AIAA-94-4325-CP

MULTIDISCIPLINARY OPTIMIZATION METHODS FOR AIRCRAFT PRELIMINARY DESIGN

by Ilan Kroo, Steve Altus, Robert Braun,
Peter Gage, and Ian Sobieski*

Stanford University, Stanford, California

Abstract

This paper describes a research program aimed at improved methods for multidisciplinary design and optimization of large-scale aeronautical systems. The research involves new approaches to system decomposition, inter-disciplinary communication, and methods of exploiting coarse-grained parallelism for analysis and optimization. A new architecture, that involves a tight coupling between optimization and analysis, is intended to improve efficiency while simplifying the structure of multidisciplinary, computation-intensive design problems involving many analysis disciplines and perhaps hundreds of design variables. Work in two areas is described here: system decomposition using compatibility constraints to simplify the analysis structure and take advantage of coarse-grained parallelism; and collaborative optimization, a decomposition of the optimization process to permit parallel design and to simplify interdisciplinary communication requirements.

Introduction

The design of complex systems often involves the work of many specialists in various disciplines, each dependent on the work of other groups. When a single chief designer or core team is able to develop and apply design tools in all disciplines, difficulties in communication and organization are minimized. As design problems become more complex, the role of disciplinary specialists increases and it becomes more difficult for a central group to manage the process. As the analysis and design task becomes more decentralized, communications requirements become more severe. These difficulties with multidisciplinary design are particularly evident in the design of aerospace vehicles, a process that involves complex analyses, many disciplines, and a large design space. Advances in disciplinary analyses in the last two decades have only aggravated these problems, increasing the amount of shared information and outpacing developments in interdisciplinary communications and system design methods.

Aerospace design methods based on the model of a central designer have been widely used in aircraft conceptual design for decades and have proven very effective when restricted to simple problems with very approximate analyses. These large, monolithic, analysis and design codes are truly multidisciplinary, but as analyses have become more complex such codes have grown so large as to be incomprehensible and difficult to maintain. Since few know what is included in the code, results are hard to explain and not credible. When simplified to be manageable, the analysis becomes simplistic and again produces results that are incredible. Moreover, complex interconnections between disciplines, even in simpler programs make modification or extension of existing analyses difficult. The problems are evidenced by the use of very simplified aerodynamics in synthesis codes that deal with hundreds of different analyses. Discussions in the literature of multidisciplinary design problems involving more complex analyses are almost invariably restricted to two or (rarely) three codes. The difficulties involved in dealing with large multidisciplinary problems lead to such results as limited design "cycling", cruise-designed wings with off-design "fixes", and an inability to efficiently consider innovative configurations. A need exists, not simply to increase the speed of the analyses, but rather to change the structure of such design problems to simultaneously improve performance and reduce complexity.

This paper introduces new architectures for the design of complex systems and describes tools that may aid in the implementation of these schemes. The approaches considered here include, first, the simplification and decomposition of analyses using numerical optimization and, next, the transformation of the design problem itself into parallel, collaborative tasks. A decomposition tool based on a genetic algorithm is introduced to aid in problem formulation.

Simplification and Decomposition of Analyses Using Compatibility Constraints

Summary

Although one often thinks of multidisciplinary optimization of an aircraft configuration as proceeding logically from geometrical description to disciplinary analyses to performance constraint evaluation, the actual structure of analyses in an aircraft conceptual design method is much more complex. Figure 1 shows the connections between just some of the subroutines in one such program (Ref. 1).

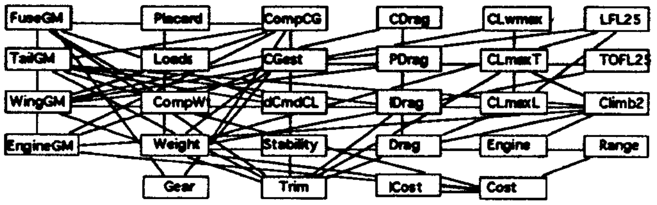

Figure 1. Connections between analyses in an aircraft design problem.

* Department of Aeronautics and Astronautics

Figure 2 illustrates the connections among analyses in a more structured format as described in reference 2. Here, it is assumed that routines are executed serially, from the upper left to the lower right. Lines connecting one routine to another on the upper right side thus represent feed-forward, while lines to the left or below represent an iterative feed-back.

