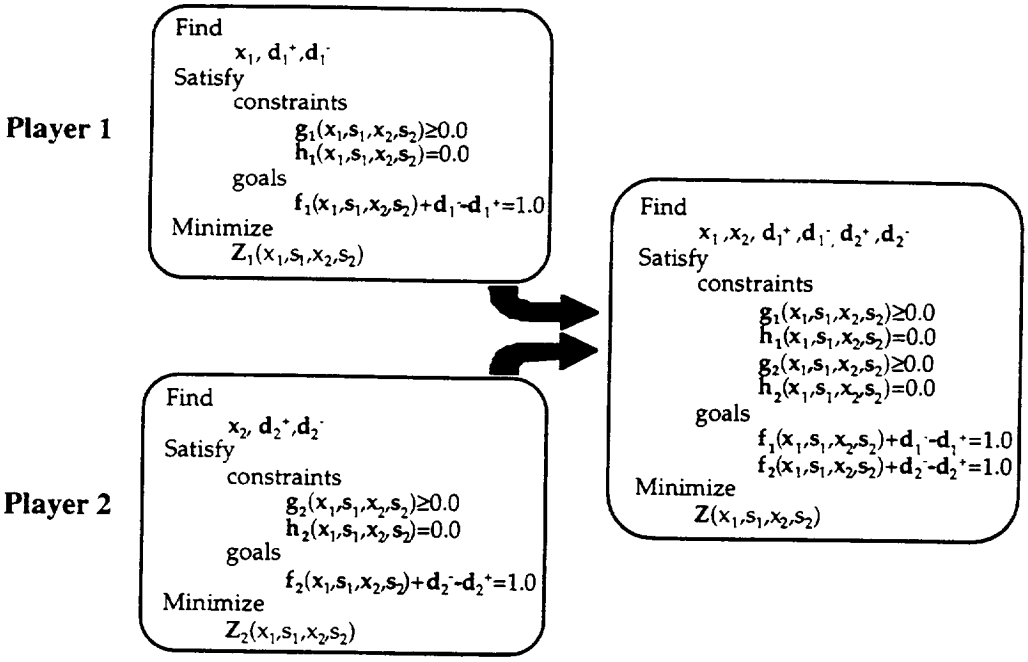


Figure 5.4. Full Cooperation: Pareto Solutions

$$F(X_A,X_B,X_C) = w_A f_A + w_B f_B + w_C f_C$$

where

$$\sum w_i = 1$$

$$0 \leq w_i \leq 1,$$

is typically impractical. Furthermore, combining separate models is often computationally impossible due to the sheer size of the models and analysis routines. Therefore, the definition of a cooperative solution in complex systems design can be extended to *approximate* cooperation where models can remain separate but linked through approximations of the coupling variables which are needed by more than one discipline.

Approximate Cooperation

The notion of cooperation in complex systems design is one of *approximate* cooperation. Approximate cooperation is achieved using approximations of the state variables, including

constraints and goals needed from the other players. However, approximation of every constraint, goal, and state variable is unrealistic. Only, the coupled equations (i.e., equations which are functions of the design variables of two or more players) are approximated. Therefore, required nonlocal information about the other players is approximated in each player's model. This approximation is accomplished using the Global Sensitivity Equations (GSE) method first proposed in (Sobieszczanski-Sobieski, 1988) and successfully used in the design of complex systems (Bloebaum, et al., 1992, Renaud and Gabriele, 1991, Renaud and Gabriele, 1993, Renaud and Gabriele, 1994). The fundamental constructs used in modeling approximate cooperation are introduced in Section 3.3.4. In this work, the full derivatives from the GSE method are used in a Taylor series expansion to approximate nonlocal variables. The steps to modeling and solving an approximate cooperation formulation are given as follows.

❶ Construct approximations of nonlocal behavior variables
- ❶a. Perform an initial analysis and take partial derivatives of behavior variables 1) with respect to the other behavior variables, matrix [M], and 2) with respect to the local design variables, matrix [B].
- ❶b. Set up and solve the GSE matrices.
- ❶c. Use the full derivatives in a first-order Taylor series approximation.

$$s(x_a, x_b, x_c) \approx s^o + \frac{ds}{dx_a}(x_a^o - x_a) + \frac{ds}{dx_b}(x_b^o - x_b) + \frac{ds}{dx_c}(x_c^o - x_c) \qquad (5.14)$$

❷ Solve disciplinary models using nonlocal approximations.

❸ If all models have converged, then stop. If not, update GSE matrices, and goto Step ❶b.

In effect, each player uses an approximation of the coupled equations of the other players. This is the essence of approximate cooperation in design. In Figure 5.5, the schematic for the implementation of approximate cooperation in the context of the compromise DSP is shown. Step 1 is performed completely within the compromise DSP formulation of each

player. Once nonlocal approximations have been made, the ALP Algorithm is used to solve the model in step 2. It is important to note that since derivatives are used to approximate the variables, discrete or integer variables are not used in this protocol. In step 3, convergence of each player is checked, and if met, the solution is found.

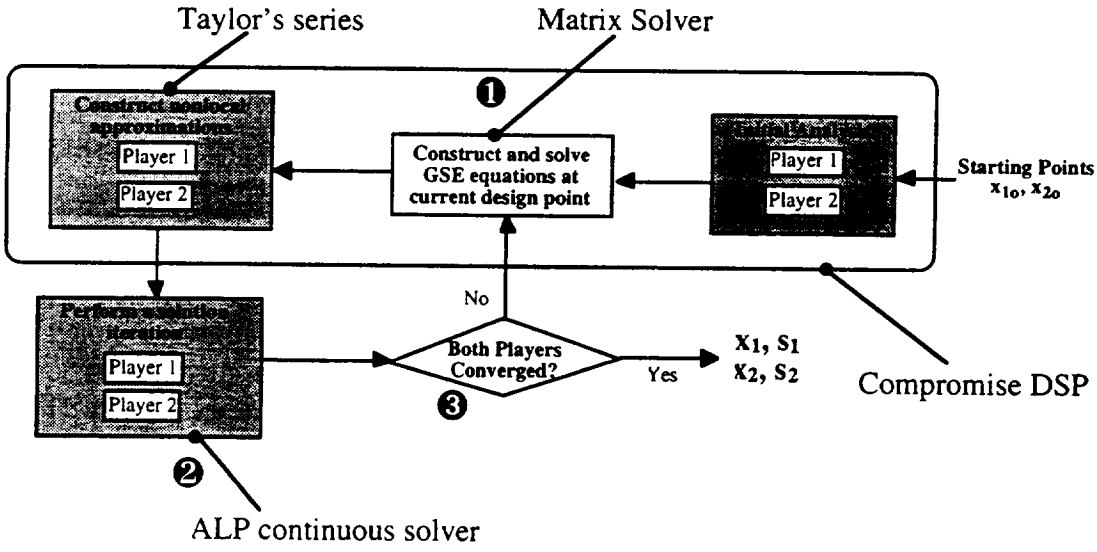


Figure 5.5 Construction and Solution of the Approximate Cooperation Formulation

In Figure 5.6, the compromise DSPs of two players in the approximate cooperative game formulation are shown. The values of the required nonlocal design variables, **x**, are used, along with approximations of the nonlocal state variables, **s**. With these representations, Player I is able to solve his compromise DSP using x_{II} and an approximation of s_{II} as part of his *given* information (left side of Figure 5.6), and vice versa for Player II (right side of Figure 5.6).

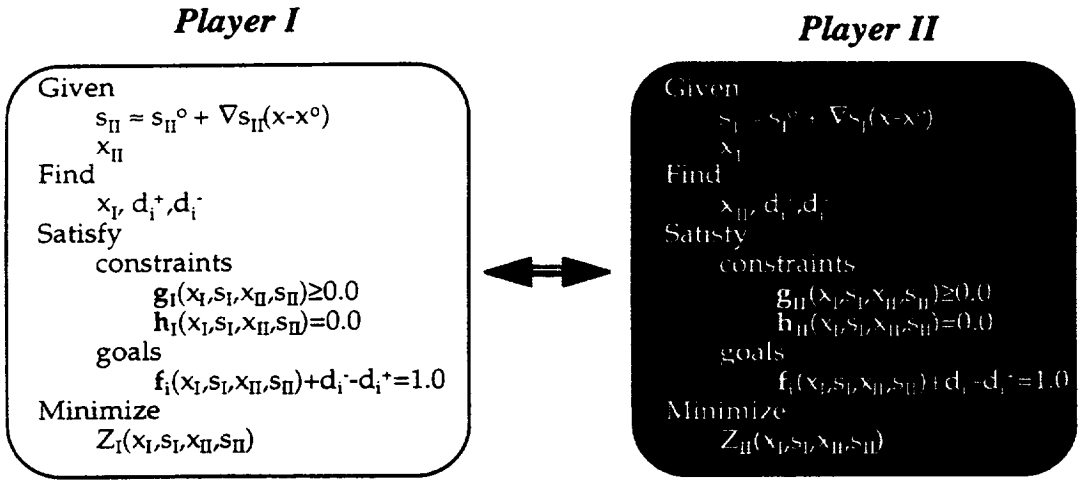


Figure 5.6 Compromise DSPs in Approximate Cooperation

5.5.2 Nash or Noncooperative solutions

The Nash or noncooperative formulation occurs when coalition among players is not possible due to organizational, information, or process difficulties. Players in design usually would not choose to "not cooperate" but because of the lack of information, the scenario can be modeled using noncooperative notions. This is often the case in designing large systems, and when players act independently and must make assumptions concerning the other players' actions. This is also the case when *information availability* plays a role in a design process. If the appropriate information is not available to the designers, assumptions will have to be made. To ensure a functional and safe design, designers often must construct a set of solutions according to any decision the other players make. In other words, designers usually have to assume worst case scenarios of the other players. This parallels the Nash formulation where players do not cooperate and must make decisions assuming the other decision makers could make any decision.

Nash Solutions in the context of Complex Systems Design

Similar to the cooperative protocol where ideal and complete cooperation is not realistic in the design of complex systems, finding the exact mathematical RRS of the two players is

not practical. Each player's model consists of multiple, nonlinear constraints and objectives that require advanced nonlinear programming and/or heuristic techniques in order to solve for the independent variables. To develop a closed form equation for one or more independent variables as functions of other independent variables is computationally difficult and theoretically extremely laborious. Therefore, the RRS of each player is found by using approximation techniques. Specifically, the RRS of each player is approximated by using design of experiments and second-order response surfaces. A response surface equation is used to approximate

$$\mathbf{X}^{1N} = f(\mathbf{X}^2) \quad \text{(5.15)}$$

as

$$\mathbf{x}_1, \mathbf{s}_1 = f(\mathbf{x}_2, \mathbf{s}_2) = \mathbf{A} \cdot \mathbf{x}_2 + \mathbf{B} \cdot \mathbf{s}_2 + \mathbf{C} \cdot (\mathbf{x}_2 \times \mathbf{x}_2) + \mathbf{D} \cdot (\mathbf{s}_2 \times \mathbf{s}_2) + \mathbf{E} \cdot (\mathbf{x}_2 \times \mathbf{s}_2) \quad \text{(5.16)}$$

In Eqn 5.16, the coupled design, x_1, and state, s_1, variables of player 1 are approximated as functions of the required design, x_2, and state, s_2, variables of player 2. A response surface is constructed for each variable, design and state, which is needed by another player. The steps to construct the RRS of each player is as follows.

Constructing the Rational Reaction Set

❶ Use NORMAN® as the design of experiments driver.

1a) Based on the number of input variables, set-up the Central Composite Face-Centered Design.

1b) Set the input variables (variables of the other player which are required) constant in P1's compromise DSP and call DSIDES.

❷ In DSIDES solve P1's compromise DSP using the *ALP Algorithm* and send the values of the design and state variables to NORMAN®

❸ Determine if the full experiment is finished.

- If not, continue by moving to the next experiment point and repeat Step ❷.
- If so, construct the response surfaces.

It is stressed that the ALP Algorithm is used to solve a compromise DSP at each simulation point. Even though the compromise DSP model may contain discrete and continuous

design variables (as illustrated in Chapter 7), to construct the RRS's the design variables are assumed to *all be continuous*. Creating response surfaces of functions of discrete variables is a difficult, if even feasible, task. To illustrate schematically, the specific steps to construct the RRS approximation for P2 are shown in Figure 5.7. In step 1, based on the range of the variables from P1, NORMAN® is used to construct an experimental design to sample the design space of P1. In step 2, the ALP Algorithm in DSIDES is called to solve P2's compromise DSP at each hypothetical point in P1's design space. In step 3, P2's solution information is used to construct response surfaces equations of P2's control variables as function of P1's control variables needed by P2. So, in effect, NORMAN® is used to build a function that embodies how P2 will behave for any value of P1's variables by sampling the design space as defined by the variable ranges and solving P2's model throughout the space.

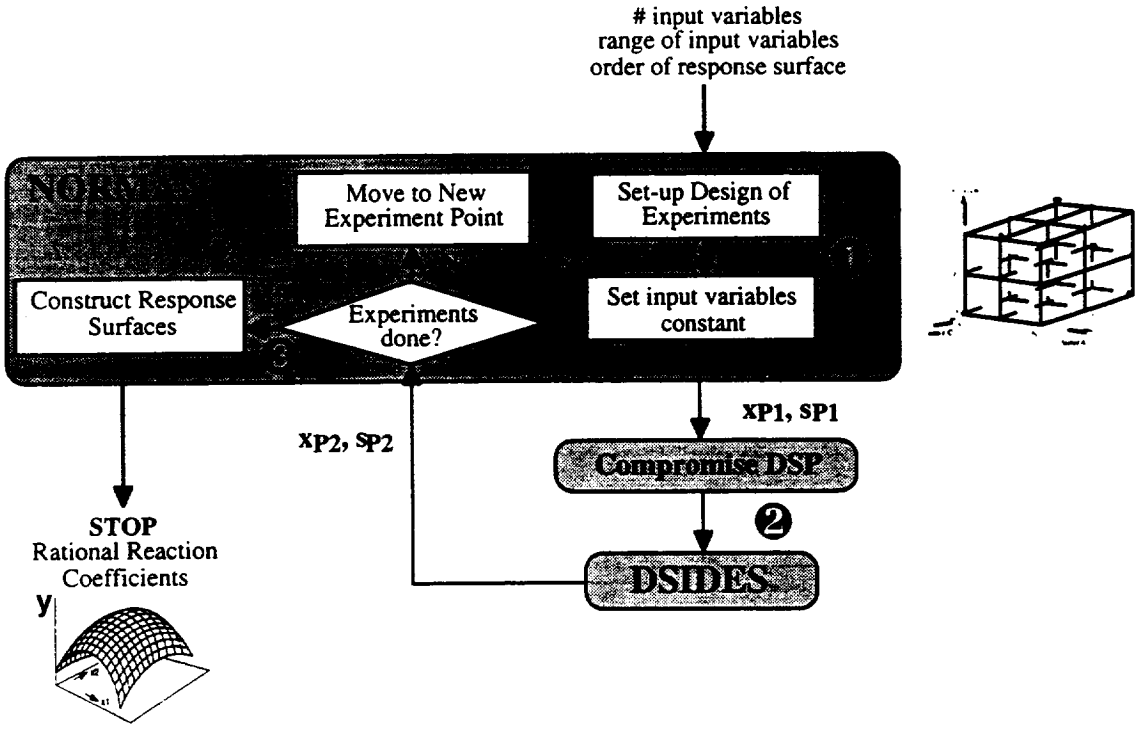


Figure 5.7. NORMAN/DSIDES Interface

Conceptually, the idea of constructing an approximate rational reaction set is illustrated in Figure 5.8. The aim is to construct the RRS of player 2, P2. Therefore, in step 1, points are taken from player one, P1's, design space (defined by x_{P1} and s_{P1}) through the CCD experimental design. In step 2, at each of these points (x_{P1}, s_{P1}), P2's compromise DSP is solved using the numerical values of (x_{P1}, s_{P1}) as input parameters. Then in step 3, the resulting set of solutions, [(x_{P2}, s_{P2})] are taken as the output parameters to construct response surface equations of the form x_{P2}, s_{P2} = f(x_{P1}, s_{P1}) which approximate the rational reaction set of P2.

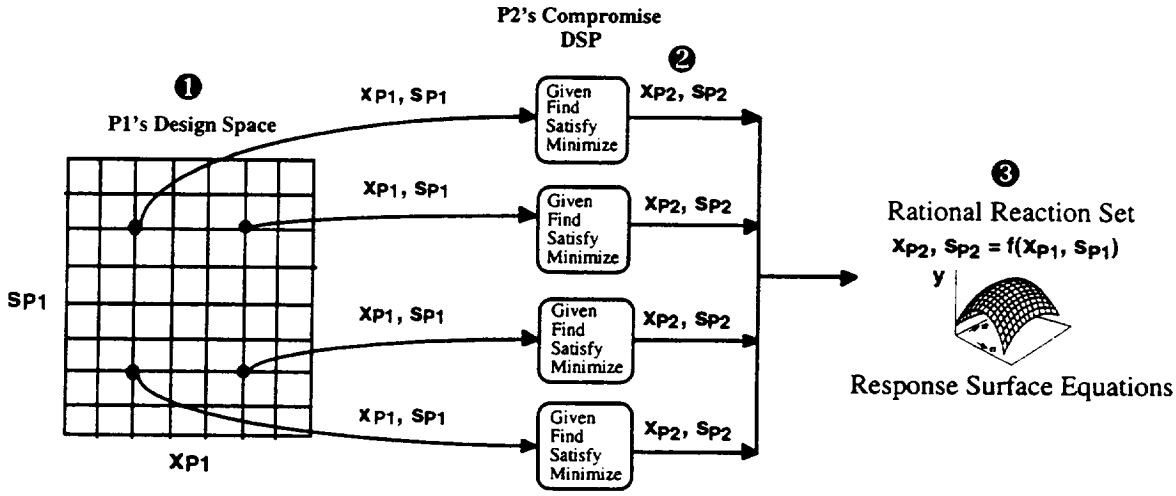


Figure 5.8 Conceptual Outline of RRS Construction

In Figure 5.9, compromise DSPs of two players in a noncooperative formulation are shown. The given information is that information required from the other player is *unknown*. Therefore, a player's solution must be found using unknown variables. Constructing an approximation of the RRS of each player is found using the compromise DSPs of Figure 5.9. Construction of the RRS's is the first step to solving the noncooperative game formulation. The steps to solve the noncooperative formulation are as follows.

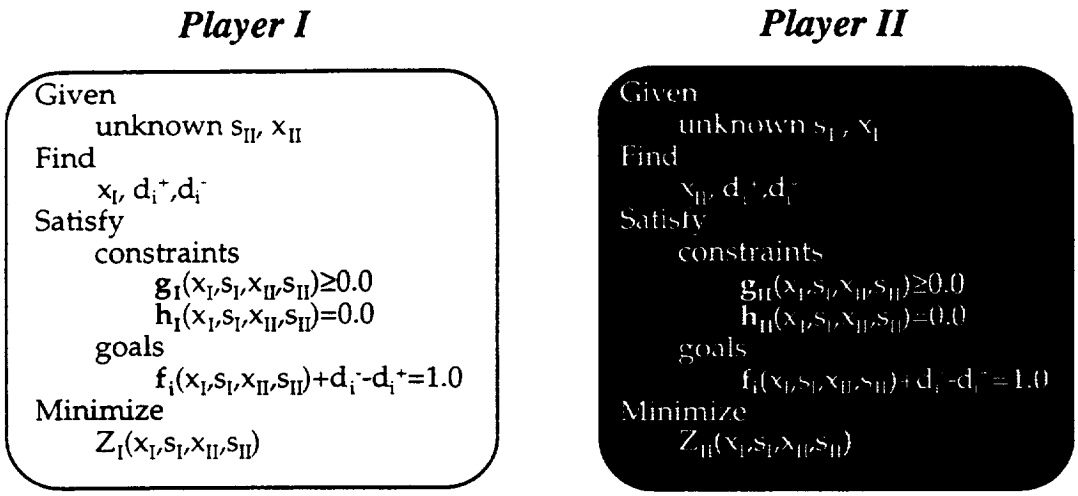


Figure 5.9. Noncooperative Compromise DSPs

Solution to the Noncooperative Protocol

❶ Construct Rational Reaction Set of Each Player, $\mathbf{D_i}$

❷ Using appropriate technique, find the intersection points of the RRS's of each player.

$$\mathbf{X^* = D_1 \cap D_2} \tag{5.17}$$

❸ Determine which solutions fall in the ranges of the design variables.

Finding the rational reaction sets of a player is paramount to game theory not only in the noncooperative protocol, but also in the Stackelberg Leader/Follower protocol.

5.5.3 Stackelberg Leader/Follower solutions

In the Stackelberg leader/follower formulation (presented in Section 3.3.3), the leader may be able to use knowledge of the followers' response to his advantage in minimizing his own deviation function. The followers may also benefit from having a leader in that they do not have to guess what the leader will do. Therefore, neither the leader nor the followers necessarily have an advantage. This behavior of the follower is dictated by his strategy and embodied by his rational reaction set, which describes how the follower will behave in response to *any* decision made by the leader.

Leader/Follower Solutions in the Context of Complex Systems Design

Again, in the Stackelberg protocol, the rational reaction sets are crucial for finding the leader/follower solution. The rational reaction sets are constructed using the same procedure as in Section 5.5.2. The process for solving the leader/follower formulation is as follows.

❶ Construct the RRS of the follower.
❷ Allowing the leader access to the follower's RRS in the leader's compromise DSP, solve the leader's compromise DSP.
❸ Allowing the follower access to the leader's solution, solve the follower's compromise DSP.

In Figure 5.10, the steps to solve the leader/follower formulation are shown schematically. Each players' compromise DSP can be solved using the Foraging-directed Adaptive Linear Programming Algorithm (see Chapter 6), since the players' models may contain discrete and continuous design variables.

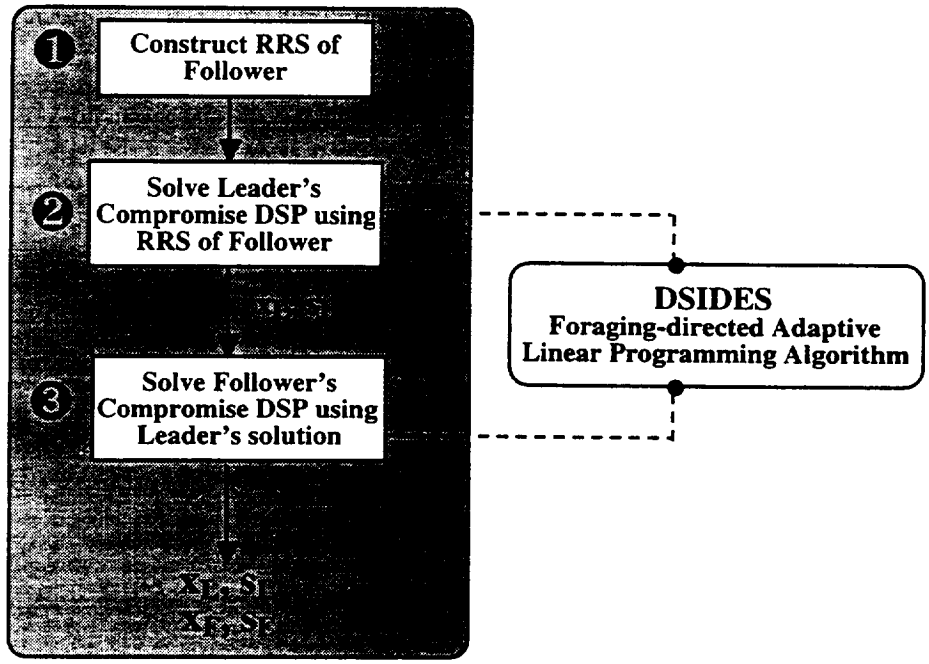


Figure 5.10. Leader/Follower Solution Process

In Figure 5.11, the compromise DSPs of the leader and follower are shown. The leader's given information includes the RRS of the follower, while the follower's given information includes the leader's control variable solution, $\mathbf{x_L}$ and $\mathbf{s_L}$.

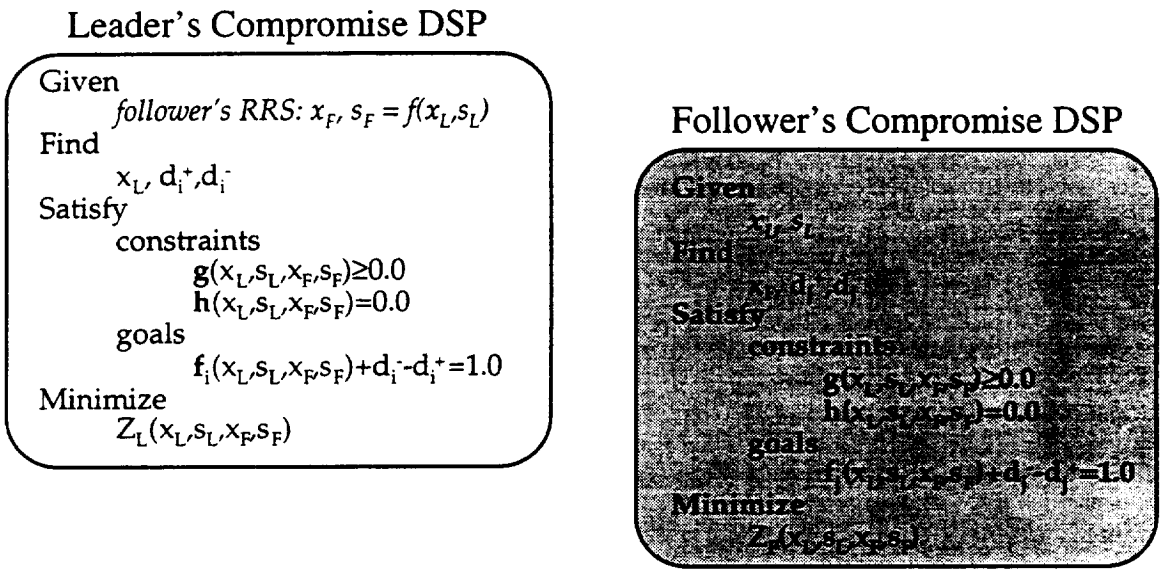


Figure 5.11. Leader/Follower Compromise DSPs

The implementation of the three protocols is studied and verified in the next section using the design of a pressure vessel as the verification study.

5.6 VERIFICATION STUDY: THE DESIGN OF A PRESSURE VESSEL

As a simple verification study, the design of a thin-walled pressure vessel which has hemispherical ends as shown in Figure 5.12 is used. The nomenclature for this example is presented in Table 5.3. This case study is derived from the example presented in Section 4.3 and studied in (Karandikar and Mistree, 1992b, Rao, et al., 1996). The design

variables are the radius R, the length L, and the thickness T. The vessel is to withstand a specified internal pressure P and the material is also specified. There are two objectives: to minimize the weight and to maximize the volume of the cylinder, both subject to stress and geometry constraints. It is recognized that this example is not naturally a multi-player problem, and that a single-player multiobjective formulation is normally used. A multi-player formulation is used in this chapter to verify the game theoretic developments of this dissertation. Two players are used: 1) player VOL who wishes to maximize the volume and thus controls R and L, and 2) player WGT who wishes to minimize the weight of the vessel and controls T.

Table 5.3. Nomenclature for the Pressure Vessel Example

W	Weight of the pressure vessel, lbs.
V	Volume, $in.^3$
R	Radius, in.
T	Thickness, in.
L	Length, in.
P	Pressure inside the cylinder, Klb.
St	Allowable tensile strength of the cylinder material, Klb.
ρ	Density of the cylinder material $lbs./in.^3$
σ_{circ}	Circumferential stress $lbs./in.^2$
TV	Target Value for a goal

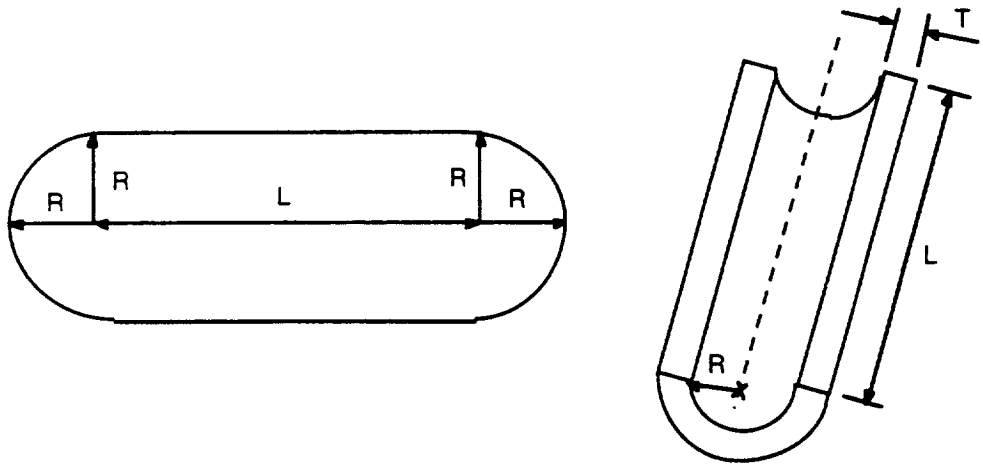


Figure 5.12. Thin-Walled Pressure Vessel

The compromise DSPs of the two players are shown below:

PLAYER WGT

Given

Weight: $W(R,T,L) = \rho\left[\frac{4}{3}\pi(R+T)^3 + \pi(R+T)^2 L - (\frac{4}{3}\pi R^3 + \pi R^2 L)\right]$

Find

Design Variable: T

Overachievement Deviation Variable associated with the weight goal, d_W^+

Satisfy

Stress constraint: $\sigma_{circ} = \frac{PR}{T} \leq S_t$

Geometric constraints: $5T - R \leq 0$

$R + T - 40 \leq 0$

$L + 2R + 2T - 150 \leq 0$

Bounds: $T_l \leq T \leq T_u$

Weight Goal: $W - d_W^+ = W_{TV}$

Minimize

d_W^+

PLAYER VOL

Given

Volume $V(R,L) = \left[\frac{4}{3}\pi R^3 + \pi R^2 L\right]$

Find

Design Variables: R and L

Underachievement Deviation Variable associated with the volume goal, dv^-

Satisfy

Stress constraint: $\sigma_{circ} = \frac{PR}{T} \leq S_t$

Geometric constraints: $5T - R \leq 0$

$R + T - 40 \leq 0$

$L + 2R + 2T - 150 \leq 0$

Bounds: $R_l \leq R \leq R_u$

$L_l \leq L \leq L_u$

Volume Goal: $V + dv^- = V_{TV}$

Minimize

dv^-

The specific data (problem constants) for this problem is given in Table 5.4.

Table 5.4. Pressure Vessel Parameters

P	3.89 klb
S_t	35.0 klb
ρ	0.283 lbs/in^3
L_l	0.1 in.
L_u	140.0 in.
R_l	0.1 in.
R_u	36.0 in.
T_l	0.5 in.
T_u	6.0 in.
W_{TV}	0 lbs.
V_{TV}	775,000 in^3

This example is studied as a multi-player formulation in (Rao, et al., 1996). Since this example is quite simple, exact analytical solutions for different protocols are found in (Rao, et al., 1996). The motivation in this chapter is the notion of *complex* systems such as

aircraft, automobiles, and ships. In other words, the developments of this chapter would typically not be used to design the pressure vessel studied in this section. The pressure vessel is *only* being used to verify the developments presented in this chapter. The work presented in this chapter, as illustrated in Chapter 7, are primarily applicable to complex systems, but certainly could be applied to smaller, less complex systems as well. The developments in this chapter include strategies to approximate constructs and solutions in game theory. Therefore, the approximated solutions are compared to the exact results from (Rao, et al., 1996) as a means to verify the developments of this chapter in the context of game theory in complex systems design.

5.6.1 The Cooperative Formulation

In (Rao, et al., 1996), the cooperative or Pareto solutions are found symbolically. That is, the equations are simple enough to find an analytical set of equations describing the set of Pareto solutions. In the compromise DSP, a *single numerical* solution is given. Therefore, to generate the set of Pareto solutions, multiple compromise DSPs are run for various values of W_1 and W_2, where W_i is the weight corresponding to the deviation function of Player i. The two players' compromise DSPs are combined into one, and the overall goal becomes:

$$W_1\rho\left[\frac{4}{3}\pi(R+T)^3 + \pi(R+T)^2 L - (\frac{4}{3}\pi R^3 + \pi R^2 L)\right] + W_2\left[\frac{4}{3}\pi R^3 + \pi R^2 L\right] \quad (5.18)$$

and the deviation functions becomes

$$Z = W_1 * d_W^+ + W_2 * d_V^-$$

where

$$W_1 + W_2 = 1 \text{ and } 0 \le W_i \le 1$$

To replicate the symbolic solutions in (Rao, et al., 1996), various values of the W's are used in the cooperative compromise DSP. In (Rao, et al., 1996), the extreme points of the Pareto solution set are given as

$$(R,T,L)=\left(\frac{40S_t}{P+S_t},\frac{40P}{P+S_t},70\right),(5T_l,T_l,L_l). \tag{5.19}$$

These two solutions are replicated using $W_1 = 0.0$ and $W_2 = 1.0$, and $W_1 = 1.0$ and $W_2 = 0.0$, respectively in the compromise DSP. The two numerical solutions from the two weighting schemes are:

$$(R,T,L)=(36,4,70),(2.5,0.5,0.1)$$

$$(Weight,Volume)=(39475\ lbs,480385\ in^3),(13.73\ lbs,67.41\ in^3).$$

These two solutions correspond to Eqn. 5.19 when the specific input parameter values from Table 5.4 are used. Three starting points are used for each case, each converging to the same solution, as shown in Appendix A. Varying W_1 and W_2 on the interval [0,1] results in a set of Pareto solutions corresponding to the cooperative protocol. These solutions correspond to the solutions identified on the interval in Eqn. 5.19. Therefore, using the compromise DSP and the ALP Algorithm, the set of exact Pareto solutions reported in (Rao, et al., 1996) is replicated. The other two protocols are now studied for the same problem. In each protocol, the fundamental mathematical construct is the Rational Reaction Set.

5.6.2 The Noncooperative Formulation

As discussed in Section 3.3.3 the noncooperative solution occurs at the intersection of the players' Rational Reaction Sets. The Rational Reaction Sets of the two players from (Rao, et al., 1996) are:

$$\text{DVOLUME:}\quad \begin{aligned} R(T) &= \min\left\{40 - T, \frac{TS_t}{P}\right\} \\ L(T) &= 150 - 2T - 2R(T) \end{aligned} \tag{5.20}$$

$$\text{DWEIGHT:}\quad T(R,L) = \begin{cases} \dfrac{PR}{S_t} & \text{if } R \le \frac{40S_t}{P+S_t} \text{ and } L + 2R\left(1 + \frac{P}{S_t}\right) \le 150 \\ () & \text{otherwise} \end{cases} . \tag{5.21}$$

Using the process described in Section 5.5.2 to approximate the RRS of each player using NORMAN and DSIDES, the RRS of each player are approximated as:

$$\begin{aligned} \text{DVOLUME:}\quad & R(T) = 29.29 + 14.75*T - 10.01*T^2 \\ & L(T) = 85.45 - 34.45*T + 20.10*T^2 \end{aligned} \tag{5.22}$$

$$\begin{aligned} \text{DWEIGHT:}\quad & T'(R,L) = 2 + 1.75*R - 2.267*10^{-5}*L + 3.15*10^{-5}*R*L + 0.2445*R^2 + \\ & 8.667*10^{-7}*L^2 \end{aligned} \tag{5.23}$$

Again, the problem parameters from Table 5.4 are used in constructing these RRS. One of the most striking differences in the two sets of RRS is the representation of L(T). In (Rao, et al., 1996), it assumed that Player VOL does not know the value of R at any given point, therefore, L(T) is not only a function of T, but also a function of R. In constructing the approximate RRS, a compromise DSP is solved for various values of (theoretically unknown) T, but the values of the local variables, R and L are known. Therefore, the approximate L(T) is only a function of T. Another subtle difference is that the independent variables in the approximate RRS (R(T), L(T), and T(R,L)) are normalized from their original ranges to [-1,1] in order to perform the experimental design and analysis.

In order to compare the accuracy of the approximate RRS, both sets of RRS are plotted. In Figures 5.13, and 5.14, the exact Rational Reaction Set of Player VOL is plotted (Eqn. 5.20).

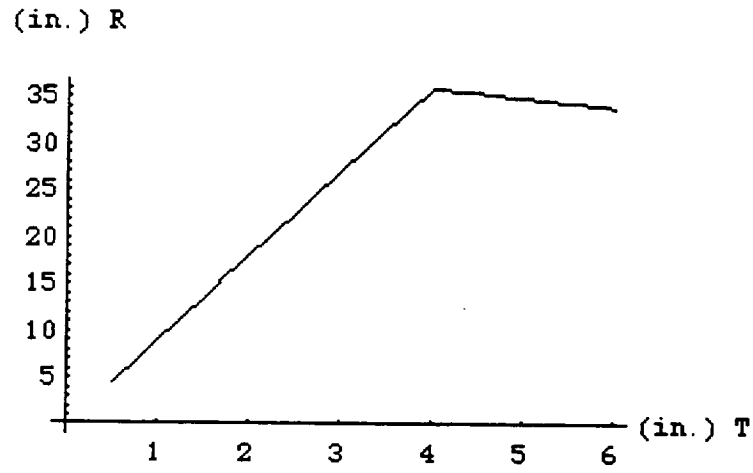


Figure 5.13. R as a Function of T

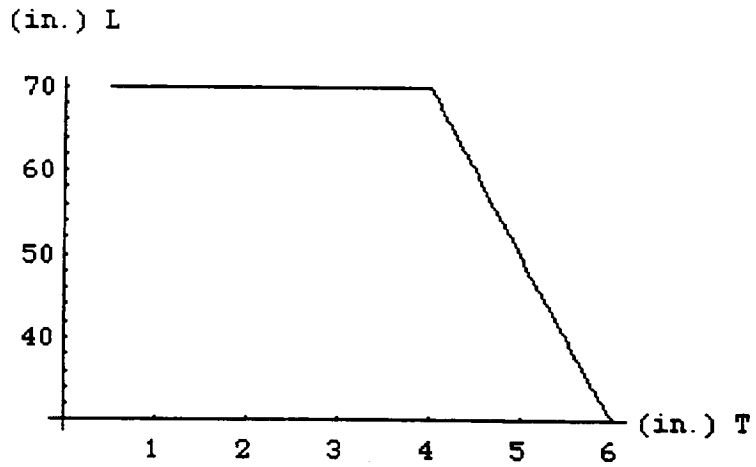


Figure 5.14. L as a Function of T

In Figures 5.15 and 5.16, the approximate RRS of Player VOL is plotted.

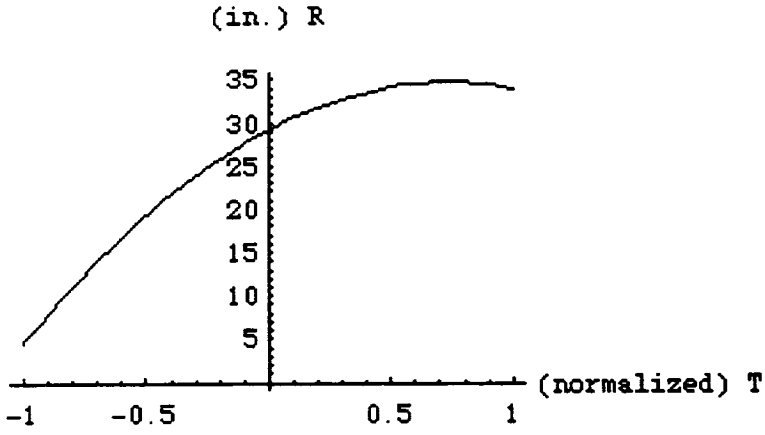


Figure 5.15. R as a Function of T: Approximation

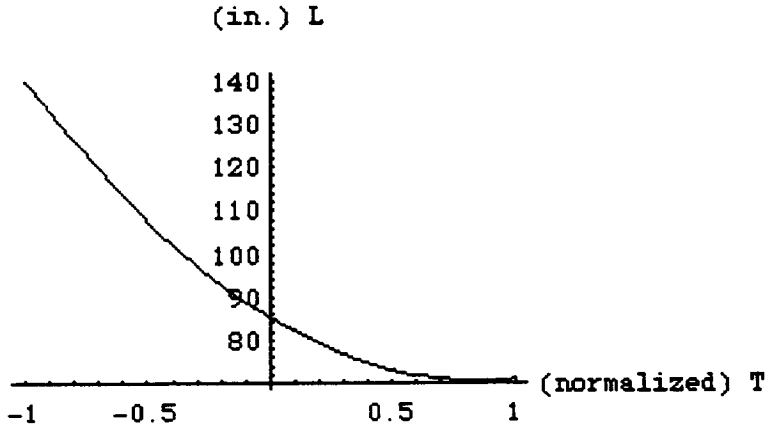


Figure 5.16. L as a Function of T: Approximation

Comparing Figures 5.13 and 5.15 (the x-axis is different because of the normalized values in Figure 5.15), it is clear that the second order approximation of R in Figure 5.15 (Eqn. 5.22) is an accurate representation of Figure 5.13 (Eqn. 5.20). The function in Figure 5.15 is a smooth, continuous function, while the one in Figure 5.13 is continuous but *not* smooth. The derivative of the function does not exist at the sharp corner. Therefore, many optimization algorithms will have difficulty handling this type of function, but would have

no trouble handling the representation of R in Eqn. 5.20. Using second order approximations of functions in game theory models and solutions is an interesting area for future research.

The plots of L in Figures 5.14 and 5.16 are quite different. This is primarily due to the representation of R as a *symbolic* variable in the equation for L in Eqn. 5.20, while in the approximation of L in Eqn. 5.22, R is *numerically* known. Therefore, the two functions do look different, but as is shown in Sections 5.6.2 and 5.6.3, both rational reaction sets produce the same noncooperative and leader/follower solutions. The exact RRS of Eqn. 5.20 is correct from a theoretical perspective, but from a practical design perspective, the designers do know and control the values of the local design variables. Therefore, the approximate RRS is more pragmatic.

The WGT player is now explored. The exact RRS of Player WGT (Eqn 5.21) is plotted in Figure 5.17

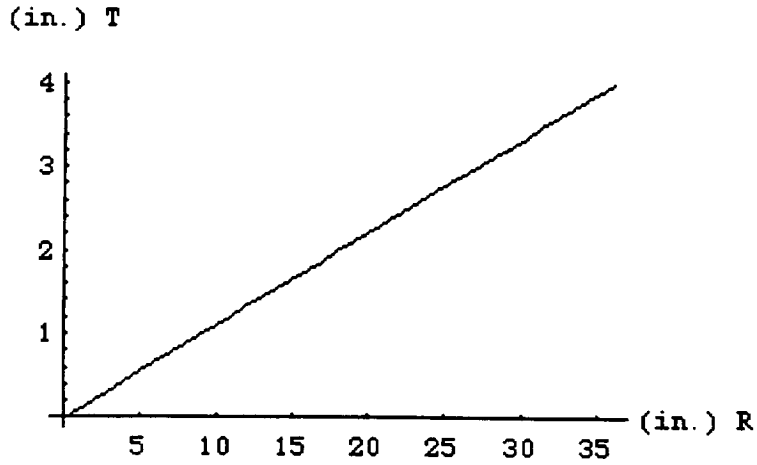


Figure 5.17. T as a Function of R

The approximate RRS of Player WGT (Eqn. 23) is plotted in Figure 5.18.

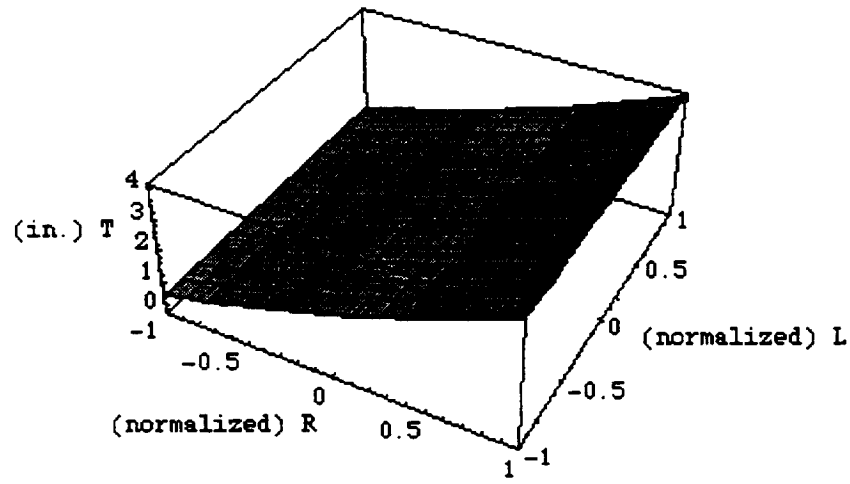


Figure 5.18. T as a Function of R and L: Approximated

The striking difference is the dimension of the two plots. The exact RRS in Figure 5.17 and Eqn. 21 is only a function of R, as L can be ignored due to active constraints and monotonicity arguments (Rao, et al., 1996). However, in the process to approximate the RRS, the players' models are typically very complex and simple active constraints and monotonicity arguments are difficult to recognize and compute. Therefore, *every* nonlocal variable is assumed to be significant in approximating the local variables. For this reason, T in Figure 5.18 is shown to be a function of R and L. With closer inspection, the exact RRS of Eqn. 5.21 and Figure 5.17 can be recovered from Figure 5.18. T is constant with respect to L, but increases with respect to R. In other words, T does not depend upon L which is exactly what is implied by Figure 5.18. Furthermore, Figure 5.17 is simply a "slice" of Figure 5.18 for *any* value of T. In Figure 5.19, a "slice" of Figure 5.18 for a constant T is shown. The similarity between Figures 5.17 and 5.19 is easily observed (the x-axis in Figure 5.19 is again normalized).

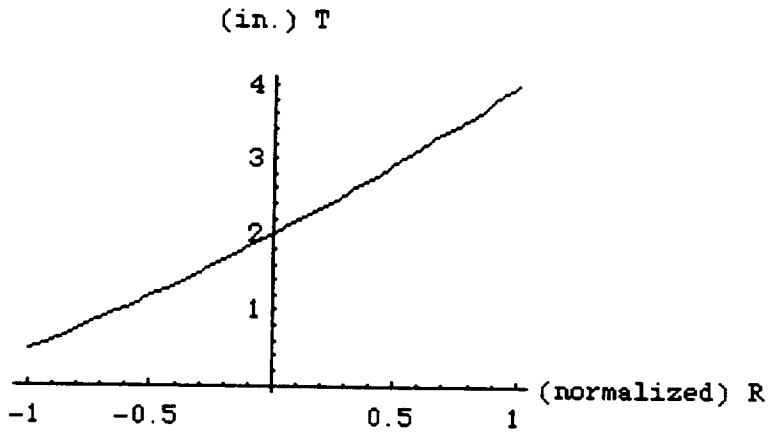


Figure 5.19. T as a Function of only R: Approximated

In Eqn. 5.23, the coefficients of the terms that include L are very small relative to the coefficients of the terms with R. This further verifies the approximation of the RRS, and validates its implementation in design.

The exact noncooperative solution occurs at the intersection of Eqns. 5.20 and 5.21. In (Rao, et al., 1996), the intersection lies along the line represented by

$$\begin{gathered}\frac{S_t(150-L_u)}{2(P+S_t)} \le R^N \le \frac{40S_t}{P+S_t} \\ L^N := 150 - 2R^N\left[\frac{P}{S_t}+1\right] \\ T^N := \frac{PR^N}{S_t}\end{gathered} \quad . \tag{5.24}$$

Using Mathematica® to solve Eqns. 5.22 and 5.23, the approximate noncooperative solution is:

$$\begin{gathered}R^N = 28.4 \text{ in.} \\ L^N = 86.9 \text{ in.} \\ T^N = 3.16 \text{ in.} \\ \text{Weight} = 24746 \text{ lbs.} \\ \text{Volume} = 316110 \text{ in}^3.\end{gathered} \tag{5.25}$$

It is readily verified that this is a solution along the line in (R, L, T) space described by Eqn. 5.24 (using the constant parameters in Table 5.4). This further verifies the approach described in Section 5.5.2 to approximate the RRS of a player, as the solution using the approximated RRS's (Eqn. 5.25) is a member of the solution set using exact RRS's (Eqn. 5.24).

5.6.3 The Leader/Follower Formulation

Player WGT as the Leader

In this protocol, player WGT dictates his strategy first by assuming or dictating that the player VOL behaves in a predetermined or rational way (i.e., player VOL must minimize his deviation function for a given thickness, T). The compromise DSP for the leader, WGT is

Given
Rational Reaction Set from VOL: {R=f(T), L=f(T)}
Weight: $W(R,T,L) = \rho\left[\frac{4}{3}\pi(R+T)^3 + \pi(R+T)^2 L - (\frac{4}{3}\pi R^3 + \pi R^2 L)\right]$
Find
Design Variable: T
Overachievement Deviation Variable associated with the weight goal, d_W^+
Satisfy
Stress constraint: $\sigma_{circ} = \frac{PR}{T} \le S_t$
Geometric constraints: $5T - R \le 0$
$R + T - 40 \le 0$
$L + 2R + 2T - 150 \le 0$
Bounds: $T_l \le T \le T_u$
Weight Goal: $W - d_W^+ = W_{TV}$
Minimize
d_W^+

where (R,L) is the solution to the follower's (VOL) problem, given by

Given
T unknown

Volume: $V(R,L) = \left[\frac{4}{3}\pi R^3 + \pi R^2 L\right]$

Find
Design Variables: R and L
Underachievement Deviation Variable associated with the weight goal, d_V^-

Satisfy

Stress constraint: $\sigma_{circ} = \frac{PR}{T} \le S_t$

Geometric constraints: $5T - R \le 0$
$R + T - 40 \le 0$
$L + 2R + 2T - 150 \le 0$

Bounds: $R_l \le R \le R_u$
$L_l \le L \le L_u$

Volume Goal: $V + d_V^- = V_{TV}$

Minimize
d_V^-

In other words, the follower constructs his Rational Reaction Set which the leader can use to solve his compromise DSP. The RRS of Player VOL is given by Eqns. 5.20 and 5.22, exact and approximate, respectively. In (Rao, et al., 1996) the solution of the leader/follower problem with WGT as the leader is reported as

$$(R,T,L) = \left(\frac{T_l S_t}{P}, T_l, 150 - 2T_l(1 + \frac{S_t}{P})\right). \tag{5.26}$$

Using the constant and parameters values (see Table 5.4), this corresponds to a solution of

$$(R, T, L) = (4.5 \text{ in.}, 0.5 \text{ in.}, 140 \text{ in.}). \tag{5.27}$$

Using the approximation of the RRS of Player VOL, and the process described in Section 5.5.3 for solving leader/follower problems in the context of the compromise DSP, the leader's solution is

$$T = 0.5 \text{ in.}$$
$$\text{Weight} = 635.6 \text{ lbs} \tag{5.28}$$

Using the leader's solution, the follower's solution is predetermined from its strategy dictated in its RRS. In other words, the follower cannot change his strategy which has been used by the leader. In the context of the compromise DSP, it would be pointless and

absurd for a designer to decide to "maximize the deviation function." Therefore, the assumption in game theory for the follower to not change strategies has a natural implication in design: a designer does not change his value structure which is inherent in his model. Player VOL's solution as follower (dictated by his RRS) is

$$\begin{aligned} R &= 4.53 \text{ in.} \\ L &= 140.0 \text{ in.} \\ \text{Volume} &= 9413.9 \text{ in}^3. \end{aligned} \tag{5.29}$$

So, the total approximate solution with WGT as the leader is

$$(R, T, L) = (4.53 \text{ in.}, 0.5 \text{ in.}, 140 \text{ in.}) \tag{5.30}$$

which is very close to the exact solution in Eqn. 5.27. The complete results for this formulation are given in Appendix A.

Player VOL as the Leader

In this protocol, player VOL dictates his strategy first by assuming or dictating that the player WGT behave in a predetermined or rational way (i.e., player WGT must minimize his deviation function for a given radius and length, R and L). The compromise DSP for the leader, player VOL is

Given
- Rational Reaction Set from WGT: {T=f(R,L)}
- Volume: $V(R,L) = \left[\frac{4}{3}\pi R^3 + \pi R^2 L\right]$

Find
- Design Variables: R and L
- Underachievement Deviation Variable associated with the weight goal, d_V^-

Satisfy
- Stress constraint: $\sigma_{circ} = \frac{PR}{T} \le S_t$
- Geometric constraints: $5T - R \le 0$; $R + T - 40 \le 0$; $L + 2R + 2T - 150 \le 0$
- Bounds: $R_l \le R \le R_u$; $L_l \le L \le L_u$
- Volume Goal: $V + d_V^- = V_{TV}$

Minimize
- d_V^-

where (R,L) is the solution to the follower's (WGT) problem, given by

Given
Unknown R and L

Weight: $W(R,T,L) = \rho\left[\frac{4}{3}\pi(R+T)^3 + \pi(R+T)^2 L - (\frac{4}{3}\pi R^3 + \pi R^2 L)\right]$

Find
Design Variable: T
Overachievement Deviation Variable associated with the weight goal, d_W^+

Satisfy

Stress constraint: $\sigma_{circ} = \frac{PR}{T} \le S_t$

Geometric constraints: $5T - R \le 0$
$R + T - 40 \le 0$
$L + 2R + 2T - 150 \le 0$

Bounds: $T_l \le T \le T_u$

Weight Goal: $W - d_W^+ = W_{TV}$

Minimize
d_W^+

The RRS of Player WGT is given by Eqns. 5.21 and 5.23, exact and approximate, respectively. In (Rao, et al., 1996) the solution of the leader/follower problem with VOL as the leader is reported as

$$(R,T,L) = \left(\frac{40S_t}{P+S_t}, \frac{40P}{P+S_t}, 70\right). \tag{5.31}$$

Using the constant and parameters values, this corresponds to a numerical solution of

$$(R, L, T) = (36.0 \text{ in.}, 70.0 \text{ in.}, 4.0 \text{ in.}). \tag{5.32}$$

Using the approximation of the RRS of Player WGT, and the process described in Section 5.5.3 for solving leader/follower problems in the context of the compromise DSP, VOL's solution as the leader's is

$$\begin{aligned} R &= 36.0 \text{ in.} \\ L &= 71.1 \text{ in.} \\ \text{Volume} &= 39772.4 \text{ in}^3 \end{aligned} \tag{5.33}$$

Using the leader's solution, the follower's solution is predetermined from its strategy dictated in its RRS. Player WGT's solution as follower (dictated by his RRS) is

$$T = 4.0 \text{ in.}$$

$$\text{Weight} = 484863.0 \text{ lbs.} \tag{5.34}$$

So, the total solution with VOL as the leader is

$$(R, L, T) = (36.0 \text{ in.}, 71.1 \text{ in.}, 4.0 \text{ in.}) \tag{5.35}$$

which is very close to the exact solution in Eqn. 5.32. The total results for this formulation are given in Appendix A.

In this section, the primary interest is to verify the game theoretical developments and associated posits that are used to support Hypothesis III of this dissertation. It is shown that for a well-studied simple problem where exact game theoretical solutions are known, the techniques presented in Section 5.5 are valid and effective.

- The set of Pareto cooperative solutions are reproduced by formulating the pressure vessel problem using two compromise DSPs and solving the cooperative formulation using the ALP Algorithm.
- The techniques to approximate the Rational Reaction Sets of each player using second order response surfaces are verified by: 1) graphical comparison, and 2) numerical verification by reproducing the exact leader/follower and noncooperative solutions found in (Rao, et al., 1996).

Full results of each protocol are presented in Appendix A. Application to a complex, large scale system where *exact solutions are unknown* is presented in Chapter 7 using the design of a passenger aircraft.

5.7 A LOOK BACK AND A LOOK AHEAD

In this chapter, the foundation for applying game theory in complex systems design is developed and presented. The work in this chapter represents one of the primary building

blocks of this work. It supports Hypothesis III and Posits 2.1, 2.4, and 2.5 presented in Section 3.1.3. Posits 4 and 6 are verified in Chapter 7. The observations relating to each posit are discussed.

Posit 2.1: Design processes can be abstracted as games where the players are multiple designers or design teams and their associated analysis and synthesis tools.

In Section 3.3.2, the similarities between a game between multiple players and a design process consisting of multiple design teams are presented. In each case, a decision-maker must make a decision to satisfy his requirements and constraints. However, this decision is affected by the decisions made by the other decision-makers in the game or design process. The notion of noncooperation is the main difference between the two. In a game, there is usually a winner and loser. Noncooperation may help a single player win. But in design ideally everyone should cooperate. Noncooperation notions occur when design teams are separated and information from another team is not available to make a decision. In this case, assumptions must be made, and worst case scenarios are often assumed. This is the notion of noncooperation in design.

Posit 2.4: Second order response surfaces can be used to approximate the Rational Reaction Sets of the disciplinary players in a design game.

In Section 5.5.2, a process of approximating Rational Reaction Sets using second order response surfaces is presented. Rational Reaction Sets of players who have complex nonlinear decision models are difficult to construct and are unknown. Therefore, a second order model is prescribed to approximate the RRS. In Section 5.6.2, the efficacy of the second order representations of the RRS of two players in a simple case, where the exact RRS's are known, is shown. The resulting noncooperative and leader/follower solutions using the approximate RRS's match those using the exact RRS's. Therefore, there is evidence to suggest that a second order approximation is shown to be an effective representation of an RRS. However, to conclude that second order RSE's are adequate for all classes of problems is wrong. It has been shown for one example. Further mathematical analysis of certain classes of problems is required to make a broad statement concerning second order RRS approximations.

Posit 2.5: The compromise DSP can be used as the fundamental construct to develop the game theory protocols and techniques.

In Section 5.5, techniques to model game theoretical protocols in complex systems design are presented, each using the compromise DSP as the fundamental decision construct. In each protocol, the players use their compromise DSPs as the basis to support their decision. However, the interactions among the players change, so the information available to each player changes. According to the information available, each players' compromise DSP gets augmented accordingly, but the role of the compromise DSP as the core decision-making construct does not change.

With reference to Figure 5.20, this chapter is the second of the three primary building blocks detailing aspects of the overall algorithm introduced in Chapter 3. The role of Chapter 5 is to provide theoretical and computational support for the second step of the algorithm (see Section 3.3) where disciplinary problems are formulated based on the game protocol among the players. The game protocol is identified and established in step one of the algorithm (Chapter 4). In the next chapter, the final building block (step three), the solution of the game theoretical formulations developed in this chapter, is presented.

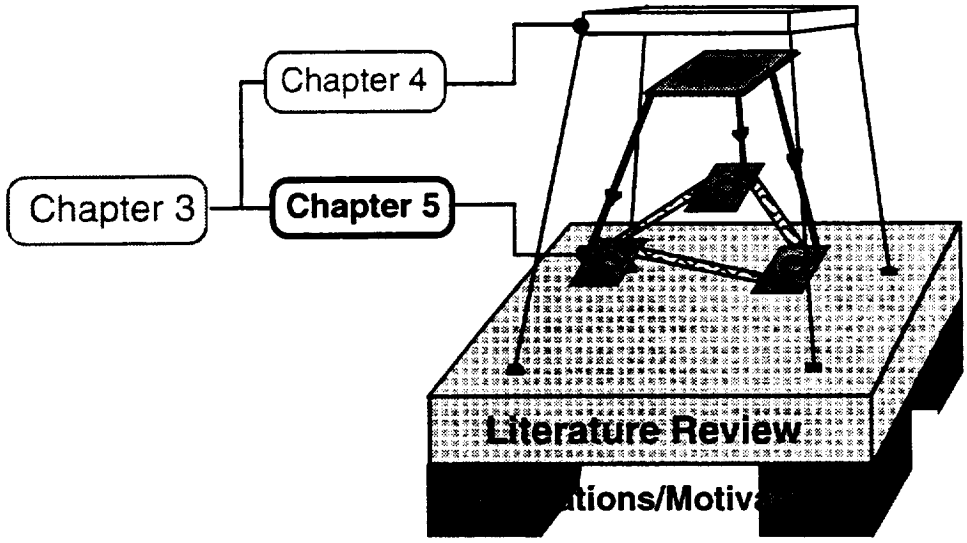


Figure 5.20. Frame of Reference: Chapter 5

CHAPTER 6

THE SOLUTION OF MIXED DISCRETE/CONTINUOUS DECISION SUPPORT PROBLEMS

Design models often contain a combination of discrete, integer, and continuous variables. Previously, the Adaptive Linear Programming (ALP) algorithm, which is based on sequential linearization, has been used to solve design models composed of continuous and Boolean variables. In this chapter, the ALP algorithm is extended to solve *subsystem models which consist of discrete and continuous variables*, is developed and verified. This new solution scheme is the vehicle with which *Step 3 of the algorithm of this dissertation* is performed. The work in this chapter supports Hypothesis III (Figure 1.7), and posits 3.1, 3.2, and 3.3 of Chapter 3, as shown below.

Hypothesis III: The notion of foraging of wild animals is a natural analogy for optimization and can be used as an effective search technique in the solution of mixed discrete/continuous models.

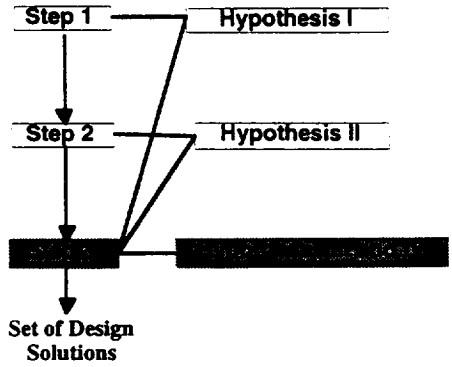
Posit 3.1: Foraging is a heuristic, under which characteristics from genetic algorithms, Tabu Search, and Simulated Annealing can be grouped.

Posit 3.2: The Tabu Search can be used as the building block for the foraging solution algorithm.

Posit 3.3: The ALP Algorithm along with foraging can be used to effectively solve mixed discrete/continuous problems.

The mixed discrete/continuous solution scheme involves extending the ALP Algorithm using a discrete heuristic based on the analogy of an animal foraging for food. This

solution scheme for mixed discrete/continuous design problems integrates ALP and the foraging search and is called Foraging-directed Adaptive Linear Programming Algorithm (FALP)[1]. In Section 6.1, a frame of reference is presented to establish the context of mixed discrete/continuous problems in design. In Section 6.2, the discrete heuristic, foraging, is detailed. In Section 6.3, the continuous solver, the ALP Algorithm is detailed. In Section 6.4, the mixed discrete/continuous scheme is developed. In Section 6.5, two design studies, the design of a compression spring and the design of a pressure vessel, are presented to illustrate the effectiveness and behavior of the solution scheme.



1 It is stressed that the FALP Algorithm is *only a portion of the overall algorithm* of this dissertation introduced in Chapter 3. The use of two "algorithms" should not cause confusion. One is an overall algorithm (as in the title of the dissertation), and the other, FALP, is the solution algorithm within the overall algorithm.

6.1 MIXED DISCRETE/CONTINUOUS OPTIMIZATION IN DESIGN

Optimization techniques have become an integral part of a design process where values for system variables must be found subject to a set of constraints, bounds, and objectives. The capability to solve a model or multiple models using optimization techniques is identified in Sections 1.1.4 and 3.1 as being necessary for the design of complex systems at both a subsystem and system level. In many cases, the design variables in optimization problems are assumed to be continuous. In design, however, this is not always the case. In a given design problem, there may exist design variables which are continuous, integer, or discrete. Examples of these are shaft lengths, number of gear teeth and gear diameters, respectively. There are well established methods for solving continuous problems (Reklaitis, et al., 1983). These are largely calculus-based and usually require evaluation of derivatives (e.g., gradient-based solvers). There are also well established methods for solving discrete problems (Arora and Huang, 1994). Application of these methods in the design of complex systems is reviewed in Section 2.3.5. These are largely based on some heuristic (e.g., branch and bound, Genetic Algorithms, Simulated Annealing, Tabu Search), since unlike its continuous counterparts, optimality criteria such as the Karush-Kuhn-Tucker conditions for discrete problems do not exist. Mixed discrete/continuous problems present mathematical programming challenges from both the continuous and discrete domains. The solution of these mixed problems is identified in (Papalambros, 1995) as being "one of the most daunting problems in design optimization."

As a starting point, a general mixed discrete/continuous optimization problem is stated as follows:

$$
\begin{aligned}
&\text{Find } X \\
&\quad x_1 \in X_C,\ x_2 \in X_I,\ x_3 \in X_D \\
&\quad x_1 \cup x_2 \cup x_3 = X \\
&\quad x_1 \cap x_2 = x_1 \cap x_3 = x_2 \cap x_3 = 0 \\
&\text{Satisfying} \\
&\quad \text{Constraints} \\
&\qquad g(X) \le 0.0 \\
&\qquad h(X) = 0.0 \\
&\quad \text{Goals} \\
&\qquad A_t(X) \left\{ \begin{matrix} \le \\ = \\ \ge \end{matrix} \right\} G_t \qquad \text{for all } t \\
&\text{Maximize} \\
&\quad A_r(X) \qquad \text{for all } r \\
&\text{Minimize} \\
&\quad A_s(X) \qquad \text{for all } s
\end{aligned}
\tag{6.1}
$$

where X_C are the continuous domain variables, X_I are the integer domain variables, and X_D are the discrete domain variables. This general model encompasses all classes of mathematical models. There are different modeling techniques to convert the general model for its solution by different codes. The philosophy for the conversion used in this dissertation is detailed in (Mistree, et al., 1994). This philosophy is implemented through the compromise Decision Support Problem (DSP), a multiobjective decision model which incorporates concepts from both traditional mathematical programming and goal programming (Mistree, et al., 1993a). In short, converting a general baseline model into a compromise DSP includes establishing priority among the goals and representing the goals as equations by using deviation variables. It is recognized that design problems are inherently multiobjective and optimizing with respect to each objective is impractical and many times impossible. Therefore, the problem is approached from a satificer's perspective (Simon, 1982). Consider a haystack with a number of needles hidden in it. An *optimizer* will continue to search the haystack until the last needle has been found. A *satisficer*, on the other hand, stops when she has found enough needles to proceed to the next step. The compromise DSP has the capability to model problems from either a

satisficing or an optimizing perspective (Mistree, et al., 1994). The perspective in this chapter is one of an *optimizer*, but the perspective of this dissertation is one of a *satisficer*. The compromise DSP is used to model mixed discrete/continuous optimization problems and focus on their single unique solutions.

The solution scheme developed in this dissertation to solve mixed discrete/continuous design problems is called the Foraging-directed Adaptive Linear Programming (FALP) Algorithm (Lewis and Mistree, 1996a). This idea is illustrated in Figure 6.1. Three primary constructs are labeled in the figure, A, B, and C. The ALP Algorithm, construct B in the lower half of Figure 6.1, uses gradients to move through the continuous design space. As part of the work in this chapter, a search engine is developed, construct A in the upper half of Figure 6.1, to intelligently search the discrete solution space for promising regions. This search engine is based on the notion of *foraging* of animals in the wild. In the animal behavior literature, foraging behavior in the wild is characterized by empirical observations and simple analytical models. Therefore, the foraging solver is not based on an accepted theory of foraging but on empirical foraging behavior observed in animals. The foraging search is not constrained by the convexity of the design space, taking a higher-level perspective of the design space and using heuristics to search it, as shown in Figure 6.1. Information is passed from the foraging discrete solver to the continuous solver using a common mathematical construct, the compromise DSP, construct C linking A and B in Figure 6.1. Therefore, a continuous solver (ALP) and discrete solver (foraging) are combined into one solution scheme (FALP) for mixed discrete/continuous problems. In Section 6.2, the three fundamental constructs of Figure 6.1 are presented. In Section 6.3, the step-wise FALP Algorithm is detailed, and in Section 6.4, the effectiveness of FALP is demonstrated using two well-studied examples.

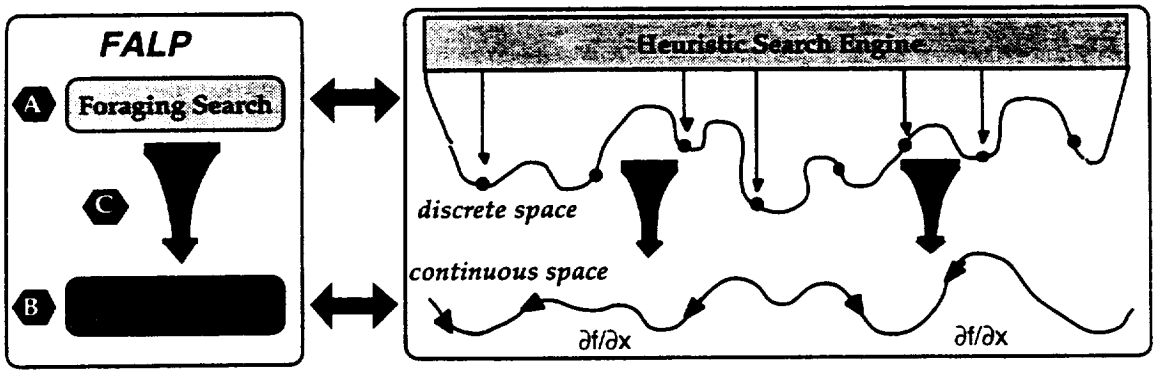


Figure 6.1. Foraging Search: Extending a Gradient Search

6.2 TECHNOLOGY BASE

In this section, each construct, A, B, and C presented in Figure 6.1 are discussed in relative detail. The integration of these constructs into an effective solution scheme for mixed discrete/continuous problems is presented in Section 6.3.

6.2.1 The Compromise DSP: The Domain Independent Interface

The compromise DSP, construct C in Figure 6.1, is the decision formulation linking the discrete and continuous solvers, foraging and ALP. In Section 1.2.1, a conceptual overview of the compromise DSP is given. In Section 3.4.4, a mathematical overview is given. The compromise DSP is a multiobjective decision model which is a hybrid formulation (Mistree, et al., 1993a), incorporating concepts from both traditional Mathematical Programming (Winston, 1995) and Goal Programming (Ignizio, 1983). The compromise DSP is used to determine the values of design variables to satisfy a set of constraints and to achieve as closely as possible a set of conflicting goals. The compromise DSP is used to model such decisions since it is *capable of handling constraints, goals, and multiple objectives* (Mistree, et al., 1994).

In this dissertation, the compromise DSP is used to model mixed discrete/continuous design problems. A general mixed discrete/continuous compromise DSP is given in Figure 3.15. The constraints, goals, and analysis routines embodied within the compromise DSP are used by the two solvers of FALP, foraging and ALP. The compromise DSP is used as the domain independent analysis interface between the foraging and ALP synthesis routines.

Because of the capabilities of the compromise DSP to handle multiple nonlinear constraints and goals, and a variety of variable types, the compromise DSP is used in this section to model single objective optimization problems, and more generally in this dissertation to model multiple objective satisficing problems. In the next section, the continuous solver for compromise DSPs, the ALP Algorithm is discussed.