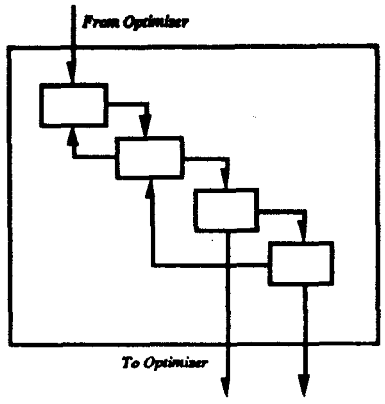

Figure 2. Connections among analyses as viewed in a dependency diagram.

Planning programs such as DeMaid (Ref. 2) can be used to re-order and arrange analyses to minimize the extent of feed-back between groups, but, in general, leave an iterative system. The desirability of removing large-scale iteration from such systems has been described in Refs. 3-4, which cite increased function smoothness and system comprehensibility as advantages. One approach to creating DAGs (directed acyclic graphs) from complex, iterative analysis structures is to use optimization to enforce the convergence constraint explicitly. That is, an auxiliary variable representing the starting "guess" for the iteration is added to the list of optimization design variables along with an additional "compatibility" constraint that represents the convergence criterion. Such an approach leads to smoother functions for the optimizer and is often more robust than conventional fixed-point iteration. The resulting analysis structure is shown on the left side of figure 3.

When one applies the same concept to the feed-forward connections as well, the structure on the right side of the figure is produced. This strikingly simple system represents a decomposition of the original problem that is not only free of iteration, but also parallel. The optimization process enforces the requirement that the results of one computation match the inputs to another, permitting serial tasks to be parallelized in a transparent manner. As the optimization proceeds, the discrepancy between assumed inputs (auxiliary design variables) and computed values is reduced to a specified feasibility tolerance.

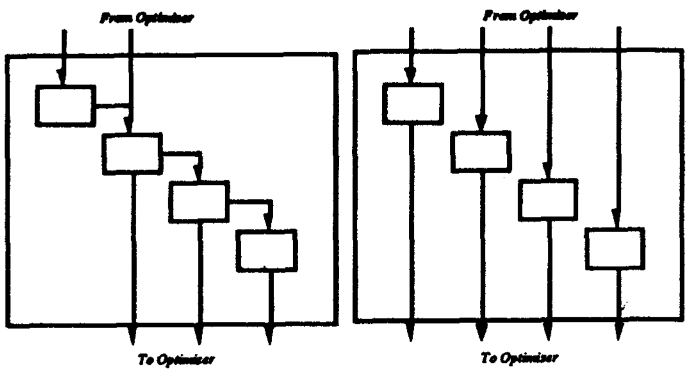

Figure 3. Decomposition using compatibility constraints

Through the compatibility constraints, the optimizer enforces the requirement that the various subproblems are using consistent data, as the auxiliary variables are driven toward their computed counterparts. This allows the subproblems to be run in parallel even though they form an overall serial task. While this can improve computational efficiency in some cases, it may have even greater impact in the development, maintenance, and extension of analyses for multidisciplinary optimization, since each subproblem (which may correspond to one discipline) communicates with other groups only through the optimizer.

Application to an Aircraft Design Problem

Decomposition with compatibility constraints was studied in an aircraft design problem using the program, PASS (Ref. 5). PASS is an aircraft synthesis code with dozens of disciplinary analysis routines, combined with NPSOL, a numerical optimizer based on a sequential quadratic programming algorithm. (Ref. 6). The problem studied here involves minimization of direct operating cost for a medium-range commercial transport, with 13 design variables and nine constraints. Design variables included: maximum take-off weight, initial and final cruise altitudes, wing area, sweep, aspect ratio, avg. t/c, wing position on fuselage, tail aspect ratio, take-off flap deflection, sea-level static thrust, max. zero fuel weight, and actual take-off weight. Constraints included: range, climb rates and gradients, field lengths, stability, surface maximum C_L's, and maximum weights.

The analysis was decomposed into three parts as shown in figure 4. Each part contains several analysis routines, with the resulting groups representing (roughly) structures, high-speed aerodynamics, and low-speed performance. The decomposition introduces five auxiliary design variables and their five associated compatibility constraints represented by the dashed lines in the figure.

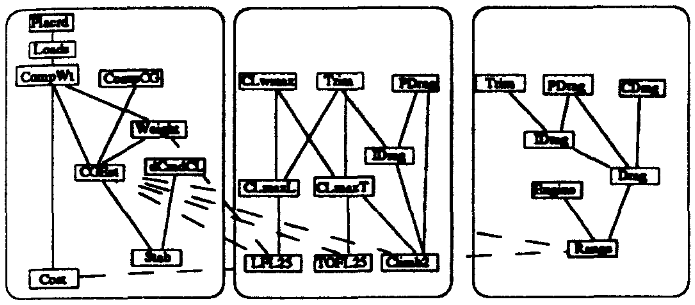

Figure 4. Decomposition of an aircraft design problem into three subproblems.

Despite the increased size of the decomposed problem, the computation time for a converged solution is actually reduced by 26%. A more interesting measure of computational efficiency is the number of calls to various analysis subroutines. PASS routines are relatively simple, and the overhead involved in the optimization can be significant compared to the actual analyses. Since the present method of decomposition is intended for analyses that are more complex, the efficiency of the scheme is best measured by the number of calls to analysis subroutines. The total number of subroutine calls is shown in figure 5. Not only is there a 29% decrease in computation for the simple, sequential execution of the decomposed problem, but parallel execution of the three subproblems decreases the computation time by 69%.

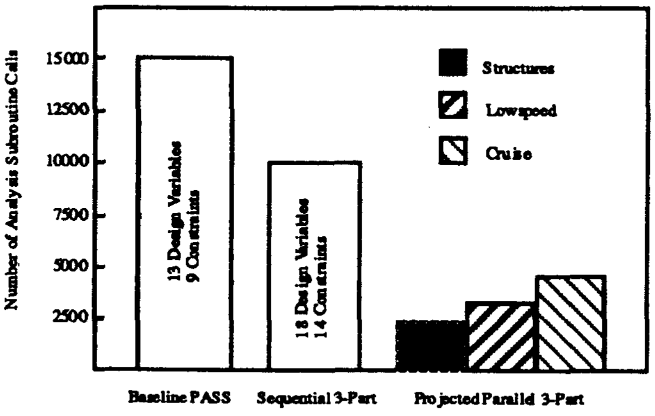

Figure 5. Results of 3-part decomposition of aircraft design problem. Solved using PASS and NPSOL on an IBM RS/6000.

<u>Solution on a Heterogeneous Distributed Network</u>

The aircraft design problem has been solved on a heterogeneous distributed network as shown in figure 6. The three analysis groups wait for the communication subroutine to write the latest values of the design variables to a file. They then compute the constraints (or objective) for which they are responsible, and write the results to another file. The communications subroutine reads the results and sends them to NPSOL, which returns a new set of design variable values. This file-sharing technique is simple but computationally expensive, and is intended not to show gains in computational efficiency, but rather to demonstrate the feasibility of parallel execution of the decomposed system.

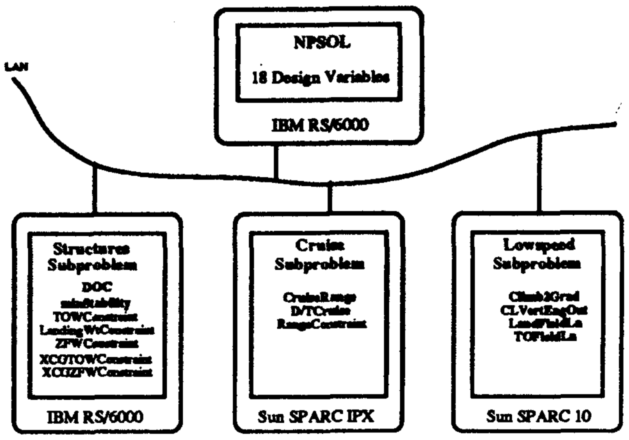

Figure 6. Aircraft design problem solved on a heterogeneous distributed network.

The simplification of the analysis by decomposition was evident in the size of the database required for each subproblem, averaging about half the size of that for the integrated problem. This makes it easier for a user to write, maintain, and modify that set of analyses without detailed knowledge of other disciplines.