6.2.2 The ALP Algorithm: The Continuous Solver

The ALP Algorithm, construct B in Figure 6.1, is used as the continuous solution portion of the FALP mixed discrete/continuous solver of this dissertation. The background of the ALP Algorithm is presented in Section 3.4.4 and the details are presented in (Mistree, et al., 1993a). The ALP Algorithm is only capable of handling continuous and Boolean variables. In this chapter, the ALP Algorithm is extended using an intelligent search engine which facilitates the handling of discrete and integer variables. There are various other methods for solving mixed optimization problems, but this work is rooted in the notion of satisficing and is applicable to *both* satisficing and optimizing problems. In this chapter, however, its applicability is only illustrated to optimization problems. The examples presented in Section 3 are single objective, and are used for comparison purposes. It is asserted that design problems are inherently multi-objective, and application to future

design problems will include multiple objectives. In Chapter 7, the design of an aircraft is studied and multiple objectives are addresses. In the next section the final component, construct ❹ from Figure 6.1, the foraging heuristic is introduced.

6.2.3 The Notion of Foraging in Optimization

The discrete solver, labeled ❹ in Figure 6.1, is modeled after the natural process of *foraging* by animals in the wild. The foundation for this search is the Tabu Search (TS) (Glover, 1989a) which is introduced and outlined in Section 3.4.2. TS is extended using constructs, which parallel the process of *foraging* by animals in the wild. The notion of using biological and evolutionary metaphors to model optimization has been explored and hypothesized upon in Glover (Glover and Greenberg, 1989). In Figure 6.2, TS and the foraging model are illustrated using a simple cartoon.

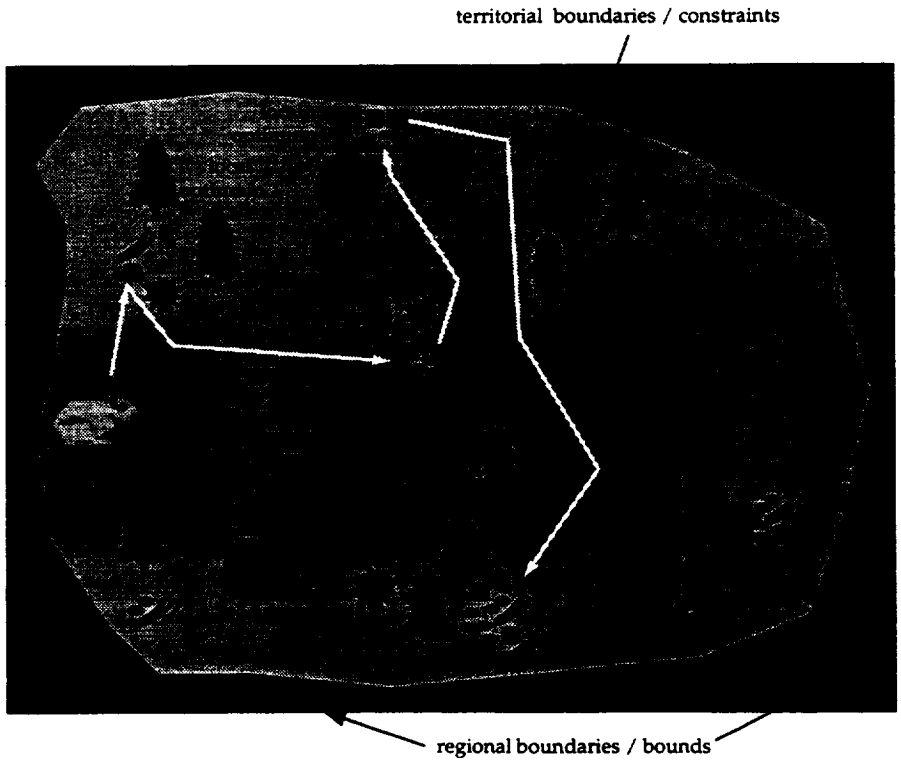


Figure 6.2. The Foraging Metaphor

In Figure 6.2, the rabbit (forager) begins the search by picking up a variety of scents (in effect, gradients). She moves in the direction of the strongest scent (largest descent gradient) until a solution (local optimum 1) is reached. The rabbit thinks, however, that places with more food exist, so she continues her search. This takes her to another site with food (local optimum 2). The search continues to another site (local optimum 3). Here, if the forager were to simply look for the strongest scent, it would lead her back to local optimum 1. However, she *remembers* that she has been there already, and continues the search elsewhere, eventually leading to the site with the most food (global optimum).

The two fundamental developments of the foraging search in this chapter are the use of a dynamic, changing memory structure and the identification of good portions of solutions that frequently occur (schema). The parallel aspects of the foraging analogy to discrete optimization are listed:

FORAGING	DISCRETE OPTIMIZATION
areas with food	*local optimum*
area with most food	*global optimum*
search steps	*discrete variables*
regional boundaries	*bounds*
territorial boundaries	*constraints*
experience	*memory structure*
site	*specific set of design variables*
neighborhood	*region of allowable moves*
increasing hunger	*dynamic memory*
characteristics of food areas	*schema identification*
diversification	*randomization*
objective: find most food in a reasonable amount of time	*objective:* *find best solution in reasonable time*

Foraging employs a set of heuristics based on empirical observations of animals, presented in Section 6.3. Searching a design space based on heuristics is common; search techniques such as genetic algorithms and simulated annealing are based on heuristics adapted from naturally occurring processes. The verification examples used to validate and verify the FALP Algorithm are single objective, and have strictly been used as a means of comparison. Therefore, in this chapter, an optimizer's perspective is taken. The results in Section 6.4 are presented from an optimization standpoint, but a satisficing standpoint has been taken in developing the solution scheme. A strength of the solution scheme for mixed discrete/continuous design problems is the capability to handle *multiple objectives,* as is demonstrated in Chapter 7.

In Section 6.3, the step-wise details of FALP are presented, which are based on the three notions introduced in this section, namely foraging, the ALP Algorithm, and the compromise DSP. In Section 6.4, two examples are presented to verify the FALP Algorithm and Hypothesis III of this work.

6.3 FALP: THE MIXED DISCRETE/ CONTINUOUS ALGORITHM, A STEP-BY-STEP APPROACH

A flowchart of the FALP algorithm in given in Figure 6.3. The foraging computer code, along with the updated files of DSIDES are given in Appendix B. Implementation of the solution scheme includes essentially 2 stages: the discrete solver (foraging), construct Ⓐ, and the continuous solver (ALP), construct Ⓒ. They are linked by the compromise DSP, construct Ⓑ. Alone, each solver would be ineffective in solving a mixed

discrete/continuous problem. The integration and interaction of the solvers provide the basis for the mixed solution scheme.

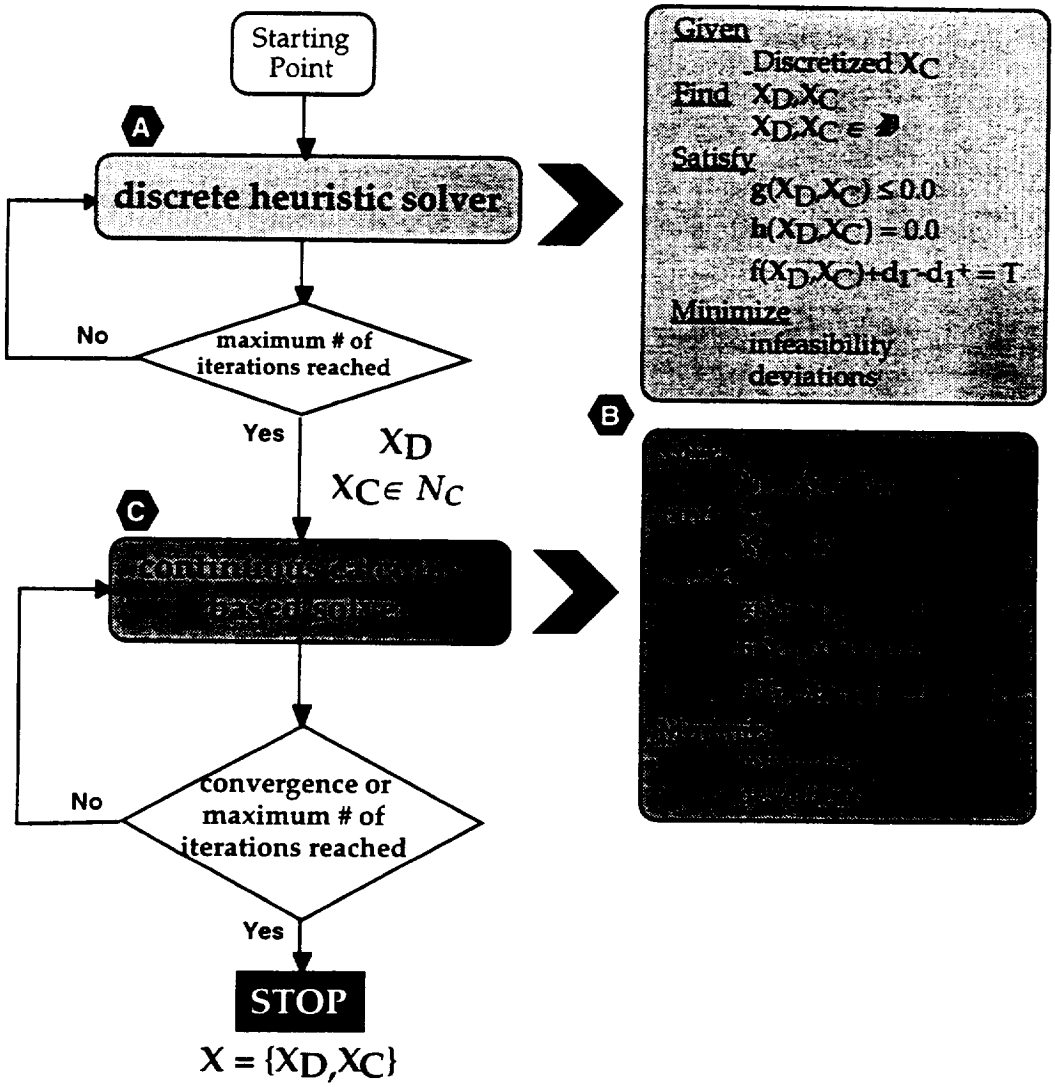


Figure 6.3. Flowchart of FALP

The basic step-wise solution procedure of FALP is summarized as follows, and a corresponding detailed schematic is shown in Figure 6.4.

Step 1: *Initialize problem and parameters (identify starting point, convergence criteria,...)*

Step 2: *Solve the discrete problem to find X_D and $X_C \in D$, using foraging search.*

- Step 2a: Discretize the continuous variables.
- Step 2b: Find best candidate solution in local neighborhood, N(x).
- Step 2c: Check dynamic memory list and aspiration criteria to allow/disallow solution.
- Step 2d: Build new solutions or enact diversification scheme based on schema, if necessary.
- Step 2e: Update list of best solutions encountered for user-interactive schema identification.
- Step 2f: If maximum number of iterations is reached, select best solution visited. If not go to step 2b.

Step 3: *Solve the continuous problem, $X_C \in R$, using ALP based on the information in Step 2.*

- Step 3a: Set discrete variables constant.
- Step 3b: Construct Linear Problem using Sequential Linearization
- Step 3c: Solve Linear Problem using Multiplex Algorithm
- Step 3d: If continuous solution has converged or reached a maximum number of iterations, stop. If not, go to step 3b.

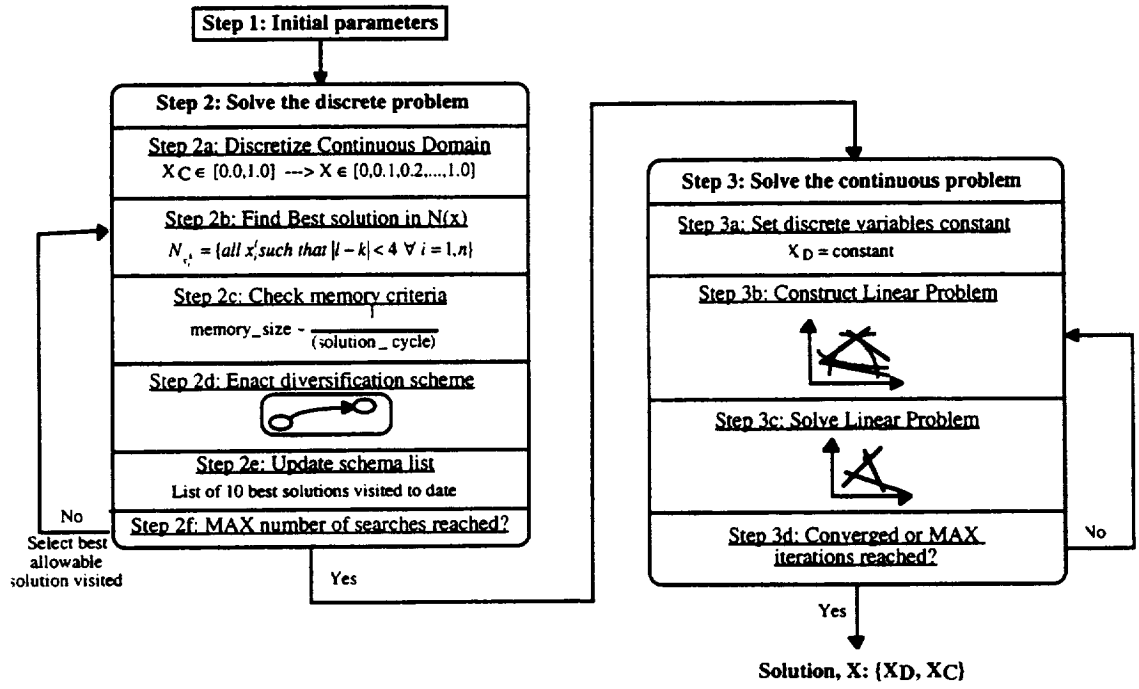


Figure 6.4. Schematic of FALP Solution Scheme

These steps are embodied within the three fundamental constructs of the solution scheme in Figure 6.4. These steps are more closely discussed.

Step 1: Initialize problem and parameters

In this step, the starting point and the lower and upper bounds on the variables are determined and the type of variable for each system variable is set. For the discrete variables, the possible values are specified. Convergence criteria and deviation function type are specified as well.

Step 2: Solve the discrete domain, X_D and $X_C \in D$, using foraging search.

Step 2a: Discretize the continuous variables.

Previous versions of the FALP solution scheme involved the isolation of the discrete solver and the continuous solver. Due to the decoupling of the two solvers, non-optimal solutions were found. The reasons for this are discussed in (Pan and Diaz, 1990). Therefore, to alleviate this difficulty, the continuous variables are discretized and used in the discrete domain search. This allows the discrete and continuous solver to interact more effectively. In the discrete solver, a discretized domain, D, is created for the continuous variables. Presently, the continuous variables are discretized using 10 discrete steps across the continuous domain specified from the lower and upper variables bounds. This low-fidelity discretization allows for exploration of the continuous domain without the expense of searching too fine of a discretization. Once the best neighborhood for the continuous variables (and best combination of the discrete variables) is found, the continuous solver can further refine the solution.

Step 2b: Find best candidate solution in local neighborhood, N(x).

As outlined in Section 3.4.2, performing local neighborhood searches is based on the Tabu Search (TS). Starting from some initial solution, the best solution in the local neighborhood is chosen by applying a greatest-descent procedure. The term *neighborhood* can be defined differently based on the size and characteristics of the problem. In this dissertation, a neighborhood size of *three* is used. That is, assume the discrete variables are ordered as $x_i^k = \{ x_i^1, x_i^2, x_i^3, \ldots, x_i^m \}$ where i is the variable number, k is the index number, and m is the number of possible discrete values for variable i. A variable value is in a local neighborhood if the *indices* of the variable values differ by less than four. Formally, the neighborhood of a point x_i^k is defined as

$$N_{x_i^k} = \{all\ x_i^l such\ that\ |l-k| < 4\ \forall\ i = 1,n\} \tag{6.2}$$

where n is the number of variables.

Step 2c: Check dynamic memory list to allow/disallow solution.

Researchers using the Tabu Search have assumed that the memory list lengths are constant. By allowing a changing list according to design space characteristics and search progress, better solutions could be generated more efficiently. This parallels the approach of Simulated Annealing (SA) where the probability of accepting a new move changes based on the progress of the search. The use of dynamic memory is evident in animals foraging for food (Benhamou, 1994, Todd and Kacelnik, 1993). The tendency for an animal to continue searching early in the search process for better "finds" as opposed to later is frequently observed (Huntingford, 1984, Todd and Kacelnik, 1993) based on relative energy levels. The mathematical description of the dynamic memory is as follows,

$$\text{memory\_size} \sim \frac{1}{(\text{solution\_iteration})} \tag{6.3}$$

where (memory_size) is the length of forbidden (tabu) moves list.

In other words, the foraging search will accept worse solutions *more frequently* (longer list of forbidden moves) earlier in the solution process. As the process continues, the length of the forbidden list becomes shorter and shorter, thus the search accepts fewer and fewer worse solutions and becomes more and more content with the best solution found thus far. In a foraging context, the dynamic memory corresponds to a forager becoming more and more hungry, thus increasingly content with the most food found to date. The dynamic memory structure is used in each neighborhood, N(x), to determine if a given design point is allowable. In other words, the search determines if the site has already been visited based on the present length of the memory list and then allows or disallows the move.

Step 2d: Enact diversification scheme based on stationarity of design variables, if necessary.

If certain variable values occur over a large number of iterations, then the solution scheme enacts a diversification scheme. The diversification scheme forces a change in the variable identified as having remained constant for a large number of iterations. Ideally, this diversification scheme would allow the search to escape from local optima. This diversification scheme has also been used in versions of the TS and is paralleled to an extent by the mutation operator in Genetic Algorithms.

Step 2e: Update list of best solutions encountered for user-interactive schema identification.

This notion parallels a similar notion in Genetic Algorithms (GA). That is, characteristics or schema (variable values) which occur frequently in the best solutions are identified and new solutions can be found based on this set of characteristics. In the foraging search, a record is kept of what discrete variable values occur as the solution of each iteration. Presently, the user can then "build" solutions using the schema as the foundation. This

could greatly simplify the problem if schema are identified. With reference to Figure 6.3, note the flowers () that are present at most of the food sites. It has been observed in the behavior of many animals that they identify certain characteristics of places where food is found and will look for these guiding characteristics in determining future sites (Menzel, 1991). Therefore, in Figure 6.3, the rabbit would ideally recognize the presence of the flowers and in the future identify the flower and then look for food in close proximity.

Step 2f: If maximum number of iterations is reached, select best solution visited. If not go to step 2a.

Stopping criteria is a very important aspect of the solution scheme. The foraging search is an unassuming search. That is, it will continue to search until a maximum number of iterations is reached. In order to ensure efficient design space search, in foraging, the maximum number of iterations is proportional to the problem size (number of variables and number of discrete values). The proportionality constant currently is determined based on simple empirical studies using classes of problems with similar size. For example, a representative rule could be:

(for 10,000-20,000 possible neighborhoods,

set maximum number of neighborhoods searched to 1,000.)

Therefore, the percentage of the design space searched in various problems may change. For example, in the preceding rule, the percentage design space searched in a problem with 10,000 possible neighborhoods would be around 10%, while for a problem with 20,000 possible neighborhoods would be around 20%. The examples in Section 6.4 are of the same order of magnitude, but the percentage of design space searched is different. Identifying a proportionality constant that results in the most effective solutions for all problem sizes is an area of future consideration.

Step 3: Solve the continuous problem, $X_C \in R$, using ALP based on the information in Step 2.

Step 3a: Set discrete variables constant.

The discrete variable values found in Step 2, X_D, are set constant. The continuous neighborhood, $X_C \in N_C$ is also determined from Step 2.

The continuous solver, labeled B in Figures 6.1 and 6.4, is the Adaptive Linear Programming Algorithm (ALP).

Step 3b: Construct Linear Problem using Sequential Linearization

Step 3c: Solve Linear Problem using Multiplex Algorithm

Step 3d: If continuous solution has converged, go on. If not, go to step 3b.

In the next section, the effectiveness of the solution scheme is illustrated using two example problems. These examples are single objective, and are used for comparison purposes. Design problems are inherently multi-objective, and application to other design problems, including the study in Chapter 7, includes addressing multiple objectives.

6.4 VERIFICATION STUDIES

The two problems in this section have been well studied by other researchers. They are used only as a means of verification, comparison, and illustration of FALP. Therefore, compromise DSPs are used, which are typically multiobjective, to model single objective

optimization problems only to illustrate and verify the FALP algorithm. In the spring design problem, it is illustrated and numerically proven that the global optimum is found by FALP. The behavior of FALP is also illustrated using the spring design problem. With certain problems, globally optimal solutions may be found, as is demonstrated in the spring design problem. In more complex design problems, finding globally optimal solutions may not be feasible or possible due to the nonlinearity of the problem and presence of multiple local optima. In the pressure vessel design problem, it is also illustrated and numerically proven that the global optimum is found by FALP. It is also illustrated how the previous studies can be improved upon using active constraint and monotonicity arguments.

6.4.1 Coil Compression Spring Design (Kannan and Kramer, 1994, Sandgren, 1990)

This is a problem involving discrete, integer, and continuous variables. A helical compression spring is to be designed as shown in Figure 6.5. The goal is to minimize the volume of the spring. The spring is to be manufactured from music wire spring steel ASTM A228. Therefore, the wire diameter can assume only the discrete values shown in Table 6.1. The design variables are D, the winding diameter (continuous), d, the wire diameter (discrete), and N, the number of spring coils (integer). The units for D and d are inches. The constraint g_1 is shear stress limit. The constraints g_2, g_3, and g_4 are geometry limits. The constraint g_5 is to ensure proper winding. The constraints g_6, g_7, and g_8 are for deflection requirements.

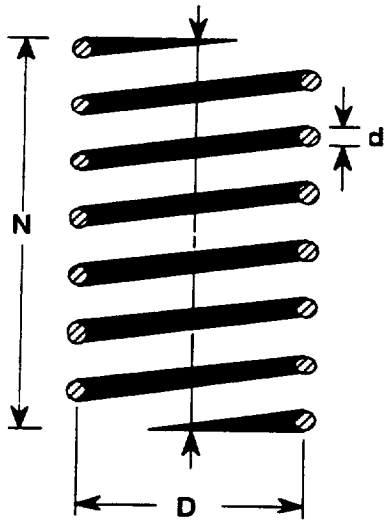


Figure 6.5. Coil Compression Spring

Table 6.1. Possible Wire Diameter for ASTM A228 (inches)

0.0090	0.0095	0.0104	0.0118	0.0128	0.0132
0.0140	0.0150	0.0162	0.0173	0.0180	0.0200
0.0230	0.0250	0.0280	0.0320	0.0350	0.0410
0.0470	0.0540	0.0630	0.0720	0.0800	0.0920
0.1050	0.1200	0.1350	0.1480	0.1620	0.1770
0.1920	0.2070	0.2250	0.2440	0.2630	0.2830
0.3070	0.3310	0.3620	0.3940	0.4375	0.5000

The problem was discussed in (Kannan and Kramer, 1994, Sandgren, 1990) to demonstrate different algorithms for nonlinear mixed discrete-continuous optimization. The compromise DSP of this problem is as follows:

Given

S = Allowable Stress	189,000 psi.
G = Shear Modulus	1.15×10^8
F_{max} = Maximum Working Load	1000 lb.
l_{max} = Maximum Free Length	14.00 in.
d_{min} = Minimum Wire Diameter	0.200 in.

D_{max} = Maximum Outside Spring Diameter 3.00 in.
F_p = Preload Compression Force 300 lb.
∂_{pm} = Maximum Deflection under Preload 6.00 in.
∂_w = Deflection from Preload to Max Load 1.25
Spring is Guided
C = D/d

$$C_f = \frac{4C-1}{4C-4} + \frac{0.615}{C}$$

$$K = \frac{Gd^4}{8ND^3} \text{ lb / in.}$$

$$\partial = \frac{F_{max}}{K} \text{ in.}$$

$l_f = \partial + 1.05(N+2)d$
$\partial_p = F_p/K$

Find

System variables
D = Coil Diameter
d = wire diameter
N = Number of Coils

Deviation Variable d_1^+

Satisfy[2]

Constraints

g_1, *shear stress*: $\frac{8KF_{max}D}{S\pi d^3} \le 1.0$

g_2, *free length limit*: $\frac{l_f}{l_{max}} \ge 1.0$

g_3, *minimum wire diameter*: $\frac{d_{min}}{d} \le 1.0$

g_4, *maximum outside diameter*: $\frac{D+d}{D_{max}} \le 1.0$

g_5, *winding limit*: $\frac{C}{3} \le 1.0$

g_6, *maximum preload deflection*: $\frac{\delta}{\delta_{pm}} \le 0.0$

g_7, *combined deflection consistency*:

$$\frac{\left[\partial_p - \frac{F_{max}-F_p}{K} - 1.05(N+2)d\right]}{l_f} \le 1.0 \qquad (6.4)$$

g_8, *deflection requirement*: $\frac{K\delta_w}{(F_{max}-F_p)} \le 1.0$

---

2 UB = Upper Bound, LB = Lower Bound, TV = Target Value

Goals

$$F:[0.25\pi^2 D d^2 (N+2)] / Vol_{TV} - d_1^+ = 0.0$$

Bounds

$D_{LB} \le D \le D_{UB}$
$d_{LB} \le d \le d_{UB}$
$N_{LB} \le N \le N_{UB}$

Minimize

Deviation Function d_1^+

The bounds on the system variables for the problem are $D_{LB} = 1.0$, $D_{UB} = 6.0$, $d_{LB} = 0.0$, $d_{UB} = 0.5$, $N_{LB} = 3$, $N_{UB} = 30$. Based on the previous results on this problem, the target value for the cost goal is taken as $Vol_{TV} = 0.5$ in^3. In Table 6.2, the results from FALP, and (Kannan and Kramer, 1994, Sandgren, 1990) are compared. While Kannan's solution is 15.5% better than Sandgren's solution, the objective function (corresponding to the deviation function) found by FALP is 58.2% lower than the Sandgren's solution. The constraint values are all feasible.

Table 6.2. Coil Spring Results

	FALP	Kannan and Kramer	Sandgren
D (in.)	1.000	1.329	1.180
d (in.)	0.283	0.283	0.283
N	3	7	10
g_1	0.874	1.054	0.971
g_2	0.129	0.318	0.382
g_3	0.707	0.707	0.707
g_4	0.428	0.537	0.488
g_5	0.849	0.639	0.720
g_6	0.016	0.089	0.089
g_7	-0.641	-0.200	-0.334
g_8	0.182	0.998	0.998
F (in.^{3})	**0.988**	**2.365**	**2.799**

6.4.2 Spring Design: Verification and Validation

Validation of Search Process

In this section, the solution found by FALP is validated and then verified as the best possible solution for this problem. Before the solution is discussed, the notion of a "cycle" in FALP warrants some definitions. Since FALP consists of 2 solvers, the total number of cycles equals those spent in the foraging search *and* the ALP solver. A cycle in FALP is defined as

the process to move from one design point to another.

However, because one is based on search heuristics and the other is based on calculus, the process of moving from one point to another is different. In the foraging search, a cycle is

the search of one neighborhood and the selection of the next design point based on the foraging protocol.

In this problem, the number of cycles (neighborhood searches) used in the foraging search is 100. In the ALP solver, a cycle is

the linearization of the nonlinear model and solution of the linear model.

Since no external analysis routines are used, this cycle correponds to a *synthesis* cycle of ALP in Figure 3.16. The number of cycles performed in the ALP routine depends on the mathematics of the model. Convergence criteria is set for the continuous variables, and when this is reached, ALP stops. If convergence is not reached, there is an upper limit of 40 cycles in ALP. Once ALP stops, FALP is finished as well.

In Figures 6.6-6.8, the search history of the foraging search is shown for the three system variables using the upper bound starting point. In this study, three different starting points, the upper bounds of the variables, the lower bounds of the variables, and points in the middle of each range. Since D and N are at their lower bounds in the final solution, the variable activity is illustrated using only the upper bounds starting point, as this starting

point is the furthest away from the solution. The other starting points also converge to the same solution and the plots look very similar and are given in Appendix B.

The one trait of the search that is clearly illustrated in Figures 6.6-6.8 is the diversification scheme (step 2d in FALP, see Section 6.3). If a system variable remains at a given value for more than a specified number of cycles, the variable is changed to another value in completely different part of the design space. For instance, in Figure 6.6, the diversification scheme was enacted around cycles 36, 57, and 80. Similar behavior can be seen in Figures 6.7 and 6.8.

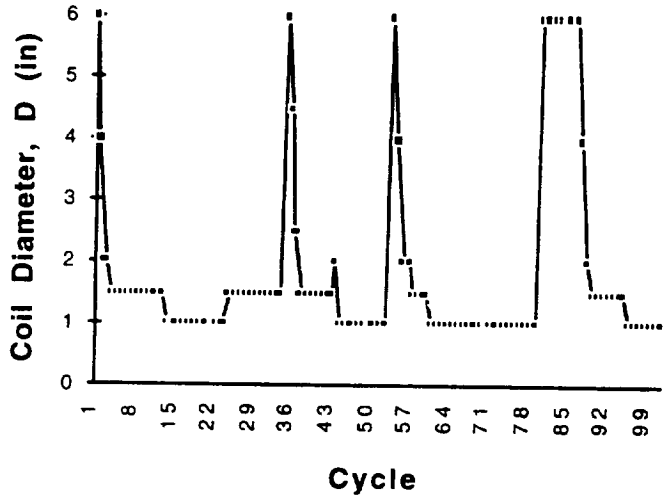


Figure 6.6. Coil Diameter Behavior

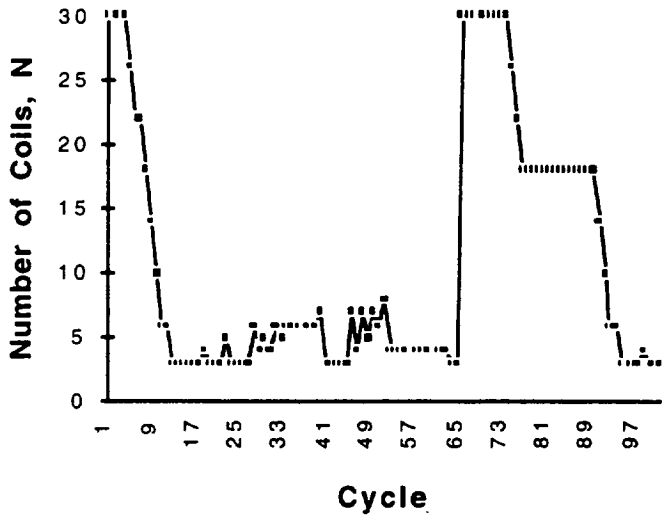


Figure 6.7. Number of Coils Behavior

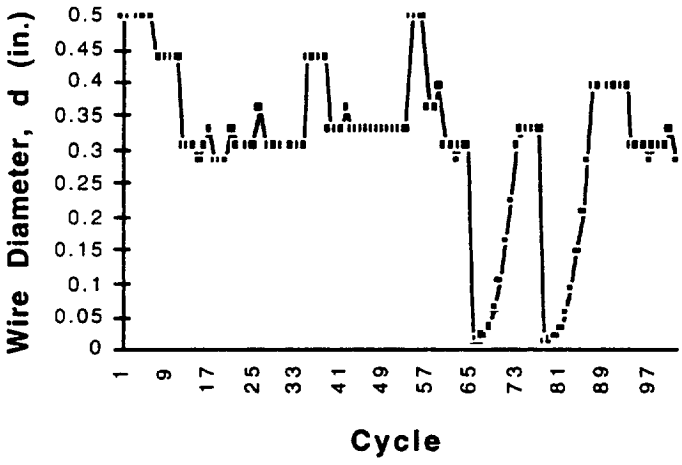
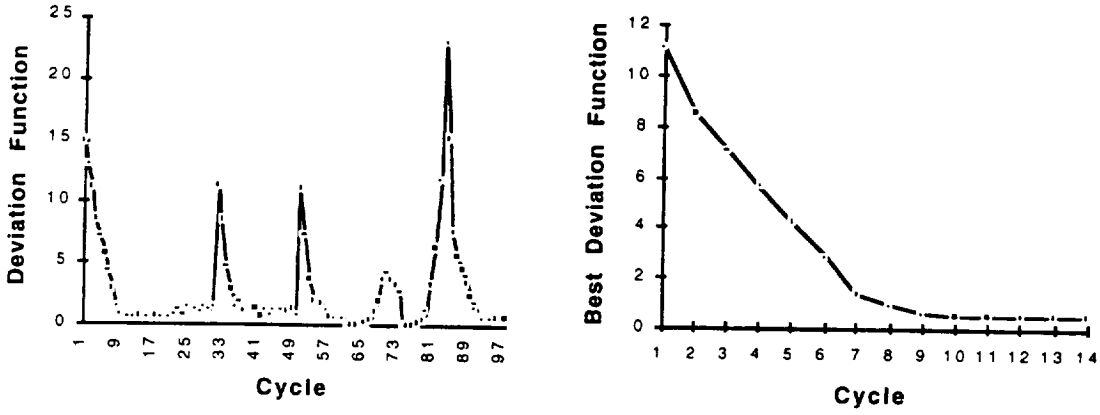


Figure 6.8. Wire Diameter Behavior

As further illustration of the foraging search, the *deviation function* at each cycle is shown in Figure 6.9 (a), and the *best* deviation function found so far is plotted in Figure 6.9 (b). It is clear that foraging is accepting *worse* solution points, in order to escape local

optimum, much like the approach in Simulated Annealing (Kirkpatrick, et al., 1983), but the *best* deviation function encountered continues to improve steadily.



(a) Current Deviation Function **(b) Best Deviation Function**

Figure 6.9. Spring Deviation Function Behavior

Two of the fundamental observations of foraging animals which are modeled are schema identification and dynamic memory (steps 2e and 2c, respectively). These two developments are illustrated using Table 6.3 and Figure 6.10. In Table 6.3, the *10 best solutions* for the spring problem as identified by the foraging portion of FALP are shown. As is shown, nine of the ten solutions include the value of 1.0 for D, while N and d vary. However, since D is a continuous variable, it is not identified as being part of the schema. If N or d consistently had a certain value in Table 6.3, they could be set constant in order to simplify the problem and build better solutions.

Table 6.3. 10 Best Spring Solutions (Schema)

Z	D	N	d
0.494	1	3	0.283
0.581	1	3	0.307
0.593	1	4	0.283
0.676	1	3	0.331
0.691	1	5	0.283
0.697	1	4	0.307
0.790	1	6	0.283
0.811	1	4	0.331
0.814	1	5	0.307
0.872	1.5	3	0.307

In Figure 6.10, the dynamic memory structure of the foraging search is validated. In Figure 6.10, the number of moves is plotted against the cycle number in the foraging search for three different starting points. Again, a cycle is a complete neighborhood search. For two of the starting points, the number of moves not allowed increases rapidly until about cycle 65, and then the search becomes more and more content with the visited solutions. In other words, earlier in the solution process, a longer memory list is enacted. In a foraging context, the scheme (or an animal in search of food) will not return to a site already visited for a considerable amount of time. As the solution process continues the memory list steadily decreases, as the scheme (animal) becomes more satisfied with what has been found and will not try new directions as often.

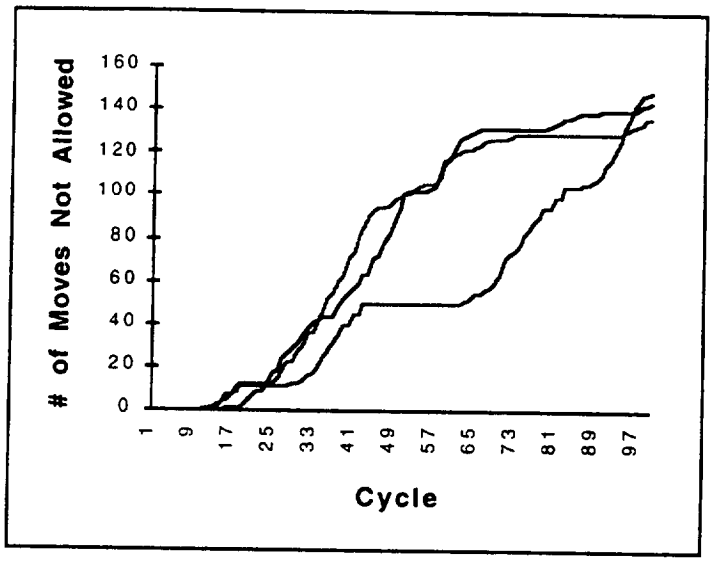


Figure 6.10. Number of Moves Not Allowed

The total number of function evaluations in this study is on the order of [n*2*(m-1)*l] where n is the number of design variables, m is the number of neighbors allowed in the local neighborhood and l is the number of foraging cycles. The total number of possible functions evaluations (a measure of the problem size) is on the order of $\prod d_i$, where d_i is the number of discrete values for design variable number i. For the spring study the number of function evaluations is

$$3*2*(4-1)*100 = 1800$$

and the total possible is

$$11*27*42 = 12474.$$

Therefore, the percentage of the discrete design space searched is approximately 14%. This is not a large percentage, but with the small size of the spring problem, even fewer foraging neighborhood searches could have been performed, as the best solution was encountered before the search ended for the lower bound, middle point, and upper bound starting points. Optimizing the length of the search based on problem size is part of the future work in this area.

Verification of Solution

The success of the solution found by FALP is interesting. As a means of verifying the solution, an exhaustive combinatorial experiment is performed using the discrete and integer variable. Since there are 28 possible values for N and 42 possible values for d, the total number of possible combinations is 28 x 42 = 1176. At each of these combinations a one-variable optimization problem was performed with respect to the continuous variable D, keeping N and d constant. Therefore, 1176 different designs are obtained. A sample of these 1176 designs is shown in Table 6.4. In Table 6.4, the first solution is infeasible (About 25% of the solutions were infeasible). Solution number 119 corresponds to the solution found by FALP. Solution number 204 corresponds to Kannan's solution. Solution number 330 corresponds to Sandgren's solution. Out of these 1176 designs, solution number 109, found by FALP, was clearly the best solution with the smallest objective function. The previous solutions from other studies were indeed found to be local optimum, but only local optimum. By performing an exhaustive search, the solution is validated and it is shown that it is indeed a global solution for this problem.

Table 6.4. Spring Validation (nfs - no feasible solution)

Solution #	N	d (in.)	D (in.)	f (vol, in^3)
1	3	0.009	nfs	nfs
.	.	.	.	.
119	**3**	**0.283**	**1.00**	**0.988**
.	.	.	.	.
204	7	0.283	1.33	2.365
.	.	.	.	.
330	10	0.283	1.10	2.799
.	.	.	.	.
1176	30	0.500	nfs	nfs

6.4.3 Pressure Vessel Design (Hsu, et al., 1995, Kannan and Kramer, 1994, Lin, et al., 1995, Sandgren, 1990)

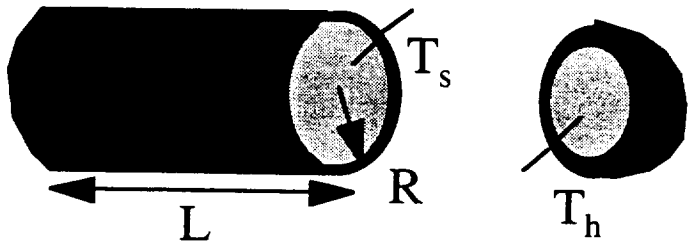


Figure 6.11. Pressure Vessel

This is a problem involving discrete and continuous variables. A cylindrical pressure vessel is capped at both ends by hemispherical heads as shown in Figure 6.11. The goal is to minimize the total cost of manufacturing the pressure vessel, including the cost of material, and cost of forming and welding. The design variables are R and L, the inner radius and length of the cylindrical section, and T_s and T_h, the thickness of the shell and head. The variables are each given in inches. The variables R and L are continuous, while T_s and T_h are integer multiples of 0.0625 inch, the available thicknesses of rolled steel plates. The constraints g_1, g_2, g_3 correspond to ASME limits on the geometry while g_4 corresponds to a minimum volume limit.

The problem was discussed in (Hsu, et al., 1995, Kannan and Kramer, 1994, Lin, et al., 1995, Sandgren, 1990) to demonstrate different algorithms for nonlinear mixed discrete-continuous optimization. In the study by Lin, Genetic Algorithms (GA) and Simulated Annealing (SA) were used as the solution algorithms. The compromise DSP of this problem is as follows:

Find

System variables

R, L, T_s and T_h

Deviation Variable d_1^+

Satisfy

Constraints

g_1, minimum shell wall thickness: $\frac{0.0193R}{T_s} \leq 1.0$

g_2, minimum head wall thickness: $\frac{0.00954R}{T_h} \leq 1.0$ (6.5)

g_3, maximum length: $\frac{L}{240} \leq 1.0$

g_4, minimum volume of tank: $\frac{(-\frac{4}{3}\pi R^3 + 1296000)}{\pi R^2 L} \leq 1.0$

Goals

$$F:[0.6224T_sRL + 1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_s^2R] / Cost_{TV} - d_1^+ = 0.0$$

Bounds

$R_{LB} \leq R \leq R_{UB}$

$L_{LB} \leq L \leq L_{UB}$

$T_{sLB} \leq T_s \leq T_{sUB}$

$T_{hLB} \leq T_h \leq T_{hUB}$

Minimize

Deviation Function d_1^+

The bounds on the system variables for the problem are R_{LB} = 25 in., R_{UB} = 150 in., L_{LB} = 25 in., L_{UB} = 250 in., T_{sLB} = 0.0625 in., T_{sUB} = 1.25 in., T_{hLB} = 0.0625 in., T_{hUB} = 1.25 in.. Based on the previous studies of this problem, the target value for the cost goal is, $Cost_{TV}$ = \$5000.00. In this problem as in the spring problem, the number of foraging cycles (neighborhood searches) used was 100. The upper limit of cycles in the continuous portion, ALP, is 40 cycles.

In Table 6.5, the results from FALP are compared with the previous studies. The objective function (corresponding to the deviation function) found by FALP is 14.6% lower than the previous best found solution. The constraint values are all feasible (≤ 1).

Table 6.5. Pressure Vessel Results

	FALP	Hsu[3]	Lin (GA/SA)	Kannan and Kramer	Sandgren
R (in.)	38.76	51.81	N/A	58.29	47.7
L (in.)	223.3	101.85	N/A	43.69	117.70
T_s (in.)	0.75	1.00	N/A	1.125	1.125
T_h (in.)	0.375	0.50	N/A	0.625	0.625
g_1	0.997	1.000	N/A	1.000	0.840
g_2	0.986	0.989	N/A	0.890	0.747
g_3	0.998	0.424	N/A	0.182	0.445
g_4	0.930	0.831	N/A	1.000	1.000
F ($)	**5869.5**	**7021.67**	**7197.7**	**7198.20**	**8129.80**

6.4.4 Pressure Vessel Design: Validation and Verification

Validation of Search Process

In this section, the solution found by FALP is validated and then verified as the best possible solution for this problem. In this study, three different starting points, the upper bounds of the variables, the lower bounds of the variables, and points in the middle of each range. Each starting point "converged" to the same solution. Since foraging is based on heuristics and not on strict convergence criteria, the "convergence" of a starting point is misleading. The solution history plots look very similar and are given in Appendix B for the pressure vessel problem. Since the value of L in the solution is close to its upper

[3] Note that g_4 is different from the Hsu's reported value of 1.000. The author was contacted about an error in his paper which led to an erroneous value of 1.000 being reported. He acknowledges this error.

bound, the variable history is shown for only the lower bound starting point, as this starting point is the furthest away from the solution of L.

In Figures 6.12-6.15, the search history of the FALP Algorithm is shown for the four system variables. For the pressure vessel problem, 125 cycles are performed, 100 in the discrete domain (foraging) and 25 in the continuous domain (ALP). As described in Section 6.3, the discrete variables are kept constant in continuous solver. Therefore, in the cycles 100-125, only the continuous variables, R and L, are changed. The diversification scheme (step 2d of FALP, see Section 6.3) is clearly illustrated again in Figures 6.12-6.15. If a system variable remains at a given value for more than a specified number of cycles, the variable is changed to another value in completely different part of the design space. For instance, in Figure 6.12, the diversification scheme was enacted around cycles 30, 60, and 80. Similar behavior can be seen in Figures 6.13-6.15.

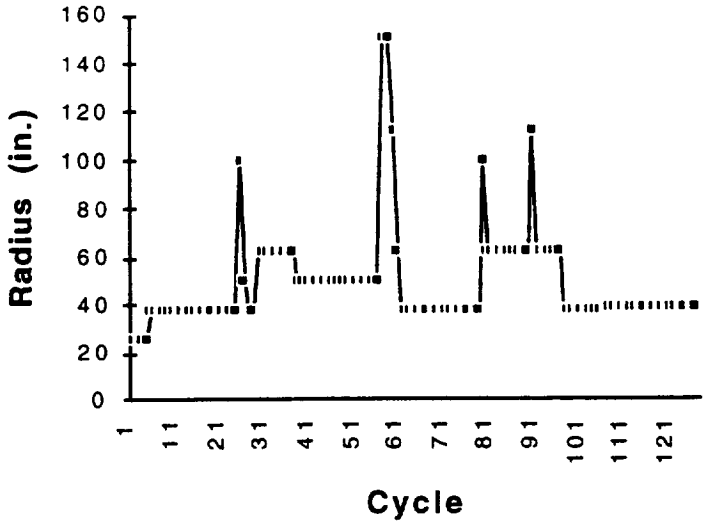


Figure 6.12. Radius Behavior

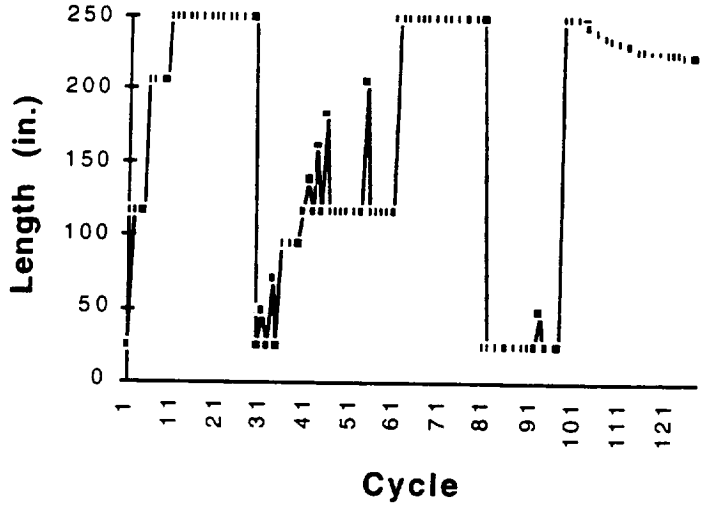


Figure 6.13. Length Behavior

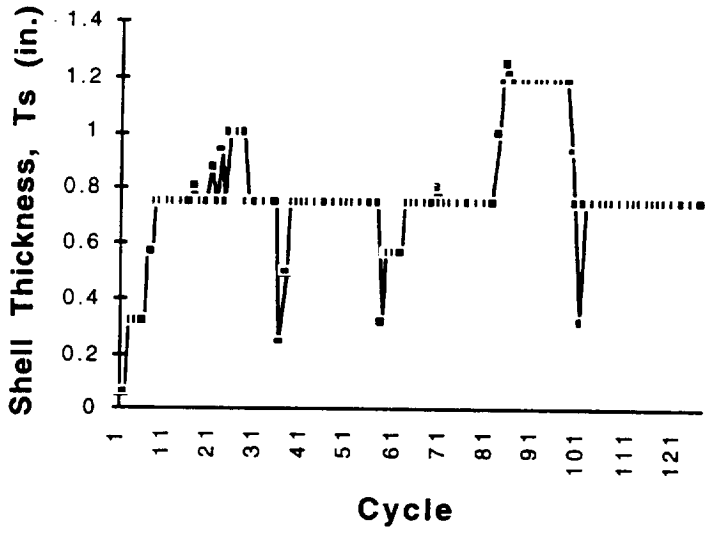


Figure 6.14. Shell Thickness Behavior

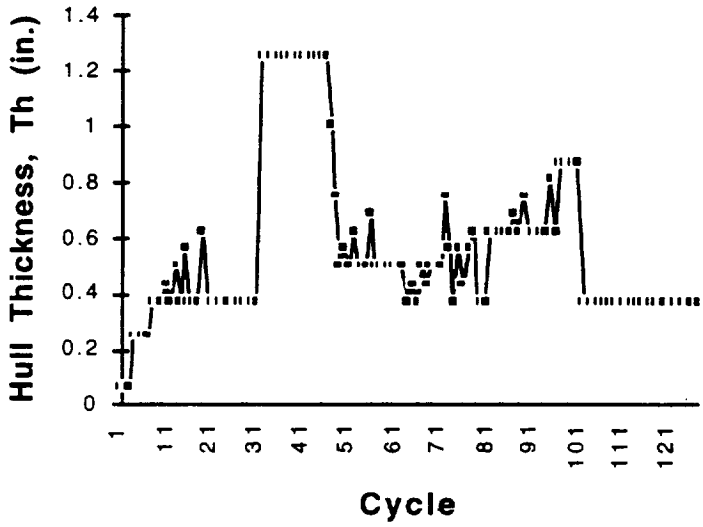
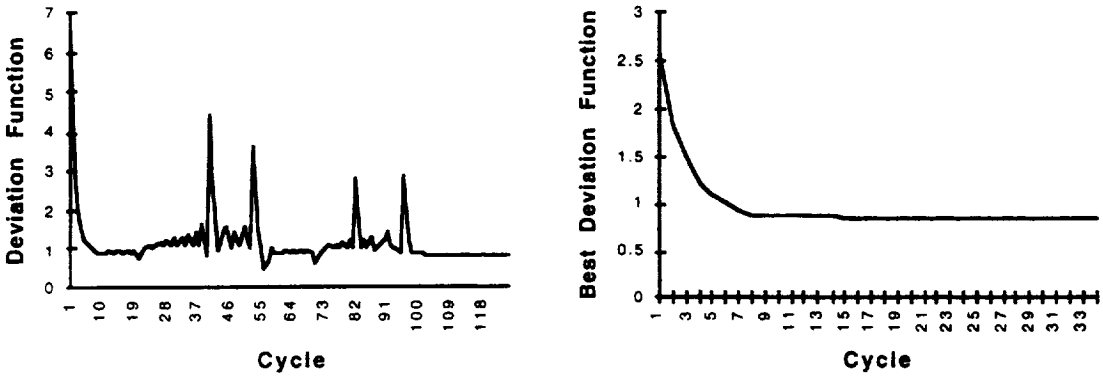


Figure 6.15. Hull Thickness Behavior

Similar to the spring example, the deviation function at each cycle is shown in Figure 6.16 (a), and the *best* deviation function found so far is plotted in Figure 6.16 (b). It is clear that foraging is accepting *worse* solution points, in order to escape local optimum, but the *best* deviation function encountered continues to improve steadily.



(a) Current Deviation Function **(b) Best Deviation Function**

Figure 6.16. Pressure Vessel Deviation Function Behavior

In Table 6.6, the 10 best solutions for the pressure vessel problem as identified by the foraging portion of the solution scheme are shown. Nine of the ten solutions include the value of 37.5 for R, and 250 for L, while T_s and T_h vary. Since R and L are continuous variables, they are not identified as being part of the schema. A designer can use this information to simplify the problem.

Table 6.6. 10 Best Pressure Vessel Solutions

Z	R	L	T_s	T_h
0.441	37.5	250	0.75	0.375
0.452	37.5	250	0.75	0.4375
0.464	37.5	250	0.75	0.5
0.475	37.5	250	0.75	0.5625
0.478	37.5	250	0.8125	0.375
0.484	50	115	0.9375	0.5
0.486	37.5	250	0.75	0.625
0.489	37.5	250	0.8125	0.4375
0.497	37.5	250	0.75	0.6875
0.500	37.5	250	0.8125	0.5

In Figure 6.17, the dynamic memory structure of the foraging search is again validated. In Figure 6.17, the number of moves is plotted against the cycle number in the foraging search for three different starting points. For all three starting points, the number of moves not allowed increases, but in the later cycles begins to level off. With the lower bound starting point, the number of moves not allowed is significantly more than the other two starting points, indicating that the better regions are found more rapidly and therefore, a greater number of moves must be not allowed. This is supported by the fact that the best solution was found at cycle number 9 using the lower bound starting point.

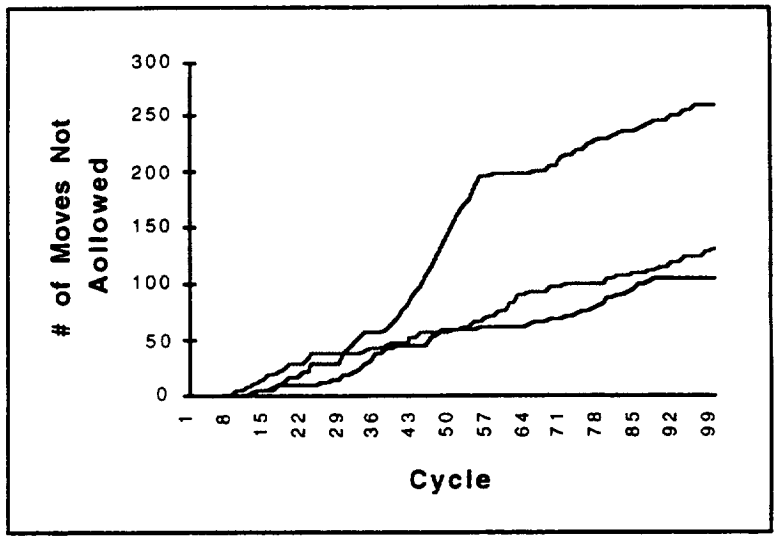


Figure 6.17. Number of Moves Not Allowed

For the pressure vessel study the number of function evaluations is

$$4*2*(4-1)*100 = 2400$$

and the total possible is

$$11*11*20*20 = 48400.$$

So, the percentage of the discrete design space searched is approximately 5%. This is even a smaller percentage than the spring problem because the size of this problem (number of variables and number of discrete values) is larger than the spring problem, but the same number of foraging cycles is used. Similar to the spring problem, even fewer foraging neighborhood searches could have been performed, as the best solution was encountered before the search ended for the lower bound, middle point, and upper bound starting points. Optimizing the length of the search based on problem size is part of the future work in this area. This will include looking at the efficiency of the search process and determining if a certain search percentage of the design space produces superior solutions regardless of problem size.

Verification of Solution

Again, in this problem favorable results are found. Various starting points are used for this problem as in the spring design problem. An exhaustive search is again performed again using 400 (20 discrete values for two variables) combinations of T_s and T_h. In this case, each discrete combination requires an optimization with respect to two continuous variables. In Table 6.7, partial results of the 400 cases are shown. Solution #1 is infeasible with T_s and T_h set at their lower bounds. In fact, 265 of the 400 (66%) solutions are infeasible. The best solution is found at solution #226 (cost = 5869.5), which corresponds to the solution found by FALP. Solution #350 corresponds to the solution found by Kannan in Table 6.5. It is interesting to note that Sandgren's solution shares the *same discrete variable values* with Kannan's solution, but the continuous variable values are different. In the 2-variable optimization results in Table 6.7, Kannan's solution (#350) was found to be the best solution *using the discrete combination*, T_s = 1.125 and T_h = 0.625. Therefore, Sandgren's solution is not even a local optimum.

Table 6.7. Pressure Vessel Validation (nfs - no feasible solution)

Solution #	T_s(in.)	T_h (in.)	R (in.)	L (in.)	f (cost)
1	0.0625	0.0625	nfs	nfs	nfs
.	.	.			.
226	**0.75**	**0.375**	**38.75**	**223.3**	**5869.5**
.	.	.			.
308	1.00	0.50	51.8	84.6	6423.40
.	.	.			.
350	1.125	0.625	58.3	43.7	7198.20
.	.	.			.
400	1.25	1.25	52.0	83.3	11401.8

Solution #308 in Table 6.7 corresponds to Hsu's solution in Table 6.5, *except* for the value of L. This can be explained by simply analyzing the active constraints and behavior of the objective function in the pressure vessel model. Specifically looking at the results from Hsu (Hsu, et al., 1995) in Table 6.5, constraints g_1 (1.000) and g_2 (0.989) are active or very close to active. Since the cost is a monotonically increasing function with respect to every variable, the only improvement would be from *decreasing* a variable. However, in g_1 and g_2, decreasing x_3, or x_4, will result in infeasibility. Therefore, any improvement must occur by changing x_1 or x_2. In constraint g_3 (0.424) the value of x_2 can be changed. However x_2 also is present in g_4, therefore caution must be used as to not cause g_4 to become infeasible. Since g_4 is not active, however, there is some slack available. Keeping x_1 constant, x_2 can be decreased to 84.6 in. where g_4 becomes active. The new solution is shown in Table 6.8, which is exactly the solution predicted in the full search of Table 6.7.

Table 6.8. Improvement in Hsu's Solution (Hsu, et al., 1995)

	Solution from Hsu	Improvement of Hsu's solution
R (in.)	51.81	51.81
L (in.)	101.85	***84.60***
T_s (in.)	1.00	1.00
T_h (in.)	0.50	0.50
g_1	1.000	1.000
g_2	0.989	0.989
g_3	0.424	0.353
g_4	0.831	***1.000***
F ($)	7021.67	***6410.80***

In this section, the effectiveness of the FALP Algorithm is illustrated using two well-studied examples. The highlights of this chapter are:

- the previously published best solutions for both problems are greatly improved upon,
- the solutions found by FALP are proven to be the global optimum for both problems,
- some of the previous solutions are shown to be local optimum while others are shown to not even be local optimum.

In the next section, the developments and contributions of this chapter are summarized, with respect to Hypothesis III and the corresponding posits.

6.5 A LOOK BACK AND A LOOK AHEAD

In this chapter, a solution scheme for mixed discrete/continuous design problems is presented, namely, the Foraging-directed Adaptive Linear Programming Algorithm. The discrete portion of the scheme is based on the notion of foraging of animals in the wild and includes a dynamic memory structure and schema identification. The effectiveness of the solution approach presented is illustrated using two examples. These examples are strictly single objective, but have strictly been used as a means of comparison. A strength of the foraging search for mixed discrete/continuous design problems is the ability to handle *multiple objectives.* The FALP algorithm presented is the solution algorithm in the computer decision support package, Decision Support in the Design of Engineering Systems (DSIDES).

This work is a step towards a broader goal. By combining these aspects of the Tabu Search, Simulated Annealing, and Genetic Algorithms, a broad class of solution algorithms can be established, under which TS, GA, and SA can be classified. This broad class is an abstraction of *nature*, where intelligence is embedded in many biological processes. Algorithms such as GAs model a natural process and capture the inherent intelligence in a

computational model. By using foraging as the foundation, it is an objective to establish a class of algorithms based on intelligence, either innate or artificial. Aspects of GAs, SA, and TS are found in the foraging analogy, and it is asserted that there is a broad class of intelligent algorithms under which these can be grouped. The work in this chapter was stimulated by the exploration of the similarities between optimization and artificial intelligence in (Glover, 1986, Glover and Greenberg, 1989).

In this chapter, the support and verification for Hypothesis III of this dissertation is presented. The developments associated with Hypothesis III constitute Step 3 of the overall algorithm of this dissertation. These developments support Posits 3.1, 3.2, and 3.3 presented in Section 3.1.3. The observations relating to each posit are discussed.

Posit 3.1: Foraging is a heuristic, under which characteristics from genetic algorithms, Tabu Search, and Simulated Annealing can be grouped.

The empirical notions with which the foraging search is based upon are presented in Section 6.3. The dynamic memory structure (step 2c) notion parallels a similar approach found in the Simulated Annealing algorithm. Identifying good portions of solutions (step 2e) parallels a similar notion in Genetic Algorithms. Diversifying a search based on its relative progress (step 2d) parallels a similar construct in the Tabu Search. Therefore, notions of foraging encompass constructs from various heuristic solvers and can be viewed as a heuristic based on a model of intelligent searching in animals.

Posit 3.2: The Tabu Search can be used as the building block for the foraging solution algorithm.

Animals, searching for food in the wild, often utilize the memory of site visits to facilitate efficient progress through the search area. This memory-based search parallels the fundamental construct of the Tabu Search (TS). The Tabu Search is presented in Section 3.4.2 and some of the constructs from TS are used in Section 6.3 in the discrete portion of FALP, or foraging.

Posit 3.3: The ALP Algorithm along with foraging can be used to effectively solve mixed discrete/continuous problems.

In Section 6.3, the ALP Algorithm is integrated with the foraging heuristic and a step-by-step solution scheme is presented. This scheme, FALP, is shown in Section 6.4 to be an effective method to solve mixed discrete/continuous design problems. In fact, the solutions found by FALP are proven to be global optimum and are significantly better solutions than previous studies of the same problems. Therefore, there is evidence to suggest that the FALP Algorithm is capable of solving mixed discrete/continuous design problems effectively. Although, the efficiency of FALP has been addressed, there is room for future studies to optimize the search efficiency of the foraging heuristic.

The role of Chapter 6 in the dissertation is shown in Figure 6.18. In Chapter 6 the final step of the algorithm presented in Chapter 3 is detailed. Together with Chapters 4 and 5, the algorithm is complete. In Chapter 7, the developments of Chapters 4-6 are integrated and applied to the design of a subsonic passenger aircraft. Chapter 6 marks the end of Phase II of the strategy for verification, developing and testing the research hypotheses. Chapter 7 marks the beginning of Phase III, exercising and further verification of the algorithm.

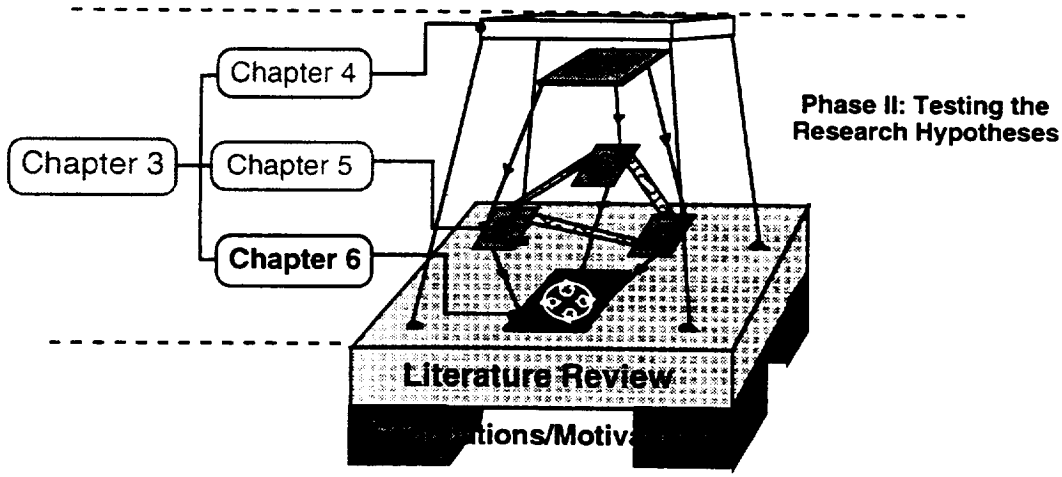


Figure 6.18. Frame of Reference: Chapter 6

CHAPTER 7

MULTIDISCIPLINARY DESIGN OF A PASSENGER TRANSPORT AIRCRAFT

In this chapter the motivating study of this dissertation, the design of a subsonic passenger transport aircraft, is presented and explored. The algorithm for integrated subsystem embodiment and system synthesis is demonstrated step by step for the aircraft study. The overall system and subsystem level requirements are presented in Section 7.1. A mathematical model of the aircraft is presented which consists of two coupled disciplines, aerodynamics and weights. The mathematical forms of the disciplinary compromise Decision Support Problems are presented in Section 7.2. The problem of integrating the design of these coupled compromise DSPs is used as the primary motivating case study to investigate the principal and secondary issues introduced in Chapter 1. In Sections 7.4 and 7.5, the compromise DSPs and the interactions between them are exercised based on the various game theory protocols introduced in Chapter 3. Through this exploration, Hypotheses I, II, and III, introduced in Chapter 3, are addressed. The Boeing 727-200 is currently flying, so why is it chosen as the focus? In (Mistree, et al., 1988), a compromise DSP model of a 727-200 is developed and exercised. It is demonstrated that the 727-200 design can be reproduced using this compromise DSP model. Although, the model used in this dissertation is modified from the one in (Mistree, et al., 1988), the 727-200 is used to illustrate the insights into the design of an existing system using an established baseline model. Aircraft are excellent examples of large scale systems, where many multi-disciplinary subsystems interact and interface continually, enabling the entire system to remain successful and profitable throughout its entire life cycle. Therefore, this work is not only applicable in the design of aircraft, but for large scale systems in general.

NOMENCLATURE

useful load fraction: The useful load is the difference between the empty weight established when the airplane is completed by the manufacturer and the gross weight, which is the maximum legal flying weight.

fuel balance: the equating of the fuel available and the fuel required.

Productivity Index: a measure of overall aircraft performance, given as $\frac{Payload * Block\ Speed}{Empty\ Weight + Fuel\ Weight}$ knots

climb gradient: the angle at which an aircraft can climb on take-off or missed approach.

aspect ratio: a geometric ratio describing the slenderness of the wing $\frac{b^2}{S}$

S	Wing area	ft^2
l	Fuselage length	ft
b	Wing span	ft
W_{to}	Take-off weight	Wto
T_i	Installed thrust	lbs
d	Fuselage diameter	ft
R	Range	nmi
AR	Aspect ratio	-
q_L	Climb gradient, landing	-
q_{TO}	Climb gradient, take-off	-
s_L	Landing field length	ft
s_{TO}	Take-off field length	ft
C_{doTO}	Drag coefficient, landing/take-off	-
C_{doC}	Drag coefficient, cruise	-
U	Useful load fraction	-
R_f	Fuel balance	-

R_{fa}	Fuel weight available	-
R_{fr}	Fuel weight required	-
W_{empty}/W_{to}	Empty weight ratio	-
W_f/W_i	Ratio of take-off weight to landing weight	-
PRI	Productivity Index	knots
V	Velocity	ft/sec
V_{br}	Best-range Velocity	ft/sec
t/c	Airfoil thickness-to-chord ratio	-
b_t	Fuel consumption	lb/lb-sec
ρ_{tl}	Density, take-off	slugs/ft^3
ρ_c	Density, cruise	slugs/ft^3
υ_{tl}	Kinematic viscosity, take-off	ft^2/sec
υ_c	Kinematic viscosity, cruise	ft^2/sec
l_{ref}	Reference length	ft
Ld_t	Lift-to-drag ratio, take-off	-
Ld_c	Lift-to-drag ratio, cruise	-
Ld_l	Lift-to-drag ratio, landing	-
e	Oswald factor	-
k	Quadratic drag polar	1/ft^2
c_L	Lift coefficient	-
S_s	Body wetted surface ratio	-
N_p	Number of passengers	-
W_{pay}	Payload	lbs
W_{fix}	Fixed equipment weight	lbs
c_{Lmax}	Maximum lift coefficient	-
N	Number of engines	-

7.1 THE SYSTEM DESIGN PROBLEM

The case study presented in this chapter is derived from the study in (Mistree, et al., 1988) where a compromise DSP template is developed for the Boeing 727-200 aircraft. In (Mistree, et al., 1988), it is shown that by using the technical template (no economical analysis), the existing 727-200 design is reproduced in an efficient and effective form. The template from the study in (Mistree, et al., 1988) is borrowed in this work with two significant additions.

- The original model is a *single-level* compromise DSP. This model is expanded into a multilevel model with two *distinct, but coupled* disciplinary compromise DSP. In order to accomplish this, *additional* analyses are developed associated with the weight ratios and overall productivity of the aircraft.
- Discrete variables are introduced for three design variables. Previously, these variables were assumed to be continuous.

In other words, the technical template used in (Mistree, et al., 1988) is augmented with more detailed weight and fuel ratio analyses, an overall measure of productivity, and the restriction of discrete design variables. The additional analyses are historically used in the design of subsonic passenger aircraft. Therefore, due to the success of the study in (Mistree, et al., 1988), it is asserted that the model used in this chapter represents the appropriate analysis in the design of a 727-200 aircraft. In Section 7.6, the 727-200 aircraft is used as a baseline for comparison

The 727-200 is a three-engined subsonic jet transport aircraft designed for short to medium ranges and short runway operations. It is considered to be one of the most successful jet transport aircraft ever produced. The aircraft is popular with the airlines because it can be operated profitably over various range segments and passenger load requirements. The 727 design was originally laid out in 1959 and 1960 - almost forty years ago. The airplane

is no longer in production, but as of September 1978 almost 1400 had been produced and the equivalent of ten billion 1978 dollars passed through the Boeing Aircraft Company.

A typical problem statement for the 727-200 could read, in part, as:

> *A three engined subsonic jet transport is to be designed. To ensure that the aircraft is operational from many airports the take-off field length should be less than 6,500 ft and the landing field length should be as close to 4,500 ft as possible. It is required that the aircraft have the capability of flying a range of 2,900 nmi.*
>
> *The aircraft will carry 188 passengers, and it is desirable that the aircraft have a useful load fraction of 0.5, a fuel balance of 1.0, and the Productivity Index should be maximized reflecting a sense of economic worth. It is also desirable that the climb gradients be as close to 3 percent as possible.*
>
> *At this early stage of design the variables to be determined are the wing span and area, fuselage length, installed thrust, and take-off weight. The solution should provide information on the size of the aircraft based on geometrical parameters, aerodynamic considerations and the Federal Air Regulations.*

The Boeing 727-200 mission requirements are summarized in Table 7.1 (Mistree, et al., 1988). Aircraft can fly many different ranges according to the payload. Transport aircraft are designed to achieve a maximum range for the desired payload. For the maximum mission requirements, a payload of 40,000 lbs. and a range of 2900 nautical miles is listed for the Boeing 727-200.

Table 7.1. Mission Requirements and System Parameters for the Boeing 727-200

Mission Requirements and System Parameters	Values
Range, R	2,900 nmi
Number of Passengers, Np	188
Payload (cargo and passengers), W_{pay}	40,000 lbs
Fixed equipment weight, W_{fix}	1100 lbs
Maximum Lift coefficient, c_{Lmax}	2.6
Number of Engines, N	3

The design constants, such as the number of engines, airfoil thickness-to-chord ratio, and number of passengers, are relatively the same for aircraft of comparable size and mission and have been principally taken from (Loftin, 1980). In complex systems design, typically system level analysis packages to design this type of aircraft are not available. However, with the availability of disciplinary level analysis packages, systems are typically partitioned into disciplinary subsystems which are designed by disciplinary design teams. The system variables, goals, and constraints are partitioned into subsystems, where detailed disciplinary analysis and synthesis packages are used to embody the subsystems. But the disciplinary design teams are rarely in isolation. Their state is one of semi-isolation; although they may be isolated geographically, by information, or organizationally, their decisions affect and are affected by the other disciplines' decisions. Therefore, some sort of coordination among their design teams and their associated analysis and synthesis routines is necessary. In the Boeing study, the system descriptors are partitioned into two disciplinary subsystems: the *aerodynamics* discipline and the *weights* discipline. The aerodynamics discipline is concerned with the aerodynamic profiles of the aircraft. Specifically, it focuses on the lift-to-drag ratios and drag coefficients of the wings and fuselage in order to maximize the lift of the aircraft from the wings and fuselage. The weights discipline is concerned with establishing a fuel balance for the aircraft by controlling the installed thrust and take-off weight. The weights discipline represents an integration of the typical aircraft disciplines, structures and propulsion. As presented in Section 5.2, it is asserted that the disciplinary design teams and their associated analysis and synthesis routines can be abstracted as players in a game. Each players model is mathematically described in Section 8.2.

The Boeing 727-200's design variables for each player and the upper and lower bounds used in exercising the aircraft template are listed in Tables 7.2 and 7.3 (Mistree, et al.,

1988) . These values were difficult to ascertain due to the many different versions of this aircraft; wherever possible the actual Boeing information was used. For the 727-200, the take-off weight varies between 175,000 to 220,000 lbs. depending on its options. The benchmark value used here was 210,000 lbs.(Maddalon, 1978). The aerodynamics player's design variables include one continuous design variable (wing area) and two discrete variables (wing span and fuselage length). The discrete variable represent a choice of available lengths for the materials of the fuselage and wings. It is assumed that the wing span and fuselage length must be *integers* between their upper and lower bounds. The weights player's design variables include one continuous variable (take-off weight) and one discrete variable (installed thrust). The discrete variable in this case represents a choice of available engines with given amounts of thrust. Therefore, design of a new engine is not required, and an existing engine can be selected. The possible thrust values for the installed thrust are given in Table 7.4.

Table 7.2. System Variables and Bounds for the Aerodynamics Player for the Boeing 727-200 Compromise DSP Template

Aerodynamic Variables	BOUNDS	BOEING 727-200
Continuous		
1. WING AREA (ft^2)	$1{,}200 \leq S \leq 2{,}500$	1,700
Discrete (Integers)		
2. FUSELAGE LENGTH (ft)	$105 \leq l \leq 150$	136
3. WING SPAN (ft)	$85 \leq b \leq 140$	108

Table 7.3. System Variables and Bounds for the Weights Player for the Boeing 727-200 Compromise DSP Template

Weight Variables	BOUNDS	BOEING 727-200
Continuous		
1. TAKE-OFF WEIGHT (lbs)	$140{,}000 \leq W_{TO} \leq 250{,}000$	210,000
Discrete		
2. INSTALLED THRUST (lbs)	$27{,}750 \leq T_i \leq 55{,}000$	48,000

Table 7.4. Possible Engine Thrust Values (lbs) for the Boeing Aircraft

27750	28500	30000	31000	33000
34000	36000	38000	40000	41000
42000	43000	45000	47000	48000
50000	51000	53000	55000	

The constraints of each player along with the limiting values are given in Tables 7.5 and 7.6. The goals of each player along with the target values are given in Tables 7.7 and 7.8. The mathematical form for the constraints and goals that constitute the technical requirements and aspirations are based on the work of (Loftin, 1980, Nicolai, 1984). The landing and take-off field length constraints and goals are common to both disciplines, since the constraints and goals are strong functions of the design variables of both disciplines. Satisfying the landing and take-off field lengths is paramount to establishing the feasibility and goodness of *both disciplines*. Likewise, the climb gradients are strong functions of the thrust and lift capacity. Therefore, both climb gradients are constraints and goals in *both disciplines*. It is not uncommon for disciplines to share the burden of satisfying one constraint (Bloebaum, 1991, Bloebaum, et al., 1992, Sobieszczanski-Sobieski, 1988). The other disciplinary constraints are strictly allocated and local to each discipline because of the innate dependence on the function of the discipline. The algorithm of this dissertation is able to handle both allocated and shared constraints and goals. The decision to allocate or share constraints is problem dependent, and must be made using domain-dependent knowledge, judgment and/or experience.

Table 7.5. Aerodynamics Player Constraints For The Boeing 727-200 Compromise DSP Template

Aerodynamics Constraints		REQUIREMENTS		
1. ASPECT RATIO	[AR]	≤	10.5	
2. CLIMB GRADIENT, landing	[q_L]	≥	2.4	percent
3. CLIMB GRADIENT, take-off	[q_{TO}]	≥	2.7	percent
4. LANDING FIELD	[S_L]	≤	4,500	ft
5. TAKE-OFF FIELD	[S_{TO}]	≤	6,500	ft
6. DRAG COEFFICIENT, take-off/landing	[Cdo_{TO}]	≤	0.02	
7. DRAG COEFFICIENT, cruise	[Cdo_C]	≤	0.02	

Note: System constraints must be satisfied for feasibility

Table 7.6. Weights Player Constraints For The Boeing 727-200 Compromise DSP Template

Weight Constraints		REQUIREMENTS		
1. USEFUL LOAD FRACTION	[U]	≥	0.3	
2. FUEL BALANCE	[R_f]	≥	1.0	
3. CLIMB GRADIENT, landing	[q_L]	≥	2.4	percent
4. CLIMB GRADIENT, take-off	[q_{TO}]	≥	2.7	percent
5. LANDING FIELD	[S_L]	≤	4,500	ft
6. TAKE-OFF FIELD	[S_{TO}]	≤	6,500	ft

Note: System constraints must be satisfied for feasibility

Table 7.7. Aerodynamics Player Goals For The Boeing 727-200 Compromise DSP Template

Aerodynamics Goals	TARGET VALUES		
1. CLIMB GRADIENT, landing	[q_{LTV}]	3.0	percent
2. CLIMB GRADIENT, take-off	[q_{TOTV}]	3.0	percent
3. ASPECT RATIO	[AR_{TV}]	10.5	
4. LANDING FIELD	[Sl_{TV}]	4,500	ft
5. TAKE-OFF FIELD	[Sto_{TV}]	4,500	ft

Note: System goals are to be achieved as far as possible

Table 7.8. Weights Player Goals For The Boeing 727-200 Compromise DSP Template

Weight Goals		TARGET VALUES	
1. PRODUCTIVITY INDEX	$[PI_{TV}]$	270	
2. USEFUL LOAD FRACTION	$[U_{TV}]$	0.5	
3. FUEL BALANCE	$[R_{fTV}]$	1.0	
4. CLIMB GRADIENT, landing	$[q_{LTV}]$	3.0	percent
5. CLIMB GRADIENT, take-off	$[q_{TOTV}]$	3.0	percent
6. LANDING FIELD	$[Sl_{TV}]$	4,500	ft
7. TAKE-OFF FIELD	$[Sto_{TV}]$	4,500	ft

Note: System goals are to be achieved as far as possible

In this next section the mathematical form of each players' compromise DSP is presented based on the information in Section 7.1.

7.2 THE SUBSYSTEM MODELS

Aircraft design involves the interaction and coordination of many disciplines and design teams. In this work, the focus is only on two of these disciplines and the interactions, results, and implications of different design scenarios. These two subsystems are aerodynamics and weights. Each subsystem model (compromise DSP) is described below. For both players, only the Archimedean form of the deviation function is used in this study. Again, the majority of the disciplinary compromise DSPs are taken from the single, system level compromise DSP in (Mistree, et al., 1988).

7.2.1 Aerodynamics Subsystem Model

Given

Airport performance requirements	
Federal Air Regulations	
Mission requirements	
aircraft maximum lift coefficient, c_{Lmax}	2.6
Number of engines, N	3.0
airfoil thickness-to-chord ratio, t/c	0.12
number of passengers, N_p	188
engine specific fuel consumption, b_t	0.00019444 lb/lb-sec
Range, R	2900 nmi, 1.762×10^7 feet
density, sea-level (take-off, landing), ρ_{tl}	0.002378 slugs/ft^3
kinematic viscosity, sea-level (take-off, landing), ν_{tl}	0.000156 ft^2/sec
kinematic viscosity, 35,000 ft (cruise), ν_c	0.000406 ft^2/sec
Velocity, sea level (take-off and landing), V_{tl}	220 ft/sec
density, 35,000 ft (cruise), ρ_c	0.000737 slugs/ft^3

Important Relationships and Equations (State Variables) **Eq. No.**

Zero lift drag coefficient

$$c_{Do} = (C_{Do})_{wing} + (C_{Do})_{body} + \Delta C_{Do} \quad [7.1]$$

wing contribution

$$(c_{DO})_{wing} = 1.1 c_{f,wing}\left(1 + 1.2\left(\frac{t}{c}\right) + 100\left(\frac{t}{c}\right)^4\right) S_{wet} \quad [7.2]$$

body contribution

$$(c_{DO})_{body} = c_{f,body}\left(1 + 0.0025\left(\frac{l}{d}\right) + 100\left(\frac{l}{d}\right)^3\right) S_S \quad [7.3]$$

skin friction coefficient

$$c_f = 0.455\left(\log_{10}\left(\frac{V_{ref} l_{ref}}{\nu}\right)\right)^{-2.58} \quad [7.4]$$

Velocity

$$V_{ref} = \begin{cases} V, \text{ if in take-off or landing} \\ V_{br}, \text{ if in cruise} \end{cases} \quad [7.5]$$

Reference length

$$l_{ref} = \begin{cases} l, \text{ for body} \\ c, \text{ for wing, } c = S / b \end{cases} \quad [7.6]$$

Body wetted surface ratio

$$S_S = \pi \frac{dl}{S}\left(1 - 2\frac{d}{l}\right)^{2/3}\left(1 + \left(\frac{d}{l}\right)^2\right) \quad [7.7]$$

Incremental drag coefficient

$$\Delta c_{Do} = 0.005 \quad [7.8]$$

Fuselage diameter

$$d = 1.83(4.325\frac{N_p}{l} + 1) \quad [7.9]$$

Oswald factor

$$e = 0.96[1 - (d/b)^2] \quad [7.10]$$

Quadratic drag polar

$$k = 1/[\pi(AR)e] \quad [7.11]$$

Lift-to-Drag, landing and take-off

$$\frac{L}{D} = \frac{c_L}{c_{DO} + kc_L^2} \quad [7.12]$$

Lift Coefficient, take-off and landing

$$c_L = \begin{cases} \frac{2W_{TO}}{\rho V^2 S}, \text{ if in take-off} \\ \frac{2W_{TO}(1-R_{fr})}{\rho V^2 S}, \text{ if in landing} \end{cases} \quad [7.13]$$

Lift-to-Drag, best cruise ratio

$$\frac{L}{D_{opt}} = \frac{1}{2}\frac{1}{\sqrt{c_{DO_{opt}} k}} \quad [7.14]$$

Velocity for best range

$$V_{BR} = \sqrt{\frac{2W_{TO}}{\rho S\sqrt{\frac{c_{DO}}{k}}}} \quad [7.15]$$

Landing Field Length

$$S_L = 400 + \frac{118 * W_{TO}(1-R_{fr})}{(c_{L\,max} S)} \quad [7.16]$$

Take-off Field Length

$$S_{TO} = \frac{20.9 * W_{TO}^2}{(c_{L\,max} S * T_i)} + 87\sqrt{\frac{W_{TO}}{(c_{L\,max} S)}} \quad [7.17]$$

Missed Approach Achievable Climb gradient, OEI, landing

$$q_L = -\left(\frac{L}{D}\right)_L^{-1} + \frac{N-1}{N}\frac{T_i}{W_{TO}(1-R_f)} \quad [7.18]$$

Achievable Climb gradient, OEI, take-off

$$q_{TO} = -\left(\frac{L}{D}\right)_T^{-1} + \frac{N-1}{N}\frac{T_i}{W_{TO}} \quad [7.19]$$

Aspect ratio

$$AR = \frac{b^2}{S} \qquad [7.20]$$

Find

The values of the **system variables**		**Units**
Continuous		
Wing area,	S	[ft^2]
Discrete		
Fuselage length,	l	[ft]
Wing span,	b	[ft]

The values of the **deviation variables** associated with
Climb Gradient, landing,
Climb Gradient, take-off,
Landing Field Length,
Take-off Field Length,
Aspect Ratio

Satisfy	**Units**	**Eq.No.**
The **system constraints** (non-normalized)		
The aspect ratio must be less than 10.5 $AR \leq 10.5$	[-]	[7.21]
The achievable climb gradient on landing must be greater than 2.4° $q_L \geq 2.4°$	[degrees]	[7.22]
The achievable climb gradient on take-off must be greater than 2.7° $q_{TO} \geq 2.7°$	[degrees]	[7.23]
The landing field length must be less than 4,500 ft. $S_L \leq 4{,}500$	[ft]	[7.24]
The take-off field length must be less than 6,500 ft. $S_{TO} \leq 6{,}500$	[ft]	[7.25]
The drag coefficient in take-off and landing must be less than 0.02 $C_{DoTL} \leq 0.02$	[-]	[7.26]
The drag coefficient in cruise must be less than 0.02 $C_{DoC} \leq 0.02$	[-]	[7.27]

The **system goals**[1] (normalized)

Missed Approach Climb Gradient, landing
$(q_L/0.03) + d_1^- - d_1^+ = 1$ [7.28]

Climb Gradient, take-off
$(q_{TO}/0.03) + d_2^- - d_2^+ = 1$ [7.29]

Landing Field Length
$S_L/4500 + d_3^- - d_3^+ = 1$ [7.30]

Take-off Field Length
$S_{TO}/4500 + d_4^- - d_4^+ = 1$ [7.31]

Aspect Ratio
$AR/10.5 + d_5^- - d_5^+ = 1$ [7.32]

The **bounds** on the **system variables**

Wing area (ft.)	1200	$\leq$	S	$\leq$	1500
Fuselage length (ft.)	105	$\leq$	l	$\leq$	150
Fuselage diameter (ft.)	85	$\leq$	d	$\leq$	140

Minimize

The sum of the **deviation variables**

$Z = \{P_1(d_1^- + d_1^+), P_2(d_2^- + d_2^+), P_3(d_3^- + d_3^+), P_4(d_4^- + d_4^+), P_5(d_5^- + d_5^+)\}$

7.2.2 Weights Subsystem Model

Given

Airport performance requirements	
Federal Air Regulations	
Mission requirements	
aircraft maximum lift coefficient, c_{Lmax}	2.6 (STO, SL)
Number of engines, N	3.0
engine specific fuel consumption, b_t	0.00019444 lb/lb-sec
Range, R	2900 nmi, 1.762×10^7 feet
payload (cargo and passengers), W_{pay}	40,000 lbs
fixed equipment weight, W_{fix}	1100 lbs

[1] It is important that the system goals are normalized so that the maximum values of the deviation variables are reasonably close.

Important Relationships and Equations (State Variables) **Eq. No.**

Fuel Weight Available

$$R_{fa} = 1 - \frac{W_{pay}}{W_{TO}} - \frac{W_{fix}}{W_{TO}} - \frac{W_{empty}}{W_{TO}} \quad [7.33]$$

Empty Weight Ratio

$$\frac{W_{empty}}{W_{TO}} = \frac{0.9592}{W_{TO}^{0.0638}} + 0.38\frac{T_i^{0.9881}}{W_{TO}} \quad [7.34]$$

Fuel Weight required for mission

$$R_{fr} = 1.1\left(1 - 0.95\frac{W_f}{W_i}\right) \quad [7.35]$$

Overall Fuel Balance

$$R_f = \frac{R_{fa}}{R_{fr}} \quad [7.36]$$

Ratio of take-off weight to landing weight

$$\frac{W_f}{W_i} = \exp\left(-\frac{Rb_t}{V_{cruise}\frac{L}{D_{cruise}}}\right) \quad [7.37]$$

Landing Field Length

$$S_L = 400 + \frac{118 * W_{TO}(1 - R_{fr})}{(c_{L\,max}S)} \quad [7.38]$$

Take-off Field Length

$$S_{TO} = \frac{20.9 * W_{TO}^2}{(c_{L\,max}S * T_i)} + 87\sqrt{\frac{W_{TO}}{(c_{L\,max}S)}} \quad [7.39]$$

Useful Load Fraction

$$U = 1.1(1 - 0.95e^{\frac{-b_t * R}{LD_c * V_{br}}}) + \frac{W_{pay}}{W_{TO}} \quad [7.40]$$

Achievable Climb gradient, OEI, landing

$$q_L = -\left(\frac{L}{D}\right)_L^{-1} + \frac{N-1}{N}\frac{T_i}{W_{TO}(1 - R_f)} \quad [7.41]$$

Achievable Climb gradient, OEI, take-off

$$q_{TO} = -\left(\frac{L}{D}\right)_T^{-1} + \frac{N-1}{N}\frac{T_i}{W_{TO}} \quad [7.42]$$

Productivity Index

$$PRI = \frac{V_{br}W_{pay}}{(1 - R_{fa} + R_{fr})W_{TO} - W_{pay} - W_{fix}} \quad [7.43]$$

Find

The values of the **system variables**		**Units**
Continuous		
Take-off weight,	W_{TO}	[lb]
Discrete		
Installed thrust,	T_i	[lb]

The values of the **deviation variables** associated with
the useful load,
the fuel balance,
the Productivity Index
the missed approach climb gradient, landing, OEI
the climb gradient, take-off, OEI
landing field length
take-off field length

Satisfy	**Units**	**Eq.No.**
The **system constraints** (non-normalized)		
The useful load must be greater than 0.3		
$U \geq 0.3$	[-]	[7.44]
The fuel available must be greater than the fuel required		
$R_f \geq 1.0$	[-]	[7.45]
The achievable climb gradient on landing must be greater than 2.4°		
$q_L \geq 2.4°$	[degrees]	[7.46]
The achievable climb gradient on take-off must be greater than 2.7°		
$q_{TO} \geq 2.7°$	[degrees]	[7.47]
The landing field length must be less than 4,500 ft.		
$S_L \leq 4{,}500$	[ft]	[7.48]
The take-off field length must be less than 6,500 ft.		
$S_{TO} \leq 6{,}500$	[ft]	[7.49]
The **system goals**[2] (normalized)		
The Productivity Index		
$PRI/270 + d_1^- - d_1^+ = 1$		[7.50]
Useful Load Fraction		

[2] It is important that the system goals are normalized so that the maximum values of the deviation variables are reasonably close.

$$U/0.5 + d_2^- - d_2^+ = 1 \qquad [7.51]$$

Fuel Balance

$$R_f + d_3^- - d_3^+ = 1 \qquad [7.52]$$

Missed Approach Climb Gradient, landing

$$(q_L/0.03) + d_4^- - d_4^+ = 1 \qquad [7.53]$$

Climb Gradient, take-off

$$(q_{TO}/0.03) + d_4^- - d_4^+ = 1 \qquad [7.54]$$

Landing Field Length

$$S_L/4500 + d_5^- - d_5^+ = 1 \qquad [7.55]$$

Take-off Field Length

$$S_{TO}/4500 + d_6^- - d_6^+ = 1 \qquad [7.55]$$

The **bounds** on the **system variables**

Installed thrust (lbs.)	27750	$\leq$	T_i	$\leq$	55000
Take-off weight (lbs.)	140,000	$\leq$	W_{TO}	$\leq$	250,000

Minimize

The sum of the **deviation variables**

$$Z = \{P_1(d_1^- + d_1^+), P_2(d_2^- + d_2^+), P_3(d_3^- + d_3^+), P_4(d_4^- + d_4^+), P_5(d_5^- + d_5^+), P_6(d_6^- + d_6^+), P_7(d_7^- + d_7^+)\}$$

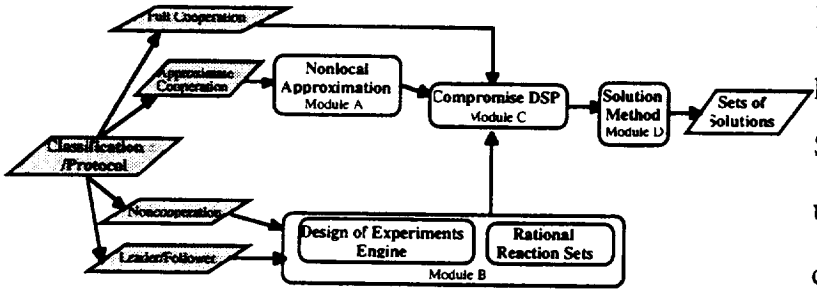
In the analytical model of the aerodynamics designer, there are three *control variables*[3] required by the aerodynamics designer from the weights designer. These are the design variables **take-off weight, W_{TO}**, and **installed thrust, T_i**, and the state variable, **fuel ratio required, R_{fr}**. In the analytical model of the weights player, there are five control variables required by the weights designer from the aerodynamics designer. These are the design variable **wing area, S**, and the state variables, **lift-to-drag ratios on take-off, cruise, and landing, Ld_t, Ld_c, Ld_l**, and the **best range velocity, V_{br}**.

---

[3] Defined in Section 5.3.

In Sections 7.3-7.5, the algorithm for integrated subsystem embodiment and system synthesis (see Chapter 3) is presented step by step for the aircraft study introduced in this section. It is stressed that even though the aircraft model has only two disciplines, the algorithm is applicable to *n disciplines*. In Section 7.3, the first step of the algorithm, the formulation and classification of the problem and process, is presented. Figure 1.8, the infrastructure of the algorithm, is used as a frame of reference through Sections 7.3-7.6.

7.3 STEP 1: FORMULATION OF PROBLEM AND PROCESS



In this section, the aircraft problem described in Section 7.2 is classified using the lexicon and classification presented in Chapter 4. The classification framework consists of three levels. In Figure 7.1, a sample of the possible classifications based on various game theory protocols with example linguistic entities at each level are shown. In no means is the classification in Figure 7.1 complete.

Level 1

Since the Boeing aircraft model consists of two disciplinary subsystems, it is a multi-level model. At the system level, since the model is solved using simultaneous analysis and design, it is SAND at the system level. At the subsystem level, the model is also solved using simultaneous analysis and design. Therefore, the Level 1 classification is Multi-SAND-SAND (shown at the top of Figure 7.1). Since it is a multi-level model, the subsystems are designated according to the design and solution process. Normally, in a

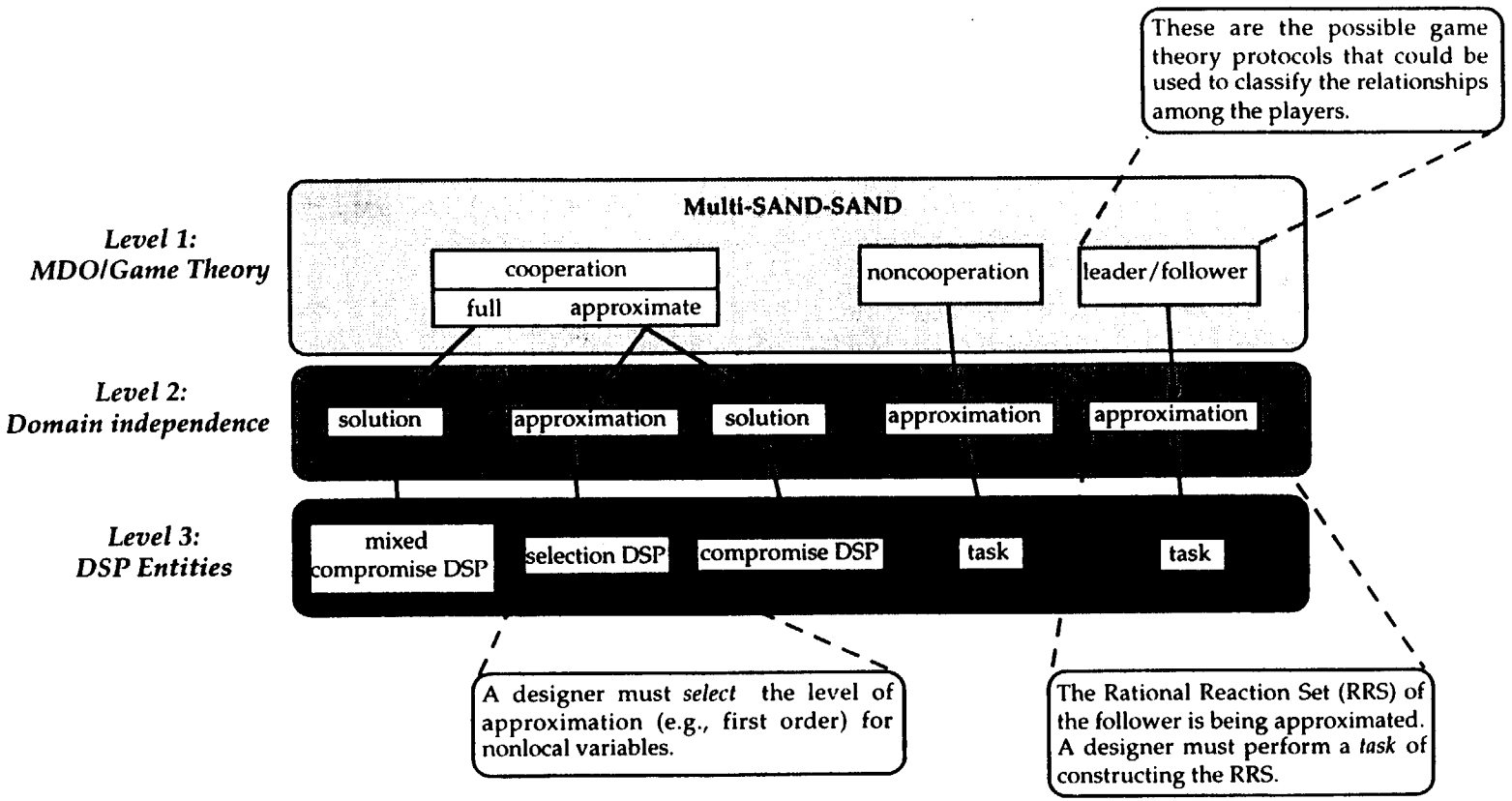


Figure 7.1 Sample Aircraft Classification

specific application of the algorithm, *one* protocol is identified and a single classification is made. Since the aircraft model is exercised in this chapter for *all* possible player protocols, there is no one classification in this study. Therefore, in Figure 7.1, all four possible game classifications based on protocol are shown as representative terms. Note that in the full cooperative protocol, the two disciplinary models are combined into one model and solved. In only this case, the designation would be *single-level.* Using the other protocols, it is a *multi-level* approach.

Level 2

The classification terms of Level 2 depend upon the classification of Level 1. For instance, in Figure 7.1, if the leader/follower formulation is used, the leader uses an *approximation* of the RRS of the follower. In the approximate cooperative formulation, each player uses an *approximation* of the other player's state variables. As another example from Figure 7.1, in the full cooperative formulation, it is required to find a *solution* to the model.

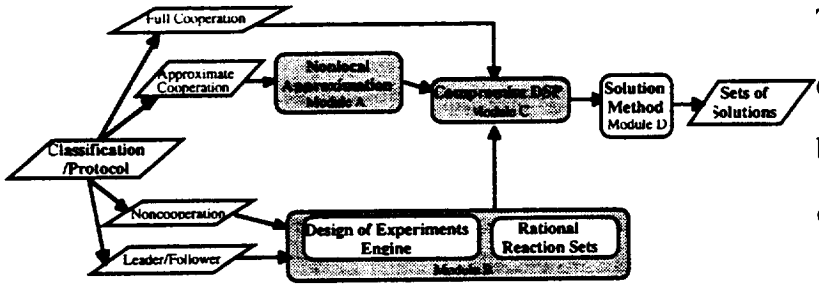
Level 3

The classification terms of Level 3 further classify the terms in Level 2 according to the DSPT entities (see Section 4.1.2). From Figure 7.1, the *approximation* of the follower's RRS in a leader/follower formulation is a *task.* Therefore, as part of the leader's decision resolution, he must perform a *task* of constructing the RRS of the follower. Construction of a RRS is illustrated in Section 7.4.2. Also, from Figure 7.1, in the approximate cooperative protocol, each player must decide how approximate to make the approximation of nonlocal state variables. In other words, they must make a *selection* decision. In this dissertation, as shown in Section 7.4.1, a first-order approximation scheme is used. Therefore, the *selection* decision has been made. In the full cooperative protocol, also in Figure 7.1, since there exists both discrete and continuous variables in the cooperative

model, solving a *mixed compromise DSP* is required. Solution of such a model is illustrated in Section 7.5.

In this chapter various game formulations of the aircraft design problem are explored. Even though the formulations are different in theory and implementation, the classification of the problem and process does not drastically change. The only part of the classification that changes is the subsystem designations in Level 1, as shown in Figure 7.1. The primary difference is the solution implementation. The solution implementation process is determined by executing the tasks and solving the decision support problems identified in Level 3. This is one of the advantages of the lexicon: even though the solution process may change drastically as a result of changes in the subsystem designations, the overall classification of the problem does not significantly change. In the next section, Step 2 of the algorithm is described using the aircraft study.

7.4 STEP 2: FORMULATE THE DISCIPLINARY PROBLEMS



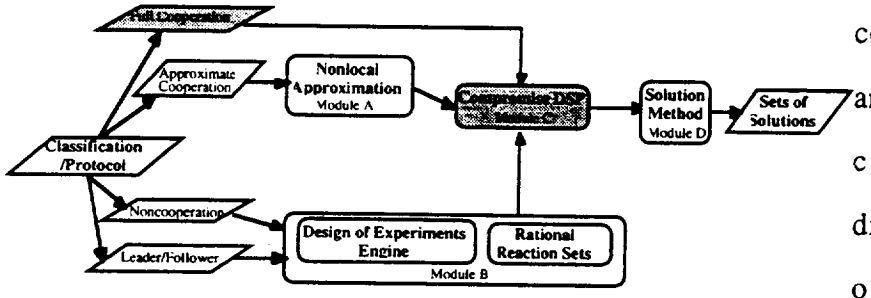
The formulation of the disciplinary problems is based on the classification of the subsystems in Step 1. In this study all possible classifications of the players are studied. With each protocol, there is a specific resolution procedure outlined in Sections 7.4.1-7.4.3. The solution of each protocol for the Boeing aircraft is outlined in Step 3 of the algorithm in Section 7.6.

7.4.1 Cooperative Protocol

In game theory, the cooperative protocol is implemented by combining the individual players' models and finding a Pareto optimal solution(s) (Vincent and Grantham, 1981). However, in complex systems design, as discussed in Section 5.1, combining the models, analysis, and synthesis routines of each player is not practical. Therefore, an approximate form of cooperation is studied along with the full cooperative formulation.

Full Cooperation

Since each discipline in complex systems design typically has their own model with the corresponding analysis and synthesis software, combining the disciplinary models into one encompassing model is highly unlikely. With the aircraft study, the disciplinary compromise DSPs, presented in Section 7.2, are combined into one compromise DSPs, as shown in Figure 7.2, in order to illustrate the ideal cooperative results. The compromise DSP, which combines the two disciplinary compromise DSP, is used as the fundamental decision-making construct. This should theoretically be the best solution, as full cooperation is achieved because each player is exposed to the other player's state and design variables (control vector) at all times. This is the ideal protocol, but rarely achievable.



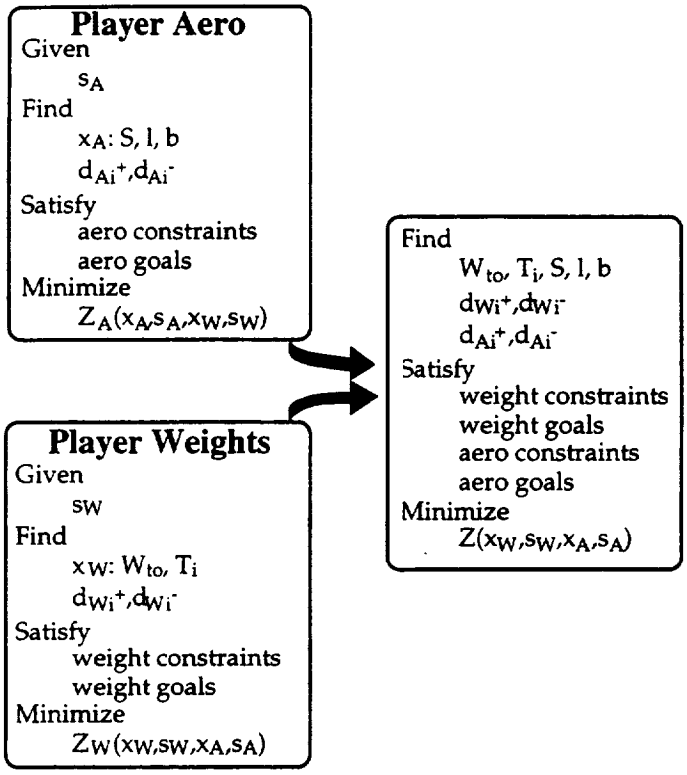
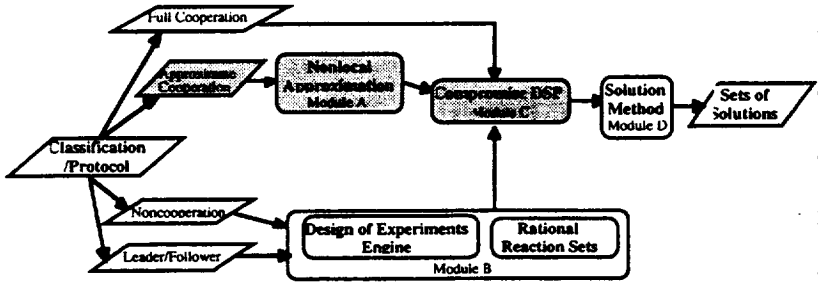


Figure 7.2. Combining the Players' Compromise DSP: Full Cooperation

Approximate Cooperation



Approximate cooperation is achieved using approximations of the required state variables of the other players. Only the coupled equations (i.e., equations that are functions of the variables of two or more players) are approximated. This approximation is accomplished using the Global Sensitivity Equations (GSE) method first proposed in (Sobieszczanski-Sobieski, 1988). Using the GSE, the total derivatives of the dependent variables can be solved for as

functions of the independent variables *from every player.* These total derivatives then are used in a Taylor's expansion of nonlocal state variables,

$$s_2(x, s_1) \approx s_2^o + \Sigma \frac{ds_2}{dx}(x_n - x_o). \tag{7.56}$$

Therefore, each player uses an approximation of the coupled state variables of the other players. Using the procedure from Section 5.5.1 to model approximate cooperation, the following steps are performed for the aircraft problem. Computer implementation of each step is given in Appendix C.

❶ Construct approximations of nonlocal behavior variables
  ❶ a. *Determine the partial derivative matrices, [M] and [B], for each player using exact calculations or appropriate approximation techniques.*

In the case of the Boeing case study, the exact derivatives in Step ❶ are determined using Mathematica®. If the derivatives require some complex analysis package and exact derivatives can not be found, certainly the derivatives can be approximated by finite differencing. In addition, with the advent of ADIFOR (Bischof, et al., 1992), representing the exact derivatives of functions written in FORTRAN is becoming computationally easier. Although ADIFOR was not used in this work, it is an area of future interest to expedite the formulation and solution of the approximate cooperative protocol.

For the aircraft case study, the symbolic [M] matrix of partial derivatives of the state variables with respect to the other state variables is given in Figure 7.3. The symbolic [B] matrix of local partial derivatives of the state variables with respect to the design variables is given in Figure 7.4. The state and design variables of each player are presented in Section 7.2.

	d	$c_{DOTO/L}$	C_{DOc}	$L/D\|_T$	$L/D\|_c$	$L/D\|_L$	V_{BR}	AR	q_L	q_{TO}	S_L	S_{TO}	R_{fr}	R_{fa}	R_f	U	PI
d	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{DOTO/L}$	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C_{DOc}	0	0	1	0	0	0	$-\frac{\partial c_{DOc}}{\partial V_{BR}}$	0	0	0	0	0	0	0	0	0	0
$L/D\|_T$	$-\frac{\partial L/D_T}{\partial d}$	$-\frac{\partial L/D_T}{\partial c_{DO/L}}$	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
$L/D\|_c$	$-\frac{\partial L/D_C}{\partial d}$	0	$-\frac{\partial L/D_C}{\partial c_{DOc}}$	0	1	0	0	0	0	0	0	0	0	0	0	0	0
$L/D\|_L$	$-\frac{\partial L/D_L}{\partial d}$	$-\frac{\partial L/D_L}{\partial c_{DO/L}}$	0	0	0	1	0	0	0	0	0	0	$-\frac{\partial L/D_L}{\partial R_{fr}}$	0	0	0	0
V_{BR}	$-\frac{\partial V_{BR}}{\partial d}$	0	$-\frac{\partial V_{BR}}{\partial c_{DOc}}$	0	0	0	1	0	0	0	0	0	0	0	0	0	0
AR	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
q_L	0	0	0	0	0	$-\frac{\partial q_L}{\partial L/D_L}$	0	0	1	0	0	0	$-\frac{\partial q_L}{\partial R_{fr}}$	0	0	0	0
q_{TO}	0	0	0	$-\frac{\partial q_{TO}}{\partial L/D_T}$	0	0	0	0	0	1	0	0	0	0	0	0	0
S_L	0	0	0	0	0	0	0	0	0	0	1	0	$-\frac{\partial S_L}{\partial R_{fr}}$	0	0	0	0
S_{TO}	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
R_{fr}	0	0	0	0	$-\frac{\partial R_{fr}}{\partial L/D_c}$	0	$-\frac{\partial R_{fr}}{\partial V_{BR}}$	0	0	0	0	0	1	0	0	0	0
R_{fa}	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
R_f	0	0	0	0	0	0	0	0	0	0	0	0	$-\frac{\partial R_f}{\partial R_{fr}}$	$-\frac{\partial R_f}{\partial R_f}$	1	0	0
U	0	0	0	0	$-\frac{\partial U}{\partial L/D_c}$	0	$-\frac{\partial U}{\partial V_{BR}}$	0	0	0	0	0	0	0	0	1	0
PI	0	0	0	0	0	0	$-\frac{\partial PI}{\partial V_{BR}}$	0	0	0	0	0	$-\frac{\partial PI}{\partial R_{fr}}$	$-\frac{\partial PI}{\partial R_{fa}}$	0	0	1

Figure 7.3. [M] Matrix of GSE

	S	b	l	T_i	W_{TO}
d	0	0	$\frac{\partial d}{\partial l}$	0	0
$c_{DOTO/L}$	$\frac{\partial c_{DO_{T/l}}}{\partial S}$	$\frac{\partial c_{DO_{T/l}}}{\partial b}$	$\frac{\partial c_{DO_{T/l}}}{\partial l}$	0	0
C_{DOc}	$\frac{\partial c_{DOc}}{\partial S}$	$\frac{\partial c_{DOc}}{\partial b}$	$\frac{\partial c_{DOc}}{\partial l}$	0	0
L/D_T	$\frac{\partial L/D_T}{\partial S}$	$\frac{\partial L/D_T}{\partial b}$	0	0	$\frac{\partial L/D_T}{\partial W_{TO}}$
L/D_c	$\frac{\partial L/D_c}{\partial S}$	$\frac{\partial L/D_c}{\partial b}$	0	0	0
L/D_L	$\frac{\partial L/D_L}{\partial S}$	$\frac{\partial L/D_L}{\partial b}$	0	0	$\frac{\partial L/D_L}{\partial W_{TO}}$
V_{BR}	$\frac{\partial V_{BR}}{\partial S}$	$\frac{\partial V_{BR}}{\partial b}$	0	0	$\frac{\partial V_{BR}}{\partial W_{TO}}$
AR	$\frac{\partial AR}{\partial S}$	$\frac{\partial AR}{\partial b}$	0	0	0
q_L	0	0	0	$\frac{\partial q_L}{\partial T_i}$	$\frac{\partial q_L}{\partial W_{TO}}$
q_{TO}	0	0	0	$\frac{\partial q_{TO}}{\partial T_i}$	$\frac{\partial q_{TO}}{\partial W_{TO}}$
S_L	$\frac{\partial S_L}{\partial S}$	0	0	0	$\frac{\partial S_L}{\partial W_{TO}}$
S_{TO}	$\frac{\partial S_{TO}}{\partial S}$	0	0	$\frac{\partial S_{TO}}{\partial T_i}$	$\frac{\partial S_{TO}}{\partial W_{TO}}$
R_{fr}	0	0	0	0	0
R_{fa}	0	0	0	$\frac{\partial R_{fa}}{\partial T_i}$	$\frac{\partial R_{fa}}{\partial W_{TO}}$
R_f	0	0	0	0	0
U	0	0	0	0	$\frac{\partial U}{\partial W_{TO}}$
PI	0	0	0	0	$\frac{\partial PI}{\partial W_{TO}}$

Figure 7.4. [B] matrix of GSE

❶ b. *Solve the GSE system of equations, [M][X] = [B] for the total derivatives, [X].*

This is performed using a matrix solver based on Gaussian elimination. The returned values are the total derivatives of the state equations with respect to the design variables, $\frac{ds}{dx}$.

❶ c. *Construct approximations of the non-local state variables, s_2, using Taylor series.*

$$s_2(x, s_1) \approx s_2^o + \Sigma \frac{ds_2}{dx}(x_n - x_o)$$

The compromise DSPs for the two players, aero and weights, in the approximate cooperative formulation are shown in Figure 7.5. The compromise DSPs are the *same* core compromise DSPs as presented in Section 7.2, except for additional *given* information in the form of nonlocal approximations. Each player is approximating the required state variables needed from the other players. Player aero approximates R_{fr}, while player weights approximates Ld_t, Ld_c, Ld_l, V_{br}.

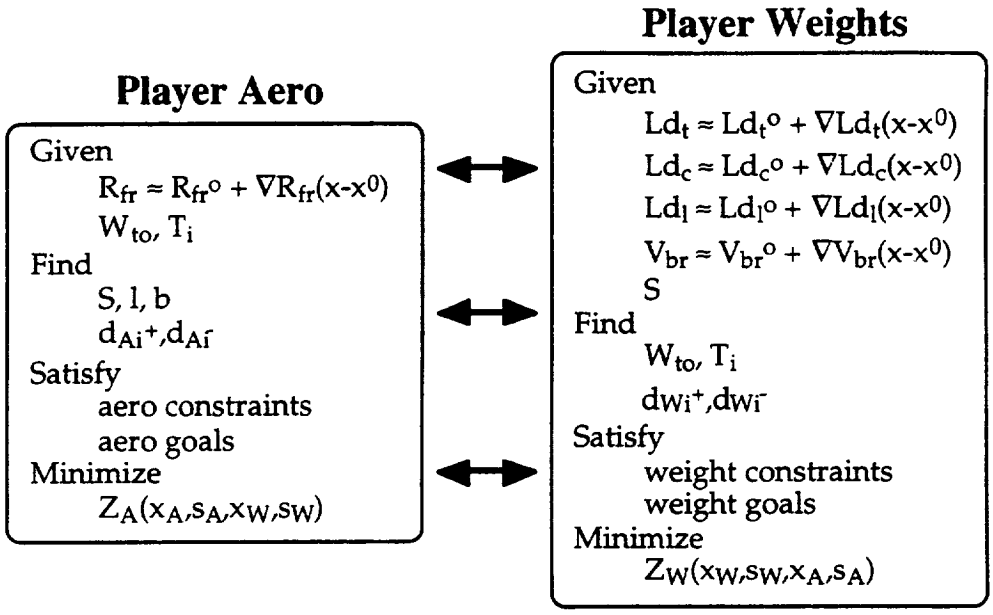


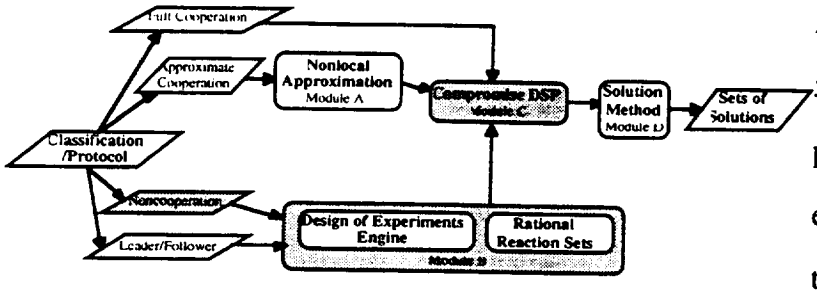
Figure 7.5. Approximate Compromise DSPs: Aircraft Study

The resulting compromise DSPs are then solved and coordinated in Step 3 of the algorithm, as described in Section 7.5.1.

❷ Solve disciplinary models using nonlocal approximations.
❸ If the models of both players have converged, then stop. If not, update GSE matrices, and goto Step ❶b.

The solution of the model using the approximate cooperation formulation is explored in Section 7.5.1.

7.4.2 Noncooperative Protocol



As defined in Section 5.5.2, the noncooperative protocol is exercised when each player does not know the information about the other players. Therefore, each player must construct his set of solutions based on unknown information about the other players. Mathematically, this is equivalent to each player constructing their own RRS which dictates how each player will react regardless of the other players' decisions. Computationally, the RRS are constructed by linking NORMAN® with DSIDES. Based on the degree of the desired response surface and design of experiments, a certain number of simulation experiments are run. This "experiment" entails solving a player's compromise DSP using DSIDES for a given set of constant parameters which are needed from and part of the other player's control vector. In Figure 7.6, a representation of the repeated DSIDES runs is shown to construct the RRS of the weights player. Different values of x_A and s_A are used in the weights' compromise DSP according to the number of experiments needed. The weights' compromise DSP is the *same* as presented in Section 7.2. The specific steps to construct the RRS approximation are presented in Section 5.5.2 and applied to the aircraft problem as follows.

❶ Use NORMAN® as the design of experiments driver.

1a) Based on the number of input variables, set-up the Central Composite Face-Centered Design.

1b) Set the input variables (variables of the follower player which are required) constant in P1's compromise DSP and call DSIDES.

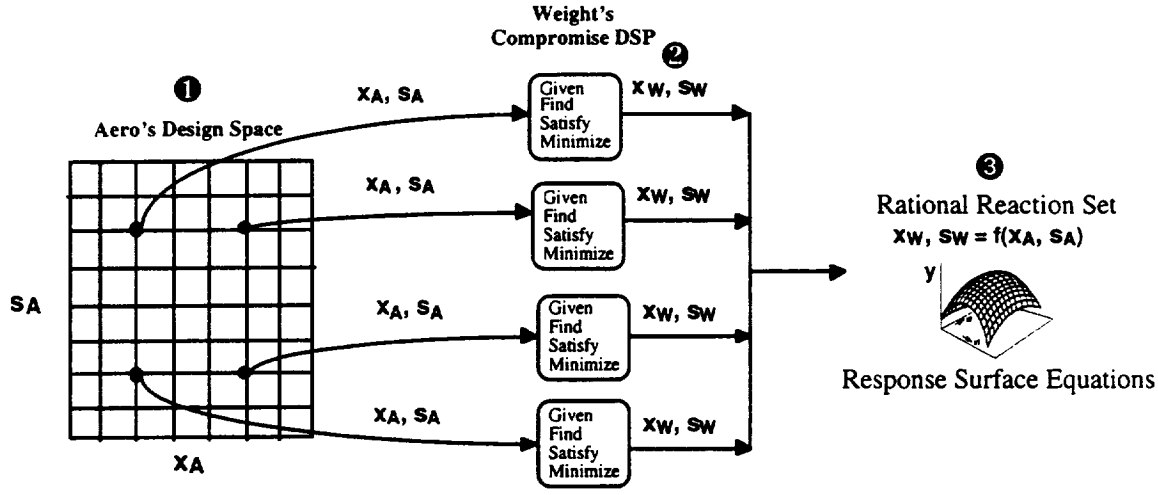


Figure 7.6. Construction of Weights' RRS

In the top half of Figure 7.7, the two players' compromise DSPs in a noncooperative protocol are given. The compromise DSPs in Figure 7.7 are the *same* ones as presented in Section 7.2. Theoretically, the information required from another player is unknown (represented by the wall in Figure 7.7). Based on the procedure presented in Section 5.5.2, the unknown information in this dissertation is simulated using the experimental design techniques. The resulting compromise DSPs of the players in the noncooperative protocol are shown in the lower half of Figure 7.7. These compromise DSPs are still the *same* core compromise DSPs (Section 7.2), except the *given* information includes simulation values of the previously unknown parameters. Each player, based on the given ranges of the nonlocal variables, uses simulated points in the design space of the other player in their local compromise DSP. The theory and heuristics behind the selection of the simulated points are embedded in the experimental design package, NORMAN.

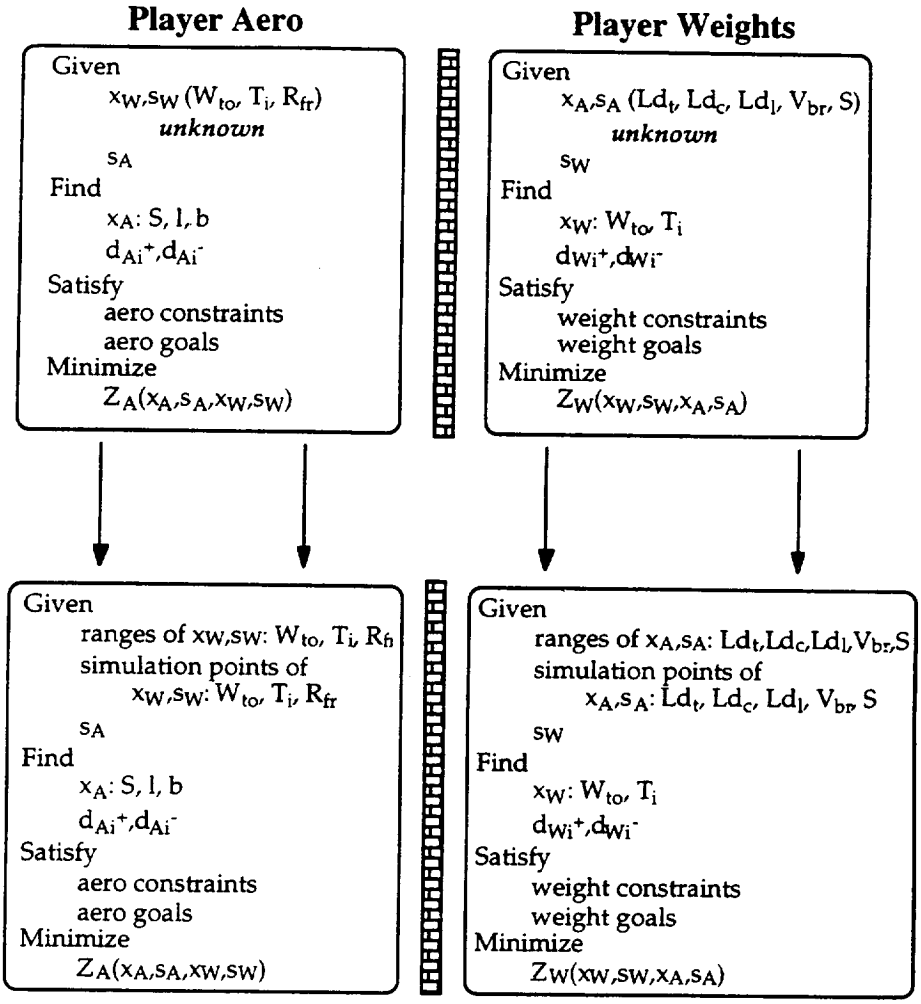


Figure 7.7. Noncooperative Compromise DSPs: Aircraft Study

❷ In DSIDES, solve P1's compromise DSP using the ALP Algorithm and send the values of the design and state variables to NORMAN®.

The compromise DSPs in the lower half of Figure 7.7 are solved at each simulation point. Although the compromise DSP may contain discrete design variables, it is assumed that the design variables are *all continuous* in constructing the RRS. This step is also depicted in Figure 7.6 where multiple compromise DSPs are solved at each point in P1's (aerodynamics) design space.

❸ Determine if the full experiment is finished.

- If not, continue by moving to the next experiment point and repeat Step ❷.
- If so, construct the response surfaces.

For the case of the Boeing aircraft there are 5 variables needed *from* the weights player *by* the aerodynamics player and 3 variables needed *from* the aerodynamics player *by* the weights player. Therefore, the aerodynamics player constructs a set of 5 equations with 3 unknowns (Eqn. 7.57), and the weights player constructs a set of 3 equations with 5 unknowns (Eqn. 7.58). The form of the approximate second order RRS's, $D_{PLAYER\ I}$, of each player are given below.

$\mathbf{D_A}$ =

$$\begin{aligned}
&\{S = C_0 + C_1W_{to} + C_2T_i + C_3R_{fr} + C_{12}W_{to}{*}T_i + C_{13}W_{to}{*}R_{fr} + C_{23}T_i{*}R_{fr} + C_{11}W_{to}^2 + C_{22}T_i^2 + C_{33}R_{fr}^2\\
&Ld_t = C_0 + C_1W_{to} + C_2T_i + C_3R_{fr} + C_{12}W_{to}{*}T_i + C_{13}W_{to}{*}R_{fr} + C_{23}T_i{*}R_{fr} + C_{11}W_{to}^2 + C_{22}T_i^2 + C_{33}R_{fr}^2\\
&Ld_c = C_0 + C_1W_{to} + C_2T_i + C_3R_{fr} + C_{12}W_{to}{*}T_i + C_{13}W_{to}{*}R_{fr} + C_{23}T_i{*}R_{fr} + C_{11}W_{to}^2 + C_{22}T_i^2 + C_{33}R_{fr}^2\\
&Ld_l = C_0 + C_1W_{to} + C_2T_i + C_3R_{fr} + C_{12}W_{to}{*}T_i + C_{13}W_{to}{*}R_{fr} + C_{23}T_i{*}R_{fr} + C_{11}W_{to}^2 + C_{22}T_i^2 + C_{33}R_{fr}^2\\
&V_{br} = C_0 + C_1W_{to} + C_2T_i + C_3R_{fr} + C_{12}W_{to}{*}T_i + C_{13}W_{to}{*}R_{fr} + C_{23}T_i{*}R_{fr} + C_{11}W_{to}^2 + C_{22}T_i^2 + C_{33}R_{fr}^2.\}
\end{aligned} \tag{7.57}$$

$\mathbf{D_W}$ =

$$\begin{aligned}
&\{W_{to} = C_0 + C_1S + C_2Ld_t + C_3Ld_c + C_4Ld_l + C_5V_{br} + C_{12}S{*}Ld_t + C_{13}S{*}Ld_c + C_{14}S{*}Ld_l + C_{15}S{*}V_{br} + C_{23}Ld_t{*}Ld_c + C_{24}Ld_t{*}Ld_l + C_{25}Ld_t{*}V_{br} + C_{34}Ld_c{*}Ld_l + C_{35}Ld_c{*}V_{br} + C_{45}Ld_l{*}V_{br} + C_{11}S^2 + C_{22}Ld_t^2 + C_{33}Ld_c^2 + C_{44}Ld_l^2 + C_{55}V_{br}^2\\
&T_i = C_0 + C_1S + C_2Ld_t + C_3Ld_c + C_4Ld_l + C_5V_{br} + C_{12}S{*}Ld_t + C_{13}S{*}Ld_c + C_{14}S{*}Ld_l + C_{15}S{*}V_{br} + C_{23}Ld_t{*}Ld_c + C_{24}Ld_t{*}Ld_l + C_{25}Ld_t{*}V_{br} + C_{34}Ld_c{*}Ld_l + C_{35}Ld_c{*}V_{br} + C_{45}Ld_l{*}V_{br} + C_{11}S^2 + C_{22}Ld_t^2 + C_{33}Ld_c^2 + C_{44}Ld_l^2 + C_{55}V_{br}^2\\
&R_{fr} = C_0 + C_1S + C_2Ld_t + C_3Ld_c + C_4Ld_l + C_5V_{br} + C_{12}S{*}Ld_t + C_{13}S{*}Ld_c + C_{14}S{*}Ld_l + C_{15}S{*}V_{br} + C_{23}Ld_t{*}Ld_c + C_{24}Ld_t{*}Ld_l + C_{25}Ld_t{*}V_{br} + C_{34}Ld_c{*}Ld_l + C_{35}Ld_c{*}V_{br} + C_{45}Ld_l{*}V_{br} + C_{11}S^2 + C_{22}Ld_t^2 + C_{33}Ld_c^2 + C_{44}Ld_l^2 + C_{55}V_{br}^2.\}
\end{aligned} \tag{7.58}$$

Since these RRS's are being approximated using design of experiments and statistical techniques, there are two issues dealing with *significance* that must be addressed. The first is the significance of the *input factors* of each response surface. It is assumed in constructing that the response surfaces that every nonlocal variable is significant in the model. This is observed in Eqns. 7.57 and 7.58 where *every* nonlocal variable is in *each* approximate model. For instance, in Eqn. 7.57, the response surfaces for S and Ld_t include *all* main factor terms (W_{to}, T_i, R_{fr}), *every* interaction term ($W_{to}*T_i$, $W_{to}*R_{fr}$, T_i*R_{fr}), and *every* second order squared term (W_{to}^2, T_i^2, R_{fr}^2). Once the response surface is constructed, some terms in the model certainly may turn out to be insignificant, but in the construction it is assumed that every term is significant.

The second issue is the significance of the *fitted response model.* The response surfaces that are used to predict the behavior of the design variables (S, W_{to}, and T_i) are approximations of quantities which have no readily available mathematical expression. There are *no* closed form expressions for design variables, as they are the indepedent variables of a model. Therefore, using the response surfaces for the design variables is a way to predict how another player will solve his model. With the state variables (Ld_t, Ld_c, Ld_l, V_{br}, R_{fr}), closed form expressions *do exist*, and are used in the local subsystem models. However, the response surfaces are being used to predict the values of the state variables as functions of *only the required nonlocal variables*. There certainly could be more *local* variables in the actual expression of the state variable which are not accounted for in the response surface approximation. For example, a representative state variable, $\mathbf{s_1}$, may be a function of the local design variables, $\mathbf{x_1}$, other local state variables, $\mathbf{s_1}$, and required nonlocal design and state variables, $\mathbf{x_2}$, and $\mathbf{s_2}$,

$$\mathbf{s_1} = f(\mathbf{x_1}, \mathbf{s_1}, \mathbf{x_2}, \mathbf{s_2}).$$

Yet, in the response surface approximation of $\mathbf{s_1}$, the value of $\mathbf{s_1}$ is being predicted only as a function of the required nonlocal design and state variables, $\mathbf{x_2}$, and $\mathbf{s_2}$,

$$\mathbf{s_1} = f(\mathbf{x_2}, \mathbf{s_2}).$$

This is, however, the essence of the rational reaction set. Therefore, the significance of the response surface regression is used to measure the significance of the effects of the nonlocal variables on a given local variable. Specific instances of the regression significance for the aircraft problem are investigated in Section 7.5.2.

In the Boeing study, the unknown parameters which the aerodynamics player *needs from* the weights player are:

W_{to} (Take-off Weight), T_i (Installed Thrust), and R_{fr} (Fuel Ratio Required).

The design of experiments is set-up based on the minimum and maximum values of these parameters, which are given below.

$$140{,}000 \leq W_{to} \leq 250{,}000$$

$$27750 \leq T_i \leq 55{,}000$$

$$0.2 \leq R_{fr} \leq 0.6$$

The number of experiments needed in a Central Composite Design is $2^n + 2n + 1$, where n is the number of input variables. With 3 input variables, 15 experiments are run for various values of W_{to}, T_i, and R_{fr} to approximate Eqn. 7.57.

Conversely, the weights player also needs some of the variables *from* the aerodynamics player. The variables needed by the weights player in this study are:

S (Wing Area), Ld_t (Lift-to-drag for take-off), Ld_c (Lift-to-drag for cruise), Ld_l (Lift-to-drag for landing), and V_{br} (Best-Range Velocity).

Only three variables are needed by the aerodynamics player, but five variables are needed by the weights player. With the increased number of input variables for the weights player

the number of experiments needed to construct the RSS of the weights player (Eqn. 7.58) increases from 15 to 43 ($2^5 + 2 \cdot 5 + 1$) using the same order of response surface. The range of the variables needed by the weights player are

$$1200 \leq S \leq 2500$$

$$5 \leq Ld_t \leq 17$$

$$12 \leq Ld_c \leq 20$$

$$8 \leq Ld_l \leq 18$$

$$500 \leq V_{br} \leq 1000.$$

The construction of the RRS of each player is the first step to solving the noncooperative formulation. The three steps to solving the noncooperative formulation, as introduced in Section 5.5.2, are given for the aircraft problem.

❶ Construct Rational Reaction Set of Each Player

❷ Using appropriate technique find the intersection points of the RRS's of each player.

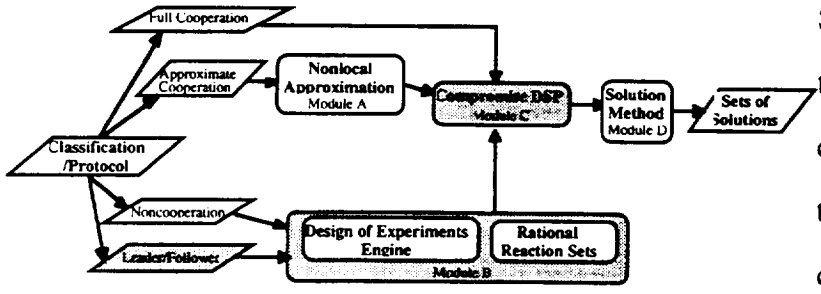
$$\mathbf{X^* = D_A \cap D_W} \qquad (7.59)$$

❸ Determine which solutions fall in the ranges of the design variables.

The noncooperative solutions for the aircraft problem are explored in Section 7.5.3.

7.4.3 Leader/Follower

The calculation of the rational reaction sets is not only paramount to the noncooperative protocol, but also is intrinsic to the Stackelberg leader/follower protocol. As introduced in Section 5.5.3, the RRS of the Follower must be constructed so the Leader in the game knows what decision the Follower is going to make. However, the Follower must construct the RRS without knowing anything



(besides the variable ranges in this case) about the Leader (by definition). The solution procedure for the leader/follower protocol is presented in Section 5.5.3 and applied to the aircraft problem as follows.

❶ Construct the RRS of the follower.

As defined in Section 5.5.2, the RRS is an approximation of the actual RRS. The procedure for constructing the RRS is the same as in the noncooperative protocol shown in Sections 5.5.2 and 7.4.2.

❷ Allowing the leader access to the follower's RRS in the leader's compromise DSP, solve the leader's compromise DSP.

The leader's compromise DSP is augmented using the RRS of the follower. The decision-making strategy of the follower is embodied in the RRS and now available to the leader.

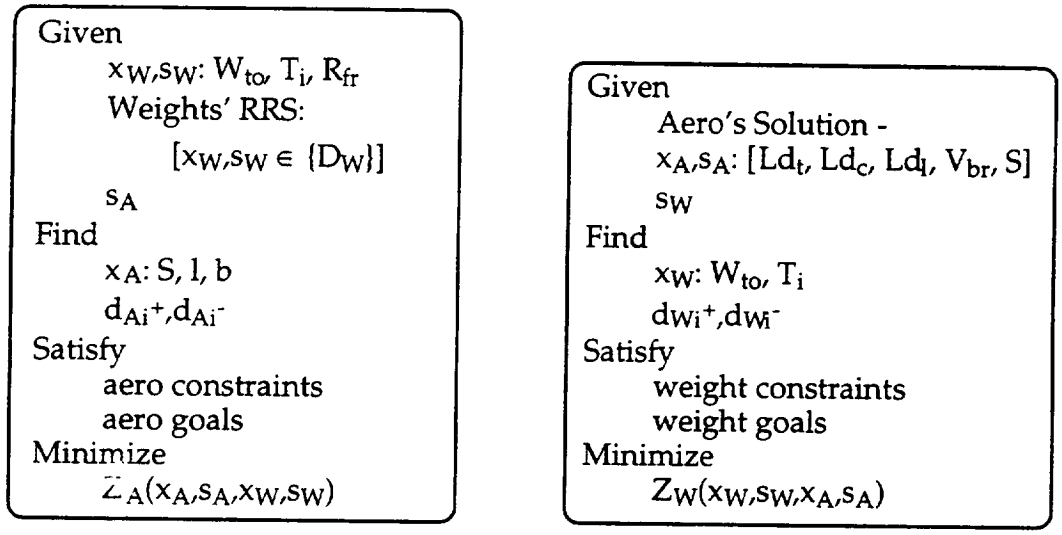
❸ Allowing the follower access to the leader's solution, solve the follower's compromise DSP.

The follower's compromise DSP is solved using the now known values from the leader. Since the RRS embodies the decision-making strategy of the follower, the resulting solution should match the strategy of the RRS. The two implementations of the leader/follower protocol in the aircraft study are the aerodynamics player as leader, and the weight player as the leader. Some discussion of each is given.

Aerodynamics as Leader/Weight as Follower

The aerodynamics design team takes the lead in a design process, and makes their decision about the aircraft profiles based on some information (in the form of the RRS) from the other disciplines. Then the weights player makes his decisions based upon the leader's solution. The compromise DSPs of the two players in this game are shown in Figure 7.8.

The compromise DSPs are the *same* as the core compromise DSPs presented in Section 7.2 except for the *given* information of each player. The leader has access to the follower's RRS ([W_{to}, T_i, $R_{fr} \in \{D_W\}$]), and the follower knows (but has to wait for) the leader's solution ({Ld_t, Ld_c, Ld_l, V_{br}, S}), and then uses that information in his model.



(a) Leader: Aero **(b) Follower: Weight**

Figure 7.8. Leader/Follower Compromise DSPs: Aero as Leader

Weight as Leader/Aerodynamics as Follower

This formulation corresponds to a design process where designers choose an engine configuration first, based on the assumption that the other disciplines will behave rationally. Then, the other disciplines have to design according to the engine specification. This is fundamentally different from the formulation with weight as the follower. The compromise DSPs of the two players in this game are shown in Figure 7.9. Again, the compromise DSPs are the *same* as the core compromise DSPs presented in Section 7.2 except for the *given* information of each player. The leader has access to the follower's RRS ([Ld_t, Ld_c, Ld_l, V_{br}, $S \in \{D_A\}$]), and the follower knows (but has to wait for) the leader's solution ({W_{to}, T_i, R_{fr}}), and then uses that information in his model.

Given
x_A, s_A: Ld_t, Ld_c, Ld_l, V_{br}, S
Aero's RRS:
$[x_A, s_A \in \{D_W\}]$
s_A
Find
x_W: W_{to}, T_i
d_{Wi}^+, d_{Wi}^-
Satisfy
weight constraints
weight goals
Minimize
$Z_W(x_W, s_W, x_A, s_A)$

(a) Leader: Weight

Given
Weights' Solution -
x_W, s_W: $[W_{to}, T_i, R_{fr}]$
s_A
Find
x_A: S, l, b
d_{Ai}^+, d_{Ai}^-
Satisfy
aero constraints
aero goals
Minimize
$Z_A(x_A, s_A, x_W, s_W)$

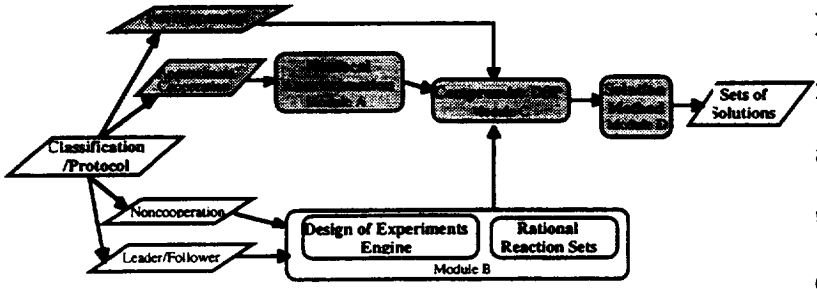
(b) Follower: Aero

Figure 7.9. Leader/Follower Compromise DSPs: Weight as Leader

The solutions to each implementation of the leader/follower protocols are discussed in Section 7.5.2. The third step of the algorithm, the solution of the game formulations, is illustrated in the next section.

7.5 STEP 3: SOLVE SUBSYSTEM AND INTEGRATION PROBLEMS

7.5.1 Cooperative

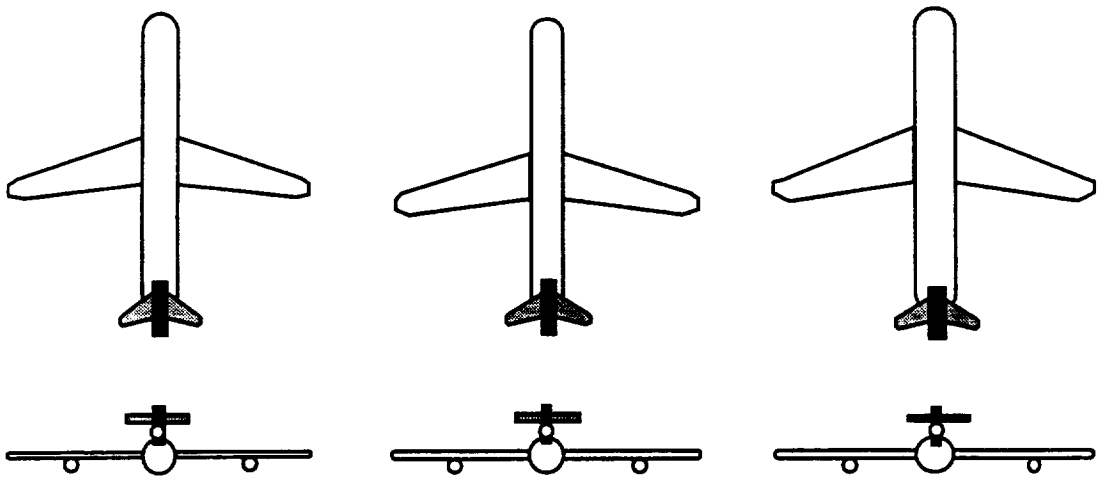


Both the cooperative formulations, full and approximate, are studied in this section as a means to compare forms of cooperation. Comparison to the other protocols is presented in Section 7.6. The approximate cooperative formulation, because it utilizes derivatives, can only handle

continuous variables. Therefore, in order to make comparisons to the (continuous) approximate cooperative formulation, a full cooperative formulation *only using continuous variables* is studied. A full cooperative model using mixed discrete/continuous design variables is studied as well. The three solutions depending upon the level of cooperation are shown in Table 7.9. The approximate cooperative and continuous version of the full cooperative formulations are solved using the ALP Algorithm (continuous variables only). The mixed version of the full cooperative formulation is solved using the discrete extension of the ALP Algorithm, the FALP Algorithm, introduced in Chapter 6. Both the ALP Algorithm and its extension, the FALP Algorithm are part of the decision support package, DSIDES. The resulting aircraft configurations are shown in Figure 7.10, and the deviation functions (Archimedean form) of the two players in each protocol solution are plotted in Figure 7.11. The configurations in Figure 7.10 look very similar, as they should since the results using the approximate cooperative protocol (see Table 7.9) are very close to the full cooperative results (continuous). There is a slight change in configuration when the mixed formulation of full cooperation is used. This is due to the discrete variable restriction in the b, l, and Ti variables. Full results of the three cooperative formulations, including solution history and corresponding DSIDES input files are given in Appendix C.