<u>Form of the Compatibility Constraints</u>

The compatibility constraint, y = y', may be posed in several forms. Since numerical optimizers can be sensitive to such details of problem formulation, a systematic study of the effect of changing the form of compatibility constraints was undertaken. The following examples were considered:

1. $y - y' \leq \pm \varepsilon_1$
2. $(y - y')^2 \leq \varepsilon_2$
3. $y/y' = 1 \pm \varepsilon_3$
4. $\ln(y/y') = 1 \pm \varepsilon_4$

The limits ε_1 - ε_4 were adjusted to demand a consistent level of accuracy, but the compatibility constraint scaling is not easily established and so the optimization was run with a variety of scalings. The data shown in Table 1 is thus a measure not only of the relative computation times associated with the various forms of the compatibility constraints, but also of the sensitivity of each to scaling. The convergence rates are high enough for each form that it may be assumed that a "good" scaling can be found for any of the forms. The results show that differences of squares are faster than the other forms, possibly because it avoids switching of active constraint sets.

Form	Total Runs	%Converged	CPU Avg.	CPU Min.
y-y'	86	87	22.0	17.5
$(y-y')^2$	27	89.3	18.1	13.6
y/y'	20	100	21.0	18.3
ln(y/y')	36	88.9	20.7	17.4

Table 1. Results with different forms of constraints

Decomposition of the Design Process: Collaborative Optimization

Summary

Although the decomposition of analyses using compatibility constraints simplifies the complex analysis structure in MDO problems and can be used to parallelize the process, it is still subject to the following criticism: 1. The individual disciplines provide analysis results but do not have a clear mechanism for changing the design. The actual design work is relegated to a central authority (the single optimizer) and the disciplines are not permitted to apply their design expertise to satisfy discipline-specific problems. 2. The communication requirements are severe. Information on all constraints and gradients must be passed to the system level optimizer. 3. All design variables, constraints, and analysis interconnections must be established *a priori* and described to the system optimizer.

Especially in cases that involve weak interdisciplinary coupling and a large number of discipline specific constraints, this centralized design approach is not appropriate. In certain special cases a hierarchical decomposition is possible. Such cases involve local variables that have no effect on other disciplines and have been widely reported (Refs. 7-8). Unfortunately most problems do involve substantial interdisciplinary coupling and non-hierarchical decomposition schemes (Ref. 9) similar to the method described in the previous section are suggested.

One may, however, extend the idea of compatibility constraints to produce a two-level decomposition from an arbitrarily connected set of analyses. This section describes how this may be done so as to decompose, not just the analyses, but the design process itself. The basic structure is illustrated in figure 7. Individual disciplines involve both analysis and design responsibilities, communicating (directly in a single program, or over a network) with a system-level coordination routine.

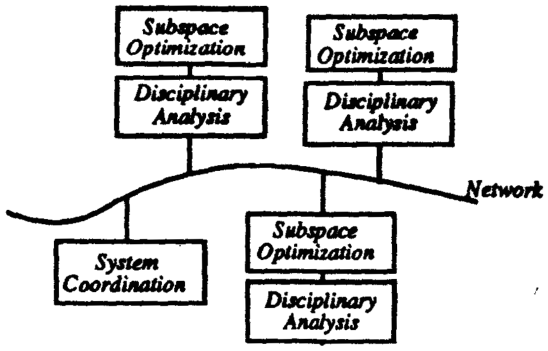

Figure 7. Basic structure for collaborative optimization.

The basic approach requires that the decentralized groups satisfy their local constraints so that discipline-specific information need not be communicated to other groups. In order to assure that the local groups can succeed in this task, they are permitted to disagree with other groups during the course of the design process. The objective of the subspace optimizers is to minimize these interdisciplinary discrepancies while satisfying the specified constraints. The system level coordination optimizer is responsible for ensuring that these discrepancies are made to vanish. More specifically, the system level optimizer provides target values for those parameters that are shared between groups. The goal of each group is to match the system target values to the extent permitted by local constraints.

This approach, termed collaborative optimization, has several desirable features. Domain-specific design variables, constraints, and sensitivities remain associated with a specific discipline and do not need to be passed among all groups. This permits disciplinary problems to be addressed by experts who understand the physical significance of the variable or constraint as well as how best to solve the local problem. Subspace optimization problems may be changed (e.g. adding constraints or local variables) without affecting the system-level problem. The method provides a general means by which the design process may be decomposed and parallelized, and in some implementations does not depend on the use of gradient-based optimization.

Several versions of the basic collaborative optimization scheme have been investigated and the specific approach is perhaps best described by example. The subsequent sections provide both a very simple example, intended to further describe the methodology, and a more realistic multidisciplinary aircraft design problem.