Table 7.9. Results of Cooperative Protocols

Player	**Protocol**	**Design Variables, x**			**Deviation Function**
		S (ft^2)	b (ft)	l (ft)	Z_{aero}
Player Aero	Approx. Coop	1554	122.4	119.1	**0.242**
	Full Coop: Cont.	1557	122.7	116.2	**0.242**
	Full Coop: Mixed	1613	126	120	**0.242**
		Ti (lbs)	Wto (lbs)		Z_{weight}
Player Weights	Approx. Coop	33960	196800		**0.213**
	Full Coop: Cont.	33910	196700		**0.214**
	Full Coop: Mixed	33000	197700		**0.220**



a. Approximate Coop. b. Full Coop.: Continuous c. Full Cooperation: Mixed

Figure 7.10. Cooperative Solutions (Approximately 1:1500 scale)

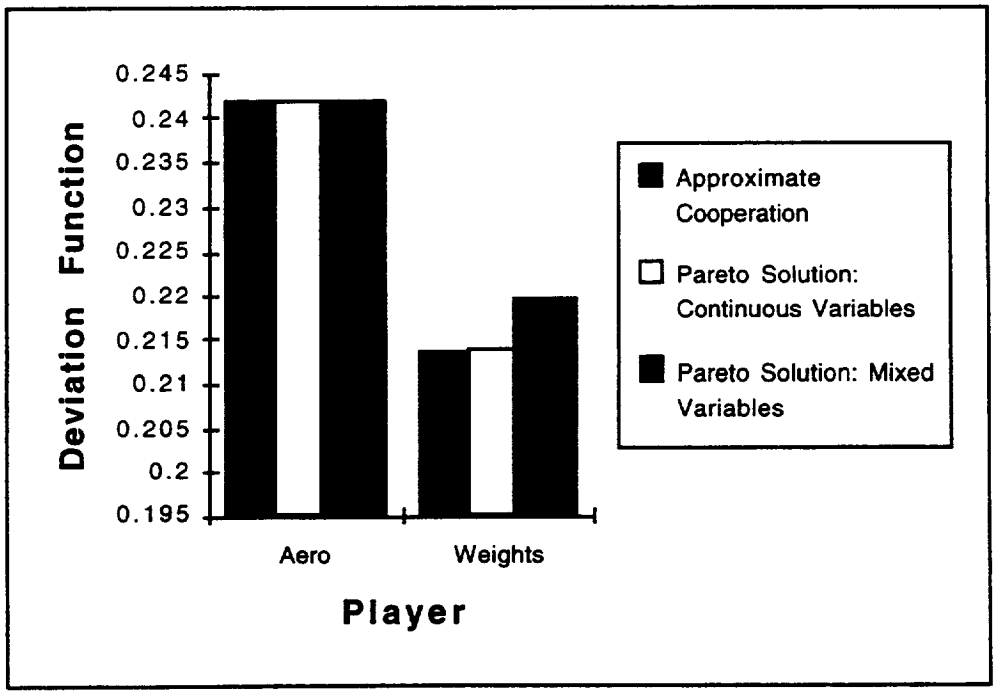


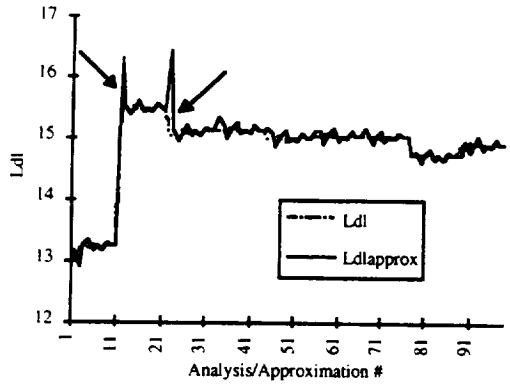
Figure 7.11. Cooperative Deviation Functions

Comparing the approximate cooperative with the mixed full cooperative solution, it is apparent that the approximate cooperative formulation solution is a better solution than the mixed solution. The fundamental reason for this is the presence of the discrete variables in the mixed formulation. The capability of "fine-tuning" a solution in the continuous domain does not exist for discrete variables in the mixed problem. A similar discrepancy is observed between the continuous and mixed full cooperative solutions. In essence, the comparison of the solution of a continuous problem with the solution of a mixed problem is like comparing apples to oranges. It is shown in Section 7.6 that in general, the cooperative formulations result in better solutions than the other protocols.

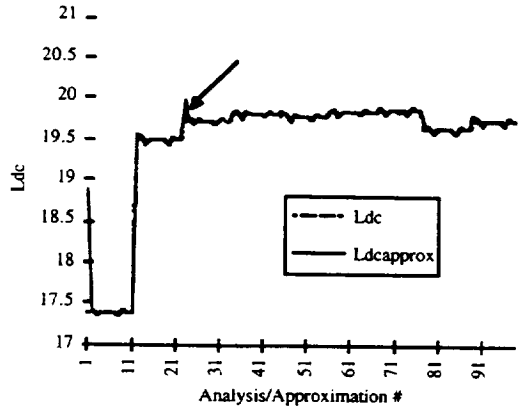
Comparing apples with apples, the *approximate* cooperative (continuous) and the continuous *full* cooperation solutions in Table 7.9 and Figure 7.11 are shown to be very similar. The deviation function of the weight player in the approximate cooperative formulation is slightly less than the full cooperative scenario (0.213 compared with 0.214), but this is due only to round-offs in the solution processes. Indeed the design variables of the weight player in each protocol are virtually the same. The only other major difference is the value of the fuselage length (l) in the aerodynamics player's problem. The difference in values (119.1 and 116.2) does not make a significant difference in the deviation function. This is an important result, as *multiple* values of the fuselage length have been identified where the deviation function of the aerodynamics player does not change. In other words, there is not *one optimal* solution, but *multiple satisficing* solutions. This is advantageous in the design of open engineering systems, where multiple solutions can be explored along a design timeline. When continuous variables are assumed, the deviation functions of the full and approximate cooperative formulations are virtually identical. Therefore, the nonlocal approximations made by the players in the approximate cooperative formulation seem to give accurate results. These nonlocal approximations are now explored.

Accuracy of the Taylor series approximations

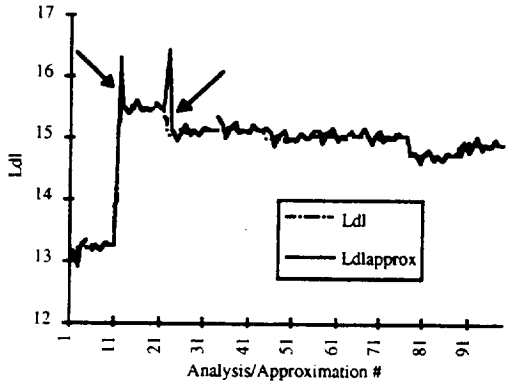
By using first order derivatives in the GSE, a first order approximation of nonlocal state variables can be constructed by the players. In previous studies, second order Taylor series expansions have been used to better approximate the nonlocal state variable. With this increase in accuracy, however, comes an increase in computation and information transfer as well. In order to formulate second order Taylor series expansions, second order derivatives are needed. In (Renaud, 1993), a second order GSE approach is presented. In this work of this dissertation, it is asserted that the first order approximations are good approximations of the nonlocal variables. In the Boeing study, five state variables, Ld_t, Ld_c, Ld_l, V_{br}, and R_{fr}, are approximated using GSE and Taylor series. In Figure 7.12 (a)-(e), plots are shown which reflect the actual values of the variables and the approximated values of the variables. In Figure 7.12, the approximations in general are very close to the actual values. The only large deviation occurs when the actual value jumps a large amount, corresponding to a large jump in the design variables. This occurs around approximation numbers 11 and 24 for Ld_l in Figure 7.12 (a) (indicated by the arrows). In these instances, the solution algorithm (ALP) makes a large step across the design space resulting in a significant increase of Ld_l from 15.4 to 16.3. Since the difference in design variables (X_n - X_o) is a multiplier in the Taylor series, if this difference is large, then the approximated variable will be approximated by a large linear step. Similar tendencies are found in the other plots in Figure 7.12 (b-e), but throughout the solution process, the approximations are very close to the actual values. In fact, the average percentage error over all the approximations for every variable is less than 1.0%. This information is summarized in Table 7.10.



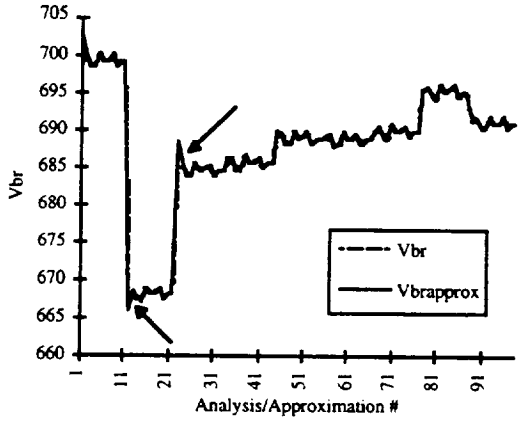
(a) Lift-to-Drag$_{take\text{-}off}$



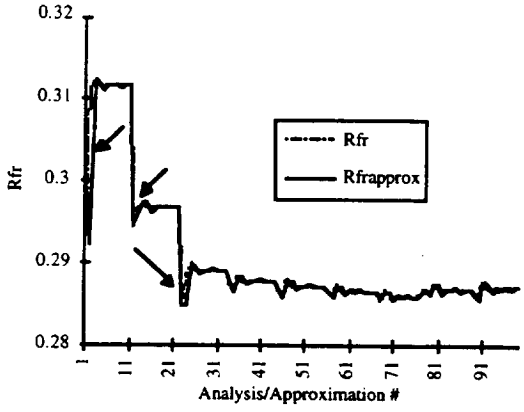
(b) Lift-to-Drag$_{cruise}$



(c) Lift-to-Drag$_{landing}$



(d) Velocity$_{best\ range}$



(e) Required Fuel Ratio

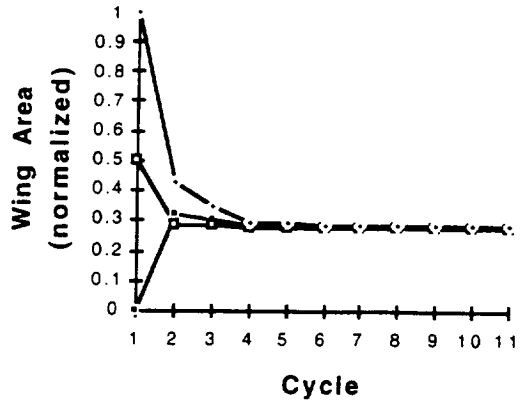
Figure 7.12. Nonlocal Approximations

Table 7.10. Average Error for Each Nonlocal Approximation

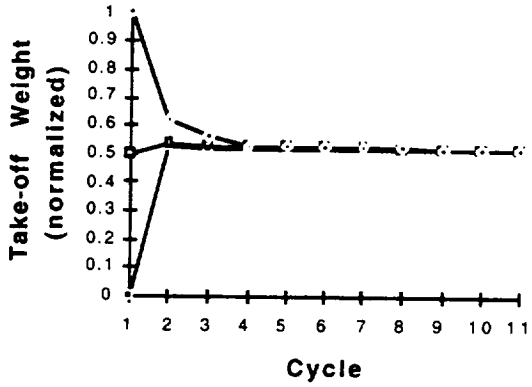
Approximated State Variable	Average Percent Error
Approximation in Aerodynamics	
R_{fr}	0.15%
Approximation in Weights	
Ld_t	0.43%
Ld_c	0.14%
Ld_l	0.50%
V_{br}	0.03%

The occasional "jumps" in the approximations in Figure 7.12 do lead to slight instabilities in the convergence history of the solution algorithm. Consider the convergence history of three different starting points of the *continuous formulation of full cooperation* shown in Figure 7.13. In Figure 7.13, there is rapid convergence for all three starting points for all of the design variables. In Figure 7.13 (f), the deviation function steadily decreases and the constraint violation decreases to zero. The only reason the convergence takes longer than 6-7 cycles is because of the Fuselage Length (Figure 7.13 (c)) takes about 9 more cycles to converge to its final value.

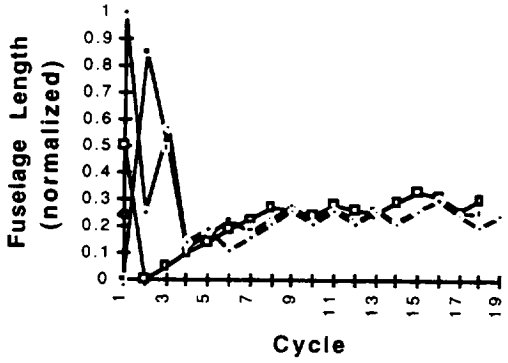
Now consider the convergence history of the *approximate cooperation formulation* shown in Figure 7.14. In these convergence plots, it is shown that one of the starting points converges to a *different* solution than the other two starting points. This lack of convergence to one common solution can be attributed to the occasional inaccuracies in the nonlocal Taylor series approximations. Because the ALP Algorithm is derivative based, when the derivatives are inaccurate, it may lead to erroneous or nonoptimal solutions. This is observed for one of the starting points of this formulation.



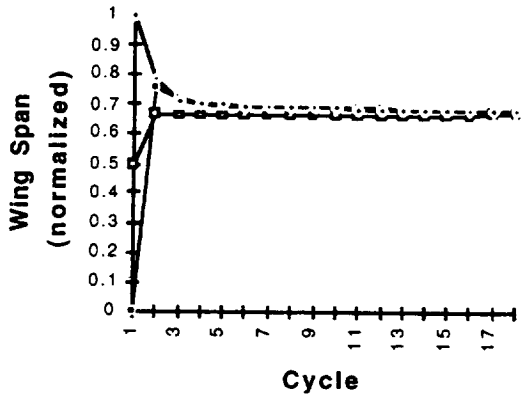
(a) Wing Area (normalized)



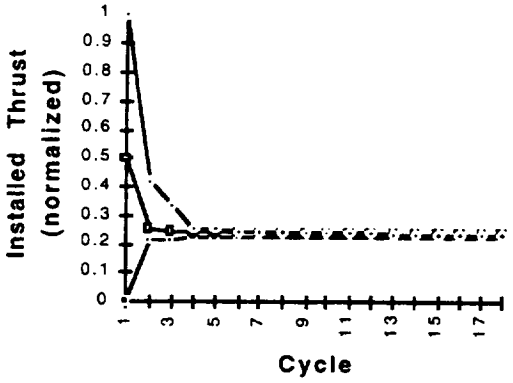
(b) Take-off Weight (normalized)



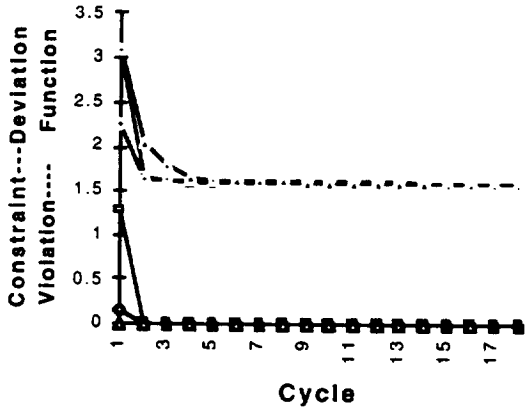
(c) Fuselage Length (normalized)



(d) Wing Span (normalized)

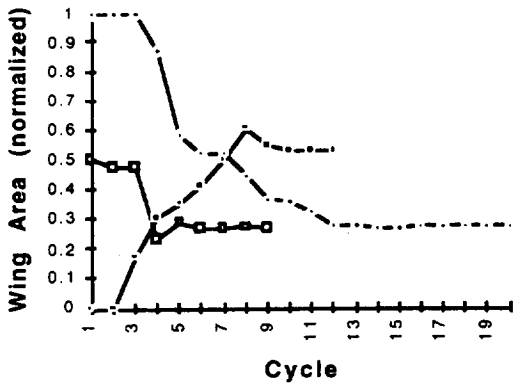


(e) Installed Thrust (normalized)

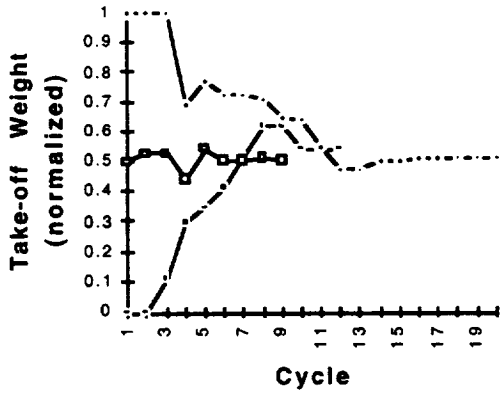


(f) Constraint Violation and Deviation Function

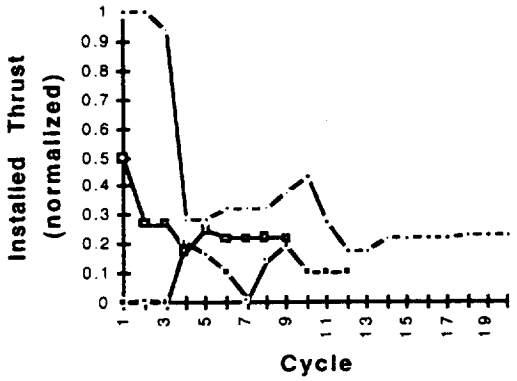
Figure 7.13. Design Variable History: Full Cooperation (Continuous)



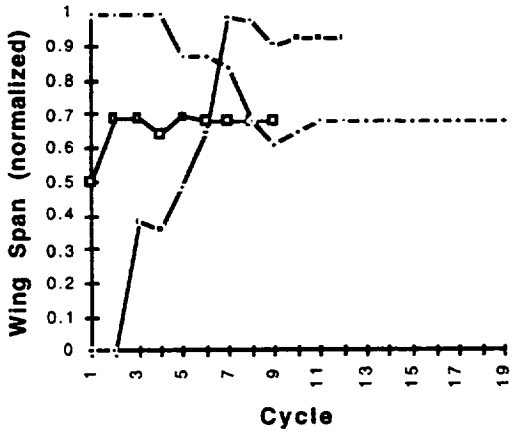
(a) Wing Area (normalized)



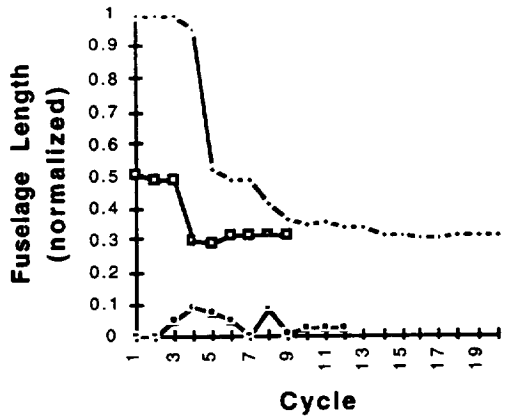
(b) Take-off Weight (normalized)



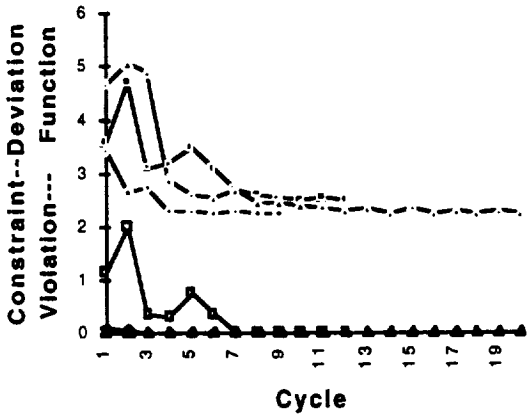
(c) Installed Thrust (normalized)



(d) Wing Span (normalized)



(e) Fuselage Length (normalized)



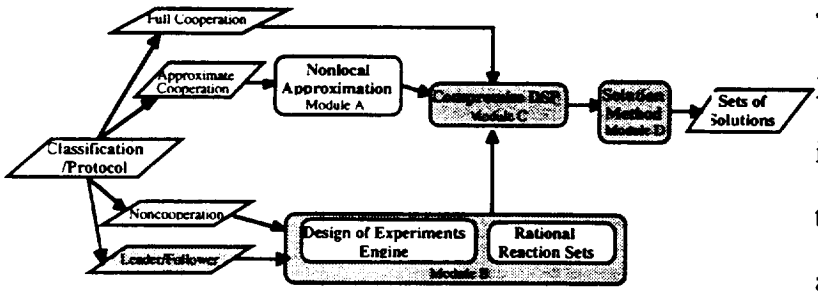
(f) Constraint Violation and Deviation Function

Figure 7.14. Design Variable History: Approximate Cooperation

Therefore, although the general accuracy of the first order Taylor series approximations are good, and the resulting solution is virtually identical to the full cooperative solution, stability in the solution process may be sacrificed. Since multiple starting points are used to verify the best solution, it is concluded from Figure 7.12 and Table 7.10 that first order Taylor series give a good approximation of the nonlocal state variables in the approximate cooperation formulation in this study. Another important issue of the Taylor series approximations is the requirement of *differentiability* since they are used with a derivative-based solution scheme. The derivatives of the state variables used in the GSE matrices are continuous as the state variables are themselves continuous functions across the domain of interest (Section 7.2). The Taylor series approximations are sums of continuous and differentiable functions, and therefore are differentiable as well. Therefore, the first order approximations in this study can be used in a derivative-based solution scheme.

In this study "approximate" cooperation is modeled. But how approximate is "approximate"? An interesting future application is to use more terms in the Taylor series as a means to model closer cooperation. Along these lines, the limit of the Taylor series as the number of terms reaches infinity would be full cooperation. With fewer terms, the level of cooperation decreases, as the approximation becomes worse.

7.5.2 Leader/Follower



The first step to solving the leader/follower formulation is constructing the RRS of the follower. Both player aerodynamics and player weights take their turn as the leader and follower in this section. The rational reaction sets

of the two players, as constructed using the process and information presented in Sections 5.5.2 and 7.4.2, are shown as follows. Again, these RRS's are constructed assuming all design variables are continuous as prescribed in Sections 5.5.2 and 7.4.2.

Player Aerodynamics

$$
\begin{aligned}
&\mathbf{D_A} = \\
&\{ \\
&S = 1448 + 444.4*Wto + 175.8*Ti - 107.5*Rfr - 155.8*Wto*Ti - 83.01*Wto*Rfr - \\
&\quad 83.01*Ti*Rfr + 186.5*Wto^2 + 97.04*Ti^2 + 15.27*Rfr^2, \\
&LDc = 18.06 - 1.878*Wto - 1.380*Ti + 0.3684*Rfr + 0.1019*Wto*Ti + 0.19*Wto*Rfr + \\
&\quad 0.19*Ti*Rfr - 0.4238*Wto^2 + 0.1319*Ti^2 - 0.685*Rfr^2, \\
&Vbr = 744.9 + 6.421*Wto - 52.37*Ti + 7.532*Rfr + 15.79*Wto*Ti + 6.924*Wto*Ti + \\
&\quad 6.924*Ti*Rfr - 31.64*Wto^2 - 9.419*Ti^2 + 7.828*Rfr^2, \\
&LDl = 14.12 + 1.47*Wto - 2.142*Ti + 2.601*Rfr - 0.2179*Wto*Ti + 0.2556*Wto*Rfr + \\
&\quad 0.0879*Ti*Rfr + 0.0998*Wto^2 + 0.4169*Ti^2 - 0.6684*Rfr^2, \\
&LDt = 9.698 - 0.9576*Wto - 1.97*Ti + 0.04071*Rfr - 0.336*Wto*Ti - 0.05205*Wto*Rfr \\
&\quad - 0.05205*Ti*Rfr + 0.1815*Wto^2 + 0.7128*Ti^2 - 0.899*Rfr^2\}
\end{aligned}
\tag{7.60}
$$

Player Weights

$$
\begin{aligned}
&\mathbf{D_W} = \\
&\{ \\
&Wto = 216000 + 15040*S - 11300*Vbr - 163.4*LDl - 5318*LDc + 20930*LDt - \\
&\quad 305.0*S*Vbr - 148.6*S*LDl + 2694*S*LDc + 13580*S*LDt - 158.9*Vbr*LDl + \\
&\quad 2512*Vbr*LDc - 9457*Vbr*LDt + 425.8*LDl*LDc - 173.6*LDl*LDt - \\
&\quad 3912*LDc*LDt - 22370*S^2 - 1624*Vbr^2 + 11730*LDl^2 + 2571*LDc^2 - 24730*LDt^2, \\
&Ti = 39120 - 284.1*S - 2565*Vbr-547.2*LDl - 1558*LDc - 10170*LDt + 1680*S*Vbr - \\
&\quad 559.6*S*LDl + 1525*S*LDc - 784.5*S*LDt - 447.1*Vbr*LDl + 1409*Vbr*LDc - \\
&\quad 2194*Vbr*LDt - 149*LDl*LDc - 581.4*LDl*LDt - 1355*LDc*LDt - 4058*S^2 - \\
&\quad 229.7*Vbr^2 + 2212*LDl^2 + 988.2*LDc^2 + 6190*LDt^2, \\
&Rfr = 0.3145 - 0.0833*Vbr - 0.06146*LDc + 0.01412*Vbr*LDc + 0.00003624*S^2 + \\
&\quad 0.02323*Vbr^2 + 0.00003624*LDl^2 + 0.01261*LDc^2 + 0.00003624*LDt^2\}
\end{aligned}
\tag{7.61}
$$

Significance issues associated with these response surfaces warrant investigation. First, it is assumed that *every term* in the regression is significant, and therefore, every term occurs in each equation. Some coefficients may certainly turn out to be insignificant are could be

eliminated using screening experiments (Chen, 1996), but in this dissertation every term is kept. With larger problems, it would be advantageous to perform screening experiments to reduce the computation demand of creating the response surfaces.

Second, the significance of the regression is investigated. In Table 7.11, the R^2 values of each response surface are given. In essence, the R^2 value of a regression for a variable, X, approxmiated as a function of a set of variables, **Y**, implies that the percentage variability of the variable X that is accounted for by variables **Y** is R^2. The highest value of R^2 for this study occurs for the state variable R_{fr}. This means that virtually all of the variation in R_{fr} is being accounted for by the required nonlocal variables, S, V_{br}, Ld_t, Ld_l, and Ld_c. The three design variables (S, W_{to}, and T_i) have fairly high R^2 values, implying that much of the variability in these variables is being accounted for by the nonlocal variables. The lowest values of R^2 occur in the state variables (V_{br}, Ld_t, Ld_l, and Ld_c). This implies that other factors, not accounted for in the response surface regression, contribute to the variation in these variables. This makes perfect sense, as illustrated in Section 7.4.2, as the state variables could be functions of other design and state variables which are not accounted for in the model. The response surface model of a rational reaction set is used *only to predict the effect of nonlocal variables on a local variable*. In general, the R^2 in Table 7.11 are acceptable and with a high level of confidence, it can be concluded that the regression for each variable is significant. In other words, there is significant nonlocal coupling that can be accounted for in the approximations of the rational reaction sets (Eqns. 7.60 and 7.61).

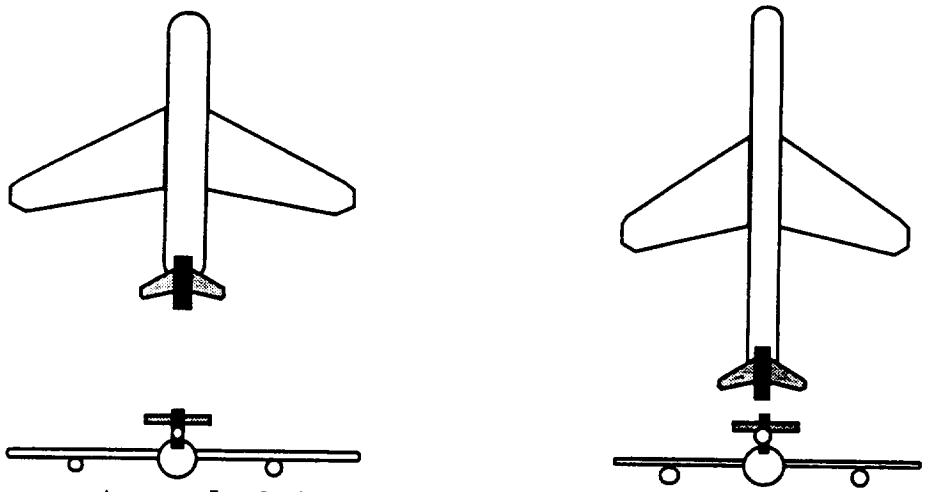
Table 7.11. R^2 Values for Each Coupled Variable

Variable	R^2 value
S	94.9
V_{br}	76.7
Ld_t	74.8
Ld_l	83.3
Ld_c	82.2
W_{to}	90.0
T_i	89.2
R_{fr}	99.9

The two solutions of the two leader/follower protocols which utilize the RRS's are shown in Table 7.12. Full results of each leader/follower formulation, including solution history and corresponding DSIDES input files are given in Appendix C. Both protocol formulations are solved using the FALP Algorithm in DSIDES (Chapter 6), since the compromise DSPs of both players have discrete and continuous variables. The resulting aircraft configurations are shown in Figure 7.15 and the deviation functions of both players are shown in Figure 7.16. The resulting configurations of the two solutions are quite different. The differences in the configurations are explored in this section.

Table 7.12. Stackelberg Solutions

Player	**Protocol**	**Design Variables, x**			**Deviation Function**
		S (ft^2)	b (ft)	l (ft)	Z_{aero}
Player Aero	As Leader	1870	136	107	**0.241**
	As Follower	1644	114	150	**0.252**
		Ti (lbs)	Wto (lbs)		Z_{weight}
Player Weights	As Leader	41000	208126		**0.201**
	As Follower	36725	224206		**0.255**



a. Aero as Leader/
Weights as Follower

b. Weights as Leader/
Aero as Follower

Figure 7.15. Stackelberg Solutions (Approximately 1:1500 scale)

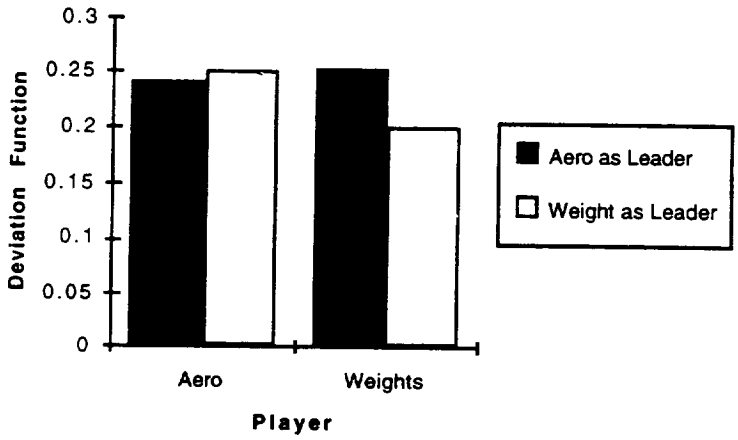


Figure 7.16. Deviation Functions for Both Stackelberg Formulations

In both cases, each player would rather be the *leader*, as the deviation function of each player is smaller when he is the leader in the game. This can be explained directly from looking at the nature of the RRS of the follower. In the leader/follower game (defined in Section 3.3.3), the follower is constrained to the strategy embodied by his RRS. Since the leader knows the RRS of the follower, he can make his decision and know how the

follower will react. The follower does have the freedom to control the local variables not coupled with the leader's problem, although he is constrained to behave as the leader has assumed he will behave. That is, the values of the control variables which are predicted by the RRS as a function of the leader's control variables *are set*. The large discrepancy between the two deviation functions of the weight player and the smaller discrepancy between the two deviation functions of the aerodynamics player can be explained by the following.

- As the leader, once the aerodynamics player's solution is found, the basic geometric parameters are set (wing area, S, wing span, b, fuselage length, l, and fuselage diameter, d). The weights player is constrained to these values, and must obtain a fuel balance based on the configuration. Therefore, depending upon the configuration geometry, the weights player is restricted to allocate the fuel weight required and fuel weight available. Once the configuration is set, the weights player is greatly constrained by the geometry. In fact, the weights player is constrained *fully* by the aerodynamics player's solution. In the case where the aerodynamics player is the leader, both design variables of the weight player, T_i and W_{to}, are needed. Therefore, the RRS of the weight player dictates what values the weight player will choose for T_i and W_{to} according to what values the aerodynamics player chooses for his control variables. In other words, once the aerodynamics player solves his model, the weight player's solution is given also. This is because the aerodynamics player requires *all the* design variables of the follower.
- When weights is leader, W_{to} is set but the aerodynamics player still has the freedom to allocate the weight to the wing or fuselage. The aerodynamics player still has freedom to size the aircraft given a certain total weight. He is constrained somewhat by the thrust available, T_i, which dictates the size of the wings to an

extent. Comparatively, however, the aerodynamics player has more freedom to meet his requirements as the follower than the weights player does as the follower. In addition, since the weights player only needs *one* of the aerodynamics player's design variables, the wing area, S, the aerodynamics player still has the freedom to control and change the other design variables, b and l.

Insights into Process Structure

These two strategies model two completely different practices in design process structure. The resulting aircraft configurations carry rich insights into the differences between the two strategies. To investigate the characteristics of each aircraft, the state variables of each aircraft are explored. The influential state variables of the aircraft for each protocol are shown in Table 7.13.

Table 7.13. State Variables for the Leader/Follower Solutions

State Variable	Aero as Leader / Weight as Follower	Weight as Leader / Aero as Follower
U	0.46	0.48
R_f	1.13	1.00
PRI	155	174
Ld_l	16.0	12.9
Ld_t	12.6	9.9
Ld_c	20.7	18.3
AR	9.89	7.91

The following observations are made, which explain the difference in configuration of the two protocols.

- The goals of player weight include bringing U (useful load fraction) close to 0.5, R_f (Fuel Balance) close to 1.0, and maximizing the PRI (Productivity Index). From Table 7.13, clearly when the weights player is the leader, these objectives are met better in the configuration of Figure 7.15 (b) than when the aerodynamics player is the leader (Figure 7.15 (a)). U is closer to 0.5, R_f is 1.0, and PRI is larger. Therefore, the weights player is able to satisfy his goals and constraints while knowing how the aerodynamics player will react. The weights player can be confident that the aerodynamics player will not adversely affect his own solution too much because of the rationality assumptions in the RRS. Although the difference in U (0.46 and 0.48) is not significant, the differences in R_f and PRI are significant. Two aircraft with these values certainly behave differently.
- The aerodynamics player strives to maximize the lift-to-drag ratios of the aircraft. From Table 7.13, each lift-to-drag ratio (Ld_l, Ld_t, Ld_c) is greater in the configuration in Figure 7.15 (a) than when aerodynamics is the leader (Figure 7.15 (b)). As the leader, the aerodynamics player has the freedom to change the aircraft profile and dimensions to meet his requirements. The differences in the values of the lift-to-drag ratios are quite significant. An average difference of around 2.8 in the Ld's result in aircraft which behave differently and have much different lift characteristics.
- One of the goals of the aerodynamics player is to bring the aspect ratio (AR) as close to 10.5 as possible. As the leader, the aerodynamics player is able to bring the AR closer to 10.5 than as the follower. The freedom to change the profile of the wing simply does not exist when the aerodynamics player is constrained by his RRS and the solution from the weights player. The difference in the AR values is significant. An aircraft with an AR of 9.89 has the potential to behave

much differently than one with an AR of 7.91. The lift and drag characteristics of the aircraft are effected by the AR.

Two significantly different aircraft are produced depending upon the protocol exercised. The aircraft look *and* behave differently. Clear understanding of the aircraft requirements and prioritization of objectives must exist in order to structure a design process accordingly to maximize the overall goodness of the aircraft.

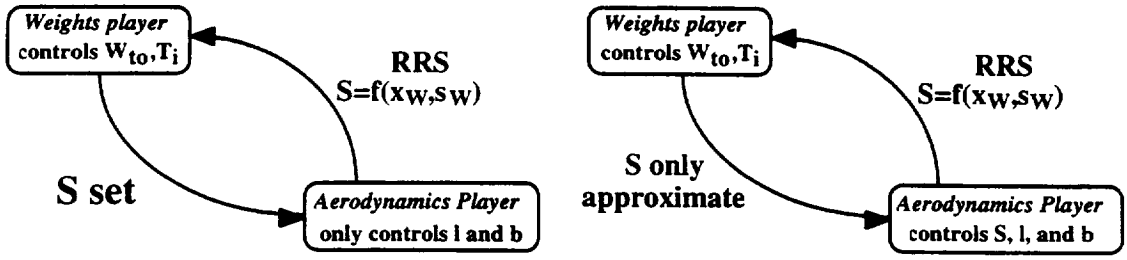
Verification of the Rational Reaction Sets

In order to verify the RRS's of the players, it is required to measure how well the RRS represents the decision making strategy of each player. In Sections 3.3.3 and 5.5.3, it is assumed that the follower is *constrained* to behave as dictated by his rational reaction set. This constraint is now relaxed in order to investigate the results when the follower is free to change *all* of his design variables. Therefore, two cases are performed:

(1) Restricting the follower to behave as his RRS predicts (already presented in Table 7.12), and

(2) Allowing the follower to solve his full model.

These two cases are illustrated with the aerodynamics player as the follower in Figure 7.17. In both cases, the weights player knows the value of the aerodynamics player's wing area, S, from his RRS. In case (1) (Figure 7.17 (a)), once the weight player solves his problem, the wing area, S, of the aerodynamics player is set. However, the value of S dictated in the aerodynamics player's RRS is only an approximation. The aerodynamics player still has the freedom to change l and b since they are not needed by the weights player. In case (2) (Figure 7.17 (b)), this constraint is relaxed and the aerodynamics player is free to change *all* of his design variables. Therefore, the leader only sees an *approximation* of how the follower will react to the leader's decision.



a. RRS Restriction **b. No RRS Restriction**

Figure 7.17. Difference in RRS Implementations

Theoretically, if the follower's RRS is a close enough approximation of how he will react, the solutions of the two cases should be similar. Case (1) is presented in the beginning of this section (see Table 7.12). In Table 7.14, the solutions for case (2) are given for each leader/follower formulation. Comparing Tables 7.14 (case 2) and 7.11 (case 1), it is obvious that the solutions are very similar. The deviation functions in Figures 7.18 and 7.18 are very similar as well. It is concluded that the RRS of each player gives a very good approximation of how each player will react to the leader's decision. This is verified as well by the high values of R^2 for each of the response surfaces in Table 7.11. The nonlocal variables are acting as significant predictors of the local variables in the response surface approximations of the rational reaction sets.

Table 7.14. Stackelberg Solutions: Relaxed RRS Constraint

Player	Protocol	Design Variables, x			Deviation Function
		S (ft^2)	b (ft)	l (ft)	Z_{aero}
Player Aero	As Leader	1870	136	107	**0.246**
	As Follower	1640	114	150	**0.252**
		Ti (lbs)	Wto (lbs)		Z_{weight}
Player Weights	As Leader	41000	208126		**0.201**
	As Follower	36000	222106		**0.256**

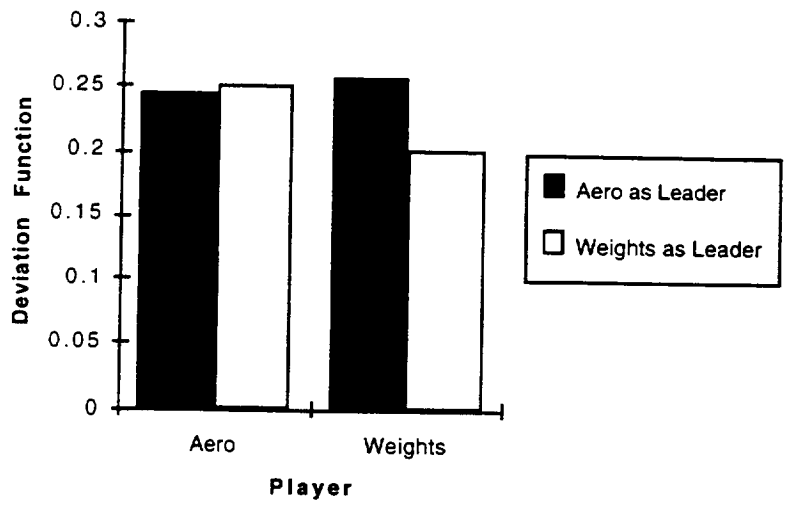
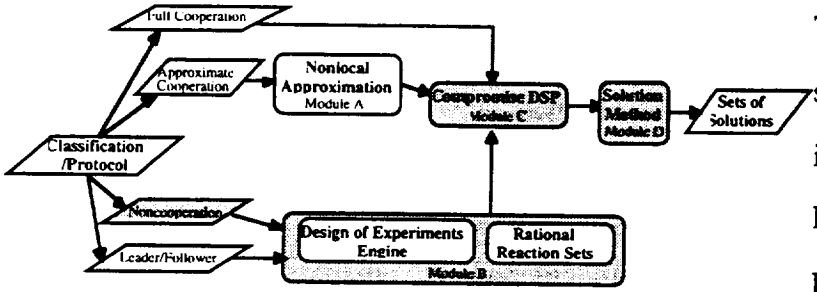


Figure 7.18. Deviation Functions for Both Stackelberg Formulations: No RRS Restriction

7.5.3 Noncooperative



The noncooperative solution occurs at the intersection of the players' RRS's, $D_A \cap D_W$, or Eqns. 7.60 and 7.61. The RRS's are constructed assuming continuous variables, and since it is not likely that the intersection of multiple nonlinear 2nd-order surfaces will occur at an acceptable discrete variable value, only continuous variables can be handled in the noncooperative formulation. There are five variables needed by the weight player from the aerodynamics player and three variables needed by the aerodynamics player from the weight player. Therefore, the combined RRS's of the two players are comprised of eight equations with eight unknown variables. Finding the intersection of these eight equations is equivalent to finding the point intersection of eight nonlinear n-dimensional surfaces. The solution to this system of

equations can be found using either a closed-form solution method, or approximate root-finding method. If the equations were linear, finding a solution to a system of eight equations with eight unknowns would be feasible. However, since the equations are second order with first order, squared, and interactions terms, set of equations of this type is extremely difficult to solve with even the most sophisticated mathematical software. Therefore, in order to solve the equations, some assumptions have to be made in order to simplify the set of equations. The largest set of equations (second order as in Eqns 7.60-61) that can be solved with Mathematica® is a set of 3 equations and 3 unknown variables. Therefore, five variables in the combined RRS's must be set constant. Since the coupled *design* variables are S (Wing Area), Ti (Installed Thrust), and Wto (Take-off Weight), these are chosen to be the variables solved for. The remaining five *state* variables are set constant at different values to explore different noncooperative solutions. Seven scenarios are explored using seven sets of constant input values. These seven scenarios correspond to the following conditions.

Scenario 1	Midpoints of the variable ranges.
Scenario 2	Lower Bounds of the variable ranges.
Scenario 3	Upper Bounds of the variable ranges.
Scenario 4	The values from the Stackelberg formulation with Aerodynamics as leader, Weight as Follower.
Scenario 5	The values from the Stackelberg formulation with Weight as leader, Aerodynamics as Follower.
Scenario 6	The values from the approximate cooperative formulation.
Scenario 7	The values from the full cooperative formulation.

By using the values of the state variables from these scenarios and only solving for the remaining design variables in the RRS's, the simplified combined RRS's take the form of:

$$\{S = C_0 + C_1W_{to} + C_2T_i + C_{12}W_{to}{*}T_i + C_{11}W_{to}^2 + C_{22}T_i^2,$$
$$W_{to} = C_0 + C_1S + C_{11}S^2,$$
$$T_i = C_0 + C_1S + C_{11}S^2\} \qquad (7.62)$$

The results of the noncooperative formulations using the assumptions of the seven scenarios are given in Table 7.15. Configurations of the seven resulting aircraft are shown in Figure 7.19. The seven configurations are very different, from short and wide to long and thin configurations. Full results of each scenario, including the simplified RRS (Eqn. 7.62) for each scenario are given in Appendix C. Some interesting observations can be made from the noncooperative results.

- Only one of the scenarios produces a feasible solution (convio = 0.0) for both players, scenario 4, when the values are taken from the leader/follower formulation when aero is the leader. Therefore, without communication and cooperation among the players, a feasible design using this aircraft model is not likely to be found. Mathematically, this occurs because of two reasons

  1) the solution is constrained to lie on the intersection of three nonlinear surfaces. There may be better solutions elsewhere in the design space, but because of this restriction, they cannot be used.

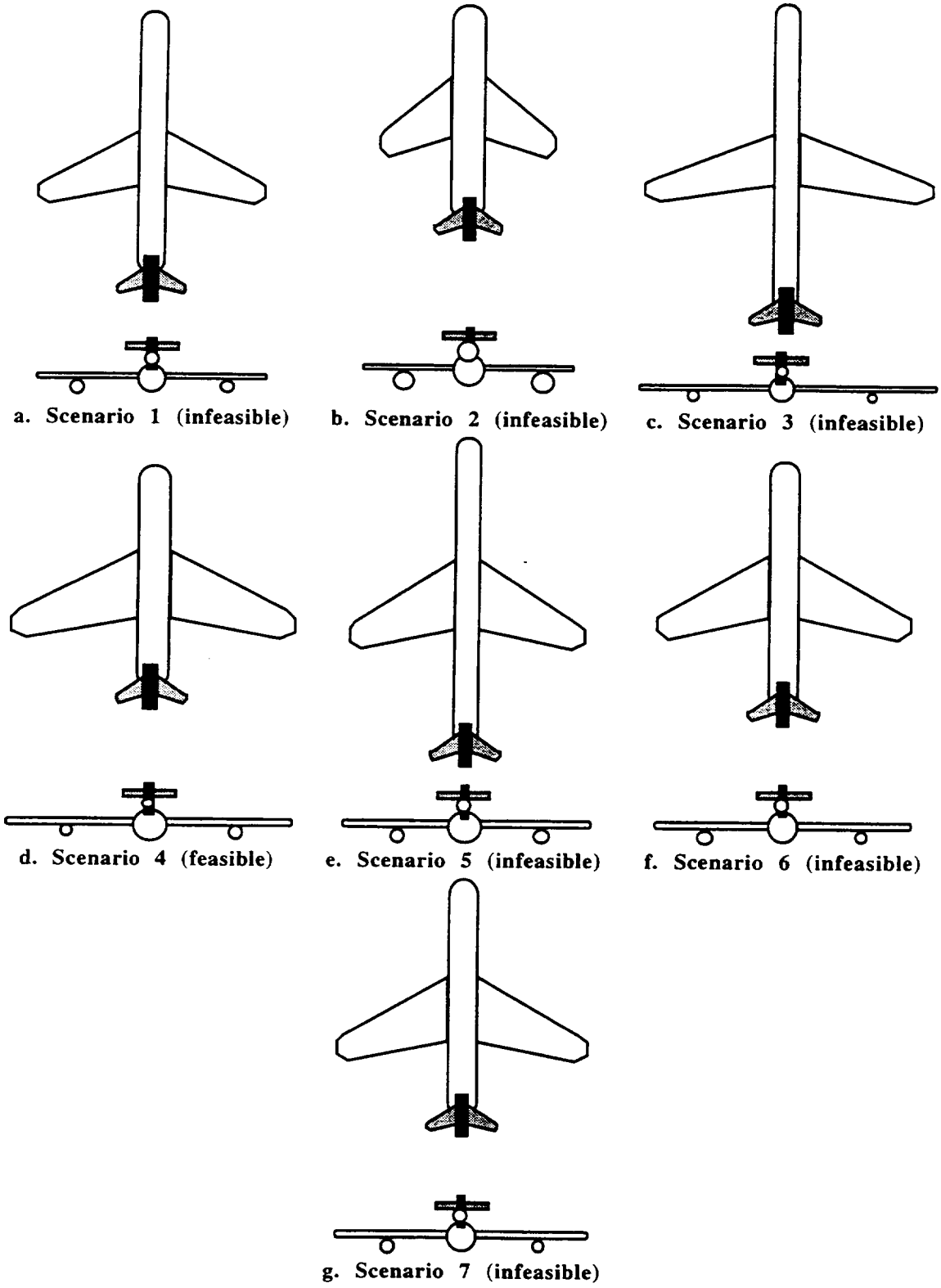
  2) the nonlinear surfaces are approximations of each player's RRS. Therefore, the approximation of the prediction of the exact behavior of each player may not be effective enough. Also, the intersection of the approximate RRS surfaces may not match the intersection of the exact RRS surfaces. However, the exact RRS surfaces are very difficult to compute. Therefore, the quality of the solution may be sacrificed for efficiency in constructing the approximate RRS surfaces.

- The thrust in scenario 2 (60460 lbs.) is greater than the upper bound for the allowable thrust. This high value occurs because of the lack of communication

and cooperation and not because of the aircraft configuration. In scenario 2, the aircraft is relatively small, having the smallest take-off weight, wing span and fuselage length of all the scenarios. The thrust required for this configuration should not be as large as it is, but since the players do not cooperate, an inferior (and infeasible) solution is found.

Table 7.15. Noncooperative Solutions

Player	Protocol	Design Variables, x			Deviation Function	Constraint Violation
		S (ft^2)	b (ft)	l (ft)	Z_{aero}	Convio
Player Aero	Scenario 1	1600	112.5	127.5	**0.326**	**-0.242**
	Scenario 2	1584	85	105	**0.519**	**0.0**
	Scenario 3	1529	140	150	**0.393**	**-0.221**
	Scenario 4	1938	136	107	**0.257**	**0.0**
	Scenario 5	1571	114	150	**0.281**	**0.0**
	Scenario 6	1818	122.4	119	**0.319**	**-0.211**
	Scenario 7	1822	122.7	116	**0.320**	**-0.211**
		Ti (lbs)	Wto (lbs)		Z_{weight}	Convio
Player Weights	Scenario 1	38622	206800		**0.252**	**-0.242**
	Scenario 2	60460	185600		**0.393**	**-0.352**
	Scenario 3	28814	176638		**0.314**	**-0.08**
	Scenario 4	36716	225960		**0.262**	**0.0**
	Scenario 5	39970	199829		**0.227**	**-0.04**
	Scenario 6	37620	218461		**0.272**	**-0.211**
	Scenario 7	37597	218723		**0.272**	**-0.211**



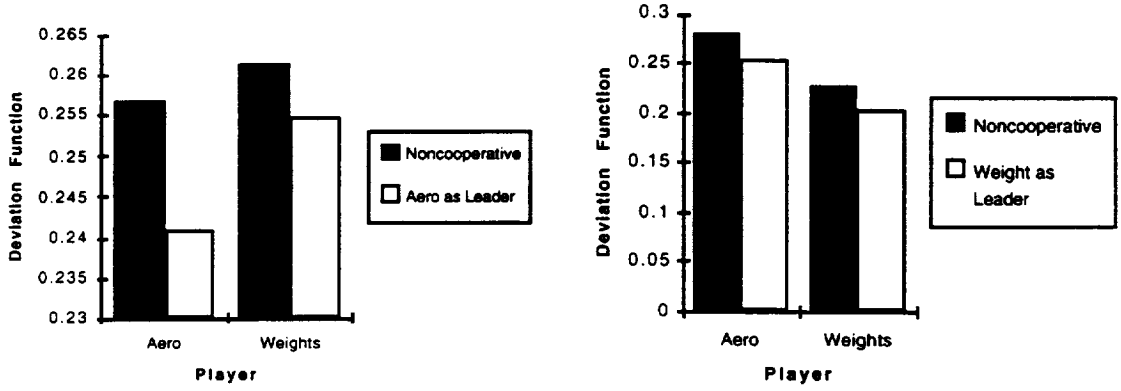
a. Scenario 1 (infeasible)
b. Scenario 2 (infeasible)
c. Scenario 3 (infeasible)
d. Scenario 4 (feasible)
e. Scenario 5 (infeasible)
f. Scenario 6 (infeasible)
g. Scenario 7 (infeasible)

Figure 7.19. Nash Noncooperative Solutions (Approximately 1:1500 scale)

In scenarios 4-7, the noncooperative solution shares the value of multiple design and state variables with the leader/follower and cooperative formulation. *Only three* of the coupled variables are found using the intersection of the players' RRS's. It is intuitive to think that if the noncooperative formulation shares multiple variable values with another protocol, the resulting solutions would be similar. But with the restriction of belonging to the intersection of the RRS's, the quality of the solution is greatly decreased even when only three variables are solved for. In fact the average increase in deviation function values from in Scenarios 4-7 is 17.2%.

Scenarios 4 and 5: Nash vs. Stackelberg

In Figure 7.20, the deviation functions for each player are plotted for the Stackelberg formulations (from Figure 7.16) and the respective Nash solutions, from Scenarios 4 and 5 in Table 7.15, which use variable values from each Stackelberg formulation. From Table 7.15, the noncooperative solution, using values from the leader/follower formulation with the aerodynamics player as the leader, is indeed feasible but the noncooperative solution, using values from the leader/follower formulation with the weights player as leader, is not feasible. The deviation functions of each player in the noncooperative formulation (shown in Figure 7.20) are worse for both players in both scenarios.



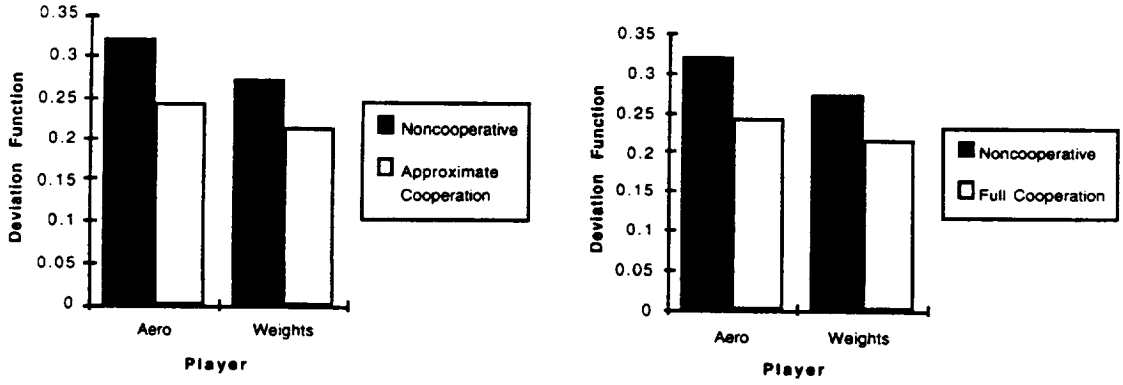
a. Aero as Leader **b. Weight as Leader**

Figure 7.20. Deviation Function for Nash vs. Stackelberg

Even though the noncooperative formulations are using five of the same variables from each Stackelberg formulation, because the noncooperative solution is constrained to lie on the intersection of the RRS's, the player's control over the remaining three variables is limited to only rationality not optimality.

Scenarios 6 and 7: Nash vs. Approximate Cooperation

In Nash Scenarios 6 and 7, variables from the approximate and full cooperative solutions are used, respectively, in the noncooperative formulations. As shown in Table 7.15, the noncooperative solutions in both cases are not feasible. In Figure 7.21, the deviation functions for each player are plotted for the cooperation formulations (from Figure 7.11) and respective Nash, from Scenarios 6 and 7 in Table 7.15, which use variable values from the approximate and full cooperative formulations. In these scenarios as well, both players' deviation functions increase in the noncooperative formulations for similar reasons as in the previous scenarios.

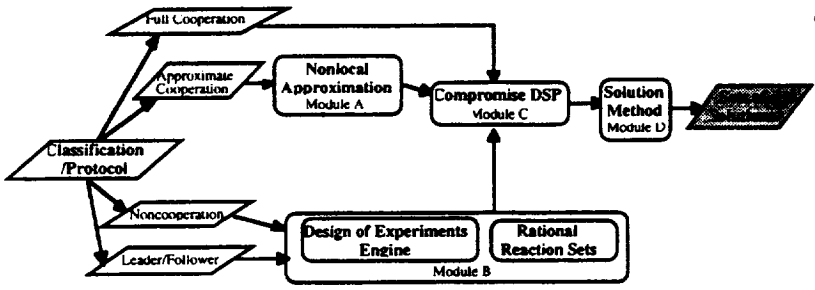


a. Approximate Cooperation **b. Full Cooperation**

Figure 7.21. Nash and Approximate Cooperative Deviation Functions

In summary, the noncooperative solutions are inferior to the solutions from the other protocols. In other words, *both* players do considerably worse when noncooperation is exercised. The results from all the protocols are compared to each other and to the existing 727-200 aircraft in the next section.

7.6 DISCUSSION OF RESULTS: COMPARISON OF PROTOCOLS

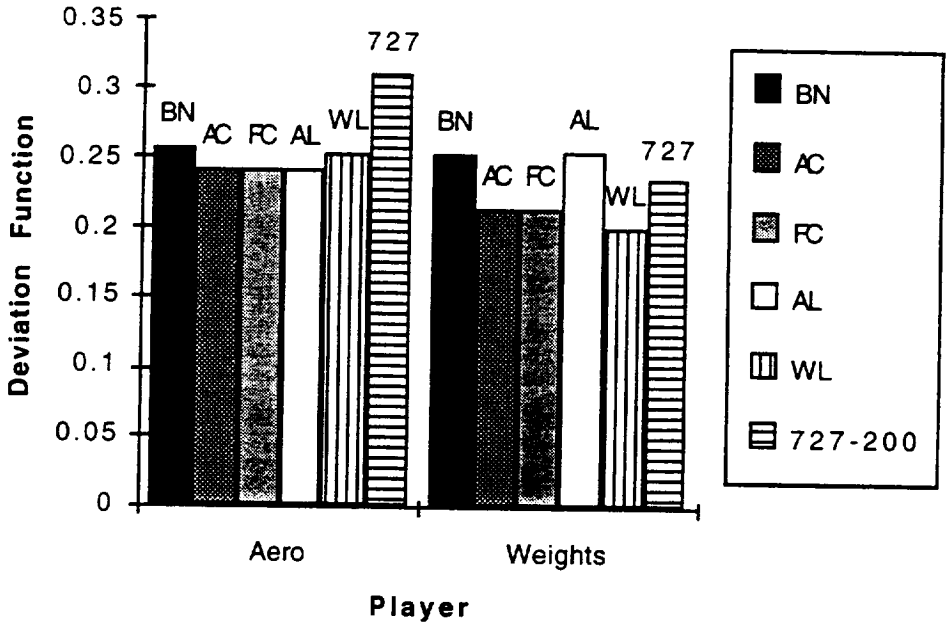


The purpose of this section is to compare the results of the individual protocol results in the previous section, Section 7.5, and gain some insight into the nature of each protocol in design. A summary of the results is given in Table 7.16.

Table 7.16. Comparison of Solutions and Existing Design

System Variable	BN	AC	FC	AL	WL	Existing Aircraft
Wing Area (ft^2)	1571	1554	1557	1870	1644	1700
Wing Span (ft.)	114	122	123	136	114	108
Fuselage Length (ft.)	150	119	116	107	150	136
Installed Thrust (lbs.)	39971	33903	33906	36725	41000	48000
Take-off Weight (lbs.)	199829	196512	196687	224206	208126	210000

BN: Best Noncooperative Solution AC: Approximate Cooperative FC: Full Cooperative
AL: Aerodynamics as Leader WL: Weights as Leader



BN: Best Noncooperative Solution AC: Approximate Cooperative FC: Full Cooperative
AL: Aerodynamics as Leader WL: Weights as Leader

Figure 7.22. Sample of Protocol Results as Compared to an Existing Design

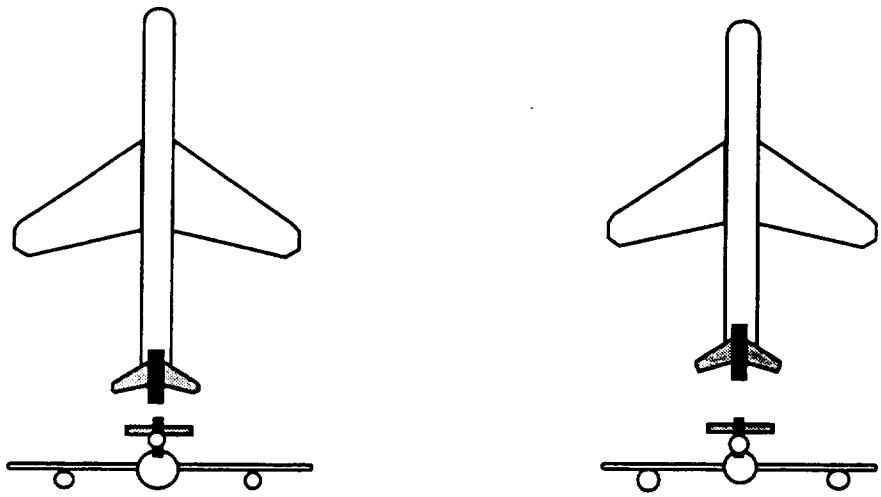
Since there were seven noncooperative solutions, depending upon the assumptions made, only the best noncooperative solution (Scenario 4) is shown in Table 7.16. Also included are the values for the existing 727-200 aircraft. In Figure 7.22 the deviation functions corresponding to the protocols are shown. Some interesting observations can be made from the results.

- The best "overall" results occur, as expected, when cooperation exists among the players. The term "overall" is meant to imply that both players cumulatively do well. It is interesting to note that whether full or approximate cooperation is exercised does not affect the result significantly. Using approximate cooperation

provides effective results with less computational information transfer than full cooperation.

- Player aerodynamics does very well (same as in the cooperative formulations) when he is leader in the leader/follower formulation, but at the expense of the weight player. Player weight as the leader in the leader/follower formulation actually does *better* than he does in the cooperative formulations, but at the expense of the aerodynamics player.
- In the existing 727-200, the aerodynamics player fares *worse* than *every* other scenario. Player weight only fares worse when he is the follower in the leader/follower formulation (AL) and in the best noncooperative formulation (BN). This result is not supposed to be used in any means to suggest that the 727-200 aircraft is inferior in any sense. It only shows that using *this* model of an aircraft (aerodynamics and weights player), the 727-200 is inferior. Certainly, aircraft design involves other disciplines as well, such as structures and controls. These disciplines were not accounted for in this work.
- The existing 727-200 values do not match exactly with any one protocol exercised in this work. In the original study of the 727-200 aircraft (Mistree, et al., 1988), the existing 727-200 values are reproduced using a single-level, simplified model with continuous variables. However, the model used in this dissertation is a *multi-level*, more detailed model that also uses *discrete* variables. Therefore, the fact that the 727-200 design does not match one protocol solution exactly is not surprising because of the partitioning of the system level problem in (Mistree, et al., 1988), into smaller problems, the existence of updated analyses for each discipline, and the restriction of discrete design variables for each discipline. The study in this chapter is used to illustrate the rich insights and benefits that could be generated when the behavior of the disciplines is modeled as strategic interactions

using game theory. It is interesting to note that if one aircraft had to chosen as being closest to the 727-200 aircraft, it would be the aircraft from the *leader/follower formulation with weights as the leader.* Configurations of the 727-200 and the weight as leader formulation are shown in Figure 7.23. The significant differences are the values of the wing span, fuselage length, which are larger in the weight as leader aircraft and thrust, which is larger in the 727-200.



a. Weight as Leader **b. Existing 727-200**

Figure 7.23. Aircraft Configurations (Approximately 1:1500 scale)

Further insight into the different aircraft can be gained by exploring the values of the state variables that describe the behavior of the aircraft. In Table 7.17, the significant state variables for the aircraft are given.

Table 7.17. State Variables of Various Solutions

State Variable	BN	FC	AC	AL	WL	727-200
U	0.48	0.49	0.49	0.46	0.48	0.49
R_f	1.0	1.0	1.0	1.1	1.0	0.94
PRI	158	177	177	155	174	174
Ld_l	13.1	15.07	15.0	16.0	12.9	11.7
Ld_t	10.0	11.8	11.8	12.6	9.9	8.8
Ld_c	18.0	19.9	19.9	20.7	18.3	17.2
AR	7.2	9.66	9.65	9.89	7.91	6.86
q_L	0.11	0.09	0.09	0.09	0.11	0.13
q_{TO}	0.03	0.03	0.03	0.03	0.03	0.04
s_L (ft)	3942	4492	4498	4306	4473	4336
s_{TO} (ft)	5124	6497	6500	6474	5774	4944

BN: Best Noncooperative Solution AC: Approximate Cooperative FC: Full Cooperative
AL: Aerodynamics as Leader WL: Weights as Leader

From the system requirements and compromise DSPs presented in Sections 7.2-7.3, the desired values of the state variables are as follows.

Weight Player

U = 0.5 R_f = 1.0 Maximize PRI

Aerodynamics Player

Maximize Ld_l, Ld_t, Ld_c AR = 10.5

Both Players

q_L = 0.03 q_{TO} = 0.03 s_L = 4500 ft. s_{TO} = 4500 ft.

Each state variable is investigated for the various cases.

Useful Load, U

The useful load fractions for each player in the FC, AC, WL, and 727-200 cases are close to 0.50, but in the other aircraft they are less than 0.50. This is intuitive because in the AL

case, the aerodynamics player does not leave the weight player enough freedom to improve U. In the noncooperative protocol, the two players do not reach a suitable compromise, and therefore U is sacrificed. The differences in the values of U are not significant enough to constitute different aircraft behavior, but used to illustrate the differences in protocols.

Fuel Balance, R_f

The fuel balance goal is satisfied in the BN, AC, and FC cases, as well as in the WL case. In the other cases, the fuel balance is not 1.0. In the AL case, again aerodynamics player does not leave the weight player enough freedom to improve R_f. It is interesting to note that the 727-200 value of R_f is the furthest away from 1.0. The differences in the values of R_f are significant and would result in different aircraft behavior and/or configurations.

Productivity Index, PRI

In the FC, AC, WL, and 727-200 cases, the productivity index is the maximum, while in the others it is significantly less. When weights is the leader (WL), the PRI is high because PRI is a state variable of the weight player, and he strives to maximize it. In both cooperative formulations, the players cooperate and achieve the highest PRI of the scenarios. The differences in the values of PRI are significant and would result in different aircraft behavior and/or configurations.

Lift-to-Drag ratios, Ld's

The lift-to-drag ratios are maximum when aerodynamics is leader (AL). This is interesting, as when the players cooperate (FC and AC), the aerodynamics player sacrifices some of the lift-to-drag to benefit the weight player in U, R_f, and PRI. When aerodynamics is only concerned with his own requirements, the lift-to-drag ratios are maximum, but this adversely affects the weight player, and in turn the "goodness" of the overall aircraft. The differences in the values of the lift-to-drag ratios are significant and would result in significantly different lift characteristics of the different aircraft.

Aspect Ratio, AR

Similar to the lift-to-drag ratios, AR is closest to 10.5 in the AL case. When cooperation is exercised, player aerodynamics realizes that he can sacrifice the AR to benefit both players. The differences in the values of AR are significant and would result in different aircraft aerodynamic behavior and/or configurations.

Climb Gradients, q_L and q_{TO}

The climb gradients are both closest to 0.03 in the FC, AC, and AL cases. The climb gradients are strong functions of the lift-to-drag ratios which are largely controlled by the aerodynamics player. Therefore, the values of q_L and q_{TO} in the AL case are close to the cooperative cases. The differences in the values of q_L and q_{TO} are not significant enough to constitute different aircraft behavior, but used to illustrate the differences in protocols.

Landing and Take-off Field Lengths, s_L and s_{TO}

The landing and take-off field lengths are closest to 4500 ft. in the existing 727-200 case. It is interesting to note that the BN case does fairly well in this regard as well. However, since the BN case is inferior to the other scenarios in each of the previous behavior variables, the BN case is certainly the worst case result. In the cooperative formulations, the players sacrifice s_{TO} to satisfy the other requirements more closely. The differences in the values of s_L and s_{TO} are significant and would result in different aircraft behavior and could possibly effect the capability of an aircraft to land and/or take-off from various airports.

7.7 OBSERVATIONS AND IMPLICATIONS IN DESIGN

The results in Sections 7.5 and 7.6 have computational and theoretical implications in modern design processes.

- The leader/follower protocol embodies a *sequential* philosophy that principles such as concurrent engineering (CE), and integrated product and process development (IPPD) strive to make obsolete. However, with the design of complex systems where design teams are located throughout the world and governed by different management with different objectives and priorities, true concurrency is very difficult. Therefore, tools and methods that accept and engage in some form of sequential processes have important roles in complex systems design.
- The noncooperative case (BN) used in Section 7.6, is the *best* noncooperative case, but is still inferior to the other solutions. Therefore, noncooperation should be avoided at all costs. Even largely sequential processes, as modeled in the leader/follower protocol are shown to be more advantageous to the final design than the noncooperative case.
- The computational requirement of constructing a player's rational reaction set is a direct function of the number of variables *needed* from another player. In the aircraft study, player aerodynamics needed 3 variables from player weights, and therefore 15 simulations were required to span the unknown design space. Player weights needed 5 variables from the aerodynamics player, and therefore 43 simulations were required. If the analysis code is expensive to run, then running 43 versus 15 simulations may prove to be costly. Of course, the number of interactions should be minimized, but completely decoupling a problem in complex systems design such as aircraft is virtually impossible.
- The noncooperative protocol embodies design scenarios where the design groups must make some rational assumptions about the other groups. This is one end of the cooperation spectrum. Although it is not explored here, introducing another player, a higher level "performance" or system-level player, whose primary duty

is to ensure that the players strive to meet overall system objectives along with their own local objectives would be a feasible strategy to model the management-engineer hierarchy at any given industry.

- Boeing in its recently publicized success in the design of the 777 aircraft has made philosophical and computational strides to ensuring cooperation at both the personal level and at the mathematical (analysis/synthesis) level. Although it is not a seamless, fully cooperative process, it certainly could be considered a form of approximate cooperation, which is shown in this work to be a worthy implementation.

The results and observations presented in this chapter have been driven largely by *descriptive* motivations, as opposed to *prescriptive* motivations. In other words, in this work the resulting designs are described when various design process structures are used, or when different strategies are used by different design teams. In this work, the intention is not to prescribe remedies to the noncooperative or leader/follower relationships, but describe the results if these relationships exist. And since relationships such as these certainly exist and will continue to exist in modern design of complex systems, the descriptive power of this work is beneficial to explore certain scenarios and the inherent tradeoffs between them.

7.8 A LOOK BACK AND A LOOK AHEAD: A SUMMARY OF OBSERVATIONS

In this chapter, Phase III of the strategy for verification and implementation of this dissertation is accomplished. This represents the final piece of the puzzle of this

dissertation, as represented in Figure 7.24. In this chapter, an aircraft case study is exercised using the algorithm established in Chapters 4-6. Chapters 1-6 have provided the basis, foundations, hypotheses, and developments for the exercising and verification of Chapter 7.

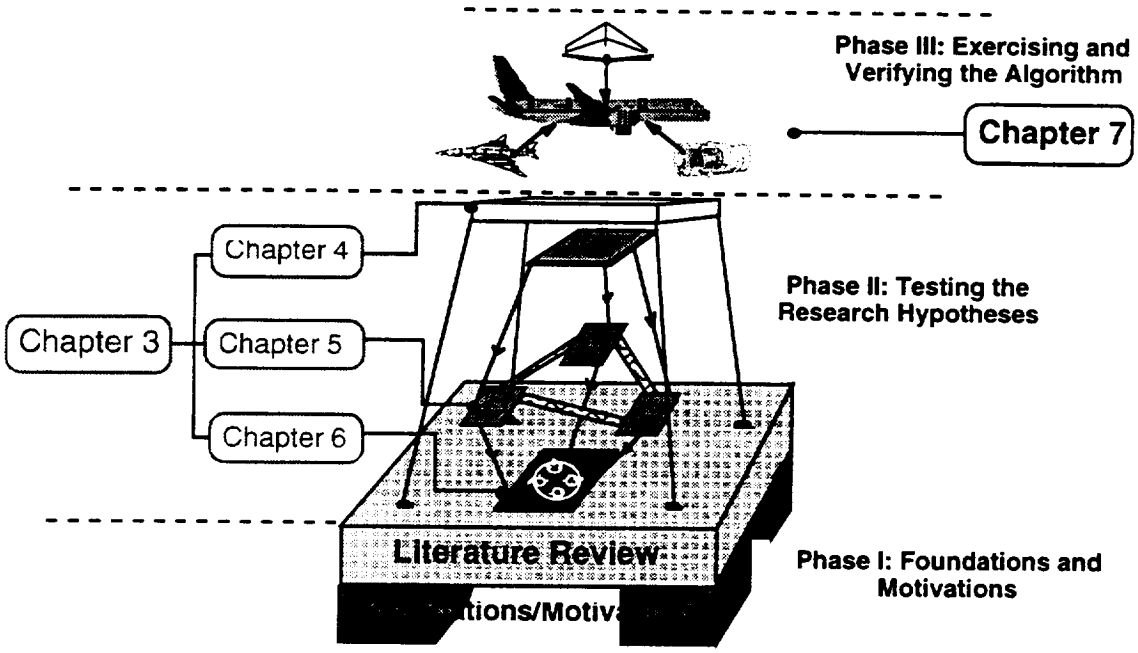


Figure 7.24. Frame of Reference: Chapter 7

The following observations are summarized based on the demonstration and verification of the algorithm for a passenger aircraft. Similar to the summaries in Chapters 4, 5, and 6, the observations are classified into categories for verifying different posits.

Hypothesis I Posits (Lexicon classification)

Verifying Posit 1.1 - domain-independent lexicon for multidisciplinary design

In Section 7.3, the lexicon is demonstrated as applied to the aircraft study. Its applicability to complex systems characterized by multiple design teams each with their own analysis, synthesis, and optimization strategies is demonstrated.

Verifying Posit 1.2 - Game Theory in multidisciplinary design problem formulation

In complex, multidisciplinary design, the problems are too large to handle with one design team. Typically multiple design teams that may be separated geographically are used. Game theoretical constructs are used to classify the teams and their computer decision support tools based on their role in the design process. In Section 7.3, this is demonstrated for the aircraft study.

Hypothesis II Posits (Game Theory formulations)

Verifying Posit 2.1 - Design processes abstracted as games

In Section 7.2, the models (compromise DSPs) of the two players are given. Each players' model is a function of variables, both design and state, from the other player. In other words, the decisions made by one designer affect the decisions made by the other player. This is precisely the definition of a game, therefore applying game theory to complex design problems is a natural extension.

Verifying Posit 2.2 - Approximate cooperation

In Section 7.4.1, the procedure for constructing approximations of nonlocal variables using the GSE and Taylor series is demonstrated. Since the actual values of the nonlocal variables are not used, but only approximated, it is considered to be an approximation of full cooperation.

Verifying Posit 2.3 - Taylor series approximation of nonlocal state equations

In Section 7.5.1, the accuracy of using first order Taylor series is shown. The first order approximations are shown to be very good representations of the actual variables of another player for the aircraft design problem. There is evidence to suggest that first order approximations may be adequate representations of nonlocal information, needed by a designer. To empirically or theoretically prove this posit would require significant mathematical investigations of a large number of

problems. In any case, the high cost of transferring analysis routines and equations is avoided by using approximations of the expensive nonlocal analyses.

Verifying Posit 2.4 - Response surfaces used to approximate the Rational Reaction Sets

In Section 7.5.2, the RRS's of the players are presented which are constructed using response surfaces. The effectiveness of a designer's ability to use the RSS's to embody and predict another decision maker's strategy is verified for the aircraft design problem in Section 7.5.2. Therefore, in complex systems design, it is possible to approximate an otherwise unknown player's RRS using second order response surfaces. The RRS of a player in complex systems design is very difficult to compute exactly. From the aircraft study, there is evidence to suggest that using second order response surfaces may be adequate. A proof of any kind of this posit would require significant mathematical investigations of a large number of problems.

Verifying Posit 2.5 - The compromise DSP as the fundamental construct

The core compromise DSP of each player is presented in Section 7.2. For each game protocol, the core compromise DSP of each player is *massaged* in the appropriate manner to account for the interaction (or lack of interaction) between the players. *Only the given* information of each player is changed in the various protocols according to the players' roles in the design process. The *find, satisfy, and minimize* constructs along with the local analysis remain the *same* as in Section 7.2. The augmented compromise DSPs of each player in each protocol are shown in Figures 7.2, 7.5, 7.6, 7.7, 7.8, and 7.9 in Sections 7.4.1-7.4.3. Although only the Archimedean formulation of the deviation functions is exercised in this study, the preemptive formulation will be exercised in future studies. The compromise DSPs of each player *do not change* using either the Archimedean or preemptive forms. Only the solution scheme in ALP or FALP gets adjusted. This is a major

advantage of the compromise DSP: to capability to model multiobjective design problems and explore the tradeoffs. The compromise DSP is used as a domain independent, fundamental mathematical model for each player for every protocol.

Hypothesis III Posits (Solution scheme)

Verifying Posit 3.3 - The Foraging-directed ALP Algorithm

The FALP Algorithm has been used in Section 7.5 to solve the full cooperative and leader/follower protocol formulations. Both players' models consist of discrete and continuous variables, and the capability to handle both in an optimization context is demonstrated.

In the next chapter, the dissertation is summarized. The primary contributions, a critical evaluation, and areas for future work are presented.

CHAPTER 8

ACHIEVEMENTS AND RECOMMENDATIONS

This dissertation is motivated by the need to understand, classify, model, and solve design problems of complex systems characterized by multiple interacting disciplinary design teams who may or may not cooperate. This dissertation represents efforts to incorporate the concepts of Game Theory, Multidisciplinary Design Optimization, and Decision-Based Design into a framework for decision support in the design of large scale systems. In this chapter, a review of the achievements is presented, a critical evaluation is provided, and areas of future work are identified.

8.1 ACHIEVEMENTS

A summary of the dissertation is provided in Section 8.1.1. The achievement of the principal goals of the dissertation is given in Section 8.1.2. In Section 8.1.3, the fundamental questions and the answers that are provided throughout the dissertation are reviewed.

8.1.1 A Summary of this Dissertation

This dissertation is entitled "An Algorithm for Integrated Subsystem Embodiment and System Synthesis." By using the algorithm, complex, multidisciplinary systems are classified, and the subsystem problems are solved and coordinated using various interaction protocols in order to obtain a system level design. Based on the motivation and background of this work, the foundations of this dissertation are laid in Chapter 1. Based on the needs identified for developing an algorithm to handle subsystem embodiment and system synthesis, the principal goal for this dissertation is

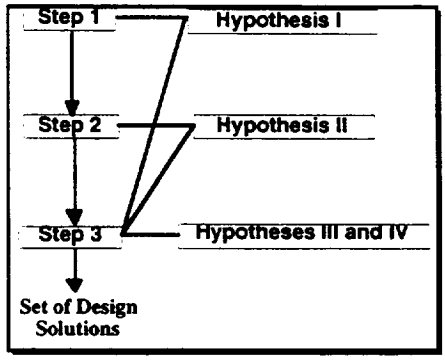
> Develop a framework for the decision support of formulating a multidisciplinary design problem, decomposing the problem, modeling the resulting interactions according to realistic assumptions, and solving and coordinating the disciplinary mathematical models.

To achieve this goal, the ideal aspects of such an algorithmic framework are identified by examining the nature of complex systems design in the context of multidisciplinary design optimization. This framework is developed by focusing on four research areas, namely, Subsystem Interaction, Mixed Discrete/Continuous Optimization, Problem and Process Classification, and Nonconvexity. For each of these areas along with related research areas, the research background, state-of-the-art, and opportunities are identified in Chapter 2.

Based on the needs and research opportunities in complex design identified in Chapter 2, four hypotheses and eleven supporting posits are identified in Chapter 3 as the theoretical foundations and assumptions for the development of the algorithm. These hypotheses involve the integration of game theoretical constructs, first and second order approximation concepts, and a hybrid solution scheme all with the multiobjective mathematical construct, the compromise Decision Support Problem (DSP). In Chapter 3, for each hypothesis, ramifications are provided, the necessary literature background is presented, and verification guidelines for each supporting posit are discussed. Based on the hypotheses and their supporting posits, the overall structure of the algorithm is also presented in Chapter 3. In Chapters 4-6 the research hypotheses are demonstrated and verified according to steps 1-3 of the algorithm, respectively. In Chapter 4, various examples, including the design of a pressure vessel and passenger aircraft are used to verify the first step of the algorithm, problem and process classification. In Chapter 5, the design of a pressure vessel is used to verify the second step of the algorithm, modeling the subsystem interactions. In Chapter 6, the design of a compression spring and a pressure vessel are used to verify the third and final step of the algorithm, the solution of mixed discrete/continuous design problems.

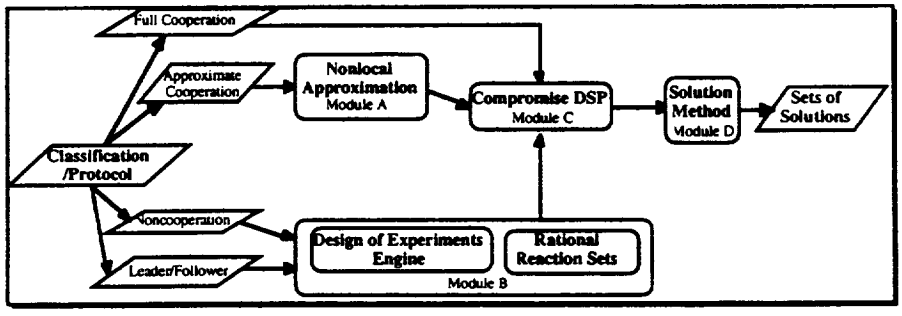
Having tested the hypotheses using example problems, the design of a subsonic transport aircraft is used in Chapter 7 as the motivating study to further demonstrate and verify the application of the algorithm in a complex systems domain. Each step of the algorithm, as applied to the aircraft problem, is presented and the results are explored in the context of complex systems design in a modern design environment. Chapter 8 is the closure of the dissertation.

8.1.2 Achieving the Principal Goal



Consistent with the principal goal as identified in Sections 1.3.1 and 8.1.1, the algorithm developed in this dissertation is a three-step framework for realizing complex systems when cooperation may or may not exist. A detailed description of the algorithm is provided in Section 3.1. The major steps of the algorithm and their relationships are illustrated (Figures 1.6 and 3.2). Associated with the development of the algorithm are techniques for decision support of designers in designing complex systems:

- a three-level lexicon for the classification of the design of complex systems and their associated design processes (Section 4.2).
- techniques for implementing game theoretical protocols in the design of complex systems characterized by multiple disciplinary design teams (Section 5.5).
- an effective solution scheme for mixed discrete/continuous design problems (Section 6.4).
- a formal nonlinear optimization proof of the characteristics of the g-function (Section 3.5).



The partial computer infrastructure for implementing the algorithm is illustrated in Figures 1.7 and 3.4. The major components of the existing computer infrastructure include four processors (a nonlocal approximation processor, a design of

experiments/response surface/rational reaction set processor, and a solution processor), each integrated with the primary processor, the compromise DSP (see Section 3.1.2).

The usefulness of the algorithm is illustrated by discussing types of applications that could benefit through the use of its various techniques (Section 1.4.1). The usefulness is also illustrated using an example application, the design of a passenger aircraft (Section 7.5). Specifically, the algorithm can be used to

- classify different approach to formulating the design of a complex system, including product and process descriptors,
- formulate disciplinary problems according to the disciplines' realistic roles in a design process,
- effectively solve disciplinary problems which consist of discrete and continuous variables,
- resolve the coordination of various disciplinary problems based on game theory principles of strategic interactions.

The implementations of these different activities have been demonstrated using various verification examples and a motivating case study. From the achievements reviewed in this section, it is concluded that the principal goal is achieved.

8.1.3 Addressing the Fundamental Questions

The achievements documented in this dissertation are also highlighted using the four fundamental questions introduced in Section 1.3.1. To address these questions, the research opportunities are identified in Chapter 2. The research hypotheses, which, along with supporting posits, address these questions, are presented in Chapter 3. In Chapters 4, 5, and 6 the research hypotheses are developed and verified using simple examples in

order to understand the implications and results. In Chapter 7, the hypotheses are further demonstrated and verified using a case study.

1) ***How can complex system design problems and processes be described and classified using an intuitive decision support lexicon?***

   It is discussed in Section 1.1.1 that it is necessary to perform a type of "meta-design" before actually formulating and solving complex design problems. In this dissertation, this meta-design takes the form of a lexicon used to classify the product and process under consideration. The classification provides a basis of comparison and communication among researchers and designers in multidisciplinary design. The use of entities from the Decision Support Problem Technique and Game Theory is proposed as a means to augment a classification system. In Section 4.2, the overall, three-level classification lexicon is presented. In Section 4.3, various examples are classified and mapped into an existing classification to illustrate the efficacy of the lexicon. Domain and time-independence, two requirements of useful lexicons, are demonstrated. In Section 7.3, representative classifications of a subsonic passenger aircraft are shown. The classification can be rapidly changed to reflect changes in system and process structure. The classification can be used as a decision support tool to guide the design process.

2) ***How can realistic interactions among design teams and their associated analysis and synthesis tools be modeled and incorporated into a design process?***

   It is discussed in Sections 1.1.1 and 1.1.2 that the design of complex systems involves multiple design teams who each use their associated analysis and synthesis tools. Coordination of these teams is not a trivial task. Many times, although total cooperation is ideal, more practical relationships exist. These

relationships, it is asserted in Section 3.3.3, can be modeled using game theoretical constructs. The four game theory protocols applicable to design, full cooperation (ideal), approximate cooperation (practical), noncooperative (worst case), and leader/follower (sequential), are demonstrated in Section 3.3.4. In Section 5.5, techniques to formulate each protocol in the context of the compromise DSP are developed and presented. Various approximation techniques and solution schemes are used for each protocol according to the amount and type of information available to each player. In Section 5.6, these developments are verified using the game-theoretical design of a pressure vessel. In Section 7.4, the developments are illustrated for a representative complex system. Using these techniques, different designs can be constructed according to different design scenarios and design process structures. Rich benefits, as illustrated in Sections 7.6 and 7.7, can be generated from comparing the results.

3) ***How can mathematical models which consist of continuous, discrete, and integer variables be solved and coordinated?***

It is discussed in Section 1.1.3 that typically in complex systems, the system variables are continuous, integer, and discrete. Solving models with these types of variables presents mathematical challenges in classical optimization theory. A solver is developed based on the notion of animals foraging for food in the wild. The empirical observations with which the foraging heuristic solver is built upon are illustrated in Section 6.2. In Section 6.3, the integration of the discrete solver, foraging, and the continuous solver, the ALP Algorithm, is detailed. In Section 6.5, the effectiveness of the FALP Algorithm is demonstrated using two well-studied examples, the design of a compression spring and the design of a pressure vessel are given. In Section 7.5, the approximate cooperative and the

leader/follower formulations of the aircraft case study are solved using the FALP Algorithm.

4) ***Is the g-function of the ALP Algorithm a good transformation of nonconvex functions into well-behaved convex functions?***

It is discussed in Section 1.1.3 that behavior or state equations which describe complex systems are often highly nonlinear. Handling models with these equations presents mathematical challenges in optimization theory. The g-function (Mistree, et al., 1993a) has been asserted as being an effective transformation of nonconvex functions into well-behaved convex functions. In Section 3.5, a formal proof of induction is given, demonstrating that the g-function does not, in theory, transform nonconvex functions into convex functions. This is not to say that in small regions, the g-function numerically may produce valid results, but across large domains, and in general, the g-function theoretically is not effective.

8.2 CRITICAL EVALUATION AND RECOMMENDATIONS

The four contributions documented in this dissertation, corresponding to the four hypotheses are given as follows.

① A three-level lexicon for the classification of the design of complex systems and their associated design processes.
② Techniques for implementing game theoretical protocols in the design of complex systems characterized by multiple disciplinary design teams.
③ An effective solution scheme for mixed discrete/continuous design problems.
④ A formal nonlinear optimization proof of the characteristics of the g-function.

In this section, each contribution is evaluated, the limitations of application are presented, and recommendations are made for improvements.

① Classification

In Section 4.2 a three-level classification system for complex, multidisciplinary design problems is presented. The developments associated with the classification scheme are used in Step 1 of the algorithm documented in this dissertation (Section 3.1.1). The advantages of the classification include:

- The classification can be used to classify the types of analysis and synthesis tools being used by different design teams and the relationships among the design teams, as illustrated in Sections 4.3 and 7.3.
- By using domain independent linguistic entities, presented in Section 4.2, it is applicable to a large set of systems, regardless of the level of technology present.

There are certain limitations of the classification, which are described in the following.

- There does not exist a computing infrastructure for this portion of the algorithm. There are pieces for the other parts of the algorithm, but one interactive infrastructure for the entire algorithm does not exist. Some assertions are made about the development of such an infrastructure in Chapter 4, but it is acknowledged that to have industrial application capability, a computer infrastructure is necessary. The computational support exists for each step of the algorithm, but for the most part, they are isolated entities.
- The classification is applicable *at a given point* in a design process. No information concerning the sequence of decisions required in a complete design process is given.

Development and application of the classification also helped stimulate future areas of exploration, both conceptually and at an implementation level, including:

- The integration with an existing design guidance system would create a useful interface between the designer and the associated analysis and synthesis tools supporting the designer, as presented in Section 4.4.
- The linguistic entities describing coupled DSPs in the DSP Technique embody the *same* information as the entities used in the Balling-Sobieski scheme. This is demonstrated by mapping the two sets of entities onto each other using various examples in Section 4.3.

② Game theory interactions

In Section 5.2, it is asserted that complex systems design can be abstracted as a series of games among players who are embodied by disciplinary design teams and their associated analysis and synthesis tools. In Section 5.5, the techniques to model the interactions between design teams and their tools in complex systems design are presented. These techniques are applied to design problems in Sections 5.6 and 7.4. The developments associated with the game theory techniques are used in Step 2 of the algorithm documented in this dissertation (Section 3.1.1). The advantages associated with this contribution include:

- The techniques can be used to abstract the interactions among design teams as a series of games, as defined in Section 5.3. This abstraction occurs at a mathematical level though, as opposed to a personal level. Team building, TQM, and CE principles are used to help bridge the personal interaction gaps, while in this dissertation, game theory constructs are used to bridge the gaps at the mathematical analysis and synthesis levels.

- The results provide a foundation with which to build strategies in order to design and build the best system subject to product and process constraints. The consequences of different strategies are explored in Chapter 7.
- This contribution is one of the primary, original contributions documented in this dissertation.

There are certain limitations identified with the developments and techniques associated with this contribution which are described in the following.

- The first concern is the *practical implementation* of the work. In this dissertation, verification studies of the hypotheses and posits are performed using various example problems and one case study to support the work presented. Because of the novelty of developing game theory constructs in complex systems design, however, application to actual industrial systems is not accomplished. The case study in Chapter 7 consists of *only two disciplines*, aerodynamics and weights, but actual aircraft design consists of these disciplines along with others such as structures and controls.
- There typically is a system-level coordinator at the engineering or management level who tries to ensure communication and cooperation among the design teams and their associated analysis and synthesis routines. In this work, it is assumed *that the design teams are carrying out a strategy dictated by the structure of the management, organization, or geography.* In an industrial context it would be pragmatic to add another player to the games studied here, a system level "overall performance" player whose sole purpose is to ensure that the players satisfy the system level requirements along with their own disciplinary requirements. Possible formulations to facilitate this type of arrangement are given in Section 8.3.
- There are *limitations* of using some the approximation techniques in this dissertation. One limitation in the approximate cooperative formulation stems from

the fact that, since derivatives are used in a Taylor series approximation (Section 3.3.4), discrete variables cannot be handled. Derivatives of functions with respect to discrete variables do not exist. Therefore, in the approximate cooperative formulation, only continuous variables can be used. Another limitation is the fact that the Rational Reaction Set of a player is constructed *assuming* continuous variables (Section 3.3.4). The aircraft study includes discrete variables, but constructing response surfaces of functions of discrete variables is a difficult task without an effective solution. In the noncooperative formulation the solution is found by taking the intersection of a set of smooth, continuous nonlinear response surfaces that make up the players' RRS's. However, since discrete variables are being solved for using the surfaces, there is no guarantee that the intersection will lie at one of the discrete points in the design space. Therefore, only continuous variables can be used in the noncooperative formulation. The RRS is used in the leader/follower formulation as well. Therefore, when the follower is constrained to behave as her RRS dictates, it may not result in an allowable discrete value.

- Evidence to suggest acceptance of the posits supporting this hypothesis is developed in Chapters 5 and 7, but at this point, the supporting posits and Hypothesis II cannot be *proven* per se. The techniques and developments are shown to work well for certain verification studies. This does not mean that they will necessarily work for all types of problems or even a large class of problems. For instance, first order Taylor series approximations are shown to be effective representations of nonlocal state variables in the aircraft study in Chapter 7. However, this does not rule out the possibility of needing second order approximations for more complex analyses. Further, second order response surfaces are shown to be effective representations of a player's rational reaction set in Chapters 5 and 7. However, this also does not rule out the possibility of needing

third or fourth order response surfaces for more complex problems. The conditions under which first order Taylor series or second order response surfaces are applicable and effective has not been established. Extensive verification studies both from empirical and theoretical standpoints must be conducted to further verify the developments. Representative studies of this type are prescribed in Section 8.3.

Development and application of the game theoretical techniques also helped stimulate future areas of exploration, both conceptually and at an implementation level, including:

- In Section 7.6, the set of solutions for the game theoretical protocols are discussed. It is shown that the cooperative protocol, with deviation functions 0.242 for aerodynamics and 0.214 for weights, is the best solution for both players. If one player deviates from this solution, it will adversely affect the other player's solution. For instance, the leader/follower protocol with weights as the leader, the weight player does better than in the cooperative protocol (0.201), but at the expense of the aerodynamics player, whose deviation function increases (0.253). However, if the deviation functions of both players are *added* to produce an overall deviation function of the system, given as

$$Z_{overall} = Z_{aero} + Z_{weights}$$

then the overall deviation functions in the cooperative and leader/follower formulations are

$$\text{Cooperative: } Z_{overall} = 0.453$$

$$\text{Leader/Follower with Weights as Leader: } Z_{overall} = 0.456$$

and the formulation with weights as the leader is the *best* scenario. However, the aerodynamics player may not be pleased with this decision, as she can improve her status. It is interesting to formulate the measure of overall goodness in different ways:

- ❑ choose the solution with collective stability (cooperative),

- allow individual players to suffer at the expense of the overall system, or
- give priority to certain player's deviation functions as being more significant in the success of the system.

Of course, the deviation functions of each player must be normalized in such a way to provide a consistent means of comparison. The problem of how to evaluate the overall goodness of the system is left to further study of the management/engineering hierarchy. In this work, it is assumed that the players' deviation functions are somewhat in isolation. That is, although the players are trying to maximize the overall goodness of the aircraft, the local deviation functions are not compatible. The goodness of the overall aircraft is measured using the *individual* deviation functions of the players.

- Another practical issue is the *scalability* of the work. In preliminary studies, *increasing the number of players in a game* (the number of disciplinary design teams) does not pose difficult theoretical or computational problems. Game Theory is as applicable to n players, as it is to 2 players. A game could have multiple leaders and multiple followers, who may cooperate or not. The same constructs and developments documented in this dissertation can be applied to multi-player games. Possible applications to three-player games are discussed in Section 8.3.
- Another scaling issue, increasing the fidelity of each player's analysis model, however, may pose greater difficulty, since the approximation concepts used in this dissertation assume that access to the analysis *is available*. With black-box analysis codes, the accuracy of the local and nonlocal derivative representations may decrease. This issue is related to the possible theoretical and empirical investigations discussed in Section 8.3

③ FALP solution scheme

In Section 6.3, the Foraging-directed Adaptive Linear Programming Algorithm is presented. The notions of foraging in the context of optimization are discussed in Section 6.2.3. The FALP Algorithm is used to solve design models that consist of both discrete and continuous design variables in Sections 5.6 and 7.5. The developments associated with the FALP scheme are used in Step 3 of the algorithm documented in this dissertation (Section 3.1.1). The advantages of the FALP Algorithm include:

- Application of the FALP Algorithm has been shown to produce effective results, compared to previous studies of mixed discrete/continuous problems. These results are documented in Section 6.4.
- FALP has the capability of handling multiple goals in either an Archimedean or preemptive formulation. The Archimedean form of a deviation function is used in Section 7.5.
- This is one of the primary contributions documented in this dissertation.

There are certain limitations identified with the developments and techniques associated with this contribution which are described in the following.

- The FALP solution scheme is based on heuristics that dictate how to search the design space. The FALP Algorithm is certainly a "smart" algorithm, as it based on the behavior of intelligent animals (Sections 6.2.3 and 6.3). However, it sacrifices efficiency for intelligence. Since foraging is an unassuming algorithm, it will continue to search until it reaches a maximum number of neighborhood searches. Determining the best stopping criteria has not been fully investigated. Possible studies to determine the best stopping criteria both empirically and theoretically are discussed in Section 8.3.

- Evidence to suggest acceptance of the posits supporting this hypothesis is developed in Chapters 6 and 7, but at this point, the supporting posits and Hypothesis III cannot be *proven* per se. The techniques and developments are shown to work well for certain verification studies. This does not mean that they will necessarily work for all types of problems or even a large class of problems. Extensive verification studies both from empirical and theoretical standpoints must be conducted to further verify the developments. Representative studies of this type are prescribed in Section 8.3.

Development and application of the FALP Algorithm also helped stimulate future areas of exploration, both conceptually and at an implementation level, including:

- The foraging heuristic is based on empirical observations of animals foraging for food. It would be interesting to update the foraging heuristic solver using new constructs from actual foraging observations, in order to make the algorithm "smarter" or more efficient in its search processes.
- The foraging heuristic combines notions from the Tabu Search, Genetic Algorithms, and Simulated Annealing, all heuristic solution schemes. It would be interesting to classify the *fundamental* assumptions and constructs of each and strive to create a class of heuristics which are based on the same meta-heuristics at an abstract level.

④ Convexity

In Section 1.2.2, the g-function of the ALP Algorithm is introduced. In Section 3.5, a proof is developed concerning the capability of the g-function to transform nonconvex equations into well-behaved convex functions. The advantages of this contribution include:

- A formal proof, disproving the hypothesis that the g-function of the ALP Algorithm is developed. Although it discounts an earlier assertion, the g-function is currently

a recommended option in the ALP Algorithm. Therefore, the quality of the solutions in the dissertation is not compromised in any way.

- By using the relaxed convexity condition present in the ALP Algorithm, a stronger version of the proof is developed.

The limitation with the proof is that only strictly convex or concave functions are investigated. Functions which are neither convex or concave are not investigated, but are common in complex systems design. Further, it would be beneficial to the ALP (and FALP) Algorithm to utilize an effective transformation function to handle nonconvex functions. In *small regions, numerically* the g-function may perform well, although the theoretical basis of the function has been disproved in this work. Further investigations of changes to the g-functions or development of a new function are warranted.

In the next section, areas of future work to address many of the issues addressed in this section are presented. Some are conceptual in nature, while others are largely empirical and the processes with which to conduct the studies are prescribed.

8.3 FUTURE WORK

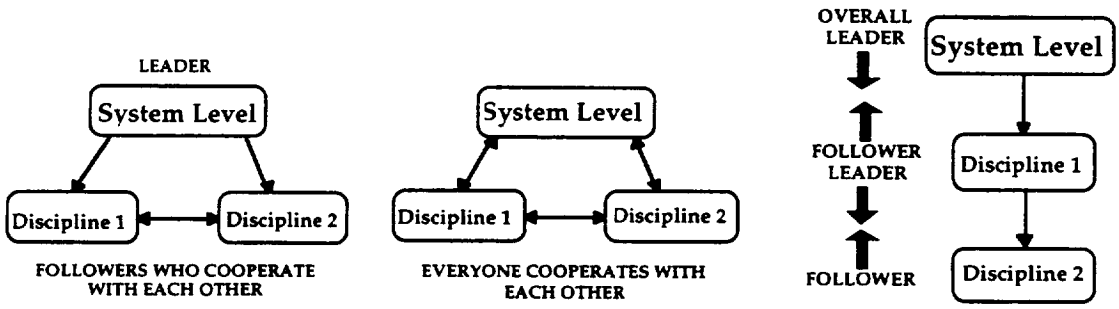
Based on the critical evaluation in Section 8.2, some areas of future work are recommended in this section:

- *Moving from single company to multiple company interactions.* One of the operating assumptions of this work is that the disciplinary teams (and support tools) which are interacting each work for the same company. Therefore, their general priority is to build a good product which maximizes their company's profit. But in

modern engineering practices, design teams from *multiple companies* must often interact and coordinate in order to design a complex system. In this case, each company would like to maximize their own profit, regardless of whether such an outcome comes at the expense of another company's profits. Exploitation may replace cooperation as the best alternative. This stimulates the question, "Is there a situation when noncooperation is advantageous to a player?" In simple games such as the prisoner dilemma's (Axelrod, 1984, Gleick, 1986, Nowak, et al., 1995), a player will benefit by not cooperating for one play of the game. This single player benefit will occur at the expense of the other player, who does not fare as well. In this work, game theoretical constructs, borrowed largely from economics applications, are defined and applied in the context of complex systems design. In the future, by addressing multiple companies and the profit-making strategy of each, the gap between design engineering and management/economics may be bridged to some extent. Management could even be introduced as a player in the game, since they certainly have some influence on the design product and process. If this influence and resulting interactions with engineers could be modeled and quantified, the results could be very beneficial to companies in organizing their design processes both from a management and an engineering perspective. This is a natural extension in design as well. Consider the following excerpt from De Bono (De Bono, 1985) who seems to ponder this very idea:

> *The plain purpose of the third party is to convert a two-dimensional fight into a three-dimensional exploration leading to the design of an outcome...The third party is not an addition or an aid but an integral part of the process.*

- *Identifying natural leaders and followers.* It is shown in Chapter 7 that both players do better when they are the leader instead of the follower. In other situations, disciplines may do better as the follower. So the question is asked, under what circumstances can a discipline be identified as a good leader or a good follower? Because of the complexity of the disciplinary models, this may be an unanswerable question in most scenarios. Yet if it can be answered under certain conditions, it could benefit the organizational structure of a company that, in large part, dictates how the design teams interact. Further, one of the limitations identified in Section 8.2 of the application of the game theoretical developments, is the lack of a high-level player whose primary objective is to ensure that the disciplinary players meet subsystem as well as system level objectives. This formulation could take many possible forms depending upon the organization of the company, information transfer, or geographical location. In Figure 8.1, three possible formulations are shown.



(a) Two-level **(b) Single-level** **(c) Three-level**

Figure 8.1. Possible Formulations with a Third Player

In Figure 8.1 (a), the system level player is the overall leader, while the disciplines are both followers. However, at the follower level, the followers could cooperate with each other as well. Therefore, multiple game theory protocols could exist

among the players in one game. In Figure 8.1 (b), all three players now cooperate with each other. In this case, the responsibility of the system level player to "guide" the disciplinary players must be explicitly formulated in the mathematical model, as opposed to the previous case (Figure 8.1 (a)), when as the leader, the system level player has the advantage of knowing how the disciplinary players will react. In Figure 8.1 (c), the system level player is again the leader, but now discipline 1 is the follower to the system level leader, but is also the leader to discipline 2. Discipline 2 is the lowest level follower. This formulation would be applicable to sequential processes, and scenarios when the system level player may not know anything about discipline 2, but must rely on discipline 1 to make decisions, reflecting its knowledge of its follower, discipline 2.

- *Investigation of the applicability of the techniques to classes of problems.* Many of the techniques developed in this dissertation have been verified using a number of small verification examples. The techniques have been shown to work well for these problems, but the unanswered question is *under what conditions, or for what class of problems would the techniques work?* In order to answer this question with any kind of confidence, detailed investigations are required. It is asserted that these investigations could take one of two forms: empirical in nature or theoretical in nature. These two perspectives are found on opposite ends of the spectrum of possible investigations for this type of problem, as illustrated in Figure 8.2

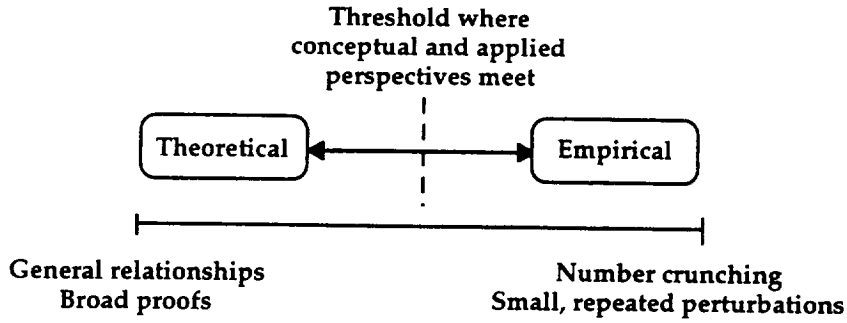


Figure 8.2. Spectrum of Investigations

On the empirical side repeated numerical experiments are typically run where small perturbations are made and results are continually generated and analyzed. While on the theoretical side, general relationships between problem parameters are constructed and broad, formal proofs are formed for types of problems. Ideally, both perspectives could be taken and at some point, there would be a threshold where the conceptual and applied investigations would meet, creating a seamless study of feasibility of a given technique to a class of problems. These types of studies could be conducted for various techniques developed in this dissertation, including:

- applicability of first order Taylor series approximations to a class of problems. When would second order approximations be necessary?
- applicability of second order response surfaces to approximate the rational reaction set of a player. When would an inflection point be advantageous?
- number of neighborhoods searched in the foraging search. The number of neighborhoods searched is a measure of the length of the foraging search. What percentage of the design space searched produces the best

results? This issue is returned to in greater detail in the next area of future work.

Issues of computational demand must be accounted for in *both* types of studies. For instance, although, in a theoretical study, it may be shown that a fourth order response surface is the best approximation for a certain class of problems, the computational demand of constructing a fourth order response surface may be too large to warrant practical implementation. Therefore, a practical perspective must always be maintained concerning analysis and synthesis computational limits.

Increasing the efficiency of the foraging search. In the current foraging search, the number of neighborhood searches in the foraging portion increases proportionally to the size of the problem (dictated by the number of variables *and* the number of possible discrete values). The number of neighborhood searches is a measure of the efficiency of foraging. If the best solution can be found in x searches as opposed to y searches, where $x < y$, then foraging is more efficient by using x searches. With problems where the best solution is *known*, it can be determined what the minimum number of neighborhood searches required to find the best solution is. However, with problems where the solution is *unknown*, determining the best number of searches is a more difficult problem. This problem can be investigated by taking one of the two approaches as discussed in the previous area of future work. From an *empirical* approach, the parameters of the algorithm can be changed repeatedly for a number of problems of varying size. The "best" set of parameters can be found by analyzing the effectiveness of the resulting solutions and efficiency of the search. From a *theoretical* approach, there are really two different approaches possible. First, since the search is based on the foraging of animals for food in the wild, observations of animals could be made and formalized in a model. However,

this approach is fundamentally flawed by the fact that animals eat what they see and continue to search for more food. Therefore, they stop when their brains receive a message that enough food has been eaten. The foraging solution scheme should stop when it has found the best or an acceptable solution. It cannot stop when it "is full." Thus, a second theoretical approach could be taken though. In this approach the defining characteristics of the foraging search could be defined as functions of the defining characteristics of a class of problems. The defining characteristics of the foraging search include:

- the number of neighborhood searches,
- the size of the neighborhood, and
- the proportion of the dynamic memory reduction.

The characteristics of a class of problems include:

- the number of variables,
- the number of possible discrete values,
- the number of constraints, and
- the number of goals.

Relationships between these sets of characteristics could be constructed and the efficiency and effectiveness of the foraging search could be investigated and established using these formalized relationships. In problems where the solution is unknown, there will be tradeoff issues involving the efficiency of the search to the effectiveness of the solution. When is a solution "good enough" for now? Or, how is "good enough" defined for these problems?

- *Further development of a Design Guidance System.* The classification lexicon presented in Chapter 4 is conceptual in nature. It is hypothesized that it could be integrated with an existing Design Guidance System (DGS) (Bras, et al., 1990) or IMAGE computing infrastructure (Hale, et al., 1996) as a means to classify the

product being designed and the process to design it. The classification of a given system may change throughout its realization process as more information is generated. Designers, interacting with the DGS, would be able to update the classification and in turn prescribe the appropriate methods and tools to help embody the system in terms of entities on a computer.

- *Interface to virtual/rapid prototyping to explore designs.* Different designs are found according to different protocols among the design teams. It would be advantageous to be able to rapidly build a prototype, virtual or actual, to explore the ease of manufacturing and mass and space-related properties of the different designs corresponding to the game protocols. The "goodness" of the examples in Chapter 6 and 7 are measured largely by technical and economic goals. Although a product may be "good" on paper using technical performance measures, it may not necessarily translate to manufacturing and operating "goodness". Measures of manufacturing considerations, and operating and interference limitations can be efficiently quantified using prototypes.

- *Consideration of different design stages as players.* Considering the typical product realization process shown in Figure 1.4, only interactions along the y-axis are considered in this dissertation. An interesting application of the developments would be to consider the interactions along the x-axis. Consider the well-used phrase, *design for manufacture.* Semantically, this phrase connotes the manufacturing phase as the leader in this process. A design process is constrained by the manufacturing capabilities available. However, what if it is *manufacturing for design*? This is a completely different situation with drastically different product and process implications as well. By exploring the interactions among

product realization phases, valuable insight can be gained into the structure and relationship between phases when cooperation (concurrent engineering), noncooperation, and leader/follower (over-the-wall) scenarios exist.

Design is an amorphous entity, always changing shape and form in the presence of outside influence. To capture the full essence of design is an impossible task; the theoretical and practical motivations of design sometimes even reside on different sides of a chasm. A primary motivation behind this dissertation is to help bridge such a chasm, to use design theory to model practical design processes and products as close as possible. However, once this bridge is constructed, it will inevitably be archaic itself in time. New problems will arise, and new solutions will be found. This idea is summarized well by the following:

"Every problem was once a solution to a previous problem."
- Bob Mandel

The solutions developed in this dissertation will help create future problems for further study and exploration. This is the nature of scientific discovery: continual questioning, hypothesizing, and testing. The fire embodied by this dissertation will live on to fuel future explorations of new problems and solutions, and more problems and more solutions.

APPENDIX A

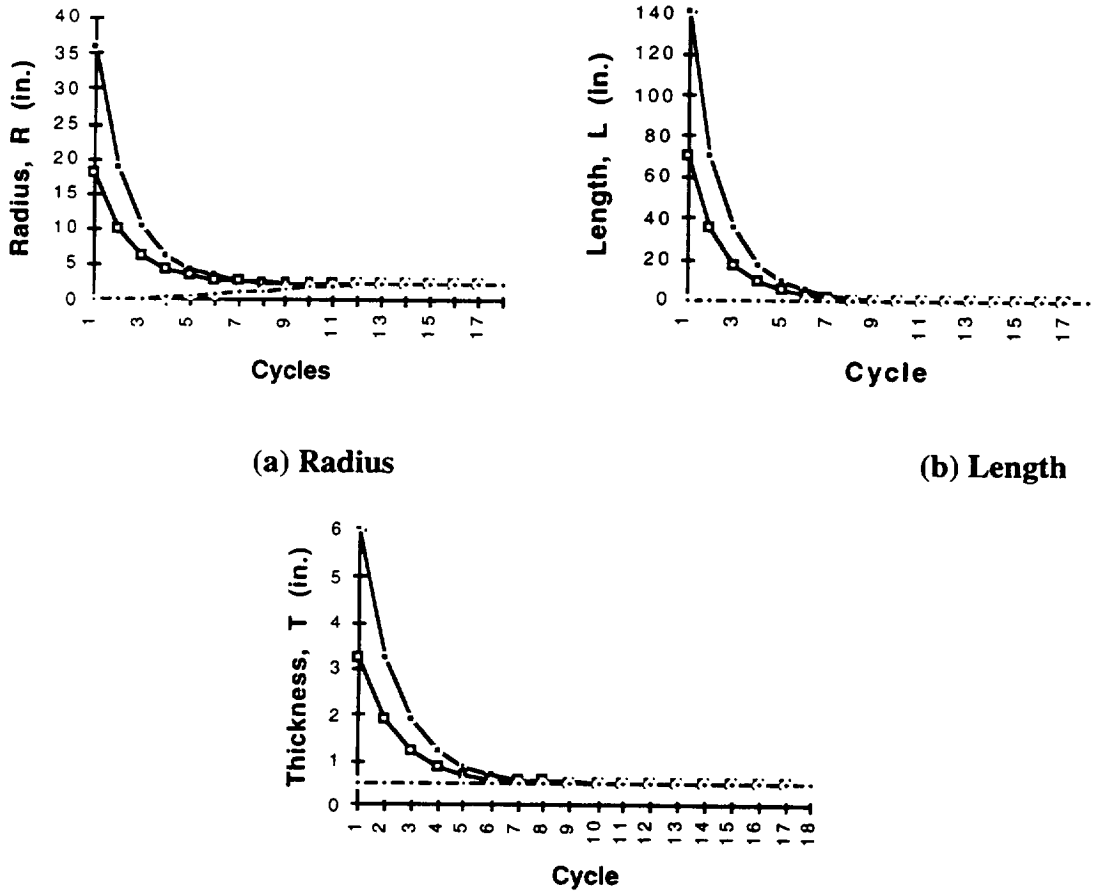
THE PRESSURE VESSEL PROBLEM, VERIFICATION OF GAME THEORY TECHNIQUES

In this Appendix, the full results for the pressure vessel verification example presented in Chapter 5 are given. Included are full results for the cooperative and leader/follower protocols.

Cooperative Protocol

Deviation Function = $W_1 * d_W^+ + W_2 * d_V^-$ where d_W^+ is the deviation function associated with the weight goal, and d_V^- is the deviation function associated with the volume goal.

Case 1: $W_1 = 1.0$ and $W_2 = 0.0$ *(emphasis on minimizing weight)*



(a) Radius

(b) Length

(c) Thickness

Figure A.1 Solution History: Cooperative (Minimizing Weight)

In Figure A.2, the weight is minimized, and since the weight player is the leader, the volume is constrained by the weight's solution. Therefore, as shown in Figure A.3, the volume is subsequently minimized as well (the volume player wants to *maximize* volume).

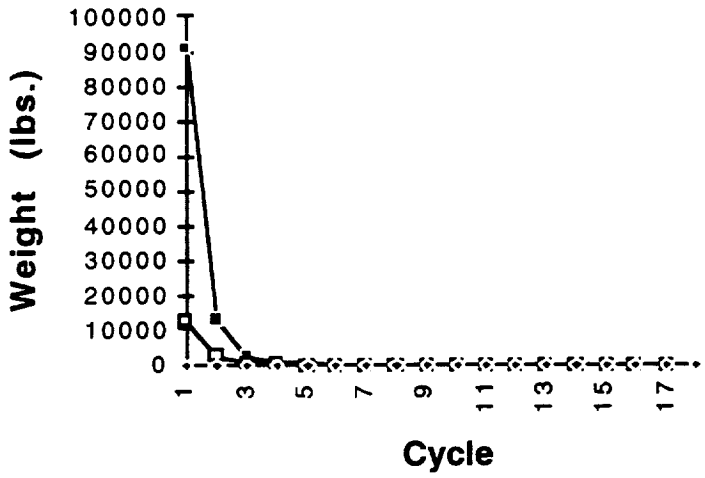


Figure A.2 Weight History: Cooperative (Minimizing Weight)

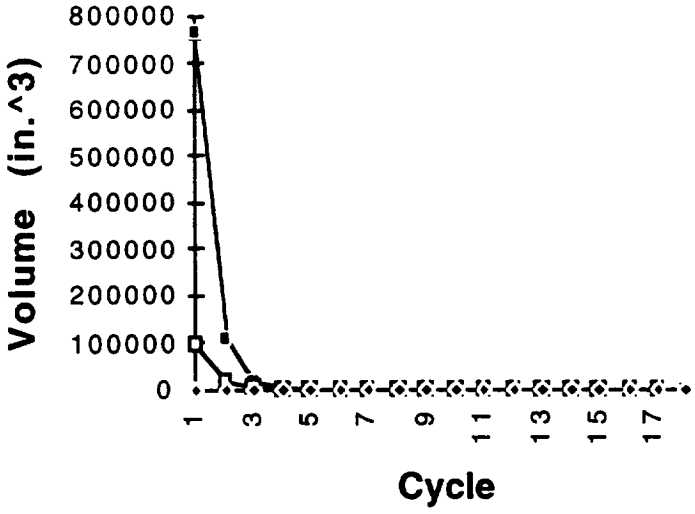
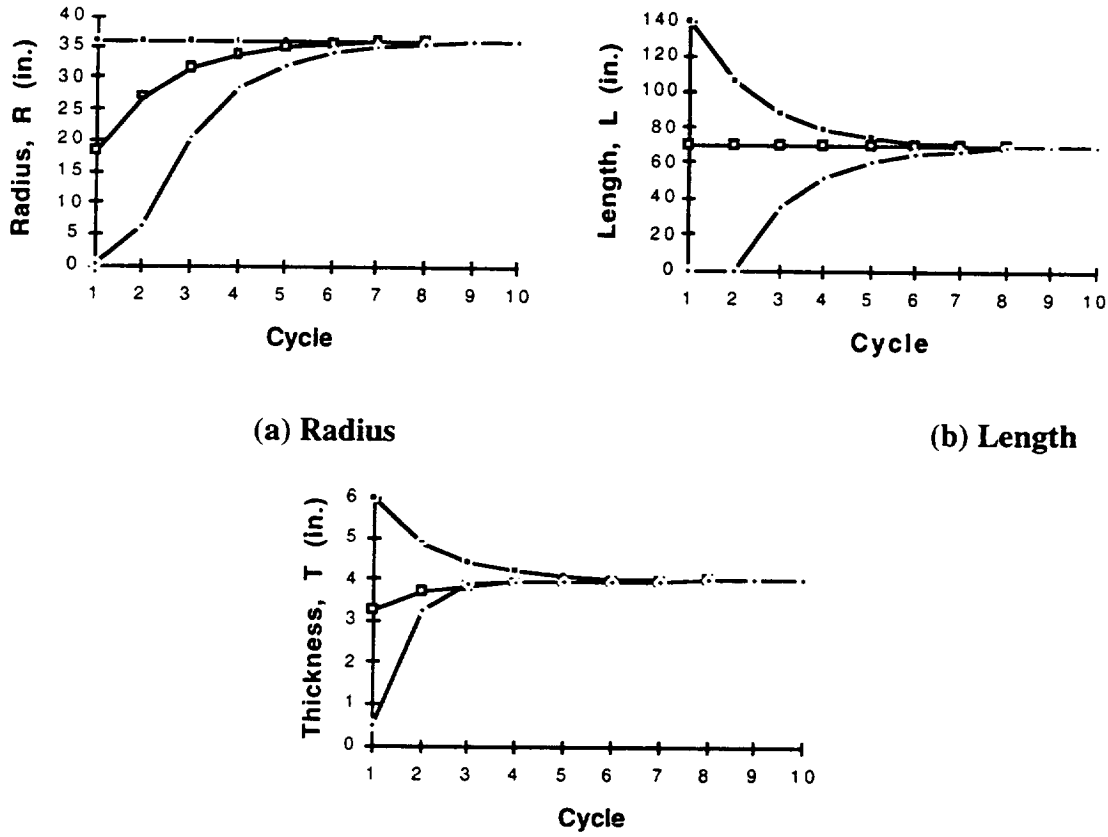


Figure A.3 Volume History: Cooperative (Minimizing Weight)

Case 2: $W_1 = 0.0$ and $W_2 = 1.0$ *(emphasis on maximizing volume)*



(a) Radius

(b) Length

(c) Thickness

Figure A.4 Solution History: Cooperative (Maximizing Volume)

In Figure A.6, the volume is maximized, and since the volume player is the leader, the weight is constrained by the volume's solution. Therefore, as shown in Figure A.5, the weight is subsequently increased compared to Figure A.3 (the weight player wants to *minimize* weight).

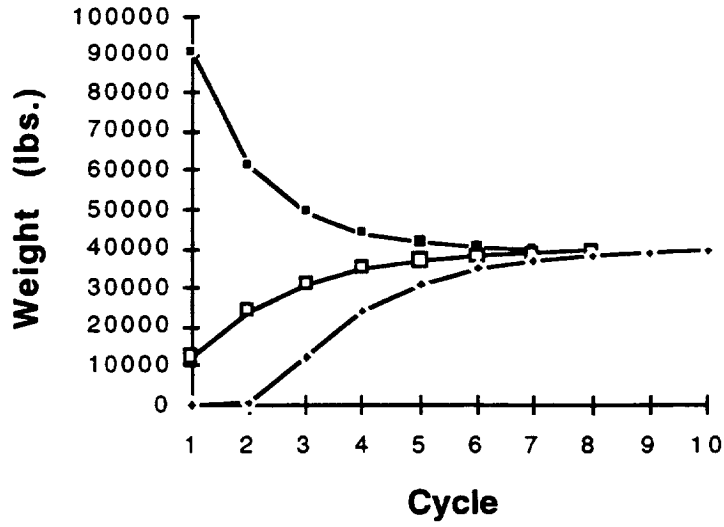


Figure A.5 Weight History: Cooperative (Maximizing Volume)

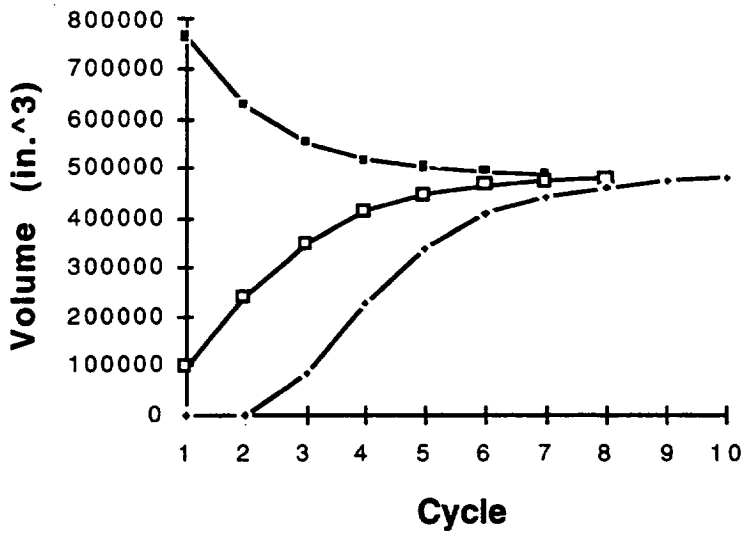


Figure A.6 Volume History: Cooperative (Maximizing Volume)

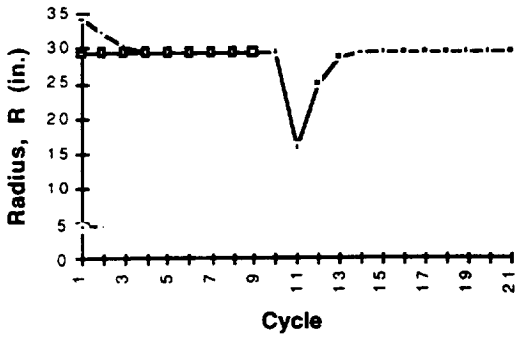
Leader Follower Protocol

Weight as the leader

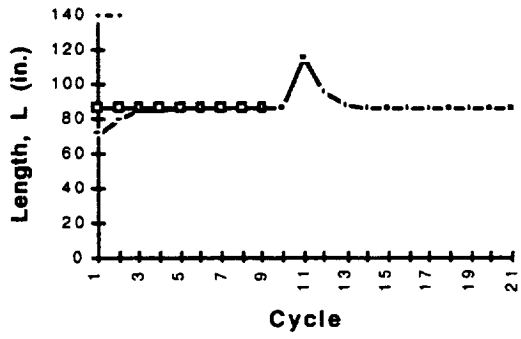
The three different starting points converge to two different solutions. The best solution is found when the lower bound of the thickness is used by the Weights player (Figure A.7 (c)). The corresponding solution of the Volume Player, as dictated by his RRS, is shown in Figure A.7 (a) and (b). The best solution is found when T = 0.5 in., R = 4.0 in., and L=140 in.. The weight corresponding to this solution is shown in Figure A.8 (Weight = 635 lbs.). As is illustrated in Figure A.8, the other two solutions have significantly higher weights. Convergence to the same solution is not achieved for all three points because of the existence of two separate, feasible design regions. The best solution corresponds to the solution reported in 5.6.3.

Volume as Leader

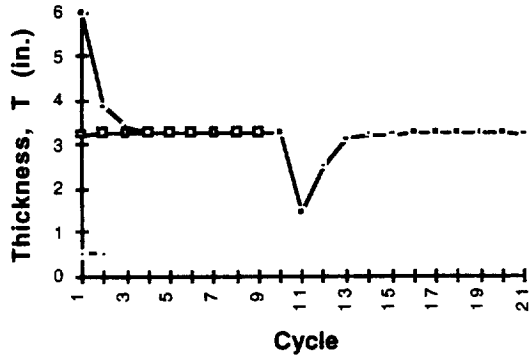
Again, the three different starting points converge to two different solutions. The best solution is found when the upper bounds of the radius and length are used as the starting point by the Volume player (Figures A.7 (a) and (b)). The corresponding solution of the Weight Player, as dictated by his RRS, is shown in Figure A.7 (c). The best solution is found when R = 36.0 in., L = 70.0 in., and T = 4.0 in.. The volume corresponding to this solution is shown in Figure A.10 (Weight = 39800 in^3.). As is illustrated in Figure A.10, the other two solutions have significantly lower volumes. Convergence to the same solution is not achieved for all three points because of the existence of two separate, feasible design regions. The best solution corresponds to the solution reported in 5.6.3.



(a) Radius



(b) Length



(c) Thickness

Figure A.7 Solution History: Weight as Leader

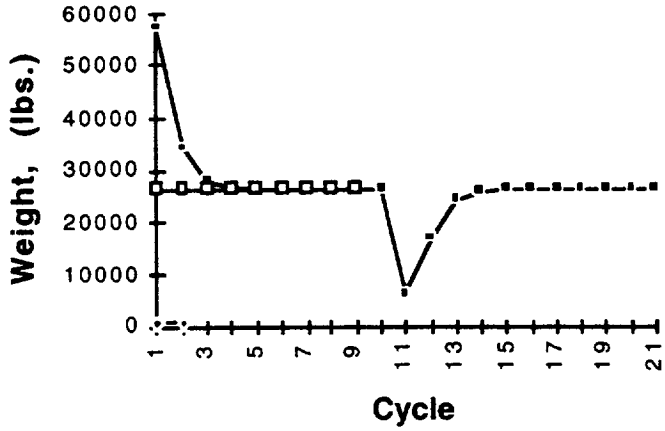
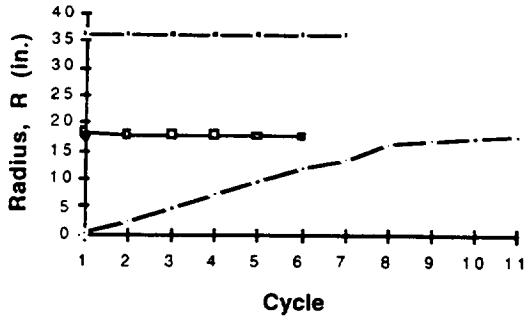
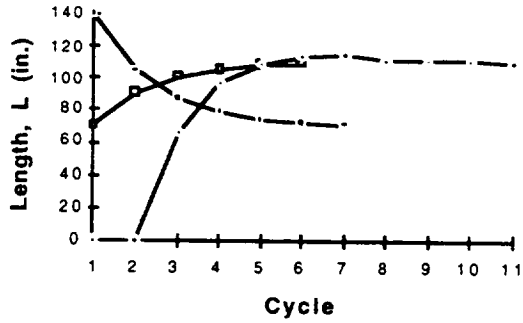


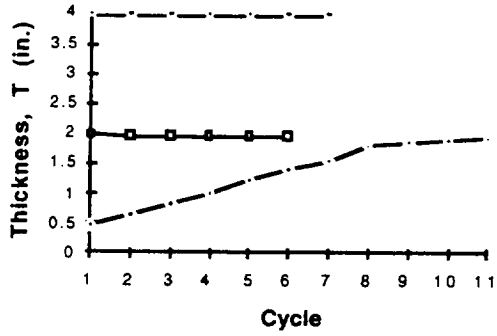
Figure A.8 Weight History: Weight as Leader



(a) Radius



(b) Length



(c) Thickness

Figure A.9 Solution History: Volume as Leader

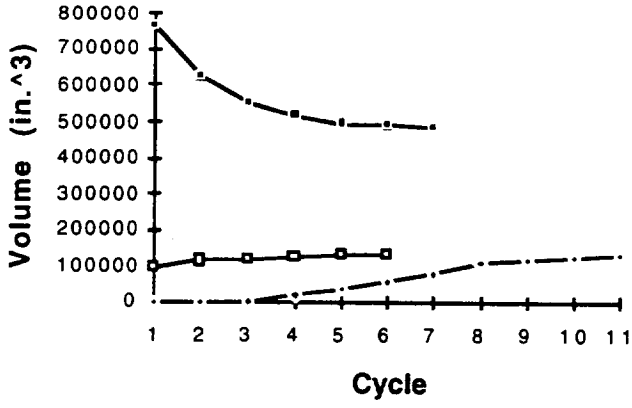


Figure A.10 Volume History: Weight as Leader

APPENDIX B

THE FORAGING-DIRECTED ADAPTIVE LINEAR PROGRAMMING ALGORITHM: ASSOCIATED CODE AND RESULTS

In this Appendix, the computer code associated with the Foraging-directed Adaptive Linear Programming Algorithm is given, along with full results for the verification examples used in Chapter 6. Firstly, the computer code, including updated files for the DSIDES manual and source code, and the course code for the foraging scheme, are given. Secondly, full results from the spring design problem, including the DSIDES data file, are given. Lastly, full results from the pressure vessel design problem, including the DSIDES data file, are given.

COMPUTER CODE: UPDATED DSIDES MANUAL

LIST OF DATA BLOCKS

Mandatory Blocks

PTITLE	1	Problem title
NUMSYS	2	Number of System Variables
SYSVAR	3	Description of System Variables - name, type, bounds and guess value
NUMCAG	4	Number of Constraints and Goals
LINCON	5	Linear Constraints - names and data *(if specified in NUMCAG)*
LINGOL	6	Linear Goals - names and data *(if specified in NUMCAG)*
DEVFUN	7	Deviation Function - number of levels and weights of deviation variables
STOPCR	8	Stopping Criteria (run and principal print flags, NITER, EPSZ, EPSX)

Optional Blocks

NLINCO	9	Names of Nonlinear Constraints (default names: NLCO##)
NLINGO	10	Names of Nonlinear Goals (default names: NLGO##)
INITFS	11	Automatic Generation of Initial Feasible Solution
ALPOUT	12	Flags for Output Level, Post Processor and Time Statistics
USRMOD	13	Flags for User Modules (USRINP, USROUT, USRMON, USRLIN)
USRDAT	14	User Data Block for Access From USRINP
OPTIMP	15	Optimization Parameters (VIOLIM, REMO, STEP)
ADPCTL	16	Nonlinear Inequality Constraint Adaption Flag (LADAP)
USERAN	17	Information for USRANA (maximum cycles - NANCY, NSYCY)
FIXVAR	18	Fixing of Variables
SUPCON	19	Suppression of Nonlinear Constraints
PVALFX	20	Particular Values for Stationarity of System Variables
PVEPSZ	21	Particular Values for Stationarity of Deviation Function Levels
PVSTEP	22	Particular Values for STEP
PVCVIL	23	Particular Values for VIOLIM
PVREMO	24	Particular Values for REMO
PVDISC	25	Particular Values for DISCRETE variables
ADREMO	26	Adaptive Reduced Move Parameters
XPLORE	27	Explore the design space for best initial points
ENDPRB	28	End of Problem Definition

2. NUMSYS (Mandatory)

Purpose: Define number of system variables – real, discrete, and boolean.

Format:

i j k

Variables:

i: integer	Number of real variables
j: integer	Number of discrete variables (including integers)
k: integer	Number of boolean (selection) variables

Example:

```
NUMSYS     : Number of system variables
  3   2   2   : real, discrete, boolean
```

Notes:

- If you do not have any variables of one type, you must indicate this by specifying a value of zero. In other words, three integers (i, j, and k) must be specified on the second line of the block.
- Number of discrete variables includes the number of integer variables.
- If you have discrete variables that are not integer-valued, you must specify the possible values in the block PVDISC.

3. SYSVAR (Mandatory)

Purpose: Define system variable information.

Format:

name k min max guess

Variables:

name: string	Name of variable (6 characters long)
k: integer	Serial number of variable
min: (real/integer)	Lower bound for variable
max: (real/integer)	Upper bound for variable
guess: (real/integer)	Initial guess value for variable

Example:

```
SYSVAR    : System variable information
weight  1  10.    100.0  33.5   :  weight of assembly
length  2  2.0    25.0   18.0   :  length of assembly
height  3  0.0    100.0  15.0   :  height of assembly
nteeth  4  20     60     45     :  number of teeth
gerdrv  5   0      1      1     :  use gear drive
bltdrv  6   0      1      0     :  use belt drive
```

Notes:

- Real variable must precede the integer/discrete variables and the boolean variables should follow the integer/discrete variables.
- If the variable name is not given, a default name is assigned to the variable. This is of the form X## where ## is the serial number of the variable, e.g., X1, X45 etc.
- Variable types are assigned based on the serial number of the variable and are shown in the output file.
- Lower and upper bounds for boolean variables are 0 and 1, respectively. The guess value can be either 0 or 1.
- If initial guess value is out of the specified bounds, a default value of $\left[\frac{\text{min+max}}{2}\right]$ is assumed and a warning is printed in output file.

25. PVDISC (Optional)

Purpose: Particular values for discrete variables.

Format:

k
i n guess
discrete values

Variables:

k: integer	Number of discrete variables to follow
i: integer	Serial number of variable
n: integer	Number of possible discrete values to follow
guess: integer	Serial number of initial guess value for discrete variable
discrete values:	n possible values of discrete variable number i

Example:

```
PVDISC : Particular values for discrete variables
2: 2 variables have discrete values
3  8  1: variable number 3 has 8 possible values, initial
        value is value number 1
2.5  4.0  7.4  9.0  12.1  13.0  14.8  16.1
4  4  4: variable number 4 has 4 possible values, initial
        value is value number 4
0.234  0.576  0.856  1.125
```

Note:

- This is used when discrete or integer variables can have values other than only integers between the MIN and MAX from the SYSVAR block.
- The guess value from this block overrides the one set in SYSVAR for the discrete variables specified in this block.
- The maximum number of discrete values for a given variable is set at 50.