Simple Application of Collaborative Optimization

As a first example of collaborative optimization, consider minimization of Rosenbrock's valley function:

$$\min: \; J(x_1,x_2) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Solution of this two-variable, unconstrained, quartic problem with various optimization techniques is presented in Ref. 1. In the present analysis, all optimization is performed with the sequential quadratic programming algorithm, NPSOL. From a starting point of [0.0, 0.0], using analytic gradients, NPSOL reached the the solution in 13 iterations. For demonstration purposes, assume that we are dealing with a more complex, multidisciplinary optimization problem than the Rosenbrock valley in which the design components are computed by different groups. Although this computationally-distributed problem may be reassembled and solved by a single optimizer, one of the significant advantages of collaborative optimization is that this integration is not necessary. For example, suppose computation of the Rosenbrock objective function is decomposed among two groups as,

$$\begin{aligned} \min: &\quad J = J_1 + J_2 \\ \text{where:} &\quad J_1\,(x_1,x_2) = 100\,(x_2 - x_1^2)^2 \\ \text{and:} &\quad J_2(x_1) = (1 - x_1)^2 \end{aligned}$$

Let one analysis group be responsible for computing J_1 and another J_2. Since the computation is distributed, collaboration is required to ensure that the groups have a consistent description of the design space. Among numerous strategies which achieve the necessary coordination, two example approaches are illustrated in figure 8. In both cases, the system-level optimizer is used to orchestrate the overall optimization process through selection of system-level target variables. A system-level target is needed for each variable which is used by more than one analysis group (e.g., y_1 which is computed in analysis group 1 and input to analysis group 2). The subspace optimizers have as design variables all inputs required by their analysis group. Note that for analysis group 1, this includes the local variable x_2 which is not used in analysis group 2. The goal of each subspace optimizer is to minimize the discrepancy between the local versions of each system-level target variable and the target variable itself. Treatment of this discrepancy minimization is the source of the differences between the two collaborative strategies. In the first approach (left side of figure 8), a summed square discrepancy is minimized and the collaborative framework does not introduce any additional constraints to the analysis group. Note that this subspace objective function is at least quadratic but is generally more nonlinear. In the approach depicted on the right, a constraint is added for each system-level target needed by the analysis group (Ref. 10). If the system-level target is represented locally as an input, this added constraint is linear; otherwise, the added constraint is nonlinear. In this formulation, an extra design variable (x_3) is required and the objective is linear.

Using each of these formulations, collaborative solutions to the Rosenbrock valley function were obtained. Results of the simulations are highlighted in Table 2. In the first formulation of these problems, the system-level objective gradient and constraint Jacobian information is obtained through finite-differencing of the optimized subproblems. Furthermore, each subspace optimization is initiated without the benefit of information from previous runs.

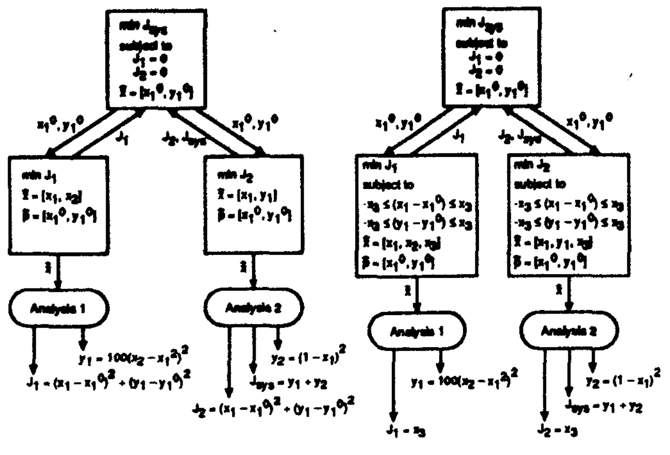

Figure 8. Two collaborative optimization approaches for Rosenbrock's valley function