COMPUTER CODE: UPDATED DSIDES ROUTINES

```
C+
C*************************************************************
C
C Program ALPCTL
C
C Purpose:  Main program for DSIDES: SLIPML Version 4.80 /
C                                    ALP Release 1.0
C
C*************************************************************
C-
      COMMON/ADINTE/ NRELV, NDISV, NVSEL, NDESV, NDVUSR, NDEVAR,
     &               NLINCO, NLINGO, NMPRI,
     &               NNLINQ, NNLEQU, NNLCON, NNLGOA, NNLTOT
      INTEGER NRELV, NDISV, NVINT, INDEX(MDESV),
     &        NVSEL, NDESV, NDVUSR, NDEVAR, NDSCC(MDESV)
      INTEGER NLINCO, NLINGO, NMPRI
      INTEGER NNLINQ, NNLEQU, NNLCON, NNLGOA, NNLTOT
C
      REAL DESVAR(MDESV), DUMVAR(MDESV), CONDEV, DEVFUN(MLEVEL),
     &     DEVVAR(MDEVV), GVAL(MNLNCG), Z2(MLEVEL),
     &     TABUN(MDESV,MDSCV)

      LOGICAL LPRCOV, LCOVIL, LVDISC
.
.
.
.
.
.
.
.
C
C     Read in control information and initialize values
C     - Call ALPDAT
C
      OPEN (UNIT=NUINP, FILE='ALPINP.DAT', ACCESS='SEQUENTIAL',
     &      FORM='FORMATTED', STATUS='OLD')
C
      OPEN (UNIT=NUSER, STATUS='SCRATCH',
     &      FORM='FORMATTED', ACCESS='SEQUENTIAL')
C
      CALL ALPDAT (NUOUT, NUINP, NUSER,
     &       NRELV, NDISV, NVINT, TABUN, NDSCC, INDEX, NVSEL,
     &       NDESV, NDEVAR,
     &       NLINCO, NLINGO, NMPRI,
     &       NNLCON, NNLEQU, NNLGOA, NNLINQ, NNLTOT,
     &       NANCY, NSYCY,
     &       NHJMAX, NADREM, IACTVR, IADCON, LISIGN,
```

```
     &        NPTGEN, NPTBST, IGSEED, IGENFX,
     &        IDDESV, IDDEVR, IDLICO, IDLIGO, IDNLCO, IDNLGO,
     &        PTITLE, COFLIN, RHSLIN, DFNCOF,
     &        PESTEP, REDMOV, VBOUNS, DESVAR,
     &        HJEXPA, HJCONT, HJSTEP, HJEPSY, HJDELT, DELREM,
     &        FRACZ, FRACX, VILCN,
     &        LFATAL, LDRYRN, LPRFIN,
     &        LPROUT, LPPROC, LTIME, LADREM, LADAP, LINIT,
     &        LMON, LUINP, LUOUT, LVCOF, LXPLOR, LPRGEN, LVDISC)
C
      CLOSE (NUINP)
C
C     STOP program if fatal errors encountered during reading
.
.
.
.
.
.
.
C
C        SYNTHESIS CYCLES ONLY
C        '''''''''''''''''''''
C
C        Obtain and record current timer values.
C
         CALL TIMER (NUOUT, 2, TIMCOM, CURTIM, INITIM, EXETIM )
C
C******Discrete Part of solution -> Call FORAGING Algorithm
C
     IPATH = 1

     IF( LVDISC ) THEN
          INTFLAG = 1
     ELSE
          INTFLAG = 50
     ENDIF

 83  IF (INTFLAG.LT.14) THEN

     CALL FORAGEMV(INTFLAG,DESVAR,NDESV,NRELV,NDISV,TABUN,
     &          NDSCC,IPATH,NNLTOT,NOUT,NNLCON, NNLGOA,
     &          DFNCOF, NMPRI, INDEX, VBOUNS, IACTVR)


C    SET INACTIVE vars to discrete ones

          DO 82 K=NRELV+1,NRELV+NDISV
               IACTVR(K) = 0
 82       CONTINUE
     ENDIF
```

```
C
C
C
C     Perform Synthesis cycles
C      - Call ALPMOD
C      - Returned DESVAR corresponds with best nonlinear
         solution
C
         INUMAN = 0
C
         CALL ALPMOD (NANCY, INUMAN, NTITER, NUOUT, NUPPI,
     &           NRELV, NDISV, NVSEL, NDESV, NDEVAR,
     &           NLINCO, NLINGO,
     &           NNLINQ, NNLEQU, NNLCON, NNLGOA, NNLTOT,
     &           NMPRI, NSYCY, IACTVR, IADCON, LISIGN,
     &           JSYCY, NADREM,
     &           CONDEV, DELREM,
     &           COFLIN, DESVAR, DFNCOF,
     &           DEVFUN, DEVVAR, FRACX, FRACZ, GVAL,
     &           PESTEP, REDMOV, RHSLIN, VBOUNS, VILCN,
     &           LADAP, LADREM, LMON, LVCOF, LPRFIN, LPROUT,
     &           LPPROC, LCONDF, LCONSV, LIMPRV,
     &           IDDESV, IDDEVR, IDLICO, IDLIGO, IDNLCO, IDNLGO)

C
C    This is to check for discrete number of synthesis cycles
C    complete
C
     WRITE (NUOUT,*) "*****************************************"
     WRITE (NUOUT,*) "This is the end of synthesis cycle
    &      ",INTFLAG
     WRITE (NUOUT,*) "*****************************************"

     IF (INTFLAG.LT.1) THEN
          INTFLAG = INTFLAG + 1
          goto 83
     ENDIF
C
C        Obtain timer results for current analysis cycle.
C
     CALL USRSET(IPATH, NDESV, MNLNCG, NOUT, DESVAR,
     &              CONSTR, GOALS)

      IF ( LTIME ) THEN
       TIMCOM = 'Time required to complete synthesis cycles:'
       CALL TIMER ( NUOUT, -3, TIMCOM, CURTIM, EXETIM, EXETIM )
      ENDIF
C
C
.
.
```

```
C+
C*****************************************************************
**
C
C Subroutine ALPDAT
C
C Purpose:  This routine reads the ALP data file for
C           compromise DSPs and sets the necessary defaults.
C
C-----------------------------------------------------------------
C Arguments    Name     Type   Description
C ---------    ----     ----   -----------
C Input:       NUINP    int    unit number of input data file
C              NUOUT    int    unit number of output data file
C              NUSER    int    unit number of user data file
C
C Output:      NDESV    int    number of design variables
C              NDEVAR   int    number of deviation variables
C              NRELV    int    number of real (continuous)
C                                    variables
C              NDISV    int    number of discrete variables
C                                    (inc. integer)
C              DSTEP    real   step for discrete variables (1
C                                   default)
C              NVALUS   int    number of discete values (overide
C                                   step) MAX = 50
C              NUMNGH   int    number of variables to have tabu
C                                   Nghbrhd
C              INDEX    int    index of initial discrete vars
C                                   (for tabu)
C              NEIGH    int    counter to set up tabu with DSTEP
C                                   = 1.0
C              TABUN    real   tabu neighborhood (MDESV, 50)
C              NVINT    int    number of integer variables
C
C*****************************************************************
C-
      SUBROUTINE ALPDAT(NUOUT, NUINP, NUSER,
     &     NRELV, NDISV, NVINT, TABUN, NVALUS, INDEX, NVSEL,
     &       NDESV, NDEVAR,
     &     NLINCO, NLINGO, NMPRI,
     &     NNLCON, NNLEQU, NNLGOA, NNLINQ, NNLTOT,
     &     NANCY, NSYCY,
     &     NHJMAX, NADREM, IACTVR, IADCON, LISIGN,
     &     NPTGEN, NPTBST, IGSEED, IGENFX,
     &     IDDESV, IDDEVR, IDLICO, IDLIGO, IDNLCO, IDNLGO,
     &     PTITLE, COFLIN, RHSLIN, DFNCOF,
     &     PESTEP, REDMOV, VBOUNS, DESVAR,
     &     HJEXPA, HJCONT, HJSTEP, HJEPSY, HJDELT, DELREM,
     &     FRACZ, FRACX, VILCN,
     &     FATAL, LDRYRN, LPRFIN,
```

```
     &      LPROUT, LPPROC, LTIME, LADREM, LADAP, LINIT,
     &      LMON, LUINP, LUOUT, LVCOF, LXPLOR, LPRGEN, LVDISC)
C
      INCLUDE 'alplim.cmm'
C
C-------------------------------------
C     Arguments:
C-------------------------------------
C     Logical Unit numbers for I/O
C
      INTEGER NUOUT, NUINP, NUSER
C
C
    INTEGER  NRELV, NDISV, NVALUS(MDESV), DSTEP, INDEX(MDESV),
C               NUMNGH,
     &      NVINT, NVSEL, NDESV, NDEVAR,
     &      NLINCO, NLINGO, NMPRI,
     &      NNLCON, NNLEQU, NNLGOA, NNLINQ, NNLTOT,
     &      NANCY, NSYCY(MNANCY),
     &      NHJMAX, NADREM, IACTVR(MDESV), IADCON(MNLNCG),
     &      LISIGN(MLINCG),
     &      NPTGEN, NPTBST, IGENFX(MDESV), IGSEED
C
     CHARACTER*6 IDDESV(MDESV), IDDEVR(MDEVV),
     &      IDLICO(MLINCG), IDLIGO(MLINCG),
     &      IDNLCO(MNLNCG), IDNLGO(MNLNCG)
C
     CHARACTER*80 PTITLE(2)
C
      REAL COFLIN(MLINCG,MDESV), RHSLIN(MLINCG),
     &      DFNCOF(MLEVEL,MDEVV),
     &      PESTEP(MDESV), REDMOV(MDESV),
     &      VBOUNS(2,MDESV), DESVAR(MDESV),
     &      HJEXPA, HJCONT, HJSTEP, HJEPSY, HJDELT, DELREM,
     &      FRACZ(MLEVEL), FRACX(MDESV), VILCN(MNLNCG),
     &      TABUN(MDESV,MDSCV)
C
      LOGICAL FATAL,
     &      LDRYRN, LPRFIN,
     &      LPROUT(8), LPPROC, LTIME,
     &      LADREM, LADAP, LINIT,
     &      LMON, LUINP, LUOUT, LVCOF,
     &      LPRGEN, LXPLOR, LVDISC
C
C-------------------------------------
C     Local variables:
C-------------------------------------
C
C
C*************************************************************
C     Initialize Logical Flags
```

```
C*************************************************************
C
      LVDISC = .FALSE.
C
C     Set other defaults
C
C
C     Beginning of main GOTO loop
C     Read next block name
C
 1111 READ(NUINP,FMT='(A)',END=9000,ERR=8888)DUM
      CALL GETNAM(DUM, BLKNAM, KST, LNONAM)
C
      IF(LNONAM)THEN
        GO TO 1111
      ENDIF
C
      WRITE(NUOUT,3)BLKNAM
    3 FORMAT(/,X,'BLOCK ',A6,X,54('-'),/)
C
C BLOCK2 ------- NUMSYS ----------------------
C
C     Read number of design variables - Real, and dicrete
C
C     NRELV = Number of real variables
C     NDISV = Number of discrete variables
C     NVINT = Number of integer variables
C     NVSEL = Number of selection (boolean) variables)
C     NDESV = Number of standard variables.
C           = NRELV+NDISV
-----------------------------------------------
C
      IF(BLKNAM.EQ.'NUMSYS')THEN
        CALL BLKCHK(NUOUT,BKIN(2))
C
        NVINT = 0
        READ(NUINP,FMT=*,ERR=8888)NRELV,NDISV,NVSEL
C
     IF (NDISV.GT.0) THEN
          LVDISC = .TRUE.
     ENDIF

        NDESV = NRELV+NDISV+NVSEL
        WRITE(NUOUT,20)NRELV,NDISV,NVSEL,NDESV
   20   FORMAT(3X,' Number of real variables      = ', I5/,
     &    3X,' Number of discrete variables = ', I5/,
     &    3X,'Number of selection (Boolean) variables = ', I5/,
     &    3X,' Total number of design variables    = ', I5)
C
        IF(NDESV.GT.MDESV)THEN
          WRITE(NUOUT,21)MDESV,NDESV
```

```
          GO TO 9999
        ENDIF
   21   FORMAT('  E ** Problem Size',/
     &  3X,' Maximum number of design variables = ',I5,/
     &  3X,'                              Specified = ',I5)
C
        GO TO 1111
      ENDIF
C
CC BLOCK3 ------- SYSVAR -----------------------------------
C     Read design variable information
C
      IF(BLKNAM.EQ.'SYSVAR')THEN
        CALL BLKCHK(NUOUT,BKIN(3))
C
C       Check if NUMSYS has been read.
        IF(.NOT.BKIN(2))THEN
          WRITE(NUOUT,700)
          GO TO 9999
        ENDIF
C
        WRITE(NUOUT,30)
   30   FORMAT(2X,'  Number  Name  Type  Minimum   Maximum  ',
     &         3X,'Guess Value',/,
     &         2X,'  ------  ------  ----  ------  ---------',
     &         3X,'------------')
C
        DO 35 K=1,NDESV
          READ(NUINP,'(A)',ERR=8888)DUM
          CALL GETNAM(DUM, DUMNAM, KST, LNONAM)
          IF (LNONAM) DUMNAM = MKNAME('X',K)
C
          READ(DUM(KST:80),FMT=*,ERR=8888)J,XMIN,XMAX,XGES
C
C         Variable type assigned using serial number
C
          IF(J.LE.NRELV)THEN
            VTYPE='R'
          ELSEIF(J.GT.(NRELV+NDISV))THEN
            VTYPE='B'
          ELSE
            VTYPE='D'
          ENDIF
C
C         Check if values are within bounds
C
          IF((XGES.LT.XMIN).OR.(XGES.GT.XMAX))THEN
            XGES=0.5*(XMIN+XMAX)
            WRITE(NUOUT,32)J
   32       FORMAT(/,' I ** Guess value out of bounds, ',
     &        'reset to (XMIN+XMAX)/2 for variable number ',I3)
```

```
            ENDIF
C
            IDDESV(J)    = DUMNAM
            VBOUNS(1,J)  = XMIN
            VBOUNS(2,J)  = XMAX
            DESVAR(J)    = XGES
C
C     THIS SETS up the tabu neighborhood from the bounds.

      DSTEP = 1

      IF (J.GT.NRELV.AND.J.LT.(NRELV+NDISV+1)) THEN
            NVALUS(J) = VBOUNS(2,J) - VBOUNS(1,J) + 1

C     This sets the index number as the guess index.
            INDEX(J) = (DESVAR(J)-VBOUNS(1,J))/DSTEP + 1

C     SETS up the temporary counter for set of discrete values.
C     STARTS at the lower bound.

            NEIGH = VBOUNS(1,J)

            DO 37 L=1,NVALUS(J)
                  TABUN(J,L) = NEIGH
                  NEIGH = NEIGH + DSTEP
 37         CONTINUE
      ENDIF
C
C           Use different formats for REAL and other variables.
C
            IF(VTYPE.EQ.'R')THEN
              WRITE(NUOUT,33)J,IDDESV(J),VTYPE,VBOUNS(1,J),
     &                       VBOUNS(2,J),DESVAR(J)
   33       FORMAT(4X,I3,5X,A6,4X,A1,2X,G12.5,3X,G12.5,3X,G12.5)
            ELSE
        IF(VTYPE.EQ.'D')THEN
              WRITE(NUOUT,38)J,IDDESV(J),VTYPE,VBOUNS(1,J),
     &                       VBOUNS(2,J),DESVAR(J)
   38    FORMAT(4X,I3,5X,A6,4X,A1,2X,G12.5,3X,G12.5,3X,G12.5)
              ELSE
                KMIN=XMIN+0.5
                KMAX=XMAX+0.5
                KGES=XGES+0.5
                WRITE(NUOUT,34)J,IDDESV(J),VTYPE,KMIN,KMAX,KGES
   34           FORMAT(4X,I3,5X,A6,4X,A1,2X,I10,5X,I10,5X,I10)
        ENDIF
              ENDIF
C
   35    CONTINUE
         GO TO 1111
       ENDIF
```

```
C
C  BLOCK4 ------- DISCRETE ---------------------------------
C     Read discrete design variable information
C
      IF(BLKNAM.EQ.'PVDISC')THEN
        CALL BLKCHK(NUOUT,BKIN(28))
C
     LVDISC = .TRUE.
     WRITE(NUOUT,285)
 285 FORMAT(' I ** User inputted discrete data will be
          used',//)

C
     READ(NUINP,FMT=*,ERR=8888)NUMNGH

     DO 284 J=1,NUMNGH

     READ(NUINP,FMT=*,ERR=8888)NUM,NVALUS(NUM),XGES
       IF (NUM .LE. NRELV) THEN
          WRITE(NUOUT,282)NUM
                GO TO 9999
       ENDIF
       IF (NUM .GT. NRELV+NDISV) THEN
          WRITE(NUOUT,282)NUM
                GO TO 9999
       ENDIF
       IF (XGES.LT.1 .OR. XGES.GT.NVALUS(NUM)) THEN
          GO TO 9999
       ENDIF
     IF (NUM .GT. 0) THEN
     READ(NUINP,FMT=*,ERR=8888) (TABUN(NUM,I), I=1,NVALUS(NUM))
C  Guess value is read is as initial DESVAR
        INDEX(NUM) = XGES
        DESVAR(NUM) = TABUN(NUM,XGES)
        WRITE(NUOUT,281)NUM,IDDESV(NUM)
        WRITE(NUOUT,*)(TABUN(NUM,I),I=1,NVALUS(NUM))
        WRITE(NUOUT,286)DESVAR(NUM)
          ENDIF
 284 CONTINUE

 281     FORMAT (4X,'Variable No. ', I2, ' (', A, ')'
    &         ' has the following discrete values possible.')
 282     FORMAT(' E ** Variable number ',I3,' is not discrete')
 286       FORMAT(5X,'and the initial value is:',F6.4)

       GO TO 1111
     ENDIF
.
.
.
      END
```

COMPUTER CODE: FORAGING SOLVER

```
C+
C**************************************************************
C
C Subroutine FORAGEMV
C
C Purpose: Solve the discrete problem
C
C-------------------------------------------------------------
C Arguments     Name Type Description
C ---------     ---- ---- -----------
C Input:   NDITER      I   Number of Calls to Discrete Routine
C          DESVAR      R   Vector of Design Variables
C          NDESV       I   Number of Design Variables
C          NRELV       I   Number of Real Variables
C          NDISV       I   Number of Discrete Variables
C          XX          R   Neighborhood Structure of Discrete
C                          Vars
C          NVALS       I   Number of Possible Discrete Values
C          IPATH       I
C          NNLTOT      I   Total number of nonlinear consts and
C                          goals
C          NOUT        I
C          NNLCON      I   Number of nonlinear constraints
C          NNLGOA      I   Number of nonlinear goals
C          DFNCOF      R   Vector of deviation function weights
C          NMPRI       I   Number of Priority Levels
C          INDEX       I   Index Marker for Discrete Variables
C          VBOUNS      R   Bounds on the design variables
C          IACTVR      I   Array of inactive variables
C
C Local variables:
C
C          MAXIT       I   Maximum number of neighborhood
C                          searches
C          NUMBST      I   Number of solution to keep in schema
C          MEMLIM      I   Memory limit, initial value
C          FEASFL      I   Flag of feasibility
C          CONT        R   Discretized continuous variable
C          CONTSTEP    R   Discretization step for continuous
C                          vars
C          MEM         I   Memory array
C          INDEX       I   Index of current variable values
C          DFLAG       I   Diversification Flag
C          DFLAG2      I   Diversification Flag: One-time check
C          FIINDX      I   Final Indices of best solution
C          DVTEMP      R   Temporary buffer for continuous vars
C          ALLBEST     R   Best deviation function
C          BESTOBJ     R   Current best objective
C          TEMPX       R   Temporary buffer for neighborhood
```

```
C                              checks
C          BESTX       R   Vector of the current best design
C                              variables
C          STARTOBJ    R   Initial deviation function value
C          STARTCON    R   Initial constraint violation
C          CONVIO      R   Current constraint violation
C          SCHEMA      R   Vector of NUMBST solutions
C          DEVFUN      R   Current deviation function
C          ABESTX      R   Final best design variables
C
C Output:
C
C Input/Output
C
C-----------------------------------------------------------------
C Common Blocks: none
C
C Include Files: alplim.cmm
C
C Calls to: LOCL, OBJ
C-----------------------------------------------------------------
C Development History
C
C Author: Kemper Lewis
C Date:   October 25, 1995
C
C Modifications:
C
C*****************************************************************
C
C*****************************************************************
C
*
C    MAIN PROGRAM:  Computes the best possible solution
C    possible for any DISCRETE problem using a
C    local improvement scheme. In particular, a
C    search of the O(n) neighorhood structure is employed.
C    A foraging search is employed (with aspiration NEWZ>BSTZ).
C    A dynamic memory is used.
C    Check all neighbors before picking best admissable move.
C    Frequency-based diversification capability added.
C    Schema list is created for each problem.
C
C*****************************************************************
*
C
C
      SUBROUTINE FORAGEMV (NDITER, DESVAR, NDESV, NRELV, NDISV,
     $                     XX, NVALS, IPATH, NNLTOT, NOUT,
     $                    NNLCON, NNLGOA, DFNCOF, NMPRI, INDEX,
     $                    VBOUNS, IACTVR)
```

```
      INCLUDE 'alplim.cmm'

C
C-------------------------------------
C     Arguments:
C-------------------------------------
C
       INTEGER NN, NDESV, NVALS(MDESV),
     $          IACTVR(MDESV),NDISV, MAXIT,
     $          NRELV, NMPRI, NDITER, NSIZE

      INTEGER I, J, K, L, M, N, NUMBST,  MEMLIM, FEASFL

C     COMMON/ADREAL/ VBOUNS(2,MDESV)

      REAL VBOUNS(2,MDESV), CONT, CONTSTEP

       PARAMETER (NN=1,
     $             NUMBST=10, NSIZE=5)
C
C*****************************************************************
*
C
*
C      NN      - number of desired random starting configurations
*
C
*
*****************************************************************
*
C
C     NOTE: This is for 50 discrete choices, can make this as
C     large as the problem is.

       INTEGER MEM(MDESV,MDSCV,MDSCV),
     $  INDEX(MDESV), NDESV, MEMPLC(MDESV), DFLAG(MDESV,MDSCV),
     $     DFLAG2(MDESV,MDSCV), NNINDX(MDESV), FIINDX(MDESV)

       INTEGER NNLCON, NNLGOA, IPATH, NOUT, NNLTOT, BESTINDX,
     $  REPEAT(MDESV,MDSCV,MDSCV), RESTART
       REAL DESVAR(MDESV), DVTEMP(MDESV), DFNCOF(MLEVEL,MDEVV),
C     $     ALLBEST

       REAL BESTOBJ, TEMPX(MDESV), BESTX(MDESV), STARTOBJ,
C     $     STARTCON,
     $     CONVIO, XX(MDESV,MDSCV), SCHEMA(NUMBST,MDESV+1),
     $     DEVFUN, ABESTX(MDESV)

      open (12,file='tpitab.out',status='unknown')
      open (13,file='points.out',status='unknown')
```

```
      open (14,file='schema.out',status='unknown')
      open (15,file='ppinfo.vars',status='unknown')
      open (17,file='ppinfo.zcon',status='unknown')
      open (16,file='debug.out',status='unknown')

*****************************************************************
*     Set up number of neighborhood searches and length of
*     memory according to the size of the problem
*****************************************************************

      IF (NRELV+NDISV.GT.4) THEN
           MAXIT = 200
           MEMLIM = 67
      ENDIF
      IF (NRELV+NDISV.GT.2 .AND. NRELV+NDISV.LE.4) THEN
           MAXIT = 100
           MEMLIM = 50
      ENDIF
      IF (NRELV+NDISV.LE.2) THEN
           MAXIT = 50
           MEMLIM = 20
      ENDIF

      DO 12 I=1,NDESV
           TEMPX(I) = DESVAR(I)
           BESTX(I) = DESVAR(I)
 12   CONTINUE

      FEASFL = 0

c************Calculate initial objective function

       CALL OBJEC(CONVIO,DEVFUN,IPATH,NDESV,DESVAR,DFNCOF,NMPRI,
     $                NNLTOT,NNLCON,NNLGOA)
      STARTOBJ = DEVFUN
      STARTCON = CONVIO
       WRITE(12,*) 'Initial design variables'
       WRITE(12,*) (DESVAR(I),I=1,NDESV)
      WRITE(12,*) 'Initial DEVFUN and CONVIO: ',DEVFUN, CONVIO

C******** loop around tabu search subroutine NN times

      ALLBEST = 10000000
       DO 2700 I=1,NN


*****************************************************************
*     Discretizing the continuous domain to explore           *
*****************************************************************
```

```
      IF (I.EQ.1) THEN

      CONTSTEP = 0.1

      DO 699 J  = 1,NRELV
           DVTEMP(J) =
     &     (DESVAR(J)-VBOUNS(1,J))/(VBOUNS(2,J)-VBOUNS(1,J))
699   CONTINUE

      DO 700 J = 1,NRELV

           IF (IACTVR(J).EQ.0) THEN
                NVALS(J) = 1
                INDEX(J) = 1
                GOTO 700
           ENDIF

           IF(DVTEMP(J).EQ.0) THEN
                INDEX(J) = 1
           ELSE
           IF(DVTEMP(J).EQ.0.1) THEN
                INDEX(J) = 2
           ELSE
           IF(DVTEMP(J).EQ.0.2) THEN
                INDEX(J) = 3
           ELSE
           IF(DVTEMP(J).EQ.0.3) THEN
                INDEX(J) = 4
           ELSE
           IF(DVTEMP(J).EQ.0.4) THEN
                INDEX(J) = 5
           ELSE
           IF(DVTEMP(J).EQ.0.5) THEN
                INDEX(J) = 6
           ELSE
           IF(DVTEMP(J).EQ.0.6) THEN
                INDEX(J) = 7
           ELSE
           IF(DVTEMP(J).EQ.0.7) THEN
                INDEX(J) = 8
           ELSE
           IF(DVTEMP(J).EQ.0.8) THEN
                INDEX(J) = 9
           ELSE
           IF(DVTEMP(J).EQ.0.9) THEN
                INDEX(J) = 10
           ELSE
           IF(DVTEMP(J).EQ.1.0) THEN
                INDEX(J) = 11
           ELSE
                INDEX(J) = 6
```

```
                DESVAR(J) = (VBOUNS(2,J) + VBOUNS(1,J))/2
            ENDIF
            ENDIF
            ENDIF
            ENDIF
            ENDIF
            ENDIF
            ENDIF
            ENDIF
            ENDIF
            ENDIF
            ENDIF
            ENDIF

            IF (INDEX(J).EQ.0) THEN
            WRITE(16,*)'You have entered an incorrect variable
     &                 guess for continuous variable number ',J
                       goto 999
                ENDIF
            NVALS(J) = 11
            CONT = 0.0

*****************************************************************
*     Set up continuous neighborhoods                           *
*****************************************************************

            DO 701 K = 1,NVALS(J)
                XX(J,K) = CONT
                CONT = CONT + CONTSTEP
                XX(J,K) = XX(J,K)*(VBOUNS(2,J)-VBOUNS(1,J))
     &                    + VBOUNS(1,J)
 701        CONTINUE

 700  CONTINUE

      ENDIF

      IF (I.EQ.2) THEN
           DO 710 J=1,NRELV
                IF(MOD(J,2).EQ.0) THEN
                DESVAR(J) = XX(J,1)*(VBOUNS(2,J)-VBOUNS(1,J))
     &                    + VBOUNS(1,J)
                INDEX(J) = 1
                ELSE
                DESVAR(J) = XX(J,NVALS(J))*
     &                (VBOUNS(2,J)-VBOUNS(1,J)) + VBOUNS(1,J)
                INDEX(J) = NVALS(J)
                ENDIF
 710        CONTINUE
            DO 713     J=NRELV+1,NRELV+NDISV
                IF(MOD(J,2).EQ.0) THEN
                DESVAR(J)=XX(J,1)
```

```
                INDEX(J) = 1
                ELSE
                DESVAR(J)=XX(J,NVALS(J))
                INDEX(J) = NVALS(J)
                ENDIF
713         CONTINUE
      ENDIF

      IF (I.EQ.3) THEN
            DO 711 J=1,NRELV
                IF(MOD(J,2).EQ.0) THEN
                DESVAR(J) = XX(J,NVALS(J))*
     &                (VBOUNS(2,J)-VBOUNS(1,J)) + VBOUNS(1,J)
                INDEX(J) = NVALS(J)
                ELSE
                DESVAR(J) = XX(J,1)*(VBOUNS(2,J)-VBOUNS(1,J))
     &                 + VBOUNS(1,J)
                INDEX(J) = 1
                ENDIF
711         CONTINUE
            DO 714     J=NRELV+1,NRELV+NDISV
                IF(MOD(J,2).EQ.0) THEN
                DESVAR(J)=XX(J,NVALS(J))
                INDEX(J) = NVALS(J)
                ELSE
                DESVAR(J)=XX(J,1)
                INDEX(J) = 1
                ENDIF
714         CONTINUE
      ENDIF

      IF (I.EQ.4) THEN
            DO 712 J=1,NRELV
                DESVAR(J) = XX(J,6)*
     &                (VBOUNS(2,J)-VBOUNS(1,J)) + VBOUNS(1,J)
                INDEX(J) = 6
712         CONTINUE
            DO 715     J=NRELV+1,NRELV+NDISV
                IF(MOD(NVALS(J),2).EQ.0) THEN
                     DESVAR(J)=XX(J,NVALS(J)/2)
                     INDEX(J) = NVALS(J)/2
                ELSE
                     DESVAR(J)=XX(J,(NVALS(J)+1)/2)
                     INDEX(J) = (NVALS(J)+1)/2
                ENDIF
715         CONTINUE
      ENDIF


      RESTART = I
```

```
c********Initialize short,long term memory and diversification

      IF (NDITER.EQ.1 .AND. I.EQ.1) THEN

            DO 2 J=1,MDESV
         DO 2 K=1,MDSCV
            DO 2 L=1,MDSCV
           MEM(J,K,L) = 0
           REPEAT(J,K,L) = 0
2           CONTINUE

C****Initialize SCHEMA array-make worst, so better ones fill up
C*** Only on the first iteration through foraging


           DO 8 K=1,NUMBST
                SCHEMA(K,1) = 100000
 8         CONTINUE

      ENDIF

      DO 21 N=1,NRELV+NDISV
           DO 21 K=1,NVALS(N)
           DFLAG(N,K) = 0
           DFLAG2(N,K) = 0
 21   CONTINUE


****** Call search routine


        CALL LOCL (NDESV,NRELV,NDISV,NVALS,STARTOBJ,STARTCON,
     $         RESTART, DESVAR, XX, BESTINDX, INDEX, MAXIT,
     $        MEM, DFLAG, DFLAG2, TEMPX, BESTX, BESTOBJ,
     $        MEMPLC, NUMBST, SCHEMA, REPEAT, DFNCOF,
     $        NMPRI, NNLTOT, NNLCON, NNLGOA, IPATH, FEASFL,
     $        MEMLIM, NNINDX, NSIZE, IACTVR)

      WRITE(14,*)'The ',NUMBST,' best solutions were found to
C               be:'

      DO 9 J=1,NUMBST
           WRITE(14,*)(SCHEMA(J,K),K=1,NDESV+1)
 9    CONTINUE
           WRITE(12,*) 'final objective is ', bestobj
        WRITE(12,*) 'at the point ',(BESTX(M),M=1,NDESV)

      PRINT *,'ALLBEST, BESTOBJ = ', ALLBEST, BESTOBJ

      IF (BESTOBJ.LE.ALLBEST) THEN
           ALLBEST = BESTOBJ
```

```
      DO 2703 M=1,NDESV
           ABESTX(M) = BESTX(M)
2703       CONTINUE


      DO 2704 M=1,NRELV+NDISV
           FIINDX(M)=NNINDX(M)
2704       CONTINUE
      ENDIF


2700    CONTINUE


      DO 500 J=1,NDESV
        DESVAR(J) = ABESTX(J)
        WRITE(12,*)'FOR ITERATION',NN,'BEST vars = ',J,DESVAR(J)
500   CONTINUE

      DO 501 J=1, NRELV+NDISV
           INDEX(J) = FIINDX(J)
501   CONTINUE


      CLOSE(UNIT=12)
      CLOSE(UNIT=13)
      CLOSE(UNIT=14)
      CLOSE(UNIT=15)
      CLOSE(UNIT=17)
999     RETURN
        END

C+
C*************************************************************
C
C Subroutine LOCL
C
C Purpose: Solve the local neighborhood problem
C
C-------------------------------------------------------------
--------
C Arguments     Name Type Description
C ---------     ---- ---- -----------
C Input:   NROW       I   Number of design variables
C          NRELV      I   Number of Real Variables
C          NDISV      I   Number of discrete variables
C          NVALS      I   Number of possible discrete values
C          OLDZ       R   The previous objective function value
C          OLDC       R   The previous constraint violation
C                         value
C          RESTART    I   Number of re-starts of foraging
```

```
C           DESVAR      R   Design variable vector
C           XX          I   Vector of possible discrete values
C           BESTINDX    I   Index of discrete variable which
C                           results in most improvement in the
C                           deviation func.
C           INDEX       I   Index of current variable
C           MAXIT       I   Maximum number of neighborhood C
                            searches
C           MEM         I   Dynamic memory vector
C           DFLAG       I   Diversification flag
C           DFLAG2      I   Diversification flag
C           TEMPX       R   Temprorary vector of design variables
C           BESTX       R   Best design variable vector
C           BSTZ        R   Best observed objective function
C           MEMPLC      I   Memory index, keep track of visited
C                           sites
C           NUMBST      I   Number of best solutions to keep in
C                           schema
C           SCHEMA      R   Nector of NUMBST solutions to
identify
C                           frequent characteristics of solution
C           REPEAT      I   Ensures solution is not repeated
C           DFNCOF      R   Vector of deviation functions weights
C           NMPRI       I   Number of priority levels
C           NNLTOT      I   Total number of nonlinear c/g's
C           NNLCON      I   Number of nonlinear constraints
C           NNLGOA      I   Number of nonlinear goals
C           IPATH       I
C           FEASFL      I   Flag of feasibility
C           MEMLIM      I   Memory size, initially
C           NNINDX      I   Number of re-starts
C           NSIZE       I   Neighborhood size
C           IACTVR      I   Inactive variable vector
C           ITBST       I
C           STCNT       I
C           ASPCNT      I   Number of times aspiration criteria
is
C                           met
C           BESTPOS     I   Index of best variable to change
C           SCHFLG      I   Flag for schema
C           DIVDUM      I   Diversification flag to find new
value
C
c
C Output:
C
C Input/Output
C
C-------------------------------------------------------------
C Common Blocks: none
C
```

```
C Include Files: alplim.cmm
C
C Calls to: OBJEC
C-----------------------------------------------------------------
C Development History
C
C Author: Kemper Lewis
C Date:   October 25, 1995
C
C Modifications:
C
C*****************************************************************
C
   SUBROUTINE LOCL (NROW, NRELV, NDISV, NVALS, OLDZ,OLDC,
     $          RESTART, DESVAR, XX, BESTINDX, INDEX, MAXIT,
     $          MEM, DFLAG, DFLAG2, TEMPX, BESTX, BSTZ,
     $          MEMPLC, NUMBST, SCHEMA, REPEAT, DFNCOF,
     $          NMPRI, NNLTOT, NNLCON,NNLGOA,IPATH,FEASFL,
     $          NMEMLIM, NINDX,NSIZE,IACTVR)

     INCLUDE 'alplim.cmm'


      INTEGER NROW, NVALS(MDESV), IACTVR(MDESV),
     $  MEM(MDESV,MDSCV,MDSCV),
     $     INDEX(MDESV), NNINDX(MDESV), BESTINDX, NUMBST, NSIZE

      INTEGER RESTART, MEMPLC(MDESV), DFLAG(MDESV,MDSCV),
     $     DFLAG2(MDESV,MDSCV),
     $     REPEAT(MDESV,MDSCV,MDSCV), NRELV, NDISV, NMPRI,
     $  NNLGOA, NNLCON, NNLTOT

      REAL DFNCOF(MLEVEL,MDEVV)

      REAL CONVIO, DEVFUN, OLDZ, OLDC,
     $          DESVAR(MDESV), TEMPX(MDESV), XX(MDESV,MDSCV)

      REAL BSTZ, BESTX(MDESV), SCHEMA(NUMBST,MDESV+1)

      INTEGER ITBST, STCNT, ASPCNT, BESTPOS, I, J, K, L, M, N,
     $     STFLAG, LTFLAG, MEMFLAG, SCHFLG, MAXIT, IPATH,
     $     FEASFL, MEMLIM, LISTL, RESTART, DIVDUM

      REAL NEWZ, DE, DZBEST, CURBST, EPS

        ITBST=0

     IF (RESTART.EQ.1) THEN
     IF (OLDC.GT.0.005) THEN
               BSTZ = 1000*OLDC
               CURBST = 1000*OLDC
```

```
            ELSE
                  BSTZ=OLDZ
                  CURBST=OLDZ
            ENDIF
      ENDIF

         STCNT=0
         ASPCNT=0
         EPS = 0.001

      LISTL = MEMLIM

      DO 100 L=1,MAXIT

            DZBEST = 10000000000.0

      FEASFL = 1

C*********************************************************
C                                                        *
C     DIVERSIFICATION SCHEME                             *
C                                                        *
C     Purpose:Identify parts of solution that            *
C     frequently show up and penalize them accordingly.*
C     This uses DFLAG                                    *
C     Diversification flag.                              *
C                                                        *
C*********************************************************

      DO 31 I = 1,NRELV+NDISV
        IF (IACTVR(I).EQ.0) THEN
           GOTO 31
        ENDIF
        DFLAG(I,INDEX(I)) = DFLAG(I,INDEX(I))+1
        IF(MOD(DFLAG(I,INDEX(I)),20).EQ.0)    THEN
           IF (MOD(NVALS(I),2).EQ.0) THEN
                DIVDUM = NVALS(I)/2
           ELSE
                DIVDUM = (NVALS(I)+1)/2
           ENDIF
                IF (INDEX(I) .LE. DIVDUM) THEN
                     INDEX(I) = NVALS(I)
                ELSE
                     INDEX(I) = 1
                ENDIF
                DESVAR(I) = XX(I,INDEX(I))
                DFLAG2(I,INDEX(I)) = 1
           ENDIF
 31   CONTINUE

            DO 24 J=1,NRELV+NDISV
```

```
                DO 26 K=1,NVALS(J)

C     This is a neighborhood of X moves in any direction

      IF (ABS(INDEX(J) - K).LT.NSIZE .AND. K.NE.INDEX(J)) THEN

      STFLAG = 0
      LTFLAG = 0
      MEMFLAG = 0

C**************Temporarily swap DESVAR with XX(j,K)
C**************Compute change in objective function

                    DO 5 I=1,NRELV+NDISV
                         TEMPX(I) = DESVAR(I)
 5                   CONTINUE
                    TEMPX(J) = XX(J,K)

      WRITE(12,*)'current design point = ',(TEMPX(I),I=1,NROW)

           CALL OBJEC (CONVIO, DEVFUN, IPATH, NROW,
     $          TEMPX, DFNCOF, NMPRI, NNLTOT, NNLCON, NNLGOA)

          IF (CONVIO.GT.0.05) THEN
               NEWZ = 1000*CONVIO
          WRITE(12,*)'current obj function (INFEAS)= ',NEWZ
          ELSE
               FEASFL = 0
               NEWZ = DEVFUN
          WRITE(12,*)'current objective function = ',NEWZ
          ENDIF


C*************************************************************
C                                                            *
C     SCHEMA INDENTIFICATION SCHEME                          *
C                                                            *
C     Purpose:Identify best set of solutions, then look to   *
C     identify parts of soluton that frequently show up.     *
C     Build future solutions on these building blocks.       *
C                                                            *
C*************************************************************

      SCHFLG = 0
      DO 12 N=1,NUMBST
      IF (SCHFLG.EQ.0) THEN
      IF (NEWZ.LT.SCHEMA(N,1)) THEN
           DO 13 M=1,N-1
           IF (N.GT.1 .AND. NEWZ.EQ.SCHEMA(M,1)) THEN
                GOTO 12
```

```
            ENDIF
 13         CONTINUE
            SCHFLG = 1
            DO 6 I = NUMBST,N+1,-1
            DO 6 M = 1,NROW+1
                 SCHEMA(I,M) = SCHEMA(I-1,M)
 6          CONTINUE

C************* update best objective function
            SCHEMA(N,1) = NEWZ

C************* update best design point
            DO 7 M = 2,NROW+1
                 SCHEMA(N,M) = TEMPX(M-1)
 7          CONTINUE
      ENDIF
      ENDIF
 12   CONTINUE

                    DE=NEWZ-OLDZ

C**************Check to see if it is the best so far
C*********NOTE: For minimization, it is .LT.
C*********       For maximization, it is .GT.

                    IF (DE.LT.DZBEST) THEN

c************** First condition tests for tabu status of (I,K)
C**************Must change STM size according to NUMDESV

      WRITE(12,*)'MEM(',J,INDEX(J),K,
      $   ')=',MEM(J,INDEX(J),K)
        IF ( MEM(J,INDEX(J),K) .GT. 0 .AND.
      $     MEM(J,INDEX(J),K) + LISTL .GT. L) THEN
            MEMFLAG = 1
            WRITE(12,*)'***MEM FLAG'
      ENDIF

      IF(MEMFLAG .EQ. 1) THEN

c*********************** IS .lt. if minimization ********
c*********************** IS .gt. if maximization ********
                         IF (NEWZ .LT. BSTZ) THEN
                     WRITE(12,*)'***ASPIRATION CRITERIA MET***'
                                ASPCNT = ASPCNT+1
                                CURBST = NEWZ
                                DZBEST = DE
                                BESTPOS = J
                                BESTINDX = K
                            ELSE
                     WRITE(12,*)'******TABU STATUS MET****'
```

```
                                    STCNT = STCNT + 1
                                ENDIF
                            ELSE

C***************** otherwise the move is not tabu

                                DZBEST = DE
                                CURBST = NEWZ
                                BESTPOS = J
                                BESTINDX = K
                    ENDIF
                    ENDIF
                   ENDIF

26               CONTINUE
24             CONTINUE

************************************************************
*     DYNAMIC MEMORY STRUCTURE.                            *
*     AS L increases, memory list size is going down.      *
************************************************************

      IF (L.GT.MEMLIM .AND. MOD(L,2).EQ.0) THEN
           LISTL = LISTL - 1
      ENDIF


      MEM( BESTPOS,INDEX(BESTPOS),BESTINDX) = L
      WRITE(12,*)'MEM(',BESTPOS,INDEX(BESTPOS),BESTINDX,') = ',
      $MEM(BESTPOS,INDEX(BESTPOS),BESTINDX)

      INDEX(BESTPOS) = BESTINDX

      WRITE(13,*)'ITERATION:',L

      DESVAR(BESTPOS) = XX(BESTPOS,BESTINDX)

      DO 27 J=1,NROW
         WRITE(13,*)'DESVAR(',J,') = ',DESVAR(J)
 27   CONTINUE


******This is postprocessing information for solution behavior

      CALL OBJEC (CONVIO, DEVFUN, IPATH, NROW,
      $           DESVAR, DFNCOF, NMPRI, NNLTOT, NNLCON, NNLGOA)

************************************************************

      WRITE(15,28)(DESVAR(J),J=1,NROW)
      WRITE(17,29)DEVFUN, CONVIO, BSTZ, STCNT, ASPCNT
```

```
 28   FORMAT(10(1X,F8.3))
 29   FORMAT(F12.3,1X,F12.3,1X,F12.3,1X,I4,1X,I4)

      IF (FEASFL.EQ.1) THEN
           WRITE(13,*)'**INFEASIBLE**(OBJ = 100*CONVIO)'
      ENDIF

      WRITE(13,*)'OBJECTIVE FUNCTION: ',CURBST


C******************Update BSTZ values

            OLDZ = CURBST
C********** NOTE: .lt. if minimization, .gt. if maximization
            IF ( CURBST .LT. BSTZ ) THEN
           WRITE(12,*)'***BESTZ being updated to :',CURBST
                  BSTZ = CURBST
                  ITBST = L
                  DO 19 M=1,NROW
                         BESTX(M) = DESVAR(M)
 19                CONTINUE

           DO 18 M=1,NRELV+NDISV
                NNINDX(M) = INDEX(M)
 18        CONTINUE
          ENDIF


     WRITE(12,*)'BESTX = ',(BESTX(I), I=1,NROW)
     WRITE(12,*) L,CURBST,BSTZ,STCNT,ASPCNT

100   CONTINUE


      WRITE(12,*) 'BEST Z ',ITBST,BSTZ
      WRITE(12,*) 'Tabu Move found ', STCNT
      WRITE(12,*) 'Aspiration criteria met ',ASPCNT
      WRITE(12,*) 'BEST variable values '
      WRITE(12,*) (BESTX(I),I=1,NROW)
      WRITE(12,*) '  '

      RETURN
      END
C
C
C*************************************************************
C*************************************************************
C
C Function OBJEC
C
C Purpose: Evaluate the deviation function and constraints
```

```
C
C-------------------------------------------------------------
C Arguments     Name Type Description
C ---------     ---- ---- -----------
C
C Input:
C          DESVAR     R   Vector of Design Variables
C          NDESV      I   Number of Design Variables
C          IPATH      I
C          NNLTOT     I   Total number of nonlinear c/g's
C          NNLCON     I   Number of nonlinear constraints
C          NNLGOA     I   Number of nonlinear goals
C          DFNCOF     R   Vector of deviation function weights
C          NMPRI      I   Number of Priority Levels
C          IACTVR     I   Array of inactive variables
C          CONVIO     R   Current constraint violation
C          DEVFUN     R   Current deviation function
C          NMPRI      I   Number of priorities
C
C
C
C Output:
C
C Input/Output
C
C-------------------------------------------------------------
C Common Blocks: none
C
C Include Files: alplim.cmm
C
C Calls to: USRSET
C-------------------------------------------------------------
C Development History
C
C Author: Kemper Lewis
C Date:   October 25, 1995
C
C Modifications:
C
C*************************************************************

      SUBROUTINE OBJEC(CONVIO, DEVFUN, IPATH, NDESV, DESVAR,
     $                 DFNCOF, NMPRI, NNLTOT, NNLCON, NNLGOA)

      INCLUDE 'alplim.cmm'
C
C-----------------------------------
C     Arguments:
C-----------------------------------
C
      INTEGER NDESV, NNLTOT, IPATH, NMPRI
```

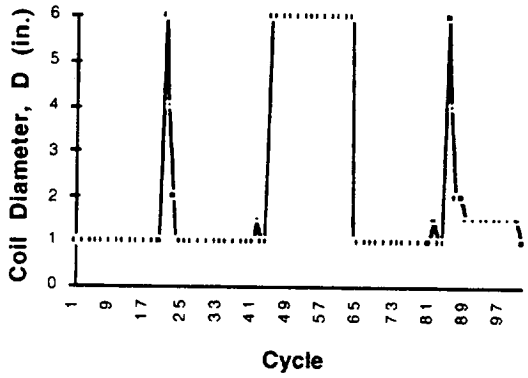
```
      INTEGER I, J, K, NOUT, NNLCON, NNLGOA

      REAL DESVAR(NDESV), CONSTR(MNLNCG), GOALS(MNLNCG),
C          GOALSUM

      REAL DFNCOF(MLEVEL,MDEVV), CONVIO, DEVFUN

c*************compute objective function based on priority

      CALL USRSET (IPATH, NDESV, MNLNCG, NOUT, DESVAR,
     &              CONSTR, GOALS)

      CONVIO = 0

      DO 56 I=1,NNLCON
           IF(CONSTR(I) .LT. 0.0) THEN
                CONVIO = CONVIO + ABS(CONSTR(I))
           ENDIF
 56   CONTINUE


      DEVFUN = 0.0

C     *** Only the first priority is used.  This is because,
C     since FALP is a two-stage solution process, the foraging
C     (discrete) portion acts as a meta-heuristic to find
C     the best neighborhood.  Then within this neighborhood,
C     the ALP portion refines the solution using 1) the
C     continuous variables and 2) multiple priority levels,
C     if applicable.

      DO 53 J=1,NNLGOA
        DO 53 K=1,2
           DEVFUN = DEVFUN + DFNCOF(1,2*J-1+K-1)*ABS(GOALS(J))
 53   CONTINUE

      DEVFUN = DEVFUN/2

 55    RETURN
       END
```

SPRING EXAMPLE: DSIDES Data File

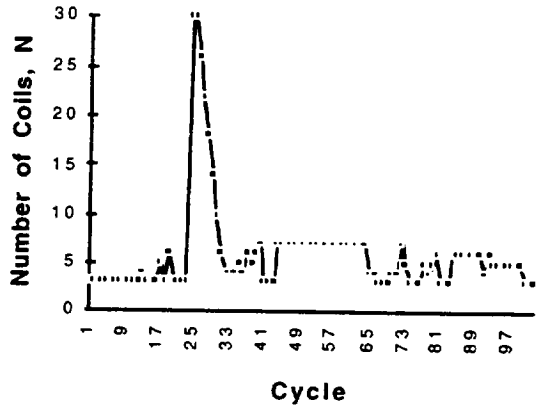
```
PTITLE: Problem Title: Design of a Compression Spring

NUMSYS:
  1  2  0: 1 continuous, 2 discrete, 0 Boolean variables

SYSVAR:
D 1 1.0 6.0  6.0:          Coil Diameter, continuous
N 2 3  30 30:              Number of Coils, integer
d 3 0.009  0.5 0.009:      Wire Diameter, discrete

NUMCAG:
0  8  0  0  1:  8 nonlinear constraints, one nonlinear goal

DEVFUN:
1:                1 level
1  2:             Level 1, 2 terms
(+1,1.0) (-1,1.0)

STOPCR:
1  0  40  0.001  0.001:

NLINCO: Names of nonlinear constraints
g1 1: shear stress
g2 2: free length
g3 3: wire diameter minimum
g4 4: outside diameter maximum
g5 5: inner coil ratio
g6 6: preload deflection
g7 7: combined deflections
g8 8: max deflection

NLINGO:
vol  1: volume goal

ALPOUT:
1  1  1  0  0  0  0  0  1  1:

PVDISC:
1     : variables with discrete values
3 42 1 : variable 3 has 42 possible values, initial value = 1st
0.009 0.0095 0.0104 0.0118 0.0128 0.0132 0.014 0.015 0.0162 0.0173
0.018 0.020 0.023 0.025 0.028 0.032 0.035 0.041 0.047 0.054 0.063
0.072 0.080 0.092 0.105 0.120 0.135 0.148 0.162 0.177 0.192 0.207
0.225 0.244 0.263 0.283 0.307 0.331 0.362 0.394 0.4375 0.500

ENDPRB:
```

SPRING EXAMPLE:Solution Search History

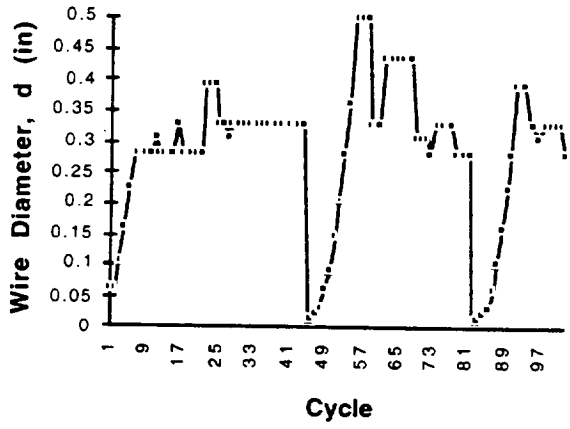
Starting Point: Lower Bound



(a) Coil Diameter



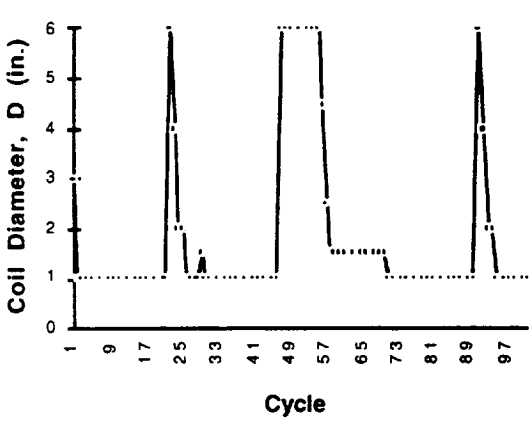
(b) Number of Coils



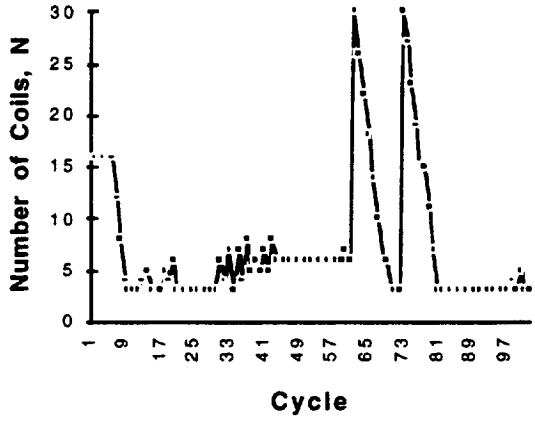
(c) Wire Diameter

Figure B.1. Design Variable History: Lower Bound Starting Point

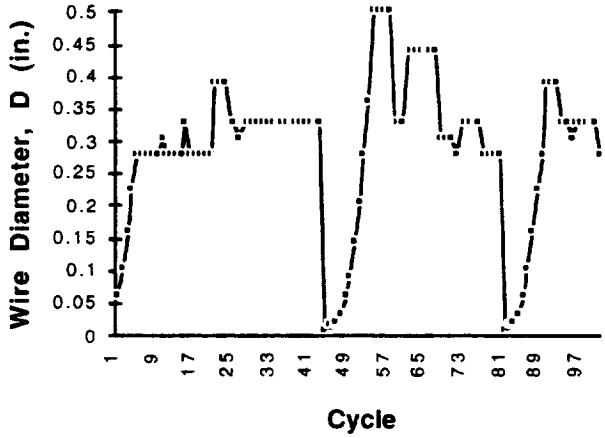
Starting Point: Mid-Points



(a) Coil Diameter



(b) Number of Coils



(c) Wire Diameter

Figure B.2. Design Variable History: Middle Point Starting Point

Starting Point: Upper Bound

GIVEN IN SECTION 6.4.2

PRESSURE VESSEL EXAMPLE: DSIDES Data File

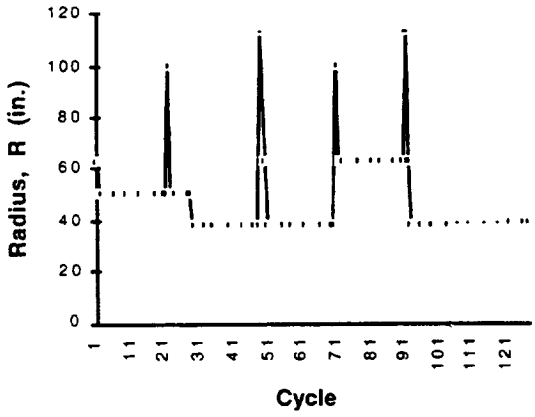
```
PTITLE: Problem Title
Press Vl problem from "A Two-Stage Sequential Approx Method for
Non-Linear Disc-Var Opt" from DAC, Bost,MA,by Hsu, pp.197-202.

NUMSYS:
  2  2  0: 2 continuous , 2 variables, 0 Boolean variables

SYSVAR:
radius 1 25  150  87.5:1       Radius, Continuous
length 2 25  250  137.5:       Length, Continuous
sthick 3 0.0625  1.25 0.0625:  Shell Thickness, Discrete
hthick 4 0.0625  1.25 0.0625:  Hull Thickness, Discrete

NUMCAG:
0  4  0  0  1: 4 nonlinear constraints, 1 nonlinear goal

DEVFUN:
1:         1 level
1  1:      Level 1, 1 term
(+1,1.0)

STOPCR:
1  0  40  0.001  0.001:

NLINCO: Names of nonlinear constraints
g1 1: Ts ratio geometry
g2 2: Th ratio geometry
g3 3: geometry limit
g4 4: space limitation

NLINGO:
cost  1: cost function

ALPOUT:
1  1  1  0  0  0  0  0  1  1:

PVDISC:
2     : variables with discrete values
3 20 10    : variable 3 has 20 possible values, guess is 10th
0.0625 0.125 0.1875 0.25 0.3125 0.375 0.4375 0.5 0.5625 0.625
0.6875 0.75 0.8125 0.875 0.9375 1.0 1.0625 1.125 1.1875 1.25

4 20 10    : variable 4 has 20 possible values, guess is 10th
0.0625 0.125 0.1875 0.25 0.3125 0.375 0.4375 0.5 0.5625 0.625
0.6875 0.75 0.8125 0.875 0.9375 1.0 1.0625 1.125 1.1875 1.25

ENDPRB:
```

PRESSURE VESSEL EXAMPLE: Solution Search History

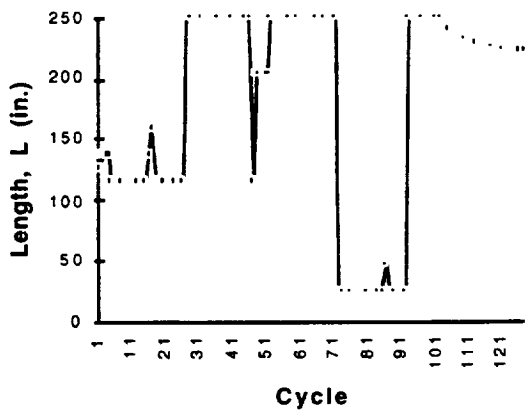
Starting Point: Lower bound

GIVEN IN SECTION 6.4.4

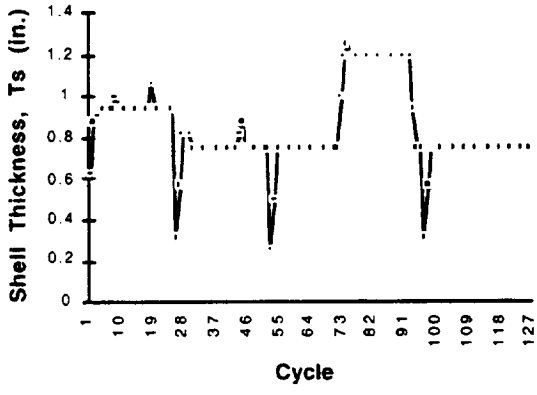
Starting Point: Mid-Points



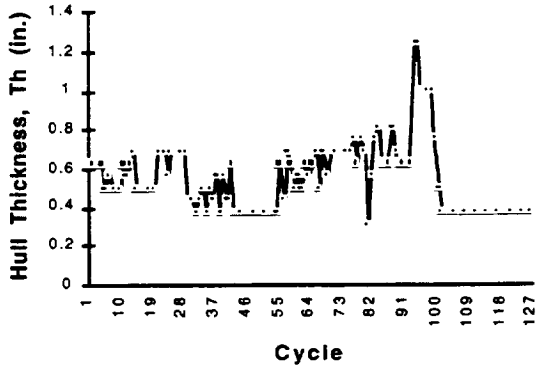
(a) Radius



(b) Length



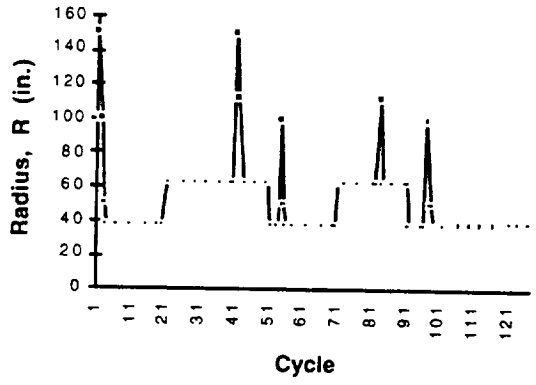
(c) Shell Thickness



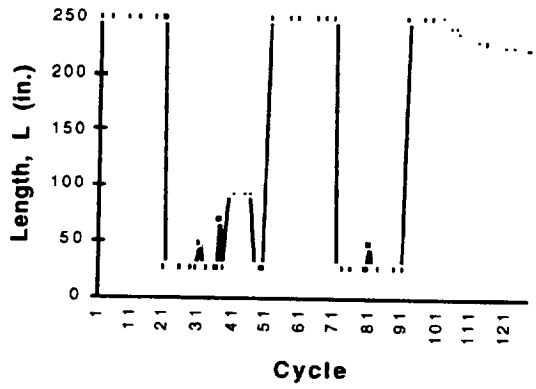
(d) Hull Thickness

Figure B.3. Design Variable History: Mid-Point Starting Point

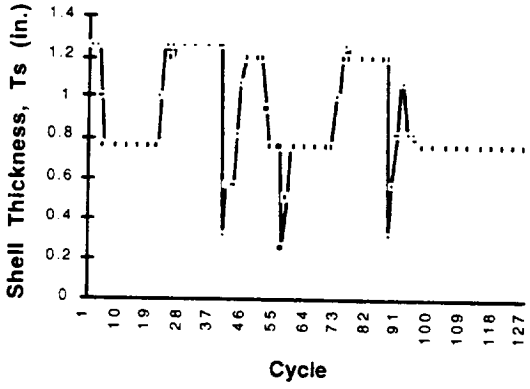
Starting Point: Upper Bound



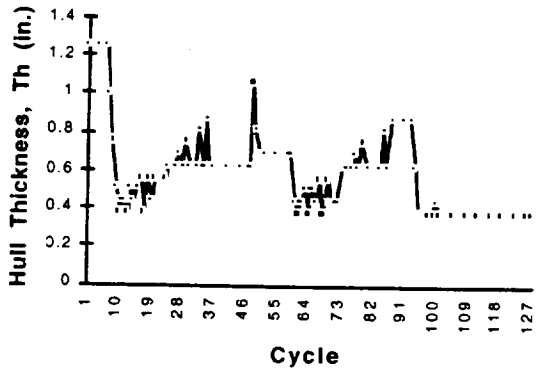
(a) Radius



(b) Length



(c) Shell Thickness



(d) Hull Thickness

Figure B.4. Design Variable History: Upper Bound Starting Point

APPENDIX C

FULL RESULTS: SUBSONIC PASSENGER AIRCRAFT STUDY

In this Appendix, the full solutions of each of the protocols and formulations presented in Chapter 7 are presented. This includes the DSIDES data files, and full solution information for the full cooperation (continuous and mixed), approximate cooperation, and leader/follower protocols. For the noncooperative formulation, full solutions and corresponding simplified rational reaction sets are given for each of the seven scenarios.

COOPERATIVE FORMULATIONS: FULL SOLUTIONS

Full Cooperative Solutions

In Table C.1, the full results for the three cooperative formulations are shown. The complete design variables, state variables, constraints, goals, deviation functions, and constraint violations are given.

Table C.1. Cooperative Solutions

(a) Full Cooperation: Continuous

Player Aero		Player Weights	
X, design vars		X, design vars	
S (ft^2)	1557	Ti (lbs)	33900
b (ft)	122.7	Wto (lbs)	196687
l (ft)	116.2		
s, state vars		s, state vars	
cdo_c	0.018	R_{fa}	0.286
cdo_{tl}	0.019	R_{fr}	0.286
d (ft)	14.64	U	0.490
Ld_l	15.1	R_f	0.999
Ld_t	11.8	PRI	177
Ld_c	19.9	q_L	0.095
V_{br} (ft/s)	688.74	q_{TO}	0.030
AR	9.66	s_L (ft)	4491
q_L	0.095	s_{TO} (ft)	6497
q_{TO}	0.030		
s_L (ft)	4491		
S_{TO} (ft)	6497		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	0.080	g_1	0.632
g_2	2.94	g_2	-0.001
g_3	0.114	g_3	2.945
g_4	0.002	g_4	0.114
g_5	0.001	g_5	0.002
g_6	0.073	g_6	0.001
g_7	0.091		
goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.683	f_1	-0.344
f_2	0.003	f_2	-0.021
f_3	-0.080	f_3	0.000
f_4	-0.002	f_4	0.683
f_5	0.444	f_5	0.003
		f_6	-0.002
		f_7	0.444
devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.242	Zweight	0.214
convio	0.0	convio	-0.001

(b) Approximate Cooperation

Player Aero		Player Weights	
X, design vars		X, design vars	
S (ft^2)	1554	Ti (lbs)	33903
b (ft)	122.4	Wto (lbs)	196512
l (ft)	119.1		
s, state vars		s, state vars	
cdo_c	0.018	R_{fa}	0.286
cdo_{tl}	0.019	R_{fr}	0.286
d (ft)	14.3	U	0.489
Ld_l	15.0	R_f	0.999
Ld_t	11.8	PRI	177
Ld_c	19.9	q_L	0.095
V_{br}	689.6	q_{TO}	0.030
AR	9.65	s_L	4498
q_L	0.095	s_{TO}	6500
q_{TO}	0.030		
s_L	4498		
S_{TO}	6500		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	0.081	g_1	0.632
g_2	2.943	g_2	-0.001
g_3	0.111	g_3	2.943
g_4	0.001	g_4	0.111
g_5	0.000	g_5	0.001
g_6	0.074	g_6	0.000
g_7	0.092		
goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.683	f_1	-0.343
f_2	0.000	f_2	-0.021
f_3	-0.081	f_3	-0.001
f_4	-0.001	f_4	0.683
f_5	0.445	f_5	0.000
		f_6	-0.001
		f_7	0.445
devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.2420	Zweight	0.213
convio	0.0	convio	-0.001

(c) Full Cooperation: Mixed

Player Aero		Player Weights	
X, design vars		X, design vars	
S (ft^2)	1613	Ti (lbs)	33000
b (ft)	126	Wto (lbs)	197717
l (ft)	120		
s, state vars		s, state vars	
cdo_c	0.018	R_{fa}	0.289
cdo_{tl}	0.018	R_{fr}	0.286
d (ft)	14.23	U	0.489
Ld_l	15.7	R_f	1.01
Ld_t	12.3	PRI	174
Ld_c	20.2	q_L	0.092
V_{br} (ft/s)	677.26	q_{TO}	0.030
AR	9.84	s_L (ft)	4369
q_L	0.092	s_{TO} (ft)	6491
q_{TO}	0.030		
s_L (ft)	4369		
S_{TO} (ft)	6499		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	0.063	g_1	0.629
g_2	2.836	g_2	0.010
g_3	0.112	g_3	2.836
g_4	0.029	g_4	0.112
g_5	0.000	g_5	0.029
g_6	0.087	g_6	0.000
g_7	0.103		
goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.674	f_1	-0.357
f_2	0.001	f_2	-0.023
f_3	-0.0628	f_3	0.010
f_4	-0.0292	f_4	0.674
f_5	0.4443	f_5	0.000
		f_6	-0.029
		f_7	0.444
devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.242	Zweight	0.220
convio	0.0	convio	0.0

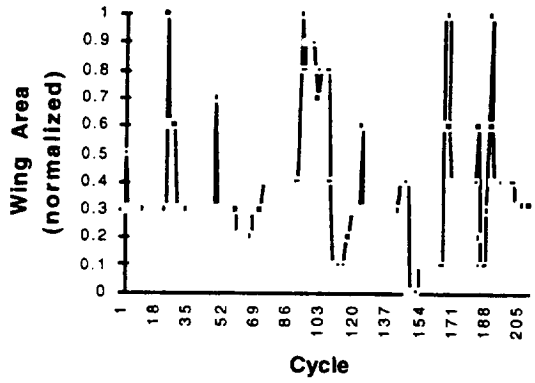
DSIDES DATA FILE: Full Cooperation, Mixed Discrete/Continuous

```
PTITLE: Problem Title, User Name and Date
        Aircraft Design, Full Cooperation: Mixed
        Kemper Lewis, October 19.1995

NUMSYS  : Number of system variables: real, discrete, boolean
    2  3  0

SYSVAR   : System variable information
winga 1  0  1  1.0     : Wing area
weigh 2  0  1  1.0 : Take-off weight
fleng 3  0  1  1.0        : fuselage length
wspan 4  0  1  1.0      : Wing Span
insth 5  0  1  1.0   : installed thrust

NUMCAG  : Number of constraints and goals
 0  9  0  0  8 :  nlinco,nnlinq,nnlequ,nlingo,nnlgoa

DEVFUN  : Achievement function
   1    : levels
   1 16   : level 1, 2 terms
  (-1,1.0) (+1,1.0) (-2,1.0) (+2,1.0) (-3,1.0) (+3,1.0)
  (-4,1.0) (+4,1.0) (-5,1.0) (+5,1.0) (-6,1.0) (+6,1.0)
  (-7,1.0) (+7,1.0) (-8,1.0) (+8,1.0)


STOPCR  : Stopping criteria
  1  0  40  0.0005 0.0005  : perform calcs, prt reslts,
                              Mcyles,sta dev, sta var

NLINCO  : Names of nonlinear constraints
aspr   1: aspect ratio
accl   2: achievable climb gradient, landing
acto   3: achievable climb gradient, take-off
ldfl   4: landing field length
tofl   5: take-off field length
cdtl   6: limit on take-off and landing Cd
cdoc   7: limit on cruise Cd
usfl   8: Useful load fraction
fuel   9: fuel balance

NLINGO  : Names of the nonlinear goals
misl  1: missed approach landing
mist  2: missed approach take-off
apcr  3: aspect ratio
ldfl  4: landing field length
tofl  5: take-off field length
prod  6: Productivity Index
usfl  7: Useful load fraction
fuel  8: fuel balance
```

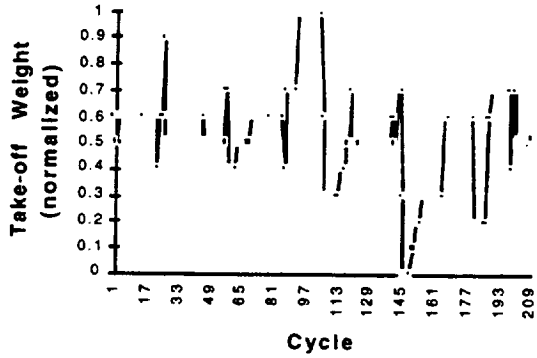
```
ALPOUT    : Output Control
     1   1   1   1  1   1   1  1  1  1

PVDISC:
3     : variables with discrete values
3 46 1     : variable 2 has 46 possible values, initial value =1
0.0 0.022 0.044 0.067 0.089 0.111 0.133 0.156 0.178 0.2
0.222 0.244 0.267 0.289 0.311 0.333 0.356 0.378 0.4 0.422 0.444
0.467 0.489 0.511 0.533 0.556 0.578 0.6 0.622 0.644 0.667 0.689
0.711 0.733 0.756 0.778 0.8 0.822 0.844 0.867 0.889 0.911 0.933
0.956 0.978 1.0
4 29 1 : variable 3 has 29 possible values, initial value =1
0.0 0.0182 0.0545 0.0909 0.1273 0.1636 0.2 0.2364 0.2727 0.3091
0.3455 0.3818 0.4182 0.4545 0.4909 0.5273 0.5636 0.6 0.6364
0.6727 0.7091 0.7455 0.7818 0.8182 0.8545 0.8909 0.9273 0.9636
1
5 19 1     : variable 2 has 19 possible values, initial value =1
0.0 0.0275 0.0826 0.1193 0.1927 0.2294 0.3028 0.3761 0.4495
0.4862 0.5229 0.5596 0.633 0.7064 0.7431 0.8165 0.8532 0.9266
1.0

ENDPRB:
```

SOLUTION HISTORY: Full Cooperation, Mixed Discrete/Continuous

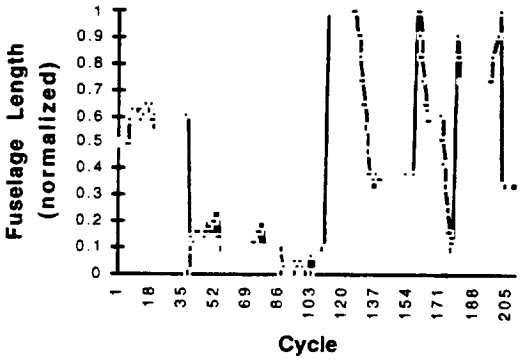
In Figure C.1, the design variable history for the full cooperative formulation is shown. In Figure C.2, the *best* deviation function encountered is shown as a function of the solution time. A large decrease in the deviation function is achieved in the first cycles, and in the later cycles, small increases (as shown in the expanded plot in Figure C.2) occur as the solution slowly gets better and better. In Figure C.3, the deviation function at each cycle is shown. Characteristic of the foraging search, worse solutions are accepted as a means to find better solutions and escape local minima. This behavior is illustrated in Figure C.3, where worse solutions are accepted throughout the solution process.



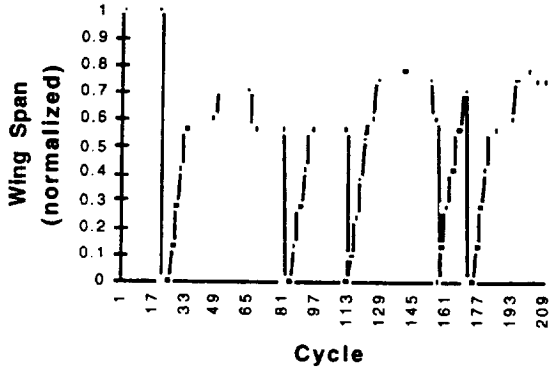
(a) Wing Area (normalized)



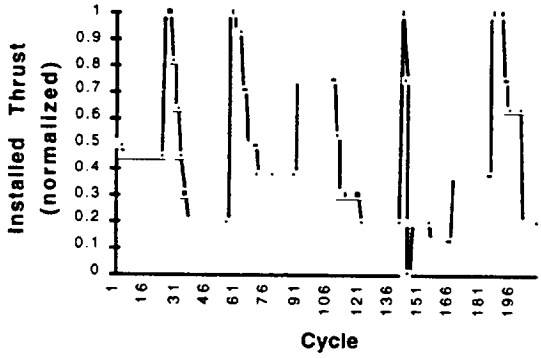
(b) Take-off Weight (normalized)



(c) Fuselage Length (normalized)



(d) Wing Span (normalized)



(e) Installed Thrust (normalized)

Figure C.1. Design Variable History: Full Cooperation (Mixed)

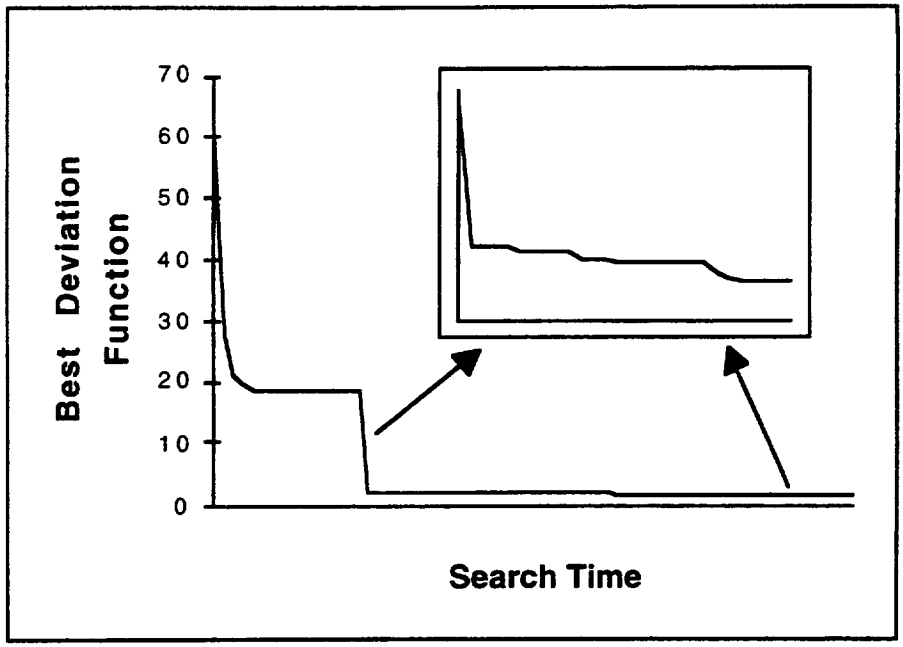


Figure C.2. Best Deviation Function

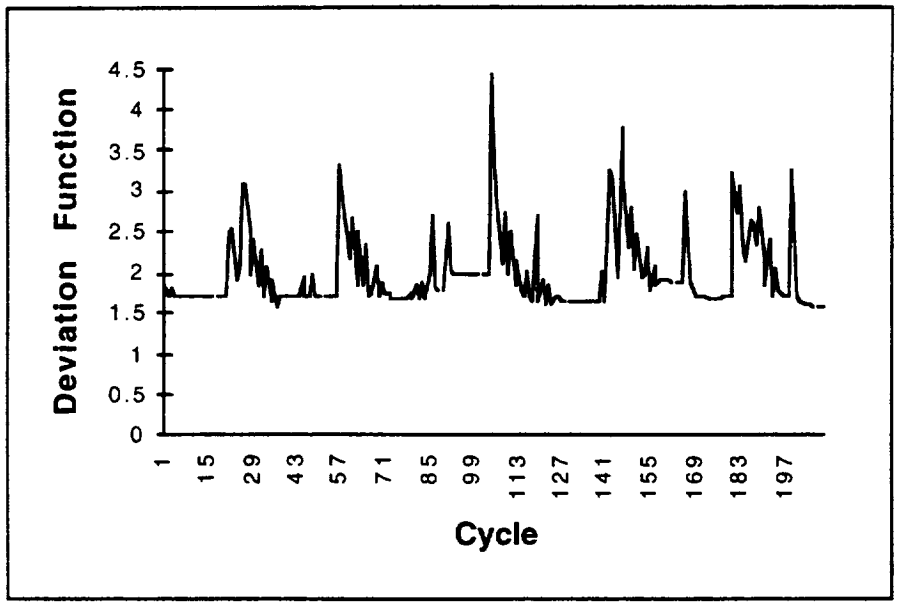


Figure C.3. Deviation Function

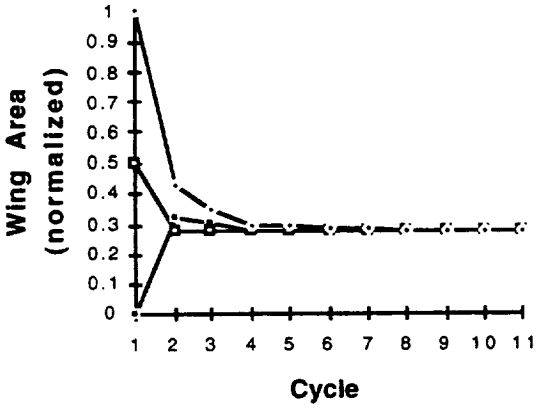
DSIDES DATA FILE: Full Cooperation, Continuous

```
PTITLE: Problem Title, User Name and Date
        Aircraft Design, Full Cooperative, Continuous
        Kemper Lewis, October 19.1995

NUMSYS  : Number of system variables: real,discrete, boolean
    5  0  0

SYSVAR   : System variable information
winga 1  0  1  1.0      : Wing area
weigh 2  0  1  1.0 : Take-off weight
fleng 3  0  1  1.0         : fuselage length
wspan 4  0  1  1.0       : Wing Span
insth 5  0  1  1.0  : installed thrust

NUMCAG  : Number of constraints and goals
 0  9  0  0  8 :  nlinco,nnlinq,nnlequ,nlingo,nnlgoa

DEVFUN  : Achievement function
   1    : levels
   1 16   : level 1, 2 terms
  (-1,1.0) (+1,1.0) (-2,1.0) (+2,1.0) (-3,1.0) (+3,1.0)
  (-4,1.0) (+4,1.0) (-5,1.0) (+5,1.0) (-6,1.0) (+6,1.0)
  (-7,1.0) (+7,1.0) (-8,1.0) (+8,1.0)


STOPCR  : Stopping criteria
  1  0  40  0.01 0.01  : perform calcs, prt reslts, Mcyles,sta
dev, sta var

NLINCO  : Names of nonlinear constraints
aspr   1: aspect ratio
accl   2: achievable climb gradient, landing
acto   3: achievable climb gradient, take-off
ldfl   4: landing field length
tofl   5: take-off field length
cdtl   6: limit on take-off and landing Cd
cdoc   7: limit on cruise Cd
usfl   8: Useful load fraction
fuel   9: fuel balance

NLINGO  : Names of the nonlinear goals
misl  1: missed approach landing
mist  2: missed approach take-off
apcr  3: aspect ratio
ldfl  4: landing field length
tofl  5: take-off field length
prod  6: Productivity Index
usfl  7: Useful load fraction
```

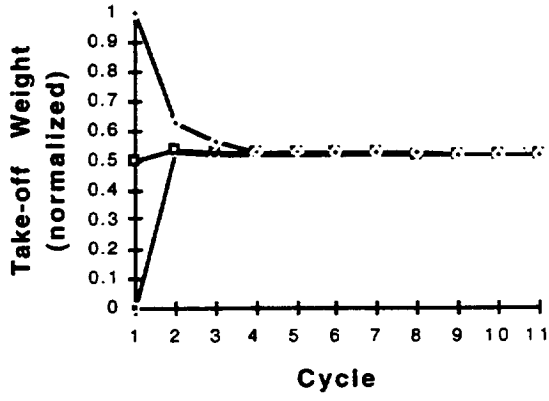
```
fuel  8: fuel balance

ALPOUT    : Output Control
     1    1    1    1  1    1    1  1  1  1


ADREMO:
 40 0.05

ENDPRB:
```

SOLUTION HISTORY: Full Cooperation, Continuous

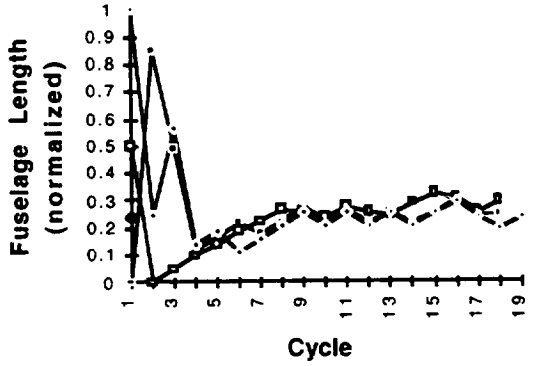
In Figure C.4, the design variable history for the full cooperative (continuous) formulation is given for three starting points. All three starting points converge to the same solution. In Figure C.5, the best deviation function and constraint violation are plotted. The best deviation function steadily decreases, and the constraint violation pregressively goes to zero, representing feasibility.



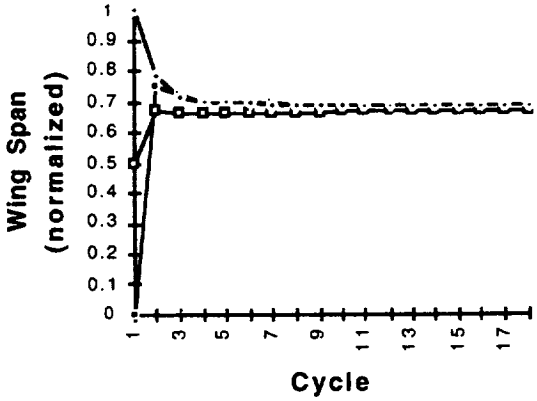
(a) Wing Area (normalized)



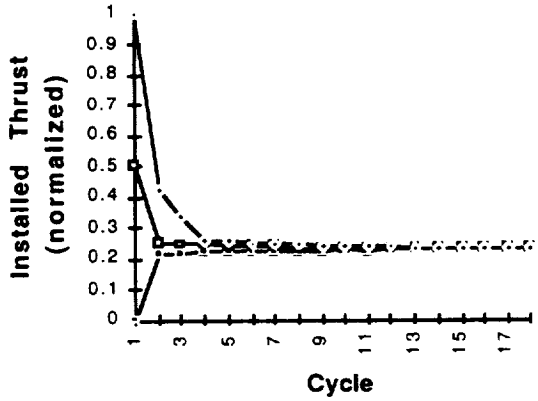
(b) Take-off Weight (normalized)



(c) Fuselage Length (normalized)



(d) Wing Span (normalized)



(e) Installed Thrust (normalized)

Figure C.4. Design Variable History: Full Cooperation (Continuous)

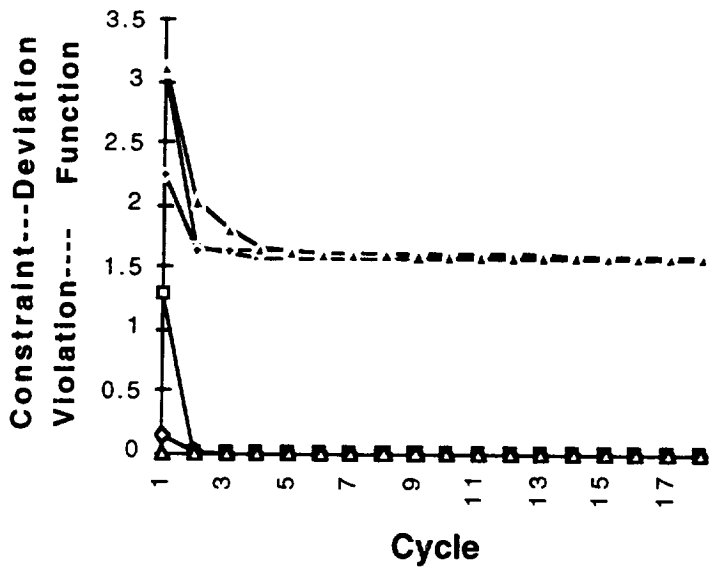


Figure C.5. Best Deviation Function and Constraint Violation: Full Cooperation (Continuous)

DSIDES DATA FILE: Approximate Cooperation

```
PTITLE: Problem Title, User Name and Date
        Aircraft Design, Approximate Cooperation
        Kemper Lewis, October 19.1995

NUMSYS  : Number of system variables: real,discrete, boolean
    5  0  0

SYSVAR   : System variable information
winga 1  0  1  0     : Wing area
weigh 2  0  1  0     : take-off weight
flgth 3  0  1  0     : fuselage length
wspan 4  0  1  0      : Wing Span
insth 5  0  1  0   : installed thrust

NUMCAG  : Number of constraints and goals
 0  12  0  0  11 :  nlinco,nnlinq,nnlequ,nlingo,nnlgoa

DEVFUN  : Achievement function
   1    : levels
   1 22   : level 1, 2 terms
  (-1,1.0) (+1,1.0) (-2,1.0) (+2,1.0) (-3,1.0) (+3,1.0)
  (-4,1.0) (+4,1.0) (-5,1.0) (+5,1.0) (-6,1.0) (+6,1.0)
  (-7,1.0) (+7,1.0) (-8,1.0) (+8,1.0) (-9,1.0) (+9,1.0)
  (-10,1.0) (+10,1.0) (-11,1.0) (+11,1.0)


STOPCR  : Stopping criteria
  1  0  40  0.01 0.01  : perform calcs, prt reslts, Mcyles,sta
dev, sta var

NLINCO  : Names of nonlinear constraints
aspa  1: aspect raito
qla   1: achievable climb gradient, landing, aero approx
qlw   2: achievable climb gradient, landing, weights approx
qto   3: achievable climb gradient, take-off
qtow   4: achievable climb gradient, take-off, weights apporx
ldfl   5: landing field length
ldfa   6: landing field length, aero approx
tofl   7: take-off field length
cdtl   8: limit on take-off and landing Cd
cdoc  9: limit on cruise Cd
usfl  10: useful load fraction
flbl  11: fuel balance

NLINGO  : Names of the nonlinear goals
qla   1: achievable climb gradient, landing, aero approx
qlw   2: achievable climb gradient, landing, weights approx
qto   3: achievable climb gradient, take-off
```

```
qtow   4: achievable climb gradient, take-off, weights approx
asra   5: aspect ratio
ldfl   6: landing field length
ldfa   7: landing field length, aero approx
tofl   8: take-off field length
prod  9: Productivity Index
usfl  10: Useful load fraction
fuel  11: fuel balance


ALPOUT    : Output Control
     1   1   1   1  1   1   1  1  1  1

OPTIMP   : Optimization parameters
   -0.05   0.5 0.005 : VIOLIM, REMO, STEP

ADREMO:
  40 0.005

ENDPRB:
```

NONLOCAL APPROXIMATION SCHEME (GSE's, Matrix Solver, and Taylor Series): Approximate Cooperation

```
C
      SUBROUTINE USRSET (IPATH, NDESV, MNLNCG, NOUT, DESVAR,
     &                   CONSTR, GOALS)
C
C-------------------------------------
C     Arguments:
C-------------------------------------
C
      INTEGER IPATH, NDESV, MNLNCG, NOUT
C
      REAL DESVAR(NDESV)
      REAL CONSTR(MNLNCG), GOALS(MNLNCG)
C
C-------------------------------------
C     Local variables:
C-------------------------------------
C
C     constants
      INTEGER Np,N,I,J, COUNT, K
      REAL bt,R,Wpay,Wfix,rhotl,mutl,Vtl,rhoc,muc,tc,clmax,pi

C     the design variables
      REAL b,l,S,Wto,Ti

C     the behaviour variables
      REAL cd0c,cd0tl,d,LDc,LDl,LDt,Vbr,Rfa,Rfr,Sto,U
      REAL ql,Rf,Sl,qto
      REAL cd0cap,cd0tlap,dap,LDcap,LDlap,LDtap,Vbrap,
     &            Rfaap,Rfrap,Uap
      REAL qla,qlw,Rfap,Sla,Slw,qtoa,qtow

C     others
      REAL H1, HH, DET pu,pd, Rfold, Rfrold,Rfaold,Uold,PRIold,
     &      Sold, bold, lold, Tiold, Wtoold
      REAL AR, ARap

C     Convergence check
      REAL EPS, cd0co, Vbro
      REAL PRI, PRIap, AACH, WACH

C     The Derivatives for the cooperative GSE formulation

      REAL dRfrdLDc, dRfrdVbr, dRfadTi, dRfadWto, dRfdRfr,
     -      dRfdrfa, dUdLDc, dUdVbr, dUdWto, dPRdVbr, dPRdRfr,
     -  dPRdRfa, dPRdWto

      REAL dcd0cdVbr, dLdtdd, dLdtdcd0tl, dLdcdd, dLdcdcd0c,
     -  dLdldd, dLdldcd0tl, dLdldRfr, dVbrdd, dVbrdcd0c, dqLdLDl,
     -  dqTOdLDt, dqLdRfr, dsLdRfr
```

```
      REAL dddl, dcd0tldb, dcd0tldS, dcd0tldl, dcd0cdS, dcd0cdb,
     -      dcd0cdl,dLdtdS, dLdtdb,dLdtdWto,dLdcdS, dLdcdb,dLdldS,
     -  dLdldb, dLdldWto, dVbrdS,dVbrdb, dVbrdWto, dARdS,dARdb

      REAL dqLdTi, dqLdWto, dqTOdTi, dqTOdWto, dsLdS, dsLdWto,
     -  dsTOdS, dsTOdTi, dsTOdWto

      REAL dold, cd0tlold, cd0cold, LDtold, LDcold,
     &      LDlold, Vbrold, ARold, qLold, qTOold,
     &      sLold, sTOold

C     The GSE Matrices GSEM*GSESOLV=GSERHS

      REAL X(5,2)

      INTEGER NUMSTAT, TOTDESV
      parameter (NUMSTAT=17)
      parameter (TOTDESV = 5)
      REAL GSEM(NUMSTAT,NUMSTAT), GSERHS(NUMSTAT,TOTDESV),
     &            GSESOLV(NUMSTAT,TOTDESV)

      HH(Np,1) = 1. + 4.325 * Np / 1
C
C*****************************************************************
C        EXPLANATION OF TERMS
C
C     AR    Aspect Ratio
C     b     wing span
C     bt    thrust-specific fuel consumption
C     cd0   zero lift drag coefficient
C     cd0c  zero lift drag coefficient, cruise
C     cd0tl zero lift drag coefficient, take-off and landing
C     clmax wing maximum lift coefficient
C     d     fuselage diameter
C     LD    lift-to-drag ratio
C     LDo   optimum lift-to-drag ratio
C     LDc   lift-to-drag ratio, cruise
C     LDl   lift-to-drag ratio, landing
C     LDt   lift-to-drag ratio, take-off
C     l     fuselage length
C     N     Number of Passengers
C     qOEI  acheievable climb gradient, one engine
C           inoperative
C     ql    achievable climb gradient, one engine
C           inoperative, missed approach condition
C     qto   dto., take-off condition
C     Rf    fuel weight balance
C     Rfa   available fuel weight ratio
C     Rfr   required fuel weight ratio
C     S     wing area
C     Sl    landing field length
C     Sto   take-off field length
C     Ti    (installed and required) thrust
C     U     useful laod fraction
C     V     velocity
```

```
C     Vbr   best range speed
C     W     aircraft weight
C     Wfix  fixed equipment weight
C     Wl    aircraft landing weight
C     Wpay  paylaod weight
C     Wto   aircraft take-off weight
C     EPS     epsilon value for convergence
C
C*****************************************************************
C
      muc = 0.000406
      clmax = 2.6
      N   = 3.0
      tc  = 0.12
      Np  = 188
      bt = 0.00019444
      R = 1.762e+7
      Wpay = 40000
      Wfix = 1100
      rhotl = 0.002378
      mutl = 0.000156
      Vtl = 220
      rhoc = 0.000737
      pi = 3.141592654
      Hl = 0.00136612
      COUNT = 40
      EPS = 0.001

C     This is the nonlocal information being read in from the
C     other players

      OPEN (1,FILE='aprxcoop.inout')

      READ(1,*)dold, cd0tlold, cd0cold, LDtold, LDcold,
     &      LDlold, Vbrold, ARold, qLold, qTOold,
     &      sLold, sTOold, Rfrold, Rfaold, Rfold,
     &      Uold, PRIold, Sold, lold,
     &      bold, Wtoold, Tiold

      CLOSE(1)


      X(1,1) = Sold
      X(2,1) = lold
      X(3,1) = bold
      X(4,1) = Wtoold
      X(5,1) = Tiold


      S   =  DESVAR(1)
      Wto =  DESVAR(2)
     l   =  DESVAR(3)
     b   =  DESVAR(4)
     Ti   =  DESVAR(5)
```

```
C     RESCALE THE DESIGN VARIABLES
C
      S = S*(2500-1200) + 1200
      l = l*(150-105) + 105
      b = b*(140-85) + 85
      Ti = Ti*(55000-27750) + 27750
      Wto = Wto*(250000-140000) + 140000

      X(1,2) = S
      X(2,2) = l
      X(3,2) = b
      X(4,2) = Wto
      X(5,2) = Ti
C
C*****************************************************************
C*** These are the derivative of player  WEIGHT for  GSE ***
C*****************************************************************

      DO 201 I=1,NUMSTAT
            DO 202 J=1,TOTDESV
                  GSERHS(I,J) = 0.0
                  GSESOLV(I,J) = 0.0
 202        CONTINUE
            DO 201 K=1,NUMSTAT
                  GSEM(I,K) = 0.0
                  IF (I.EQ.K) THEN
                        GSEM(I,K) = 1.0
                  ENDIF
 201  CONTINUE

C     OPEN (3,FILE='GSEa.out')

************** Rfr Approximation *****************

      dRfrdLdc = -1.045*bt*R/
     -     (EXP(bt*R/(LDcold*Vbrold))*LDcold**2.*Vbrold)
     -      *LDcold/Rfrold

      dRfrdVbr =-1.045*bt*R/(EXP(bt*R/(LDcold*Vbrold))*
     -      LDcold*Vbrold**2.)
     -    *Vbrold/Rfrold


************* Rfa Approximation ******************

      dRfadTi = -0.379278/(Ti**0.0019*Wto)* Ti/Rfaold


     dRfadWto = (0.38*Ti**0.9981/Wto**2 + 0.061197/Wto**1.0638 +
     -      Wfix/Wto**2 + Wpay/Wto**2.)* Wto/Rfaold

************ Rf Approximation ********************

      dRfdRfa = 1/Rfrold * Rfaold/Rfold
      dRfdRfr = -(Rfaold/Rfrold**2)* Rfrold/Rfold
```

```
************ U Apprximation **********************

      dUdLdc = -1.045*bt*R/(EXP(bt*R/(LDcold*Vbrold))
     -       *LDcold**2.*Vbrold)
     -       *LDcold/Uold

     dUdVbr = -1.045*bt*R/(EXP(bt*R/(LDcold*Vbrold))
     -       *LDcold*Vbrold**2.)
     -       *Vbrold/Uold

      dUdWto = -Wpay/Wto**2* Wto/Uold


*********** PRI Approximation ********************

      dPRdVbr = 40000/(-41100+(1-Rfaold+Rfrold)*Wto)*Vbrold/PRIold

      dPRdRfr =-40000*Vbrold*Wto/(-41100+(1-Rfaold+Rfrold)*Wto)**2*
     -       Rfrold/PRIold

      dPRdRfa = 40000*Vbrold*Wto/(-41100+(1-Rfaold+Rfrold)*Wto)**2*
     -       Rfaold/PRIold

      dPRdWto = -40000*(1-Rfaold+Rfrold)*Vbrold/
     -       (-41100+(1-Rfaold+Rfrold)*Wto)**2*Wto/PRIold

C*************************************************************
C*** These are the derivatives of player AERO for GSE     ***
C*************************************************************

********* d Approximation ***************

      dddl = -7.91475*Np/lold**2.*lold/dold

******** Cdotl Approximation ***********

      dcd0tldb = 22.2111*(1.+1.2*tc+100.*tc**4.)/
     - (bold*Log(Sold*Vtl/(bold*mutl))**3.58)*bold/cd0tlold

      dcd0tldS =
     -     -7.16109*lold*HH(Np,lold)*
     -    (1. + 367.709/(lold/HH(Np,lold))**3. +
     -      H1*lold/HH(Np,lold))*
     -    (1. - 3.66*HH(Np,lold)/lold)**0.666667*
     -    (1. + 3.3489*(HH(Np,lold)/lold)**2.)*pi/
     -   (Sold**2*Log(lold*Vtl/mutl)**2.58) -
     -  22.2111*(1. + 1.2*tc + 100.*tc**4.)/
     - (Sold*Log(Sold*Vtl/(bold*mutl))**3.58)*Sold/cd0tlold

      dcd0tldl = -18.4756*HH(Np,lold)*(1.+367.709/
     &    (lold/HH(Np,lold))**3.+ H1*lold/HH(Np,lold))*
     &      (1.- 3.66*HH(Np,lold)/lold)**
     &  0.666667*(1.+3.3489*(HH(Np,lold)/lold)**2.)*pi/(Sold*
     &      Log(lold*Vtl/mutl)**
```

```
     &     3.58) + 47.9635*lold*HH(Np,lold)*(HH(Np,lold)/lold)**1.*
     &       (1. + 367.709/
     & (lold/HH(Np,lold))**3. +  H1*lold/HH(Np,lold))*(-4.325*Np/
     & lold**3 - HH(Np,lold)/lold**2)*(1.-3.66*HH(Np,lold)/lold)**
     &       0.666667*pi/(Sold*
     &  Log(lold*Vtl/mutl)**2.58) + 4.77406*lold*HH(Np,lold)*
     &       (1. + 367.709/(lold/
     & HH(Np,lold))**3.+ H1*lold/HH(Np,lold))*(15.8295*Np/lold**3
     &     + 3.66*HH(Np,lold)/lold**2)*(1. + 3.3489*(HH(Np,lold)/
     &       lold)**2.)*pi/
     -   ((1. - 3.66*HH(Np,lold)/lold)**0.333333*Sold*
     &       Log(lold*Vtl/mutl)**2.58) -
     - 30.9717*Np*(1. + 367.709/(lold/HH(Np,lold))**3. + H1*lold/
     & HH(Np,lold))*(1.- 3.66*HH(Np,lold)/lold)**0.666667
      &      *(1.+3.3489*
     &   (HH(Np,lold)/lold)**2.)*pi/
     &   (lold*Sold*Log(lold*Vtl/mutl)**2.58)+
     &        7.16109*
     & HH(Np,lold)* (1.+367.709/(lold/HH(Np,lold))**3. + H1*lold/
     &   HH(Np,lold))*(1. - 3.66*HH(Np,lold)/lold)**0.666667*
     &       (1. + 3.3489*
     & (HH(Np,lold)/lold)**2.)*pi/(Sold*
     &   Log(lold*Vtl/mutl)**2.58) +
     &       7.16109*lold*
     & HH(Np,lold)*(1.-3.66*HH(Np,lold)/lold)**0.666667*(1.+3.3489*
     &   (HH(Np,lold)/lold)**2.)*
     & (0.00590847*Np/(lold*HH(Np,lold)**2)+H1/
     & HH(Np,lold)-1103.13*(4.325*Np/(lold*HH(Np,lold)**2) + 1./
     &       HH(Np,lold))/
     &   (lold/HH(Np,lold))**4.)*pi/(Sold*Log(lold*Vtl/mutl)**2.58)
     &        * lold/cd0tlold

******* Cdoc Approximation **************

      dcd0cdb = 22.2111*(1.+1.2*tc+100.*tc**4.)/
     - (bold*Log(Sold*Vbrold/(bold*muc))**3.58) * bold/cd0cold

        dcd0cdS =
     -      -7.16109*lold*HH(Np,lold)*
     -     (1. + 367.709/(lold/HH(Np,lold))**3. +
     -       H1*lold/HH(Np,lold))*
     -     (1. - 3.66*HH(Np,lold)/lold)**0.666667*
     -     (1. + 3.3489*(HH(Np,lold)/lold)**2.)*pi/
     -    (Sold**2*Log(lold*Vbrold/muc)**2.58) -
     -   22.2111*(1. + 1.2*tc + 100.*tc**4.)/
     -   (Sold*Log(Sold*Vbrold/(bold*muc))**3.58) * Sold/cd0cold

      WRITE(3,*)Np,lold,H1,pi,Sold,Vbrold,muc,cd0cold

      dcd0cdl = -18.4756*HH(Np,lold)*(1.+367.709/
     &   (lold/HH(Np,lold))**3.+ H1*lold/HH(Np,lold))*(1.- 3.66*
     &       HH(Np,lold)/lold)**
     &   0.666667*(1.+3.3489*(HH(Np,lold)/lold)**2.)*pi/(Sold*
     &       Log(lold*Vbrold/muc)**
     &    3.58) + 47.9635*lold*HH(Np,lold)*(HH(Np,lold)/lold)**1.*
```

```
     &       (1. + 367.709/
     &     (lold/HH(Np,lold))**3.+H1*lold/HH(Np,lold))*(-4.325*Np/
     &     lold**3 - HH(Np,lold)/lold**2)*(1.-3.66*HH(Np,lold)/
     &       lold)**0.666667*pi/(Sold*
     &   Log(lold*Vbrold/muc)**2.58) + 4.77406*lold*HH(Np,lold)*
     &       (1. + 367.709/(lold/
     & HH(Np,lold))**3.+H1*lold/HH(Np,lold))* (15.8295*Np/lold**3
     &     + 3.66*HH(Np,lold)/lold**2)*(1. + 3.3489*(HH(Np,lold)/
     &       lold)**2.)*pi/
     -    ((1. - 3.66*HH(Np,lold)/lold)**0.333333*Sold*
     &       Log(lold*Vbrold/muc)**2.58) -
     - 30.9717*Np*(1.+ 367.709/(lold/HH(Np,lold))**3. +  H1*lold/
     & HH(Np,lold))*(1.- 3.66*HH(Np,lold)/lold)**0.666667
      &   *(1.+3.3489*
     &   (HH(Np,lold)/lold)**2.)*pi/(lold*Sold*Log(lold*Vbrold/
     &       muc)**2.58)+7.16109*
     &   HH(Np,lold)*(1.+367.709/(lold/HH(Np,lold))**3. + H1*lold/
     &   HH(Np,lold))*(1. - 3.66*HH(Np,lold)/lold)**0.666667*
     &       (1. + 3.3489*
     & (HH(Np,lold)/lold)**2.)*pi/
      &    (Sold*Log(lold*Vbrold/muc)**2.58)+
     &       7.16109*lold*
     &   HH(Np,lold)*(1.-3.66*HH(Np,lold)/lold)**0.666667
      &   *(1.+3.3489*
     & (HH(Np,lold)/lold)**2.)*
      & (0.00590847*Np/(lold*HH(Np,lold)**2)+H1/
     & HH(Np,lold)- 1103.13*(4.325*Np/(lold*HH(Np,lold)**2) + 1./
     &       HH(Np,lold))/
     & (lold/HH(Np,lold))**4.)*pi/
      & (Sold*Log(lold*Vbrold/muc)**2.58)
     &        * lold/cd0cold

      dcd0cdVbr =
     -    -18.4756*lold*HH(Np,lold)*
     -    (1. + 367.709/(lold/HH(Np,lold))**3. +
     -     H1*lold/HH(Np,lold))*
     -    (1. - 3.66*HH(Np,lold)/lold)**0.666667*
     -    (1. + 3.3489*(HH(Np,lold)/lold)**2.)*pi/
     -   (Sold*Vbrold*Log(lold*Vbrold/muc)**3.58) -
     - 22.2111*(1. + 1.2*tc+100.*tc**4.)/
     - (Vbrold*Log(Sold*Vbrold/(bold*muc))**3.58) * Vbrold/cd0cold

******** LDt Approximation **************

      dLdtdcd0tl = -2.*Wto/(rhotl*Sold*Vtl**2.*(cd0tlold +
     -       4.16667*Wto**2./
     -     (bold**2.*(1.- dold**2./bold**2.)*pi*rhotl**2.*
     -       Sold*Vtl**4.))**2.)*
     -       cd0tlold/LDtold

        dLdtdd =
     -       -16.6667*dold*Wto**3./
     -   (bold**4.*(1.- dold**2./bold**2.)**2.*pi*rhotl**3.*
     -       Sold**2.*Vtl**6.*
     -     (cd0tlold + 4.16667*Wto**2./
```

```
     -     (bold**2.*(1.- dold**2./bold**2.)*pi*rhotl**2.*
     -       Sold*Vtl**4.))**2.)*
     -       dold/LDtold

      dLdtdb =
     -       -2.*Wto*(-8.33333*dold**2.*Wto**2./
     -       (bold**5.*(1.- dold**2./bold**2.)**2.*pi*rhotl**2.*
     -       Sold*Vtl**4.) -
     -      8.33333*Wto**2./
     -       (bold**3.*(1.- dold**2./bold**2.)*pi*rhotl**2.*
     -       Sold*Vtl**4.))/
     -   (rhotl*Sold*Vtl**2.*(cd0tlold +
     -        4.16667*Wto**2./
     -      (bold**2.*(1.- dold**2./bold**2.)*pi*rhotl**2.*
     -       Sold*Vtl**4.))**2.)*
     -       bold/LDtold

         dLdtdS =
     -       8.33333*Wto**3./
     -    (bold**2.*(1.- dold**2./bold**2.)*pi*rhotl**3.*Sold**3.*
     -      Vtl**6.*(cd0tlold + 4.16667*Wto**2./
     -       (bold**2.*(1.- dold**2./bold**2.)*pi*rhotl**2.*Sold*
     -     Vtl**4.))**2.)- 2.*Wto/
     -    (rhotl*Sold**2.*Vtl**2.*(cd0tlold +
     -        4.16667*Wto**2./
     -         (bold**2.*(1.- dold**2./bold**2.)*pi*rhotl**2.*Sold*
     -       Vtl**4.)))*
     -       Sold/LDtold

         dLdtdWto =
     -       -16.6667*Wto**2./
     -   (bold**2.*(1.-dold**2./bold**2.)*
      -    pi*rhotl**3.*Sold**2.*Vtl**6.*
     -      (cd0tlold + 4.16667*Wto**2./
     -       (bold**2.*(1.-dold**2./bold**2.)*pi*rhotl**2.*Sold*
     -     Vtl**4.))**2.)+ 2./
     -    (rhotl*Sold*Vtl**2.*(cd0tlold +
     -        4.16667*Wto**2./
     -         (bold**2.*(1.-dold**2./bold**2.)*pi*rhotl**2.*Sold*
     -       Vtl**4.)))*
     -       Wto/LDtold

******* Ldc Approximation **************

       dLdcdcd0c =
     -       -0.244949*bold**2.*(1.- dold**2./bold**2.)*pi/
     -   (cd0cold**2.*Sqrt(bold**2.*(1.-dold**2./bold**2.)*pi/
     -       (cd0cold*Sold))*Sold)*cd0cold/LDcold

         dLdcdd =
     -      -0.489898*dold*pi/
     -   (cd0cold*Sqrt(bold**2.*(1.-dold**2./bold**2.)*pi/
     -       (cd0cold*Sold))*Sold)
     -       *dold/LDcold
```

```
      dLdcdb =
     -      0.244949*(2.*dold**2.*pi/(bold*cd0cold*Sold) +
     -     2.*bold*(1.- dold**2./bold**2.)*pi/(cd0cold*Sold))/
     -  Sqrt(bold**2.*(1.- dold**2./bold**2.)*pi/(cd0cold*Sold))
     -      *bold/LDcold

      dLdcdS =
     -      -0.244949*bold**2.*(1.- dold**2./bold**2.)*pi/
     -  (cd0cold*Sqrt(bold**2.*(1.- dold**2./bold**2.)*pi/
     -      (cd0cold*Sold))*
     -    Sold**2.)*Sold/LDcold


******* Ldl Approximation ***************

        dLdldd =
     -   (-3.47959*10**-6*dold*(1-Rfrold)*Wto*((1-Rfrold)*Wto)**2/
     -   (bold**4*(1-dold**2/bold**2)**2*
     -   Sold**2*(cd0tlold+(0.0001*((1-Rfrold)*Wto)**2)/
     -   (bold**2*(1-dold**2/bold**2)*Sold))**2))* dold/LDlold

        dLdldcd0tl = (-0.01738*(1-Rfrold)*Wto)/
     -   (Sold*(cd0tlold+(0.0001*((1-Rfrold)*Wto)**2)/
     - (bold**2*(1-dold**2/bold**2)*Sold))**2)* cd0tlold/LDlold

      dLdldRfr =
     -  (3.47959*10**-6*dold*(1-Rfrold)*Wto**2*((1-Rfrold)*Wto)/
     -   (bold**2*(1-dold**2/bold**2)*
     -   Sold**2*(cd0tlold+(0.0001*((1-Rfrold)*Wto)**2)/
     -   (bold**2*(1-dold**2/bold**2)*Sold))**2) -
     -   (0.0173769*Wto)/
     -   Sold*(cd0tlold+(0.0001*((1-Rfrold)*Wto)**2)/
     -   (bold**2*(1-dold**2/bold**2)*Sold))**2)* Rfrold/LDlold

     dLdldS =(1.73979*10**-6*(1-Rfrold)*Wto*((1-Rfrold)*Wto)**2)/
     -  (bold**2*(1-dold**2/bold**2)*
     - Sold**3*(cd0tlold+(0.0001*((1-Rfrold)*Wto)**2)/
     -  (bold**2*(1-dold**2/bold**2)*Sold))**2) -
     - (0.01738*(1-Rfrold)*Wto)/(Sold**2*(cd0tlold+(0.0001*
     - ((1-Rfrold)*Wto)**2)/(bold**2*(1-dold**2/bold**2)*Sold)))
     -      * Sold/LDlold

        dLdldb =
     - (-0.01738*(1-Rfrold)*Wto*
      -  (-0.0002*dold**2*((1-Rfrold)*Wto)**2/
     -    (bold**5*(1-dold**2/bold**2)**2*Sold) - 0.0002*
     -      ((1-Rfrold)*Wto)**2/
     -    (bold**3*(1-dold**2/bold**2)*Sold))/
     -    (Sold*(cd0tlold + 0.0001*((1-Rfrold)*Wto)**2/
     -    (bold**2*(1-dold**2/bold**2)*Sold))**2)) * bold/LDlold

        dLdldWto =
     -    (-3.47959*10**-6*(1-Rfrold)**2*Wto*((1-Rfrold)*Wto))/
     -  (bold**2*(1-dold**2/bold**2)*
     - Sold**2*(cd0tlold+(0.0001*((1-Rfrold)*Wto)**2)/
```

```
     -   (bold**2*(1-dold**2/bold**2)*Sold))**2) +
      -   (0.01738*(1-Rfrold))/
     - (Sold*(cd0tlold+(0.0001*((1-Rfrold)*Wto)**2)/
     - (bold**2*(1-dold**2/bold**2)*Sold)))* Wto/LDlold

********** Vbr Approximation ************

      dVbrdcd0c =
     -      -0.35718*bold**2.*(1.- dold**2./bold**2.)*pi*Wto/
     - (rhoc*(bold**2.*cd0cold*(1.-dold**2./bold**2.)*pi/Sold)**
     -    (3./2.)*Sold**2.*Sqrt(Wto/
     -    (rhoc*Sqrt(bold**2.*cd0cold*(1.-dold**2./bold**2.)*
     -      pi/Sold)*Sold)))*
     -      cd0cold/Vbrold

      dVbrdd =
     -      0.71436*cd0cold*dold*pi*Wto/
     -   (rhoc*(bold**2.*cd0cold*(1.-dold**2./bold**2.)*pi/Sold)**
     -     (3./2.)*Sold**2.*Sqrt(Wto/
     -   (rhoc*Sqrt(bold**2.*cd0cold*(1.- dold**2./bold**2.)*
     -      pi/Sold)*Sold)))*
     -      dold/Vbrold

      dVbrdb =
     -      -0.35718*(2.*cd0cold*dold**2.*pi/(bold*Sold) +
     -     2.*bold*cd0cold*(1.- dold**2./bold**2.)*pi/Sold)*Wto/
     -   (rhoc*(bold**2*cd0cold*(1.- dold**2./bold**2.)*pi/Sold)**
     -     (3./2.)*Sold*Sqrt(Wto/
     -    (rhoc*Sqrt(bold**2*cd0cold*(1.- dold**2./bold**2.)*pi/
     -      Sold)*Sold)))*
     -      bold/Vbrold

      dVbrdS =
     - 0.71436*(bold**2*cd0cold*(1.-dold**2./bold**2.)*pi*Wto/
     - (2.*rhoc*(bold**2*cd0cold*(1.-dold**2./bold**2.)*pi/Sold)**
     -     (3./2.)*Sold**3.) -
     -   Wto/
     -    (rhoc*Sqrt(bold**2.*cd0cold*(1.- dold**2./bold**2.)*
     -      pi/Sold)*Sold**2.))/
     -   Sqrt(Wto/
     -    (rhoc*Sqrt(bold**2.*cd0cold*(1.- dold**2./bold**2.)*pi/
     -      Sold)*Sold))*
     -      Sold/Vbrold

      dVbrdWto =
     - 0.71436/(rhoc*Sqrt(bold**2.*cd0cold*(1.-dold**2./bold**2.)*
     -   pi/Sold)*Sold*Sqrt(Wto/
     -   (rhoc*Sqrt(bold**2.*cd0cold*(1.-dold**2./bold**2.)*pi/
     -      Sold)*Sold)))*
     -      Wto/Vbrold

**********  Aspect Ratio, AR Approximation ********

      dARdb =
     -        2*bold/Sold*bold/ARold
```

```
      dARdS =
     -        -(bold**2/Sold**2)* Sold/ARold

**********  qL Approximation **************

      dqLdLDl = 1/LDlold**2*LDlold/qLold
      dqLdRfr = (0.6667*Ti)/((1-Rfrold)**2*Wto)* Rfrold/qLold
      dqLdTi = (0.6667)/((1-Rfrold)*Wto)* Ti/qLold
      dqLdWto = (-0.6667*Ti)/((1-Rfrold)*Wto**2)* Wto/qLold

*********** qTO Approximation *************

      dqTOdLDt = 1/LDtold**2 * LDtold/qTOold
      dqTOdTi = 0.6667/Wto * Ti/qTOold
      dqTOdWto = -0.6667*Ti/Wto**2 * Wto/qTOold

*********** sL Approximation ************

      dsLdRfr = -45.3846*Wto/Sold * Rfrold/sLold
      dsLdS = (-45.3846*(1-Rfrold)*Wto)/Sold**2 * Sold/sLold
      dsLdWto = (45.3846*(1-Rfrold))/Sold * Wto/sLold

********** sTO Approximation **********

      dsTOdS = (-8.03846*Wto**2)/(Sold*Ti) -
     -            (26.9776*Wto)/(Sold**2*SQRT(Wto/Sold))* Sold/sTOold
      dsTOdTi = (-8.03846*Wto**2)/(Sold*Ti**2)* Ti/sTOold
      dsTOdWto = (16.0769*Wto)/(Sold*Ti) +
     -          (26.9776)/(Sold**2*SQRT(Wto/Sold))* Wto/sTOold


************************************************
*******     SETTING UP GSE MATRICES  *************
************************************************

      GSEM(3,7) = -dcd0cdVbr
      GSEM(4,1) = -dLdtdd
      GSEM(4,2) = -dLdtdcd0tl
      GSEM(5,1) = -dLdcdd
      GSEM(5,3) = -dLdcdcd0c
      GSEM(6,1) = -dLdldd
      GSEM(6,2) = -dLdldcd0tl
      GSEM(6,13) = -dLdldRfr
      GSEM(7,1) = -dVbrdd
      GSEM(7,3) = -dVbrdcd0c

      GSEM(9,6) = -dqLdLDl
      GSEM(9,13) = -dqTOdLDt
      GSEM(10,4) = -dqLdRfr
      GSEM(11,13) = -dsLdRfr

      GSEM(13,7) = -dRfrdVbr
      GSEM(13,5) = -dRfrdLdc
      GSEM(15,13) = -dRfdRfa
```

```
      GSEM(15,14) = -dRfdRfr
      GSEM(16,5) = -dUdLdc
      GSEM(16,7) = -dUdVbr
      GSEM(17,7) = -dPRdVbr
      GSEM(17,13) = -dPRdRfr
      GSEM(17,14) = -dPRdRfa

      GSERHS(1,2) = dddl
      GSERHS(2,1) = dcd0tldS
      GSERHS(2,2) = dcd0tldl
      GSERHS(2,3) = dcd0tldb
      GSERHS(3,1) = dcd0cdS
     GSERHS(3,2) = dcd0cdl
     GSERHS(3,3) = dcd0cdb
     GSERHS(4,1) = dLdtdS
     GSERHS(4,3) = dLdtdb
     GSERHS(4,4) = dLdtdWto
     GSERHS(5,1) = dLdcdS
     GSERHS(5,3) = dLdcdb
     GSERHS(6,1) = dLdldS
     GSERHS(6,3) = dLdldb
     GSERHS(6,4) = dLdldWto
     GSERHS(7,1) = dVbrdS
     GSERHS(7,3) = dVbrdb
     GSERHS(7,4) = dVbrdWto
     GSERHS(8,1) = dARdS
     GSERHS(8,3) = dARdb

     GSERHS(9,4) = dqLdWto
     GSERHS(9,5) = dqLdTi
     GSERHS(10,4) = dqTOdWto
     GSERHS(10,5) = dqTOdTi
     GSERHS(11,1) = dsLdS
     GSERHS(11,4) = dsLdWto
     GSERHS(12,1) = dsTOdS
     GSERHS(12,4) = dsTOdWto
     GSERHS(12,5) = dsTOdTi

     GSERHS(14,4) = dRfadWto
     GSERHS(14,5) = dRfadTi
     GSERHS(16,4) = dUdWto
     GSERHS(17,4) = dPRdWto


**********************************
*     Solve GSE                  ***
*     Construct taylor series    ***
**********************************

      call INVERSE(GSEM,NUMSTAT,GSERHS,TOTDESV,DET)
C     call mmult(NUMSTAT,NUMSTAT,TOTDESV,GSEMINV,GSERHS,GSESOLV)
C
C     do 10 i=1,NUMSTAT
C           do 11 j=1,TOTDESV
C           PRINT *,'X(',i,',',j,') = ',GSERHS(i,j)
```

```
C 11        CONTINUE
C 10  CONTINUE


      Do 802 I=1,NUMSTAT
              Do 803 J=1,TOTDESV

*   the ratio pu/pd will 'denormalize' the important sensitivities.
                  If(I.eq.1) pu=dold
                  If(I.eq.2) pu=cd0tlold
                  If(I.eq.3) pu=cd0cold
                  If(I.eq.4) pu=LDtold
                  If(I.eq.5) pu=LDcold
                  If(I.eq.6) pu=Ldlold
                  If(I.eq.7) pu=Vbrold
                  If(I.eq.8) pu=ARold
                  If(I.eq.13) pu=Rfrold
                  If(I.eq.14) pu=Rfaold
                  If(I.eq.15) pu=Rfold
                  If(I.eq.16) pu=Uold
                  If(I.eq.17) pu=PRIold

                  If(j.eq.1) pd=Sold
                  If(j.eq.2) pd=lold
                  If(j.eq.3) pd=bold
                  If(j.eq.4) pd=Wto
                  If(j.eq.5) pd=Ti

                     GSERHS(I,J)=GSERHS(I,J)*pu/pd

 803            Continue
 802    Continue


**********************************************************************
****  Taylor's Series Approximation using first order terms ****
**********************************************************************

C     X is the vector of all design variables X(i,2) = new
C     X(i,1) = old
C
C     Xold are the first terms in the taylor's series expansion

      dap = dold
      cd0tlap = cd0tlold
      cd0cap = cd0cold
      LDtap = LDtold
      LDcap = LDcold
      LDlap = LDlold
      Vbrap = Vbrold
      ARap = ARold
      Rfrap = Rfrold
      Rfaap = Rfaold
      Rfap = Rfold
      Uap = Uold
```

```
      PRIap = PRIold

      DO 801 I=1,TOTDESV
            dap = dap + GSERHS(1,I)*(X(I,2)-X(I,1))
            cd0tlap = cd0tlap + GSERHS(2,I)*(X(I,2)-X(I,1))
            cd0cap = cd0cap + GSERHS(3,I)*(X(I,2)-X(I,1))
            LDtap = LDtap + GSERHS(4,I)*(X(I,2)-X(I,1))
            LDcap = LDcap + GSERHS(5,I)*(X(I,2)-X(I,1))
            LDlap = LDlap + GSERHS(6,I)*(X(I,2)-X(I,1))
            Vbrap = Vbrap + GSERHS(7,I)*(X(I,2)-X(I,1))
            ARap = ARap + GSERHS(8,I)*(X(I,2)-X(I,1))
            Rfrap = Rfrap + GSERHS(13,I)*(X(I,2)-X(I,1))
            Rfaap = Rfaap + GSERHS(14,I)*(X(I,2)-X(I,1))
            Rfap = Rfap + GSERHS(15,I)*(X(I,2)-X(I,1))
            Uap = Uap + GSERHS(16,I)*(X(I,2)-X(I,1))
            PRIap = PRIap + GSERHS(17,I)*(X(I,2)-X(I,1))
 801  CONTINUE


      d = 1.83*(1. + 4.325 * Np / l)

      cd0c = 0.05
      Vbr = 770

C*****
C*****Iterate until converged within AERO

      DO 100 I=1,COUNT

      IF (ABS((cd0co - cd0c)/cd0c).LE.EPS) THEN
      IF (ABS((Vbro-Vbr)/Vbr).LE.EPS) THEN
            GOTO 101
      ENDIF
      ENDIF

      IF(I.EQ.40) THEN
      PRINT*,'TROUBLE CONVERGING WITH cd0c AND Vbr'
      ENDIF

      cd0co = cd0c
      Vbro = Vbr

C**********************************
C            cd0c
C**********************************
      cd0c = 0.005 + 7.16109*1*(1. + 4.325 * Np / l)*
     -    (1. + 367.709/(1/(1. + 4.325 * Np / l))**3. +
     -      H1*1/(1. + 4.325 * Np / l))*
     -    (1. - 3.66*(1. + 4.325 * Np / l)/l)**0.666667*
     -    (1. + 3.3489*((1. + 4.325 * Np / l)/l)**2.)*pi/
     -   (S*Log(1*Vbr/muc)**2.58) +
     -  8.60896*(1. + 1.2*tc + 100.*tc**4.)/Log(S*Vbr/(b*muc))**2.58

C     best range speed
      Vbr = 1.42872*Sqrt(Wto/
```

```
     -    (rhoc*Sqrt(b**2.*cd0c*(1.- d**2./b**2.)*pi/S)*S))

 100  CONTINUE

C     optimum cruise lift-to-drag ratio
 101  LDc=
     -      0.489898*Sqrt(b**2.*(1.- d**2./b**2.)*pi/(cd0c*S))
C**********************************
C               cd0tl
C**********************************
      cd0tl = 0.005 + 7.16109*1*(1. + 4.325 * Np / 1)*
     -    (1. + 367.709/(1/(1. + 4.325 * Np / 1))**3. +
     -      H1*1/(1. + 4.325 * Np / 1))*
     -    (1. - 3.66*(1. + 4.325 * Np / 1)/1)**0.666667*
     -    (1. + 3.3489*((1. + 4.325 * Np / 1)/1)**2.)*pi/
     -   (S*Log(1*Vtl/mutl)**2.58) +
     -  8.60896*(1. + 1.2*tc + 100.*tc**4.)/Log(S*Vtl/(b*mutl))**2.58


C     lift - to - drag take off
      LDt=2.*Wto/(rhotl*S*Vtl**2.*(cd0tl +
     -      4.16667*Wto**2./
     -       (b**2.*(1.- d**2./b**2.)*pi*rhotl**2.*S*Vtl**4.)))

C     lift - to - drag landing
      LDl=2.*(1-Rfrap)*Wto/(rhotl*S*Vtl**2.
     -  *(cd0tl + 4.16667*((1-Rfrap)*Wto)**2./
     -  (b**2.*(1.-d**2./b**2.)*pi*rhotl**2.*S*Vtl**4.)))

      AR=b**2/S

c********************** Weights constraints (LD's approximate)

C     fuel weight available ratio
      Rfa = 1. - 0.38*Ti**0.9981/Wto
     -      - 0.9592/Wto**0.0638 - Wfix/Wto - Wpay/Wto

C     fuel weight required ratio
      Rfr = 1.1*(1. - 0.95*EXP(-bt*R/(LDcap*Vbrap)))

C     useful load fraction
      U = 1.1*(1. - 0.95*EXP(-bt*R/(LDcap*Vbrap))) + Wpay/Wto

*     Fuel Weight Balance
      Rf = Rfa / Rfr

********************** Aero constraints (Rfr approximate)

** *  Achievable Climb Angle, OEI, LANDING
      qLa = -1/LDl + (-1+N)*Ti/(N*Wto*(1-Rfrap))

*     Achievable Climb Angle, OEI, TAKEOFF
      qTOa = -1/LDt + (-1+N)*Ti/(N*Wto)

*     Landing Field Length
```

```
      sLa = 400 + 118*Wto*(1-Rfrap)/(clmax*S)

*     Takeoff Field Length
      sTO= 20.9*Wto**2/(clmax*S*Ti)+87*Sqrt(Wto/(clmax*S))


********** Weights constraints (LD's approximate)

*     Landing Field Length
      sLw = 400 + 118*Wto*(1-Rfr)/(clmax*S)

*     Achievable Climb Angle, OEI, LANDING
      qLw = -1/LDlap + (-1+N)*Ti/(N*Wto*(1-Rfr))

*     Achievable Climb Angle, OEI, TAKEOFF
      qTOw = -1/LDtap + (-1+N)*Ti/(N*Wto)

********** Actual coupling constraints

** *  Achievable Climb Angle, OEI, LANDING
      qL = -1/LDl + (-1+N)*Ti/(N*Wto*(1-Rfr))

*     Achievable Climb Angle, OEI, TAKEOFF
      qTO = -1/LDt + (-1+N)*Ti/(N*Wto)

*     Landing Field Length
      sL = 400 + 118*Wto*(1-Rfr)/(clmax*S)

*     Takeoff Field Length
      sTO= 20.9*Wto**2/(clmax*S*Ti)+87*Sqrt(Wto/(clmax*S))


C
C     PRODUCTIVITY INDEX CALCULATION

      PRI = Vbrap*Wpay/((1-Rfa+Rfr)*Wto - Wpay - Wfix)
C
      OPEN (1,FILE ='aprxcoop.inout')

      WRITE(1,*)d, cd0tl, cd0c, LDt, LDc,
     &      LDl, Vbr, AR, qL, qTO,
     &      sL, sTO, Rfr, Rfa, Rf,
     &      U, PRI, S, l,
     &      b, Wto, Ti

      CLOSE(1)

C
C
C*******************WRITING OUTPUT TO FILE******************
C
      OPEN (8,FILE='aprxcoop.apprxs')
      OPEN (9,FILE='aprxcoop.apprxs2')

      WRITE (8,*) LDt,LDtap,LDc,LDcap,LDl,LDlap
```

```
      WRITE (9,*) Vbr,Vbrap,Rfr,Rfrap

      OPEN (11,FILE='aprxcoop.vars')

      WRITE (11,*) 'DESIGN VARIABLES:'
      WRITE (11,*) 'S   = ',S
      WRITE (11,*) 'l = ',l
      WRITE (11,*) 'b   = ',b
      WRITE (11,*) 'Wto = ',Wto
      WRITE (11,*) 'Ti = ',Ti


      WRITE (11,*)
      WRITE (11,*) 'BEHAVIOR VARIABLES:'
      WRITE (11,*) 'cd0c   = ',cd0c
      WRITE (11,*) 'cd0tl  = ',cd0tl
      WRITE (11,*) 'd      = ',d
      WRITE (11,*) 'LDl    = ',LDl
      WRITE (11,*) 'LDlap    = ',LDlap
      WRITE (11,*) 'LDt    = ',LDt
      WRITE (11,*) 'LDtap    = ',LDtap
      WRITE (11,*) 'LDc    = ',LDc
      WRITE (11,*) 'LDcap    = ',LDcap
      WRITE (11,*) 'Vbr    = ',Vbr
      WRITE (11,*) 'Vbrap    = ',Vbrap
      WRITE (11,*) 'qla     = ',qla
      WRITE (11,*) 'qlw    = ',qlw
      WRITE (11,*) 'qtoa    = ',qtoa
      WRITE (11,*) 'qtow    = ',qtow
      WRITE (11,*) 'sTO    = ',sTO
      WRITE (11,*) 'sLa    = ',sLa
      WRITE (11,*) 'sLw     = ',sLw
      WRITE (11,*) 'AR     = ',AR
      WRITE (11,*) 'Rfa    = ',Rfa
      WRITE (11,*) 'Rfr    = ',Rfr
      WRITE (11,*) 'Rfrap   = ',Rfrap
      WRITE (11,*) 'U      = ',U
      WRITE (11,*) 'Rf     = ',Rf
      WRITE (11,*) 'PRI    = ',PRI
      WRITE (11,*) 'ql     = ',ql
      WRITE (11,*) 'qto    = ',qto
      WRITE (11,*) 'sTO    = ',sTO
      WRITE (11,*) 'sL    = ',sL

      CLOSE(11)

C
C     3.0 Evaluate non-linear constraints
C
      IF (IPATH  .EQ.  1  .OR.  IPATH  .EQ.  2) THEN
C
C        ASPECT RATIO constraint.
         CONSTR(1) = -AR/10.5 + 1.0
C
C        ACHIEVABLE CLIMB GRADIENT, LANDING constraint, AERO
```

```
          CONSTR(2) = -1.0 + qla/0.024
C
C         ACHIEVABLE CLIMB GRADIENT, Weight subsystem
          CONSTR(3) = -1.0 + qlw/0.024
C
C         ACHIEVABLE CLIMB GRADIENT, TAKE-OFF constraint
          CONSTR(4) = -1.0 + qtoa/0.027
C
C         ACHIEVABLE CLIMB GRADIENT, weight subsystem
          CONSTR(5) = -1.0 + qtow/0.027
C
C         LANDING FIELD LENGTH constraint
          CONSTR(6) = 1.0 - Sla/4500
C
C         LANDING FIELD LENGTH constraint, AERO
          CONSTR(7) = 1.0 - Slw/4500
C
C         TAKE-OFF FIELD LENGTH constraint
          CONSTR(8) = 1.0 - Sto/6500
C
C         Cdotl limit
          CONSTR(9) = 1.0 - cd0tl/0.02
C
C         Cdoc limit
          CONSTR(10) = 1.0 - cd0c/0.02
C
C         USEFUL LOAD FRACTION constraint
          CONSTR(11) = U/0.3 - 1.0
C
C         FUEL BALANCE constraint
          CONSTR(12) = -1.0 + Rf
C
=
      END IF
C
=
C     4.0 Evaluate non-linear goals
C
      IF (IPATH .EQ. 1 .OR. IPATH .EQ. 3) THEN
C
C       MISSED APPROACH CLIMB GRADIENT, OEI, landing goal, AERO
          GOALS(1) = 1.0 - (0.03/qLa)
C
C       MISSED APPROACH CLIMB GRADIENT, OEI, Weights
          GOALS(2) = 1.0 - (0.03/qLw)
C
C         MISSED APPROACH CLIMB GRADIENT, OEI, take-off goal
          GOALS(3) = (qtoa/0.03) - 1.0

C         MISSED APPROACH CLIMB GRADIENT, OEI, Weights
          GOALS(4) = (qtow/0.03) - 1.0
C
C         ASPECT RATIO GOAL
          GOALS(5) = AR/10.5 - 1.0
C
```

```
C         LANDING FIELD LENGTH goal
          GOALS(6) = Sla/4500 -1.0
C
C         LANDING FIELD LENGTH goal, AERO approximation
          GOALS(7) = Slw/4500 -1.0
C
C         TAKE-OFF FIELD LENGTH goal
          GOALS(8) = Sto/4500 -1.0
C
C         PRODUCTIVITY INDEX
          GOALS(9) = PRI/270 - 1.0
C
C         USEFUL LOAD FRACTION goal
          GOALS(10) = U/0.5 - 1.0
C
C         FUEL BALANCE goal
          GOALS(11) = Rf - 1.0
C
       OPEN (18,FILE='devfuncs.out')
       AACH = ABS(GOALS(1)) + ABS(GOALS(3)) + ABS(GOALS(5)) +
      &       ABS(GOALS(6)) + ABS(GOALS(8))
       WACH = ABS(GOALS(2)) + ABS(GOALS(4)) + ABS(GOALS(7)) +
      &       ABS(GOALS(8)) + ABS(GOALS(9)) +
      &       ABS(GOALS(10)) + ABS(GOALS(11))

       WRITE (18,*) AACH, WACH
C
       END IF
C
C      5.0 Return to calling routine
C
       RETURN
       END
C
.
.
.
************************************************************
************************************************************
*      SUBROUTINE INVERSE(A,N,B,Mn,DET)
*      SUBPROGRAM FOR MATRIX INVERSION AND SIMULTANEOUS
*      LINEAR EQUATION SOLUTION.  TAKEN FROM KUO'S "COMPUTER
*      APPLICATIONS OF NUMERICAL METHODS", ADDISON WESLEY, 1972.
*      USES A GAUSS-JORDAN REDUCTION TECHNIQUE

*      A = GIVEN COEFFICIENT MATRIX, INVERSE OF A IS STORED AT
*          RETURN TO MAIN PROGRAM
*      N = ORDER OF A; N>=1
*      B = MATRIX OF CONSTANTS VECTOR, USED ONLY FOR SOLUTION OF
*          SIMULTANEOUS EQUATIONS
*      Mn = NUMBER OF COLUMN VECTORS IN THE MATRIX OF CONSTANT
*          VECTORS (M=0 IF INVERSION IS SOLE AIM; Mn=1,2,... FOR
*          SOLUTION OF SIMULTANEOUS EQUATIONS)
*    DET = DETERMINANT OF A MATRIX
************************************************************
```

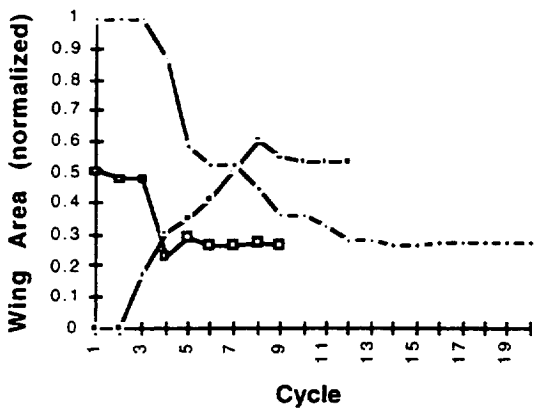
```
      DIMENSION A(17,17),B(17,5),IPVOT(30),INDEX(30,2),PIVOT(30)
      COMMON IPVOT,INDEX,PIVOT
C      INTEGER I,J,K,L,N,LI,MN,
C     &     IROW,JROW,ICOL,JCOL,INDEX, IPVOT, PIVOT
C      REAL T, A, B, DET
      EQUIVALENCE (IROW,JROW),(ICOL,JCOL)


*     INITIALIZATION

   57 DET = 1.
      DO 17 J=1,N
   17 IPVOT(J)=0
      DO 135 I=1,N

*     SEARCH FOR PIVOT ELEMENT

      T=0.
      DO 9 J=1,N
      IF (IPVOT(J).EQ.1) GO TO 9
   13 DO 23 K=1,N
      IF(IPVOT(K)-1) 43,23,81
   43 IF(ABS(T).GE.ABS(A(J,K))) GO TO 23
   83 IROW=J
      ICOL=K
      T=A(J,K)
   23 CONTINUE
    9 CONTINUE
      IPVOT(ICOL)=IPVOT(ICOL)+1

*     PUT PIVOT ELEMENT ON DIAGONAL

      IF(IROW.EQ.ICOL) GO TO 109
   73 DET=-DET
      DO 12 L=1,N
      T=A(IROW,L)
      A(IROW,L)=A(ICOL,L)
   12 A(ICOL,L)=T
      IF(Mn.LE.0) GO TO 109
   33 DO 2 L=1,Mn
      T=B(IROW,L)
      B(IROW,L)=B(ICOL,L)
    2 B(ICOL,L)=T
  109 INDEX(I,1)=IROW
       INDEX(I,2)=ICOL
      PIVOT(I)=A(ICOL,ICOL)
      DET=DET*PIVOT(I)

*     DIVIDE PIVOT ROW BY PIVOT ELEMENT

      A(ICOL,ICOL)=1.
      DO 205 L=1,N
  205 A(ICOL,L)=A(ICOL,L)/PIVOT(I)
      IF (Mn.LE.0) GO TO 347
```

```
   66  DO 52 L=1,Mn
   52  B(ICOL,L)=B(ICOL,L)/PIVOT(I)

*      REDUCE NON-PIVOT ROWS

  347  DO 135 LI=1,N
       IF(LI.EQ.ICOL) GO TO 135
   21  T=A(LI,ICOL)
       A(LI,ICOL)=0.
       DO 89 L=1,N
   89  A(LI,L)=A(LI,L)-A(ICOL,L)*T
       IF(Mn.LE.0) GO TO 135
   18  DO 68 L=1,Mn
   68  B(LI,L)=B(LI,L)-B(ICOL,L)*T
  135  CONTINUE

*      INTERCHANGE COLUMNS

  222  DO 3 I=1,N
       L=N-I+1
       IF(INDEX(L,1).EQ.INDEX(L,2)) GO TO 3
   19  JROW=INDEX(L,1)
       JCOL=INDEX(L,2)
       DO 549 K=1,N
       T=A(K,JROW)
       A(K,JROW)=A(K,JCOL)
       A(K,JCOL)=T
  549  CONTINUE
    3  CONTINUE
   81  RETURN
       END
```

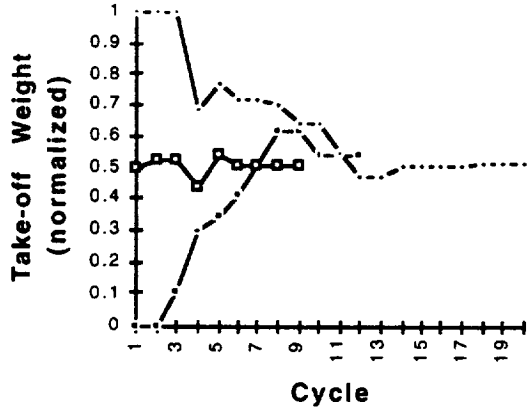
SOLUTION HISTORY: Approximate Cooperation

In Figure C.6, the design variable history for the approximate cooperative formulation is given for three starting points. Two of the starting points converge to the same solution, but the third converges to another solution. This lack of convergence to one common solution can be attributed to the occasional instabilities in the nonlocal Taylor series approximations, as discussed in Section 7.5.1.

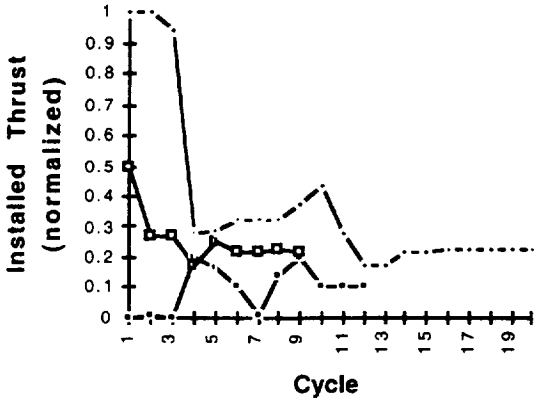
In Figure C.7, the best deviation function and constraint violation are plotted. The best deviation function steadily decreases, and the constraint violation pregressively goes to zero, representing feasibility.



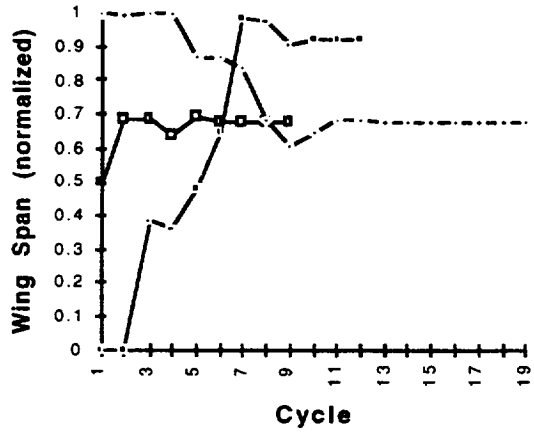
(a) Wing Area (normalized)



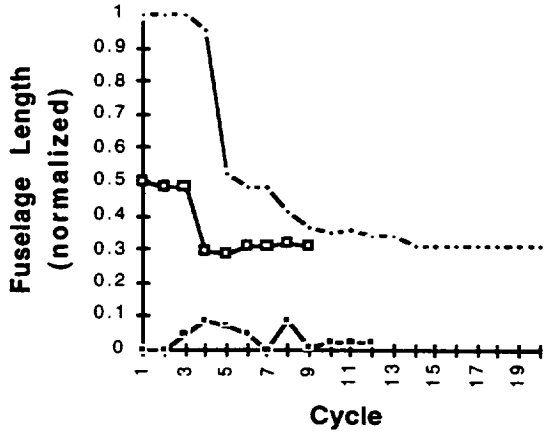
(b) Take-off Weight (normalized)



(c) Installed Thrust (normalized)



(d) Wing Span (normalized)



(e) Fuselage Length (normalized)

Figure C.6. Design Variable History: Approximate Cooperation

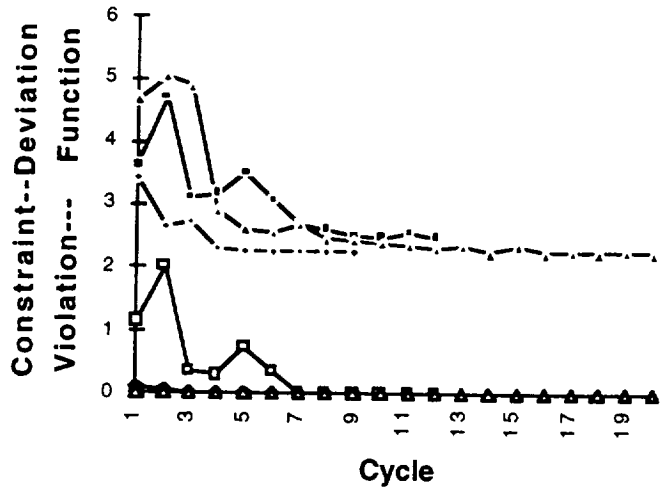


Figure C.7. Best Deviation Function and Constraint Violation: Approximate Cooperation

FULL SOLUTIONS: Leader/Follower Formulations

In Table C.2, the full solutions of the two leader/follower formulations are given. Included are the design variables, state variables, constraints, goals, deviation functions, and constraint violation of each player in both formulations.

Table C.2. Leader/Follower Full Solutions

Aero as Leader

Player Aero		Player Weights	
X, design vars		**X**, design vars	
S (ft^2)	1870	Ti (lbs)	36725
b (ft)	136	Wto (lbs)	224206
l (ft)	107		
s, state vars		**s**, state vars	
cdo_c	0.017	R_{fa}	0.319
cdo_{tl}	0.018	R_{fr}	0.282
d (ft)	15.73	U	0.461
Ld_l	16.0	R_f	1.13
Ld_t	12.6	PRI	155
Ld_c	20.7	q_L	0.090
V_{br} (ft/s)	676.2	q_{TO}	0.030
AR	9.89	s_L (ft)	4306
q_L	0.090	s_{TO} (ft)	6474
q_{TO}	0.030		
s_L (ft)	4306		
S_{TO} (ft)	6474		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	0.058	g_1	0.535
g_2	2.732	g_2	0.130
g_3	0.111	g_3	2.732
g_4	0.043	g_4	0.111
g_5	0.004	g_5	0.043
g_6	0.125	g_6	0.004
g_7	0.140		
goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.665	f_1	-0.427
f_2	0.000	f_2	-0.079
f_3	-0.058	f_3	0.130
f_4	-0.043	f_4	0.665
f_5	0.439	f_5	0.000
		f_6	-0.043
		f_7	0.439
devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.241	Zweight	0.255
convio	0.0	convio	0.0

Weights as Leader

Player Aero		Player Weights	
X, design vars		**X**, design vars	
S (ft^2)	1644	Ti (lbs)	41000
b (ft)	114	Wto (lbs)	208216
l (ft)	150		
s, state vars		**s**, state vars	
cdo_c	0.018	R_{fa}	0.290
cdo_{tl}	0.018	R_{fr}	0.291
d (ft)	11.75	U	0.483
Ld_l	12.9	R_f	0.996
Ld_t	9.9	PRI	175
Ld_c	18.3	q_L	0.108
V_{br} (ft/s)	730.2	q_{TO}	0.030
AR	7.91	s_L	4473
q_L	0.108	s_{TO}	5574
q_{TO}	0.030		
s_L (ft)	4473		
S_{TO} (ft)	5574		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	0.247	g_1	0.612
g_2	3.485	g_2	-0.004
g_3	0.106	g_3	3.485
g_4	0.006	g_4	0.106
g_5	0.112	g_5	0.006
g_6	0.098	g_6	0.112
g_7	0.120		
goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.721	f_1	-0.353
f_2	-0.044	f_2	-0.033
f_3	-0.247	f_3	-0.004
f_4	-0.006	f_4	0.721
f_5	0.283	f_5	-0.005
		f_6	-0.006
		f_7	0.283
devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.252	Zweight	0.201
convio	0.0	convio	-0.004

DSIDES DATA FILE: Aerodynamics as Leader

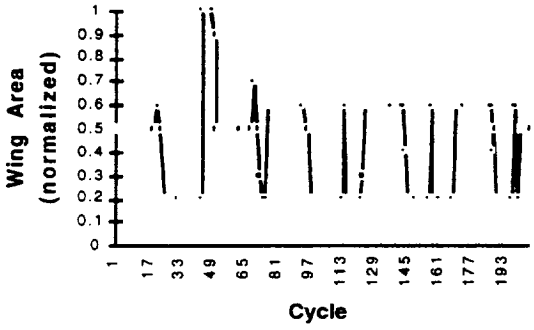
```
PTITLE: Problem Title, User Name and Date
        Aircraft Design, Leader/Follower: Aero as Leader
        Kemper Lewis, October 19.1995

NUMSYS  : Number of system variables: real,discrete, boolean
    1  2  0

SYSVAR:                 System variable information
winga 1  0  1  0:       Wing area
fleng 2  0  1  0:       fuselage length
wspan 3  0  1  0:       Wing Span

NUMCAG  : Number of constraints and goals
 0  7  0  0  5 :  nlinco,nnlinq,nnlequ,nlingo,nnlgoa

DEVFUN  : Achievement function
   1    : levels
   1 10   : level 1, 2 terms
  (-1,1.0) (+1,1.0) (-2,1.0) (+2,1.0) (-3,1.0) (+3,1.0)
  (-4,1.0) (+4,1.0) (-5,1.0) (+5,1.0)


STOPCR  : Stopping criteria
  1  0  40  0.001 0.001  : perform calcs, prt reslts,
Mcyles,sta dev, sta var

NLINCO  : Names of nonlinear constraints
aspr   1: aspect ratio
accl   2: achievable climb gradient, landing
acto   3: achievable climb gradient, take-off
ldfl   4: landing field length
tofl   5: take-off field length
cdtl   6: limit on take-off and landing Cd
cdoc   7: limit on cruise Cd

NLINGO  : Names of the nonlinear goals
misl  1: missed approach landing
mist  2: missed approach take-off
apcr  3: aspect ratio
ldfl  4: landing field length
tofl  5: take-off field length

PVDISC:
2    : variables with discrete values
2 46 1    : variable 2 has 46 possible values, initial value =1
0.0 0.022 0.044 0.067 0.089 0.111 0.133 0.156 0.178 0.2
0.222 0.244 0.267 0.289 0.311 0.333 0.356 0.378 0.4 0.422 0.444
0.467 0.489 0.511 0.533 0.556 0.578 0.6 0.622 0.644 0.667 0.689
```

```
0.711 0.733 0.756 0.778 0.8 0.822 0.844 0.867 0.889 0.911 0.933
0.956 0.978 1.0
3 29 1 : variable 3 has 29 possible values, initial value =1
0.0 0.0182 0.0545 0.0909 0.1273 0.1636 0.2 0.2364 0.2727 0.3091
0.3455 0.3818 0.4182 0.4545 0.4909 0.5273 0.5636 0.6 0.6364
0.6727 0.7091 0.7455 0.7818 0.8182 0.8545 0.8909 0.9273 0.9636
1

ALPOUT   : Output Control
     1   1   1   1  1   1   1  1  1  1

OPTIMP   : Optimization parameters
   -0.005   0.2 0.005 : VIOLIM, REMO, STEP

ENDPRB:
```

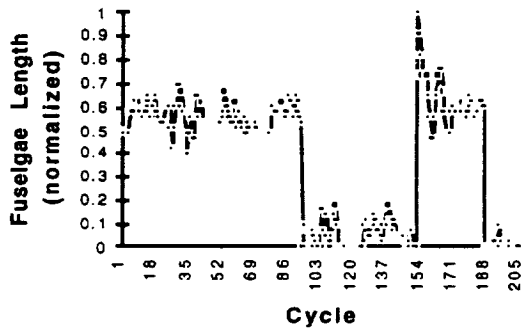
SOLUTION HISTORY: Aerodynamics as Leader

Aerodynamics Solution

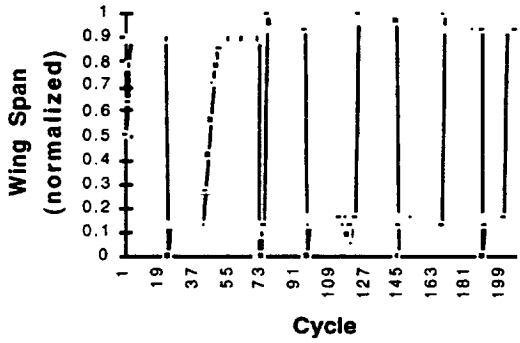
In Figure C.8, the design variable history for the leader/follower formulation with aerodynamics as leader is given for one of the three starting points. Again, since there is no "convergence" in the foraging search, the plots from the other two starting points look similar and do not contribute to the insight or results of this problem.



(a) Wing Area (normalized)



(b) Fuselage Length (normalized)



(c) Wing Span (normalized)

Figure C.8. Design Variable History: Aero as Leader

Weights Solution

Since the aerodynamics player needs both design variables from the weights player, the weight player is constricted to the solution prescribed by his RRS. Once aerodynamics solves their problem, the solution for the weights player is given as well. Therefore, no convergence plots are available for the weights problem.

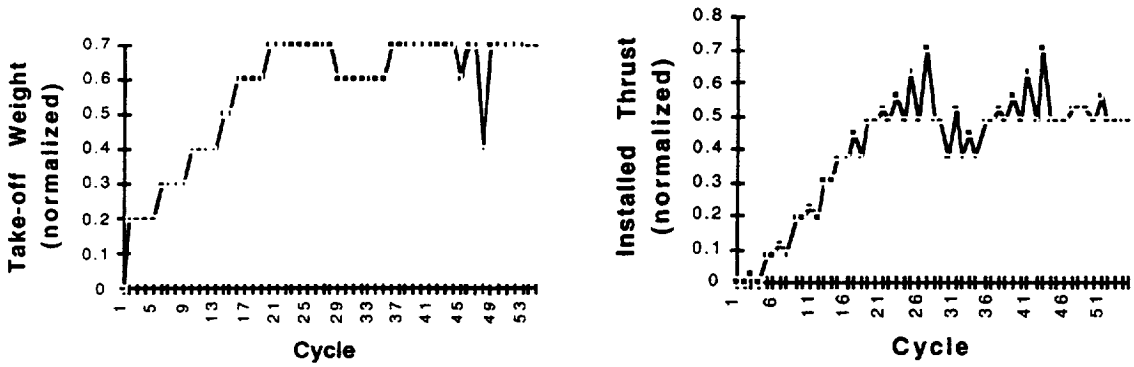
DSIDES DATA FILE: Weights as Leader

```
PTITLE: Aircraft Design, Leader/Follower: Weights as Leader

NUMSYS  : Number of system variables: real,discrete,boolean
   1  1  0: 1 real, 1 discrete

SYSVAR   : System variable information
weigh 1  0  1  0 : take-off weight
insth 2  0  1  0   : installed thrust

NUMCAG  : Number of constraints and goals
 0  6  0  0  7 :  nlinco,nnlinq,nnlequ,nlingo,nnlgoa

DEVFUN  : Achievement function
   1    : levels
   1 14   : level 1, 5 terms
  (-1,1.0) (+1,1.0) (-2,1.0) (+2,1.0) (-3,1.0) (+3,1.0)
  (-4,1.0) (+4,1.0) (-5,1.0) (+5,1.0) (-6,1.0) (+6,1.0)
  (-7,1.0) (+7,1.0)

STOPCR  : Stopping criteria
  1  0  40  0.005 0.005  :

NLINCO  : Names of nonlinear constraints
usfl   1: useful load fraction
fuba   2: fuel balance
achl   3: achievable climb, landing
acto   4: achievable climb, take-off
ldfl   5: landing field length
tofl   6: take-off field length

NLINGO  : Names of the nonlinear goals
prod  1: productivity index
usef  2: useful load fraction
fuel  3: fuel balance
achl   4: achievable climb, landing
acto   5: achievable climb, take-off
ldfl   6: landing field length
tofl   7: take-off field length

ALPOUT   : Output Control
     1   1   1   1  1   1   1  1  1  1

PVDISC:
1    : variables with discrete values
2 19 1 : variable 2 has 19 possible values, initial value =1st
0.0 0.0275 0.0826 0.1193 0.1927 0.2294 0.3028 0.3761 0.4495
0.4862 0.5229 0.5596 0.633 0.7064 0.7431 0.8165 0.8532 0.9266
1.0

ENDPRB:
```

SOLUTION HISTORY: Weights as Leader

Weight's solution

In Figure C.9, the design variable history for the leader/follower formulation with weights as leader is given for one of the three starting points. Again, since there is no "convergence" in the foraging search, the plots from the other two starting points look similar and do not contribute to the insight or results of this problem.

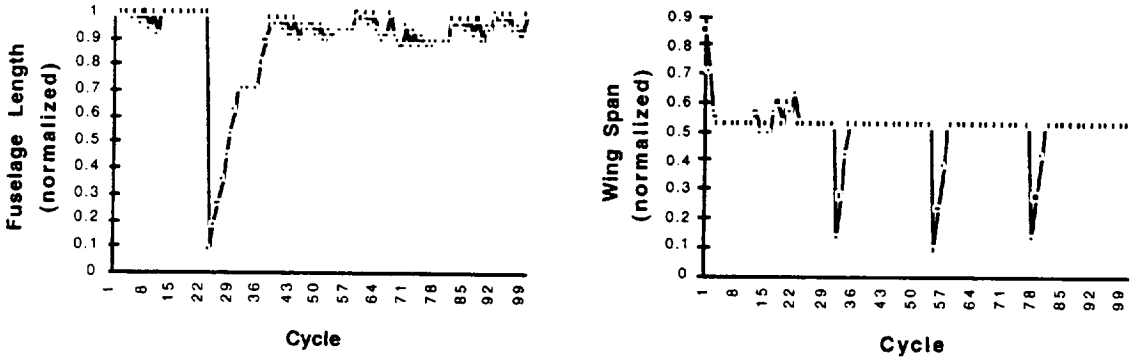


(a) Take-off Weight (normalized) **(b) Installed Thrust (normalized)**

Figure C.9. Design Variable History: Weight as Leader

Aero's follower solution

The weights player only needs the design variable wing area from the aero player. Therefore, the aero player still has the freedom to change his two other design variables, fuselage length and wing span. The value of the wing area is dictated by the aero player's RRS. The solution history for the "free" design variables of the aero player are given in Figure C.10.



(a) Take-off Weight (normalized) **(b) Installed Thrust (normalized)**

Figure C.10. Design Variable History: Aero as Follower

FULL NONCOOPERATIVE SCENARIO SOLUTIONS

The full solutions for each scenario are given in Table C.4. This includes all design variables, state variables, constraints, goals, deviation functions, and constraint violations for each player. The seven scenarios are shown in Table C.3.

Table C.3. Seven Noncooperative Scenarios

Scenario 1	Midpoints of the variable ranges.
Scenario 2	Lower Bounds of the variable ranges.
Scenario 3	Upper Bounds of the variable ranges.
Scenario 4	The values from the Stackelberg formulation with Aerodynamics as leader, Weight as Follower.
Scenario 5	The values from the Stackelberg formulation with Weight as leader, Aerodynamics as Follower.
Scenario 6	The values from the approximate cooperative formulation.
Scenario 7	The values from the full cooperative formulation.

Table C.4. Noncooperative Full Solutions: All Scenarios

Scenario 1				Scenario 2			
Player Aero		Player Weights		Player Aero		Player Weights	
X, design vars		**X**, design vars		**X**, design vars		**X**, design vars	
S (ft^2)	1598	Ti (lbs)	38622	S (ft^2)	1583	Ti (lbs)	60460
b (ft)	112.5	Wto (lbs)	206830	b (ft)	85	Wto (lbs)	185603
l (ft)	127.5			l (ft)	105		
s, state vars		**s**, state vars		**s**, state vars		**s**, state vars	
cdo_c	0.018	R_{fa}	0.292	cdo_c	0.018	R_{fa}	0.215
cdo_{tl}	0.018	R_{fr}	0.290	cdo_{tl}	0.018	R_{fr}	0.332
d (ft)	13.5	U	0.483	d (ft)	16.0	U	0.547
Ld_l	12.6	R_f	1.01	Ld_l	8.6	R_f	0.648
Ld_t	9.6	PRI	179	Ld_t	6.2	PRI	195
Ld_c	18.2	q_L	0.096	Ld_c	13.7	q_L	0.209
V_{br} (ft/s)	737.5	q_{TO}	0.021	V_{br}	811.1	q_{TO}	0.055
AR	7.92	s_L (ft)	4569	AR	4.56	s_L	3954
q_L	0.096	s_{TO} (ft)	6183	q_L	0.209	s_{TO}	3476
q_{TO}	0.021			q_{TO}	0.055		
s_L (ft)	4569			s_L	3954		
S_{TO} (ft)	6183			S_{TO}	3476		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	0.246	g_1	0.612	g_1	0.566	g_1	0.824
g_2	3.006	g_2	0.008	g_2	7.711	g_2	-0.352
g_3	-0.227	g_3	3.006	g_3	1.028	g_3	7.711
g_4	-0.015	g_4	-0.227	g_4	0.1214	g_4	1.028
g_5	0.049	g_5	-0.015	g_5	0.465	g_5	0.121
g_6	0.091	g_6	0.049	g_6	0.087	g_6	0.465
g_7	0.115			g_7	0.120		
goal values (ideal = 0.0)		goal values (ideal = 0.0)		goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.688	f_1	-0.339	f_1	0.857	f_1	-0.277
f_2	-0.304	f_2	-0.033	f_2	0.825	f_2	0.095
f_3	-0.246	f_3	0.008	f_3	-0.566	f_3	-0.352
f_4	0.015	f_4	0.688	f_4	-0.121	f_4	0.857
f_5	0.374	f_5	-0.304	f_5	-0.228	f_5	0.825
		f_6	0.015			f_6	-0.121
		f_7	0.374			f_7	-0.228
devation function and total constraint violation		devation function and total constraint violation		devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.326	Zweight	0.252	Zaero	0.519	Zweight	0.394
convio	-0.242	convio	-0.242	convio	0.0	convio	-0.352

Scenario 3				Scenario 4			
Player Aero		Player Weights		Player Aero		Player Weights	
X, design vars		X, design vars		X, design vars		X, design vars	
S (ft^2)	1529	Ti (lbs)	28814	S (ft^2)	1938	Ti (lbs)	36715
b (ft)	140	Wto (lbs)	176638	b (ft)	136	Wto (lbs)	225960
l (ft)	150			l (ft)	107		
s, state vars		s, state vars		s, state vars		s, state vars	
cdo_c	0.019	R_{fa}	0.263	cdo_c	0.017	R_{fa}	0.310
cdo_{tl}	0.019	R_{fr}	0.285	cdo_{tl}	0.017	R_{fr}	0.290
d (ft)	11.75	U	0.511	d (ft)	15.74	U	0.473
Ld_l	19.8	R_f	0.922	Ld_l	15.9	R_f	1.07
Ld_t	16.2	PRI	174	Ld_t	12.5	PRI	164
Ld_c	22.7	q_L	0.102	Ld_c	20.4	q_L	0.088
V_{br} (ft/s)	608.3	q_{TO}	0.047	V_{br}	675.0	q_{TO}	0.028
AR	12.8	s_L (ft)	4149	AR	9.54	s_L	4184
q_L	0.102	s_{TO} (ft)	6273	q_L	0.088	s_{TO}	6350
q_{TO}	0.047			q_{TO}	0.028		
s_L (ft)	4149			s_L	4184		
S_{TO} (ft)	6273			S_{TO}	6350		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	-0.221	g_1	0.705	g_1	0.091	g_1	0.539
g_2	3.233	g_2	-0.078	g_2	2.687	g_2	0.127
g_3	0.744	g_3	3.233	g_3	0.054	g_3	2.687
g_4	0.078	g_4	0.744	g_4	0.070	g_4	0.054
g_5	0.035	g_5	0.078	g_5	0.023	g_5	0.070
g_6	0.059	g_6	0.035	g_6	0.137	g_6	0.023
g_7	0.065			g_7	0.151		
goal values (ideal = 0.0)		goal values (ideal = 0.0)		goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.705	f_1	-0.354	f_1	0.661	f_1	-0.434
f_2	0.569	f_2	0.023	f_2	-0.051	f_2	-0.077
f_3	0.221	f_3	-0.078	f_3	-0.091	f_3	0.127
f_4	-0.078	f_4	0.705	f_4	-0.070	f_4	0.661
f_5	0.394	f_5	0.569	f_5	0.411	f_5	-0.051
		f_6	-0.078			f_6	-0.070
		f_7	0.394			f_7	0.411
devation function and total constraint violation		devation function and total constraint violation		devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.393	Zweight	0.314	Zaero	0.257	Zweight	0.262
convio	-0.221	convio	-0.078	convio	0.0	convio	0.0

Scenario 5

Player Aero		Player Weights	
X, design vars		**X**, design vars	
S (ft^2)	1571	Ti (lbs)	39971
b (ft)	114	Wto (lbs)	199829
l (ft)	150		
s, state vars		**s**, state vars	
cdo_c	0.018	R_{fa}	0.280
cdo_{tl}	0.018	R_{fr}	0.292
d (ft)	11.75	U	0.492
Ld_l	13.3	R_f	0.96
Ld_t	10.2	PRI	179
Ld_c	18.5	q_L	0.113
V_{br} (ft/s)	720.1	q_{TO}	0.036
AR	8.27	s_L (ft)	4489
q_L	0.113	s_{TO} (ft)	5719
q_{TO}	0.036		
s_L (ft)	4489		
S_{TO} (ft)	5719		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	0.212	g_1	0.639
g_2	3.710	g_2	-0.041
g_3	0.314	g_3	3.710
g_4	0.002	g_4	0.314
g_5	0.120	g_5	0.002
g_6	0.081	g_6	0.120
g_7	0.103		
goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.735	f_1	-0.338
f_2	0.183	f_2	-0.017
f_3	-0.212	f_3	-0.041
f_4	-0.002	f_4	0.735
f_5	0.271	f_5	0.183
		f_6	-0.002
		f_7	0.271
devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.281	Zweight	0.227
convio	0.0	convio	-0.04

Scenario 6

Player Aero		Player Weights	
X, design vars		**X**, design vars	
S (ft^2)	1819	Ti (lbs)	37620
b (ft)	122.4	Wto (lbs)	218461
l (ft)	119		
s, state vars		**s**, state vars	
cdo_c	0.017	R_{fa}	0.310
cdo_{tl}	0.017	R_{fr}	0.290
d (ft)	14.33	U	0.473
Ld_l	13.8	R_f	1.07
Ld_t	10.7	PRI	164
Ld_c	19.0	q_L	0.089
V_{br}	710.4	q_{TO}	0.021
AR	8.24	s_L	4273
q_L	0.089	s_{TO}	6198
q_{TO}	0.021		
s_L	4273		
S_{TO}	6198		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	0.215	g_1	0.576
g_2	2.723	g_2	0.070
g_3	-0.211	g_3	2.723
g_4	0.051	g_4	-0.211
g_5	0.047	g_5	0.051
g_6	0.130	g_6	0.047
g_7	0.149		
goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.664	f_1	-0.391
f_2	-0.290	f_2	-0.055
f_3	-0.215	f_3	0.070
f_4	-0.051	f_4	0.664
f_5	0.377	f_5	-0.290
		f_6	-0.051
		f_7	0.377
devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.319	Zweight	0.271
convio	-0.211	convio	-0.211

Scenario 7

Player Aero		Player Weights	
X, design vars		X, design vars	
S (ft^2)	1823	Ti (lbs)	37597
b (ft)	122.7	Wto (lbs)	218723
l (ft)	116.2		
s, state vars		s, state vars	
cdo_c	0.017	R_{fa}	0.310
cdo_{tl}	0.017	R_{fr}	0.290
d (ft)	14.64	U	0.473
Ld_l	13.9	R_f	1.07
Ld_t	10.7	PRI	164
Ld_c	19.0	q_L	0.089
V_{br} (ft/s)	709.6	q_{TO}	0.021
AR	8.26	s_L (ft)	4268
q_L	0.089	s_{TO} (ft)	6203
q_{TO}	0.021		
s_L (ft)	4268		
S_{TO} (ft)	6203		
constraint values (feasible ≥ 0.0)		constraint values (feasible ≥ 0.0)	
g_1	0.214	g_1	0.575
g_2	2.718	g_2	0.071
g_3	-0.211	g_3	2.718
g_4	0.051	g_4	-0.211
g_5	0.046	g_5	0.051
g_6	0.129	g_6	0.046
g_7	0.148		
goal values (ideal = 0.0)		goal values (ideal = 0.0)	
f_1	0.664	f_1	-0.393
f_2	-0.290	f_2	-0.055
f_3	-0.214	f_3	0.071
f_4	-0.051	f_4	0.664
f_5	0.379	f_5	-0.290
		f_6	-0.051
		f_7	0.379
devation function and total constraint violation		devation function and total constraint violation	
Zaero	0.320	Zweight	0.272
convio	-0.211	convio	-0.211

The deviation functions of each player are plotted in Figure C.11. The best solution (and the only feasible one is found in Scenario 4).

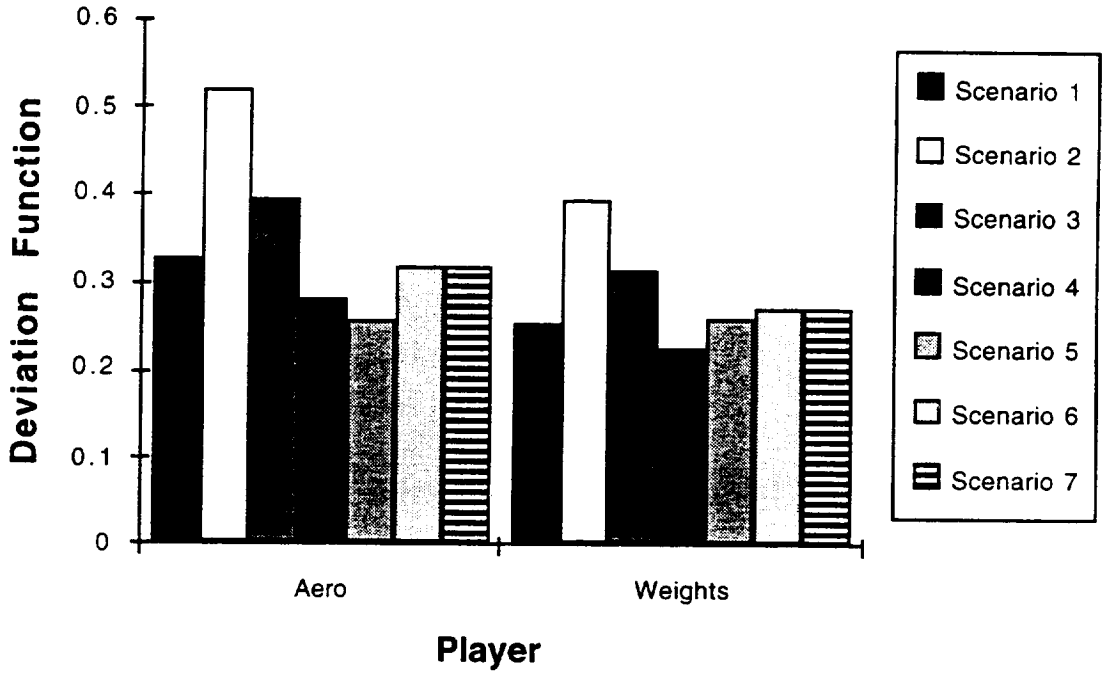


Figure C.11. Plot of Noncooperative Scenarios: Deviation Functions

RATIONAL REACTION SETS: Noncooperative Protocol

As described in Section 7.5.3, in order to solve the noncooperative formulation, certain assumptions are made in order to simplify the Rational Reaction Sets of each player. The simplified RRS's of each player are given in this section for each scenario. These RRS's only consist of 3 variables, Wing Area, S, Take-off Weight, W_{to}, and Installed Thrust, T_i.

Scenario 1

$$\{S = 1448 + 444.4*W_{to} - 175.8*T_i - 155.8*W_{to}*T_i + 186.5*W_{to}^2 + 97.04*T_i^2,$$
$$W_{to} = 216000 + 15040*S - 22370*S^2,$$
$$T_i = 39120 - 284.1*S - 4058*S^2\}$$

Scenario 2

$$\{S = 1570.77 + 527.41*W_{to} - 92.79*T_i - 155.8*W_{to}*T_i + 186.5*W_{to}^2 + 97.04*T_i^2,$$
$$W_{to} = 189035 - 780.4*S - 22370*S^2,$$
$$T_i = 60262.6 - 2145*S - 4058*S^2\}$$

Scenario 3

$$\{S = 1355.77 + 361.39*W_{to} - 258.81*T_i - 155.8*W_{to}*T_i + 186.5*W_{to}^2 + 97.04*T_i^2,$$
$$W_{to} = 197332 + 30860.4*S - 22370*S^2,$$
$$T_i = 30582.2 + 1576.8*S - 4058*S^2\}$$

Scenario 4

$$\{S = 1516.7 + 493.349*W_{to} - 126.851*T_i - 155.8*W_{to}*T_i + 186.5*W_{to}^2 + 97.04*T_i^2,$$
$$W_{to} = 223403 + 21877.4*S - 22370*S^2,$$
$$T_i = 36728.3 + 456.661*S - 4058*S^2\}$$

Scenario 5

$$\{S = 1510.96 + 489.529*W_{to} - 130.671*T_i - 155.8*W_{to}*T_i + 186.5*W_{to}^2 + 97.04*T_i^2,$$
$$W_{to} = 209958 + 14030.4*S - 22370*S^2,$$
$$T_i = 40984.9 + 625.614*S - 4058*S^2\}$$

Scenario 6

$$\{S = 1514.16 + 491.666*W_{to} - 128.534*T_i - 155.8*W_{to}*T_i + 186.5*W_{to}^2 + 97.04*T_i^2,$$
$$W_{to} = 219439 + 19386.7*S - 22370*S^2,$$
$$T_i = 37651.6 + 459.511*S - 4058*S^2\}$$

Scenario 7

$$\{S = 1514.04 + 491.582*W_{to} - 128.618*T_i - 155.8*W_{to}*T_i + 186.5*W_{to}^2 + 97.04*T_i^2,$$
$$W_{to} = 219585 + 19449.1*S - 22370*S^2,$$
$$T_i = 37622.8 + 449.045*S - 4058*S^2\}$$

REFERENCES

Collins English Dictionary, 1976.

Webster's Collegiate Dictionary, 1984.

ACSYNT Institute, 1992, "ACSYNT Overview and Installation Manual," Virginia Polytechnic Institute and State University.

Allen, J. K., Krishnamachari, R. S., Masetta, J., Pearce, D., Rigby, D. and Mistree, F., 1992, "Fuzzy Compromise: An Effective Way to Solve Hierarchical Design Problems," *Structural Optimization,* Vol. 4, pp. 21-43.

Allen, J. K., Simovich, G. and Mistree, F., 1989, "Selection Under Uncertain Conditions: A Marine Application," *Fourth International Symposium on Practical Design of Ships and Mobile Units,* Varna, Bulgaria, Bulgarian Ship Hydrodynamics Centre, pp. 80.1-80.8.

Andreasen, M. M., 1987, "Design Strategy," *Proceedings 1987 International Conference on Engineering Design,* Boston, MA, The American Society of Mechanical Engineers, pp. 171-178.

Arora, J. S. and Huang, M. W., 1994, "Methods for Optimization of Nonlinear Problems with Discrete Variables: A Review," *Structural Optimization,* Vol. 8, pp. 69-85.

Aubin, J. P., 1979, *Mathematical Methods of Game and Economic Theory*, North-Holland Publishing Company, Amsterdam.

Axelrod, R. M., 1984, *The Evolution of Cooperation*, Basic Books, New York.

Azarm, S. and Li, W. C., 1987, "Optimal Design Using a Two-Level Monotonicity-Based Decomposition Method," In *Advances in Design Automation*, S.S.Rao, ed., DE-vol 10-1, pp. 41-48.

Azarm, S. and Li, W. C., 1995, "Optimality and Constrained Derivatives in Two-Level Design Optimization," *Journal of Mechanical Design,* Vol. 112, No. 4, pp. 563-568.

Badhrinath, K. and Rao, J. R. J., 1995, "Illustration of Bilevel Models in Concurrent Design Modeling," *ASME Design Automation Conference, Boston, MA.*, DE-Vol. 82-2, pp. 189-195.

Balling, R. J., 1993, "How to Optimize Reinforced Concrete Systems," *ASCE Structures Congress XI,* Irvine, CA, pp. 1614-1619.

Balling, R. J. and Sobieski, J., 1994a, "An Algorithm for Solving the System-Level Problem in Multilevel Optimization," *5th AIAA/USAF/NASA/ISSMO Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Panama City, FL, pp. 794-809.

Balling, R. J. and Sobieski, J., 1994b, "Optimization of Coupled Systems: A Critical Overview of Approaches," *5th AIAA/USAF/NASA/ISSMO Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Panama City, FL, pp. 753-773.

Barkan, P. and Hinckley, C. M., 1993, "The Benefits and Limitations of Structured Design Methodologies," *ASME Manufacturing Review,* Vol. 6, No. 3, pp. 211-220.

Barthelemy, J.-F. and Sobieszczanski-Sobieski, J., 1983, "Optimum Sensitivity Derivatives of Objective Functions in Nonlinear Programming," *AIAA Journal,* Vol. 21, No. 6, pp. 913-915.

Bascaran, E., 1990, "A Conceptual Model for the Design of Thermal Systems: Concurrent Decisions in Designing for Concept," Ph.D. Dissertation, Department of Mechanical Engineering, University of Houston, Houston, Texas.

Bascaran, E., Bannerot, R. B. and Mistree, F., 1989, "Hierarchical Selection Decision Support Problems in Conceptual Design," *Engineering Optimization,* Vol. 14, pp. 207-238.

Batill, S. and Swift, R., 1993, "Preliminary Structural Design - Defining the Design Space," Report No. WL-TR-93-3004. Hessert Center for Aerospace Research, Department of Aerospace and Mechanical Engineering, University of Norte Dame.

Beltracchi, T., 1990, "A Decomposition Approach to Solving to Allup Trajectory Optimization Problem," *28th Aerospace Sciences Meeting,* Reno, Nevada, AIAA 90-0469, pp. 1-11.

Benhamou, S., 1994, "Spatial Memory and Searching Efficiency," *Animal Behavior,* Vol. 47, pp. 1423-1433.

Bertalanffy, L., 1968, *General Systems Theory*, George Braziller, New York, NY.

Bhattacharya, N., 1990, "A Comprehensive Computer Based Decision Support System for the Preliminary Design of Aircraft Tires," Master's Thesis, Department of Mechanical Engineering, University of Houston, Houston, Texas.

Bischof, C., Carle, A., Corlisss, G., Griewank, A. and Hovland, P., 1992, "ADIFOR-Generating Deivative Codes from Fortran Programs," *Scientific Programming,* Vol. 1, No. 1, pp. 1-29.

Bland, J. A. and Dawson, G. P., 1989a, "Application of Tabu Search to Quadratic Assignment Problems," Internal Report, Department of Mathematics, Statistics and Operational Research, Nottingham Polytechnic, UK.

Bland, J. A. and Dawson, G. P., 1989b, "Tabu Search Applied to Layout Optimisation," Internal Report, Department of Mathematics, Statistics and Operational Research, Nottingham Polytechnic, UK.

Bland, J. A. and Dawson, G. P., 1991, "Tabu Search and Design Optimization," *Computer Aided Design,* Vol. 23, No. 3, pp. 195-201.

Bloebaum, C. L., 1991, "Formal and Heuristic System Decomposition Methods in Multidisciplinary Synthesis," NASA Contractor Report 4413.

Bloebaum, C. L. and Chi, H.-W., 1994, "A Concurrent Decomposition Approach for Mixed Discrete/Continuous Variables," *AIAA/ASME/ASCE/AHS/ASC 35th Structures, Structural Dynamics, and Materials Conference,* Hilton Head, SC, AIAA 94-1363.

Bloebaum, C. L., Hajela, P. and Sobieski, J., 1992, "Non-Hierarchic System Decomposition in Structural Optimization," *Engineering Optimization,* Vol. 19, pp. 171-186.

Bond, A. H. and Ricci, R. J., 1992, "Cooperation in Aircraft Design," *Research in Engineering Design,* Vol. 4, pp. 115-130.

Box, G., 1988, "Signal-to-Noise Ratios, Performance Criteria, and Transformations," *Technometrics,* Vol. 30, No. 1, pp. 1-18.

Box, G., Hunter, W. and Hunter, J., 1978, *Statistics for Experimenters*, Wiley, Inc., New York, NY.

Box, G. E. P. and Draper, N. R., 1987, *Empirical Model-Building and Response Surfaces*, John Wiley & Sons, New York, NY.

Bras, B., Smith, W. F. and Mistree, F., 1990, "The Development of a Design Guidance System for the Early Stages of Design," *CFD and CAD in Ship Design*, Elsevier Science Publishers B.V., Wageningen, The Netherlands, pp. 221-231.

Bras, B. A., 1992, "Foundations for Designing Decision-Based Design Processes," Ph.D. Dissertation, School of Mechanical Engineering, University of Houston, Houston, TX.

Bras, B. A. and Mistree, F., 1991, "Designing Design Processes in Decision-Based Concurrent Engineering," *SAE Transactions, Journal of Materials & Manufacturing*, pp. 451-458.

Bras, B. A. and Mistree, F., 1993, "Robust Design using Compromise Decision Support Problems," *Engineering Optimization,* Vol. 21, No. 3, pp. 213-239.

Brassard, M., 1989, *The Memory Jogger Plus*, GOAL/QPC, Methuen, MA.

Brown, D. C., 1985, "Capturing Mechanical Design Knowledge," *ASME Computers in Mechanical Engineering,* Vol. 2, pp. 121-129.

Brown, D. C. and Chandrasekaran, B., 1986, "Expert Systems for a Class of Mechanical Design Activity," *Knowledge Engineering in Computer-Aided Design*, North-Holland, pp. 259-283.

Cartuyvels, R. and Dupas, L. H., 1993, "NORMAN/DEBORA: A Powerful CAD-Integrating Automatic Sequencing System Including Advanced DOE/RSM Techniques for Various Engineering Optimization Problems," *JSPE-IFIP WG 5.3 DIISM'93 Workshop (Design of Information Infrastructure Systems for Manufacturing),* Tokyo, Japan.

Chen, W., J.K. Allen, K-L Tsui , and F. Mistree, 1995a, "Integration of the Response Surface Methodology with the Compromise Decision Support Problem in Developing a General Robust Design Procedure," *ASME Design Engineering Technical Conferences,* Azarm, S., Dutta, D., Eschenauer, H., Gilmore, B., McCarthy, M. and Yoshimura, M., eds., Boston, MA, Published in *Advances in Design Automation*, pp. 485-492.

Chen, W., 1995b, "A Robust Concept Exploration Method for Configuring Complex Systems," Ph.D. Dissertation, G.W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA.

Chen, W., J.K. Allen, D., Mavris, and F. Mistree, 1996a, "A Concept Exploration Method for Determining Robust Top-Level Specifications," *Engineering Optimization,* in press.

Chen, W., J. K. Allen, and F. Mistree, 1996b, "System Configuration: Concurrent Subsystem Embodiment and System Syntheses," *Journal of Mechanical Design,* in press.

Chen, W., Allen, J. K. and Mistree, F., 1994a, "Robust Concurrent Concept Selection and System Synthesis," *ASME Design Engineering Technical Conferences,* Gilmore, B. J., Hoeltzel, D., Dutta, D. and Eschenauer, H., eds., Minneapolis, MN, Published in *Advances in Design Automation*, New York: ASME, DE-Vol. 69-1, pp. 141-149.

Chen, W., Meher-Homji, C. and Mistree, F., 1994b, "Compromise: An Effective Approach for Condition-Based Maintenance of Gas Turbines," *Engineering Optimization,* Vol. 22, No. 3, pp. 185-202.

Courtois, P.-J., 1985, "On Time and Space Decomposition of Complex Structures," *Communications of the ACM,* Vol. 28, No. 6, pp. 590-603.

Cramer, E. J., Dennis, J., J.E., Frank, P. D., Lewis, R. M. and Shubin, G. R., 1994, "Problem Formulation for Multidisciplinary Optimization," *SIAM Journal of Optimization,* Vol. 4, No. 4, pp. 754-776.

Cross, N., 1989, *Engineering Design Methods*, John Wiley & Sons, Chichester.

Cross, N., 1993, "Science and Design Methodology: A Review," *Research in Engineering Design,* Vol. 5, pp. 63-69.

Darwin, C., *Descent of Man.*

De Boer, S. J., 1989, *Decision Methods and Techniques in Methodical Engineering Design*, Academisch Boeken Centrum, De Lier, The Netherlands.

De Bono, E., 1985, *Conflicts: A Better Way to Resolve Them*, Penguin Books, Great Britain.

Dertouzos, M.L. et. al. and MIT Commission on Industrial Productivity, 1989, *Made in America: Regaining the Productive Edge,* MIT Press, Cambridge, MA.

Dhingra, A. K. and Rao, S. S., 1995, "A Cooperative Fuzzy Game Theoretic Approach to Multiple Objective Design Optimization," *European Journal of Operational Research,* Vol. 83, pp. 547-567.

Diaz, A., "A Strategy for Optimal Design of Hierarchical Systems Using Fuzzy Sets," College of Engineering, Michigan State University, Internal Report.

Dixon, J. R., Duffey, M. R., Irani, R. K., Meunier, K. L. and Orelup, M. F., 1988, "A Proposed Taxonomy of Mechanical Design Problems," *Proceedings ASME Computers in Engineering Conference,* San Francisco, California, pp. 41-46.

Dresher, M., 1981, *Games of Strategy*, Dover Publication, New York.

Duffey, M., Koenig, P., Rosen, D. and Singh, P., 1996, "Design Infrastructure in Shipbuilding and Other Heavy Industries: A Benchmarking Study," Final Report Draft, US Navy Project N0002488C4216.

Engelund, W., Stanley, D., Lepsch, R., McMillian, M. and Unal, R., 1993, "Aerodynamic Configuration Design Using Response Surface Methodology Analysis," *AIAA Aircraft Design, Systems and Operations Meeting,* Monterey, CA, AIAA 93-3967.

Falk, J. E. and Liu, J., 1993, "On Bilevel Programming, Part I: General Nonlinear Cases," The George Washington University, Washington, D.C.

Fang, H. and Azarm, S., 1994, "Multidisciplinary Design Optimization in Precast Concrete Wall Panels," *5th AIAA/USAF/NASA/ISSMO Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Panama City, FL, pp. 454-464.

Feng, X., Mukai, H. and Brown, R. H., 1990, "New Decomposition and Convexification Algorithm for Nonconvex Large-Scale Primal-Dual Optimization," *Journal of Optimization Theory and Applications,* Vol. 67, No. 2, pp. 279-296.

Finger, S. and Dixon, J. R., 1989a, "A Review of Research in Mechanical Engineering Design. Part 1: Descriptive, Prescriptive, and Computer-Based Models of Design Processes," *Research in Engineering Design,* Vol. 1, pp. 51-67.

Finger, S. and Dixon, J. R., 1989b, "A Review of Research in Mechanical Engineering Design. Part 2: Representations, Analysis, and Design for the Life Cycle," *Research in Engineering Design,* Vol. 1, pp. 121-137.

Floudas, C. A. and Visweswaran, V., 1990, "Global Optimization Algorithm for Certain Classes of Nonconvex NLPs," *Computer and Chemical Engineering,* Vol. 14, No. 12, pp. 1397-1417.

Ford, J. and Bloebaum, C., 1993, "A Decomposition Method for Design of Mixed Discrete/Continuous Systems," *ASME Advances in Design Automation,* Albuquerque, NM, DE-Vol. 65-2, pp. 367-376.

Fu, J., Fenton, R. G. and Cleghorn, W., 1991, "A Mixed Integer-Discrete-Continuous Programming Method and Its Application to Engineering Design Optimization," *Engineering Optimization,* Vol. 17, pp. 263-280.

Fudenberg, D. and Tirole, J., 1991, *Game Theory*, MIT Press, Cambridge, MA.

Fulton, R. E., Pin-Yeh, C. and Richter, K. J., 1989, "Managing Engineering Design Information," Unclassified, Institute for Defense Analyses, Alexandria, Virginia.

Gleick, J., 1986, "Prisoner's Dilemma Has Unexpected Applications," *New York Times*, June 17, pp. C1, C9.

Glover, F., 1986, "Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers and Operations Research,* Vol. 13, No. 5, pp. 533-549.

Glover, F., 1989a, "Tabu Search, Part I," *ORSA Journal on Computing,* Vol. 1, No. 3, pp. 190-206.

Glover, F., 1989b, "Tabu Search, Part II," *ORSA Journal on Computing*, Vol. 2, No. 1, pp. 4-32.

Glover, F. and Greenberg, H., 1989, "New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence," *European Journal of Operational Research*, Vol. 39, pp. 119-130.

Hajela, P., 1990, "Multiobjective Optimum Design in Mixed Discrete Design Variable Problems," *AIAA Journal*, Vol. 28, pp. 670-675.

Hajela, P., 1995, "Genetic Algorithms in Multidisciplinary Rotor Blade Design," *36th AIAA/ASME/ASCE/AHS/ASC Structural Dynamics and Materials Conference*, New Orleans, LA, AIAA-95-1144, pp. 2187-2197.

Hajela, P. and Shih, C.-J., 1989, "Genetic Search-An Approach to the Nonconvex Optimization Problem," *30th AIAA/ASME/ASCE/ASHS Structures, Structural Dynamics, and Materials Conference*, Mobile, AL, pp. 165-175.

Hale, M. A., Craig, J. I., Mistree, F. and Schrage, D. P., 1995, "On the Development of a Computing Infrastructure that Facilitates IPPD from a Decision-Based Design Perspective," *1st AIAA Aircraft Engineering, Technology, and Operations Congress*, Los Angeles, CA, AIAA-95-3880.

Hale, M. A., Craig, J. I., Mistree, F. and Schrage, D. P., 1996, "DREAMS & IMAGE: A Model and Computer Implementation for Concurrent, Life-Cycle Design of Complex Systems," *Concurrent Engineering: Research and Applications*, accepted, January 1996.

Hazelrigg, G., 1996, *Systems Engineering: An Approach to Information-Based Design*, Prentice-Hall, Upper Saddle River, NJ.

Heiberger, R. M., 1989, *Computation for the Analysis of Designed Experiments*, John Wiley & Sons, Inc., New York, NY.

Hofstadter, D. R., 1985, *Metamagical Themas*, Basic Books, New York.

Hsu, Y.-H., Sun, T.-L. and Leu, L.-H., 1995, "A Two-Stage Sequential Approximation Method for Non-linear Discrete Variable Optimization," *ASME Design Engineering Technical Conference*, Azarm, S., Dutta, D., Eschenauer, H., Gilmore, B., McCarthy, M. and Yoshimura, M., eds., Boston, MA, pp. 197-202.

Hubka, V., 1982, *Principles of Engineering Design*, Butterworth & Co. (Publishers) Ltd., London.

Hubka, V., Andreasen, M. M. and Eder, W. E., 1988, *Practical Studies in Systematic Design*, Butterworth & Co. (Publishers) Ltd., London.

Hubka, V. and Schregenberger, J., 1987, "Paths Towards Design Science," *Proceedings 1987 International Conference on Engineering Design*, Boston, MA, ASME, pp. 3-14.

Huntingford, F., 1984, *The Study of Animal Behavior*, Chapman and Hall, New York, NY.

Ignizio, J. P., 1983, "Generalized Goal Programming: An Overview," *Computers and Operations Research*, Vol. 5, No. 3, pp. 179-197.

Ignizio, J. P., 1985a, *Introduction to Linear Goal Programming*, Sage University Papers, Beverly Hills, California.

Ignizio, J. P., 1985b, "Multiobjective Mathematical Programming via the MULTIPLEX Model and Algorithm," *European Journal of Operational Research*, Vol. 22, pp. 338-346.

Kamal, S. Z., 1990, "The Development of Heuristic Decision Support Problems for Adaptive Design," Ph.D. Dissertation, Department of Mechanical Engineering, University of Houston, Houston, Texas.

Kamal, S. Z., Garson, J. and Mistree, F., 1992, "Heuristic Decision Support Problems: Integrating Heuristic Search and Expert Systems for the Design of Continuous Manufactured Products," *Artificial Intelligence in Design '92*, J. S. Gero and F. Sudweeks, eds., Kluwer Academic Publishers, Boston, Massachusetts, pp. 883-902.

Kamal, S. Z., Karandikar, H. M., Mistree, F. and Muster, D., 1987, "Knowledge Representation for Discipline-Independent Decision Making," *Expert Systems in Computer-Aided Design*, J.S. Gero, ed., Elsevier Science Publishers B.V., Amsterdam, pp. 289-321.

Kannan, B. K. and Kramer, S. N., 1994, "An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and its Applications to Mechanical Design," *Journal of Mechanical Design,* Vol. 116, pp. 405-411.

Karandikar, H. M., 1989, "Hierarchical Decision Making for the Integration of Information from Design and Manufacturing Processes in Concurrent Engineering," Ph.D. Dissertation, Department of Mechanical Engineering, University of Houston, Houston, Texas.

Karandikar, H. M., Fuchs, W. J., Mistree, F. and Eschenauer, H., 1990, "Compromise: An Effective Approach for Designing Composite Conical Shell Structures," *ASME Journal of Mechanical Design*, Vol. 112, No. 2, pp. 362-368.

Karandikar, H. M. and Mistree, F., 1992a, "An Approach for Concurrent and Integrated Material Selection and Dimensional Synthesis," *ASME Journal of Mechanical Design*, Vol. 114, No. 4, pp. 633-641.

Karandikar, H. M. and Mistree, F., 1992b, "Designing a Composite Material Pressure Vessel for Manufacture: A Case Study for Concurrent Engineering," *Engineering Optimization,* Vol. 18, No. 4, pp. 235-262.

Karandikar, H. M. and Mistree, F., 1992c, "Tailoring Composite Materials Through Optimal Selection of Their Constituents," *ASME Journal of Mechanical Design*, Vol. 114, No. 3, pp. 451-458.

Karandikar, H. M. and Mistree, F., 1993, "Modeling Concurrency in the Design of Composite Structures," *Structural Optimization: Status and Promise*, M. P. Kamat, ed., AIAA, Washington, D.C., pp. 769-806.

Khuri, A. and Cornell, J. A., 1987, *Response Surfaces: Design and Analysis*, Marcel Dekker, New York, NY.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P., 1983, "Optimization by Simulated Annealing," *Science,* Vol. 220, pp. 671-680.

Korngold, J., Renaud, J., Gabriele, G. and Kott, G., 1992, "Application of Multidisciplinary Design Optimization to Electronic Package Design," *4th AIAA/USAF/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Cleveland, Ohio, pp. 130-140.

Kroo, I., Altus, S., Braun, R., Gage, P. and Sobieksi, I., 1994, "Multidisciplinary Optimization Methods for Aircraft Preliminary Design," *5th AIAA/USAF/NASA/ISSMO Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Panama City, FL, pp. 697-707.

Kuppuraju, N., Ganesan, S., Mistree, F. and Sobieski, J. S., 1985a, "Hierarchical Decision Making in System Design," *Engineering Optimization,* Vol. 8, pp. 223-252.

Kuppuraju, N., Ittimakin, P. and Mistree, F., 1985b, "Design through Selection ... A Method that Works," *Design Studies,* Vol. 6, No. 2, pp. 91-106.

Kusiak, A. and Larson, N., 1995, "Decomposition and Representation Methods in Design," *Journal of Mechanical Design,* Vol. 117, pp. 17-24.

Kusiak, A. and Wang, J., 1993, "Decomposition of the Design Process," *ASME Journal of Mechanical Design,* Vol. 115, pp. 687-695.

Lewis, K., Lucas, T. and Mistree, F., 1994, "A Decision Based Approach to Developing Ranged Top-Level Aircraft Specifications: A Conceptual Exposition," *5th AIAA/USAF/NASA/ISSMO Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Panama City, FL, pp. 465-481.

Lewis, K. and Mistree, F., 1995, "On Developing a Taxonomy for Multidisciplinary Design Optimization: A Decision-Based Approach," *The First World Congress of Structural and Multidisciplinary Optimization*, Olhoff, N. and Rozvany, G.I.N., eds., Goslar, Germany, Pergamon Press, pp. 811-818.

Lewis, K., Mistree, F., 1996a, "Aircraft Design: A Game-Theoretic Approach," *Sixth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* Bellevue, Washington, (in press).

Lewis, K. and Mistree, F., 1996b, "Foraging-directed Adaptive Linear Programming: An Algorithm for Solving Nonlinear Mixed Discrete/Continuous Design Problems," *ASME Design Automation Conference,* (in press).

Lin, S.-S., Zhang, C. and Wang, H.-P., 1995, "On Mixed-Discrete Nonlinear Optimization Problems: A Comparative Study," *Engineering Optimization,* Vol. 23, pp. 287-300.

Livne, E., Schmit, L. A. and Friedmann, P. P., 1993, "Integrated Structure/Control/Aerodynamic Synthesis of Actively Controlled Composite Wings," *Journal of Aircraft,* Vol. 30, No. 3, pp. 387-394.

Loftin, L. K., 1980, "Subsonic Aircraft: Evolution and Matching of Size to Performance," NASA Reference Publication 1060.

Loh, H. T. and Papalambros, P. Y., 1991, "A Sequential Linearization Approach for Solving Mixed-Discrete Nonlinear Design Optimization Problems," *ASME Journal of Mechanical Design,* Vol. 113, pp. 325-334.

Loridan, P. and Morgan, J., 1988, "Approximate Solutions for Two-Level Optimization Problems," *International Series of Numerical Mathematics,* Vol. 84, pp. 181-196.

Loridan, P. and Morgan, J., 1989, "A Theoretical Approximation Scheme for Stackelberg Problems," *Journal of Optimization Thoery and Applications,* Vol. 61, No. 1, pp. 95-110.

Lucchetti, R., Mignanego, F. and Pieri, G., 1987, "Existence of Equilibrium Points in Stackleberg Games with Constraints," *Optimization,* Vol. 18, No. 6, pp. 857-866.

Luce, R. D. and Raiffa, H., 1957, *Games and Decisions*, John Wiley, New York.

Maddalon, D. V., 1978, "Estimating Airline Operating Costs," NASA Technical Memorandum 78694.

Malone, B. and Mason, W. H., 1991, "Multidisciplinary Optimization in Aircraft Design Using Analytic Technology Models," *AIAA/AHS/ASEE Aircraft Design Systems and Operations Meeting,* Baltimore, MD, AIAA-91-3187.

Marshek, K. and Kannapan, S., 1987, "Design Methodologies: A New Perspective on Approaches and Tasks," Technical Report #201, The University of Texas at Austin, Austin, TX.

Matsumoto, M., Abe, J. and Yoshimura, M., 1993, "A Multiobjective Optimization Strategy with Priority Ranking of the Design Objectives," *Journal of Mechanical Design,* Vol. 115, pp. 784-792.

McCullers, L. A., 1993, "FLight OPtimization System, User's Guide," NASA Langley Research Center, Hampton, VA.

McCulley, C. and Bloebaum, C., 1994, "Optimal Sequencing for Complex Engineering Systems Using Genetic Algorithms," *5th AIAA/USAF/NASA/ISSMOSymposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Panama City, FL, pp. 718-730.

Menzel, C. R., 1991, "Cognitive Aspects of Foraging in Japanese Monkeys," *Animal Behavior,* Vol. 41, pp. 397-402.

Messac, A. and Hattis, P., 1995, "High Speed Civil Transport (HSCT) Plane Design using Physical Programming," *36th AIAA/ASME/ASCE/AHS/ASC Structural Dynamics and Materials Conference,* New Orleans, LA, AIAA -95-1401.

Mesterton-Gibbons, M., 1992, *An Introduction to Game-Theoretic Modeling*, Addison-Wesley Publishing Company, Redwood City, CA.

Michelena, N., Jiang, T. and Papalambros, P., 1995, "Decomposition of Simultaneous Analysis and Design Models," *First World Congress of Structural and Multidisciplinary Optimization,* Olhoff, N. and Rozvany, G., eds., Goslar, Germany, Pergamon Press, pp. 845-850.

Mills, J. J., 1993, "A Taxonomy of the Product Realization Process Environment," *Research in Engineering Design,* Vol. 4, pp. 203-213.

Mistree, F., Hughes, O. F. and Bras, B. A., 1993a, "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm," *Structural Optimization: Status and Promise*, M. P. Kamat, ed., AIAA, Washington, D.C., pp. 247-286.

Mistree, F., Hughes, O. F. and Phuoc, H. B., 1981, "An Optimization Method for the Design of Large, Highly Constrained, Complex Systems," *Engineering Optimization,* Vol. 5, No. 3, pp. 141-144.

Mistree, F., Lautenschlager, U. and Erikstad, S. O., 1993b, "Simulation Reduction using the Taguchi Method," NASA Contractor Report No. CR 93-4542.

Mistree, F., Lewis, K. and Stonis, L., 1994, "Selection in the Design of Aircraft," *5th AIAA/USAF/NASA/ISSMO Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Panama City, FL, pp. 1153-1166.

Mistree, F., Marinopoulos, S., Jackson, D. and Shupe, J. A., 1988, "The Design of Aircraft using the Decision Support Problem Technique," NASA Contractor Report, CR 88-4134.

Mistree, F., Muster, D., Shupe, J. A. and Allen, J. K., 1989, "A Decision-Based Perspective for the Design of Methods for Systems Design," *Recent Experiences in Multidisciplinary Analysis and Optimization,* J. Sobieszczanski-Sobieski, ed., Hampton, Virginia, NASA CP 3031.

Mistree, F., Muster, D., Srinivasan, S. and Mudali, S., 1990a, "Design of Linkages: A Conceptual Exercise in Designing for Concept," *Mechanism and Machine Theory,* Vol. 25, No. 3, pp. 273-286.

Mistree, F., Patel, B. and Vadde, S., 1994, "On Modeling Multiple Objectives and Multi-Level Decisions in Concurrent Design," Gilmore, B.J., Hoeltzel, D., Dutta, D. and Eschenauer, H., eds., *ASME Advances in Design Automation,* Minneapolis, Minnesota, ASME, DE-Vol. 69-2, pp. 151-161.

Mistree, F., Smith, W. F., Bras, B., Allen, J. K. and Muster, D., 1990b, "Decision-Based Design: A Contemporary Paradigm for Ship Design," *Transactions, Society of Naval Architects and Marine Engineers*, Jersey City, New Jersey, pp. 565-597.

Mistree, F., Smith, W. F. and Bras, B. A., 1993c, "A Decision-Based Approach to Concurrent Engineering," *Handbook of Concurrent Engineering*, H. R. Paresai and W. Sullivan, eds., Chapman & Hall, New York, New York, pp. 451-458.

Montgomery, D., 1991, *Design and Analysis of Experiments*, John Wiley & Sons, New York, NY.

Muster, D. and Mistree, F., 1988, "The Decision Support Problem Technique in Engineering Design," *International Journal of Applied Engineering Education,* Vol. 4, No. 1, pp. 23-33.

Muster, D. and Mistree, F., 1989, "Engineering Design as it Moves from an Art towards a Science: Its Impact on the Education Process," *The International Journal of Applied Engineering Education,* Vol. 5, No. 2, pp. 239-246.

Myerson, R. B., 1991, *Game Theory: Analysis of Conflict*, Harvard University Press, Cambridge, Massachusetts.

Nash, J. F., 1951, "Non-Cooperative Games," *Ann. Mathematics,* Vol. 54, No. 2.

Nicolai, L. M., 1984, *Fundamentals of Aircraft Design*, METS Inc., San Jose, CA.

Nowak, M. A., May, R. M. and Sigmund, K., 1995, "The Arithmetics of Mutual Help," *Scientific American*, June Issue, pp. 76-81.

Olds, J., 1992, "The Suitability of Selected Multidisciplinary Design Techniques to Conceptual Aerospace Vehicle Design," *4th AIAA/USAF/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*, Cleveland, Ohio, pp. 820-844.

Olds, J., 1994, "Results of Rocket-Based Combined-Cycle SSTO Design Using Parametric MDO Methods," *SAE Aerospace Atlantic Conference*, Dayton, OH, SAE Paper 94-1165.

Olds, J. and Walberg, G., 1993, "Multidisciplinary Design of a Rocket-Based Combined-Cycle SSTO Launch Vehicle Using Taguchi Methods," *AIAA/AHS/ASEE Aerospace Design Conference*, AIAA 93-1096.

Ostrofsky, B., 1977, *Design, Planning and Development Methodology*, Prentice-Hall, Englewood Cliffs, New Jersey.

Owen, G., 1995, *Game Theory*, Academic Press, San Diego, CA.

Pahl, G. and Beitz, W., 1984, *Engineering Design*, The Design Council/Springer-Verlag, London/Berlin.

Pakala, R. and Rao, J. R. J., 1996, "Study of Concurrent Decision-Making Protocols in the Design of a Metal Cutting Tool Using Monotonicity Arugments," *Engineering Optimization*, in press.

Pan, J. and Diaz, A. R., 1990, "Some Results in Optmization of Nonhierarchic Systems," *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 112, No. 3, pp. 399-405.

Papalambros, P. Y., 1995, "Optimal Design of Mechanical Engineering Systems," *Journal of Mechanical Design, Special 50th Anniversary Design Issue*, Vol. 117, pp. 55-62.

Peplinski, J. D., Koch, P. N., Allen, J. K. and Mistree, F., 1996a, "Design for Manufacture at the Function Level of Abstraction: A Conceptual Exposition," *Flexible Automation and Intelligent Manufacturing Conference,* Atlanta, GA, Begell House, Inc., pp. 488-498.

Peplinski, J. D., Koch, P. N., Allen, J. K. and Mistree, F., 1996b, "Design Using Available Assets: A Paradigm Shift in Design for Manufacture," *Concurrent Engineering Research and Applications,* in press.

Rao, J. R. J., Badrinath, K., Pakala, R. and Mistree, F., 1996, "A Study of Optimal Design Under Conflict Using Models of Multi-Player Games," *Engineering Optimization,* in press.

Rao, J. R. J. and Chidambaram, B., 1993, "Parametric Deformations and Model Optimality in Concurrent Design," *Advances in Design Automation,* ASME, DE-Vol. 65-2, pp. 477-486.

Rao, J. R. J. and Mistree, F. M., 1995, "Recent Applications of Bilevel Models in Multidiciplinary Optimization," *First World Congress of Structural and Multidisciplinary Optimization,* Olhoff, N. and Rozvany, G., eds., Goslar, Germany, Pergamon Press, pp. 17-24.

Rao, S. S., 1987, "Game Theory Approach for Multiobjective Structural Optimization," *Computers & Structures,* Vol. 25, No. 1, pp. 119-127.

Rao, S. S. and Freiheit, T. I., 1991, "A Modified Game Theory Approach to Multiobjective Optimization," *Journal of Mechanical Design,* Vol. 113, pp. 286-291.

Raymer, D. P., 1989, *Aircraft Design: A Conceptual Approach,* AIAA, Washington, D.C.

Reklaitis, G. V., Ravindran, A. and Ragsdell, K. M., 1983, *Engineering Optimization: Methods and Applications*, John Wiley and Sons, New York, NY.

Renaud, J., 1993, "Second Order Based Multidisciplinary Design Optimization Algorithm Development," *Design Automation Conference,* Gilmore, B., Hoeltzel, D., Azarm, S. and Eshenauer, H., eds., Albuquerque, New Mexico, ASME, Vol. 65, No. 2, pp. 347-357.

Renaud, J. E., 1992, "Sequential Approximation in Non-Hierarchic System Decomposition and Optimization: A Multidisciplinary Design Tool," Doctoral Dissertation, Rensselear Polytechnic Institute, Troy, NY.

Renaud, J. E. and Gabriele, G. A., 1991, "Sequential Global Approximation in Non-Hierarchic System Decomposition and Optimization," *Advances in Design Automation, ASME,* Vol. 1, pp. 191-200.

Renaud, J. E. and Gabriele, G. A., 1993, "Improved Coordination in Nonhierarchic System Optimization," *AIAA Journal,* Vol. 31, No. 12, pp. 2367-2373.

Renaud, J. E. and Gabriele, G. A., 1994, "Approximation in Non-Hierarchic System Optimization," *AIAA Journal,* Vol. 32, No. 1, pp. 198-205.

Renaud, J. E., Sellar, R. S., Batill, S. M. and Kar, P., 1994, "Design Driven Coordination Procedure for Concurrent Subspace Optimization in MDO," *AIAA/ASME/ASCE/AHS/ASC 35th Structures, Structural Dynamics, and Materials Conference,* Hilton Head, SC, AIAA 94-1482.

Rogan, J. E. and Cralley, W. E., 1990, "Meta-Design -- An Approach to the Development of Design Methodologies," IDA Paper No. P-2152, Institute for Defense Analyses, Alexandria, Virginia.

Rogers, J. L., 1989, "A Knowledge-Based Tool for Multilevel Decomposition of a Complex Design Problem," NASA Technical Paper No. 2903.

Rohl, P. and Schrage, D., 1992, "Preliminary Wing Design of a High Speed Civil Transport Aircraft by Multilevel Decomposition Techniques," *4th AIAA/USAF/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Cleveland, Ohio, pp. 244-250.

Saaty, T. L., 1980, *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*, McGraw-Hill, New York.

Sandgren, E., 1990, "Nonlinear Integer and Discrete Programming in Mechanical Design Optimization," *Journal of Mechanical Design,* Vol. 112, pp. 223-229.

Schrage, D. P., 1992, "Concurrent Design: A Case Study," *Concurrent Engineering-Automation, Tools, and Techniques*, John Wiley & Sons, New York, NY, pp. 535-581.

Schrage, D. P. and Gordon, M., 1992, "Management Issues and Techniques in Concurrent Engineering," *AIAA Aircraft Design Systems,* Hilton Head, SC.

Sellar, R. S., Batill, S. M. and Renaud, J. E., 1994, "Optimization of Mixed Discrete/Continuous Design Variable Systems Using Neural Networks," *5th Air Force/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* Panama City, FL, AIAA-94-4348.

Sen, P. and Yang, J.-B., 1993, "A Multiple Criteria Decision Support Environment for Engineering Design," *International Conference on Engineering Design,* The Hague, pp. 465-472.

Shimuzu, K., 1985, "Optimality Conditions and Algorithms for Parameter Design Problems with Two-Level Structure," *IEEE Transactions on Automatic Control,* Vol. AC-30, No. 10, pp. 983-996.

Shimuzu, K. and Aiyoshi, E., 1981, "A New Computational Method for Stackleberg and Min-Max Problems by use of Penalty Method," *IEEE Transactions on Automatic Control,* Vol. AC-26, No. 2, pp. 460-466.

Shupe, J. A., 1988, "Decision-Based Design: Taxonomy and Implementation," Ph.D. Dissertation, Department of Mechanical Engineering, University of Houston, Houston, Texas.

Simaan, M. and Cruz, J. B., 1973, "On the Stackelberg Strategy in Nonzero-Sum Games," *Journal of Optimization Theory and Applications,* Vol. 11, No. 5, pp. 533-555.

Simon, H. A., 1982, *The Sciences of the Artificial*, The MIT Press, Cambridge, Mass.

Snavely, G. L., Pomrehn, L. P. and Papalambros, P. Y., 1989, "Toward a Vocabulary for Classifying Research in Mechanical Design Automation," Unpublished Report, General Motors Technical Center, Warren, Michigan.

Sobieszczanski-Sobieski, J., 1988, "Optimization by Decomposition: A Step from Hierarchic to Non-hierarchic Systems," *Second NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization,* NASA CP 3031.

Sobieszczanski-Sobieski, J., 1993, "Multidisciplinary Optimization: An Emerging New Engineering Discipline," *World Congress on Optimal Design of Structural Systems,* Rio de Janeiro, Brazil, Kluwer Publishers, in press, 1996.

Sobieszczanski-Sobieski, J., Berthelemy, J.-F. M. and Giles, G. L., 1984, "Aerospace Engineering Design by Systematic Decomposition and Multilevel Optimization," *14th Congress of the International Council of the Aeronautical Sciences,* Toulouse, France, pp. 828-840.

Sobieszczanski-Sobieski, J. and Chopra, I., 1991, "Multidisciplinary Optimization of Aeronautical Systems," *Journal of Aircraft, Special Edition of Multidisciplinary Optimization, part 2*, Vol. 28, No. 1, pp. 977-978.

Sobieszczanski-Sobieski, J. and Tulinius, J., 1991, "MDO Can Help Resolve the Designer's Dilemma," *Aerospace America,* Vol. 29, No. 9, pp. 32-35.

Stadler, W., 1988, *Multicriteria Optimization in Engineering & in the Sciences*, Plenum Press, New York.

Stanley, D., Engelund, W., Lepsch, R., McMillian, M., Unal, R., Braun, R. and Powell, R., 1994, "Multidisciplinary Design and Optimization of a Rocket-Powered Single-Stage Vehicle," *Fifth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* Panama City, FL, AIAA 94-4340.

Stanley, D., Unal, R. and Joyner, R., 1992b, "Application of Taguchi Methods to Propulsion System Optimization for SSTO Vehicles," *Journal of Spacecraft and Rockets,* Vol. 29, No. 4, pp. 453-459.

Stewart, R. A. and Griffith, R. E., 1961, "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," *Management Science,* Vol. 7, pp. 359-392.

Styblinski, M. A. and Tang, T. S., 1990, "Experiments in Nonconvex Optimization. Stochastic Approximation with Function Smoothing and Simulated Annealing," *Neural Networks,* Vol. 3, No. 4, pp. 467-483.

Suh, N. P., 1990, *Principles of Design*, Oxford University Press, Oxford, U.K.

Taguchi, G., 1987, *System of Experimental Design: Engineering Methods to Optimize Quality and Minimize Cost*, UNIPUB/Kraus International Publications, Dearborn, Michigan.

Takala, T., 1987, "Theoretical Framework for Computer Aided Innovative Design," *Design Theory for Computer Aided Design*, H. Yoshikawa and E. Warman, eds., North-Holland Publishers, Amsterdam, pp. 323-338.

Thach, P. T. and Konno, H., 1993, "Generalized Dantzig-Wolfe Decomposition Principle for a Class of Nonconvex Programming Problems," *Mathematical Programming, Series B,* Vol. 62, No. 2, pp. 239-260.

Todd, I. A. and Kacelnik, A., 1993, "Psychological Mechanisms abnd the Marginal Value Theorem: Dynamics of Scalar Memory for Travel Time," *Animal Behavior,* Vol. 46, pp. 765-775.

Tomiyama, T. and Yoshikawa, H., 1987, "Extended General Design Theory," *Design Theory for Computer Aided Design*, North-Holland Publishers, Amsterdam, pp. 95-130.

Townsend, J. C., Weston, R. P. and Eidson, T. M., 1994, "An Overview of the Framework for Interdisciplinary Design Optimization (FIDO) Project," NASA Langley Research Center, NASA TM 109058.

Ullman, G., 1992, "A Taxonomy for Mechanical Design," *Research in Engineering Design,* Vol. 3, pp. 179-189.

Unal, R. and Stanley, D., 1992, "The Role of Statistically Designed Experiments in Conceptual Aerospace Vehicle Design," *1992 International Engineering Managment Conference, IEEE Engineering Management Society/ASEM,* pp. 211-214.

Unal, R., Stanley, D. O., Engelund, W. and Lepsch, R., 1994, "Design for Quality Using Response Surface Methods: An Alternative to Taguchi's Parameter Design Approach," *Engineering Management Journal,* Vol. 6, No. 3, pp. 40-48.

Vadde, S., 1995, "Modeling Multiple Objectives and Multilevel Decisions in Concurrent Design of Engineering Systems," M.S. Thesis, Georgia Institute of Technology, Atlanta, GA.

Vadde, S., Allen, J. K. and Mistree, F., 1992, "Catalog Design: Design using Available Assets," *Advances in Design Automation*, Hoeltzel, D. A., ed., ASME, New York, pp. 345-354.

Vaidyanathan, R. and El-Halwagi, M., 1994, "Global Optimization of Nonconvex Nonlinear Programs via Interval Analysis," *Computer and Chemical Engineering,* Vol. 18, No. 10, pp. 889-897.

Vanderplaats, G. N., 1984, *Numerical Optimization Techniques for Engineering Design: With Applications*, McGraw-Hill, New York.

VDI, 1986, *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte (Systematic approach to the development and design of technical systems and products),* VDI-Richtlinien (VDI-Standards), VDI 2221, VDI-Verlag GmbH, Dusseldorf, Germany.

Vincent, T. L., 1983, "Game Theory as a Design Tool," *ASME Journal of Mechanisms, Transmissions, and Automation in Design,* Vol. 105, pp. 165-170.

Vincent, T. L. and Grantham, W. J., 1981, *Optimality in Parametric Systems*, John Wiley, New York, NY.

Von Neumann, J. and Morgenstern, O., 1944, *Theory of Games and Economic Behavior*, Princeton University Press, Princeton, NJ.

Welch, W. J., Yu, T. K., Kang, S. M. and Wu, J., 1990, "Computer Experiments for Quality Control by Parameter Design," *Quality Technology,* Vol. 22, pp. 15-22.

Ye, Y., 1992, "On Affine Scaling Algorithms for Nonconvex Quadratic Programming," *Mathematical Programming,* Vol. 56, No. 3, pp. 285-300.

Zabinsky, Z., Graesser, D., Tuttle, M. and Kim, G.-I., 1992, "Global Optimization of Composite Laminates Using Improving Hit and Run," *Recent Advances in Global Optimization,* Floudas, C. and Pardalos, P., eds., pp. 343-367.

Zabinsky, Z., Smith, R., McDonald, F., Romelin, H. E. and Kaufman, D. E., 1993, "Improving Hit-and-Run for Global Optimization," *Journal of Global Optimization,* Vol. 3, No. 2, pp. 171-192.

Zhang, C. and Wang, H.-P., 1993, "Mixed-Discrete Nonlinear Optimization with Simulated Annealing," *Engineering Optimization,* Vol. 21, pp. 277-291.

Zhou, Q.-J., 1988, "The Compromise Decision Support Problem: A Fuzzy Formulation," M.S. Thesis, Department of Mechanical Engineering, University of Houston, Houston, Texas.

Zhou, Q.-J., Allen, J. K. and Mistree, F., 1992, "Decisions under Uncertainty: The Fuzzy Compromise Decision Support Problem," *Engineering Optimization*, Vol. 20, pp. 21-43.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1997	Contractor Report

4. TITLE AND SUBTITLE
An Algorithm for Integrated Subsystem Embodiment and System Synthesis

5. FUNDING NUMBERS
NAG-1-1564

505-10-31-03

6. AUTHOR(S)
Kemper Lewis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Georgia Institute of Technology
Atlanta, GA 30332-0356

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-2199

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CR-201732

11. SUPPLEMENTARY NOTES
The information in this report was offered as a thesis in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia, September 1996. Langley Technical Monitor: Jaroslaw Sobieski.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 05

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

Consider the statement, A system has two coupled subsystems, one of which dominates the design process. Each subsystem consists of discrete and continuous variables, and is solved using sequential analysis and solution." To address this type of statement in the design of complex systems, three steps are required, namely, the embodiment of the statement in terms of entities on a computer, the mathematical formulation of subsystem models, and the resulting solution and system synthesis.

In complex system decomposition, the subsystems are not isolated, self-supporting entities. Information such as constraints, goals, and design variables may be shared between entities. But many times in engineering problems, full communication and cooperation does not exist, information is incomplete, or one subsystem may dominate the design. Additionally, these engineering problems give rise to mathematical models involving nonlinear functions of both discrete and continuous design variables

In this dissertation an algorithm is developed to handle these types of scenarios for the domain-independent integration of subsystem embodiment, coordination, and system synthesis using constructs from Decision-Based Design, Game Theory, and Multidisciplinary Design Optimization. Implementation of the concept in this dissertation involves testing of the hypotheses using example problems and a motivating case study involving the design of a subsonic passenger aircraft.

14. SUBJECT TERMS
Optimization
System

15. NUMBER OF PAGES
532

16. PRICE CODE
A23

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		