Collaborative Framework Characteristics			
System Objective Gradient	System Constraint Jacobian	Subspace Optimization Restart Scheme	Total Iterations
	Solution Strategy 1		
Finite-difference	Finite-difference	Fixed $x_{initial}$ Hessian reset	1556 (1262)
Finite-difference	Post-optimality	Fixed $x_{initial}$ Hessian reset	755 (94)
Finite-difference	Post-optimality	$x_{initial}$ (j) $= x^*(j-1)$ Hessian reset	390
Finite-difference	Post-optimality	$x_{initial}$ (j) $= x^*(j-1)$ Hessian retained	385
J^0-analytic	Post-optimality Hessian reset	$x_{initial}$ (j) $= x^*(j-1)$	516
	Solution Strategy 2		
Finite-difference	Finite-difference	Fixed $x_{initial}$ Hessian reset	345 (248)
Finite-difference	Post-optimality	Fixed $x_{initial}$ Hessian reset	336 (48)
Finite-difference	Post-optimality	$x_{initial}$ (j) $= x^*(j-1)$ Hessian reset	208
Finite-difference	Post-optimality	$x_{initial}$ (j) $= x^*(j-1)$ Hessian retained	141
J^0-analytic	Post-optimality Hessian reset	$x_{initial}$ (j) $= x^*(j-1)$	260

Table 2: Performance of two collaborative optimization solution strategies for Rosenbrock's valley function

Obtaining the system-level objective gradient and constraint Jacobian by finite-differencing optimized subproblems requires numerous subproblem optimizations (each with numerous subspace iterations) for every system-level iteration. Furthermore, these subspace optimizations must be tightly converged to to yield accurate derivatives. To further minimize numerical error, additional calculations were performed to select appropriate finite-difference intervals. These extra iterations are shown in parenthesis in Table 1. Without the proper choice of finite-difference interval, the convergence of the collaborative strategies was not robust to changes in the starting point.

To reduce both computational expense and numerical error, optimal sensitivity information from the converged subproblem may be used to provide system-level derivatives. (Ref. 11) This is possible since the system-level targets are treated as parameters in the subproblems and as design variables in the system optimization. Hence, the main problem Jacobian is equivalent to the subproblem dJ^*/dp. The required information is generally available at the solution of each subspace optimization problem (with little to no added cost) through the following equation. (Refs.11-13.)

$$\frac{dJ^*}{dp} = \frac{\partial J}{\partial p} + \lambda_i^{T*} \frac{\partial c_i}{\partial p}$$

Here λ^* represents the Lagrange multiplier vector at the solution. Because of the special structure of the collaborative optimization solution strategies, this equation becomes,

$$\frac{dJ^*}{dp} = \frac{\partial J}{\partial p} = -2(x - x^0) \qquad \text{for solution strategy 1}$$

$$\frac{dJ^*}{dp} = \lambda_i^{T*} \frac{\partial c_i}{\partial p} = \lambda_i^T \begin{cases} 0 \\ -1 \end{cases} \qquad \text{for solution strategy 2}$$

Note that in either of the proposed collaborative formulations, calculation of the required partial derivatives is trivial. However, although many optimizers provide an estimate of λ^* as part of the termination process, accurate estimates are only ensured when the problem has converged tightly (Refs. 12-13). For this reason, solution strategy 1 may be preferred. As shown in Table 1, using this post-optimality information results in significant computational savings by reducing both the finite-difference interval computations and the number of calls to each subspace optimizer. Note that in this case, finite-differencing is still used to estimate the system-level objective gradient.

Computational expense can be reduced further by starting the subspace optimizers from their previous solutions and with knowledge of the previous Hessian. While starting the subspace optimization runs from the previous solutions is effective in both solution strategies, use of the previous Hessian information is of greater benefit for solution strategy 2. This performance gain results because knowledge of the previous solution's active constraint set is of significance for strategy 2.

In another approach, an extra system-level design variable is added (J^0) which also serves as the system-level objective function (see Fig. 3). This concept which may be adapted to either solution strategy results in a linear system-level objective and completely eliminates the finite-difference requirements between the system and subspace levels. The analysis group which computes the actual objective function (group 2 in this case) is now also responsible for matching the target system objective. This added responsibility causes more difficulty for the subspace analysis (as evident in the total number of function evaluations). Furthermore, an increased number of system-level iterations is required. An alternate means of eliminating these finite-difference requirements is to rely on additional post-optimality information in the subspace analysis to estimate the change in actual objective function with respect to a change in the parameters as,

$$\frac{dJ_{sys}^*}{dp} = \left(\frac{\partial J_{sys}}{\partial x_i}\right)\left(\frac{\partial x_i^*}{\partial p}\right)$$

Application to an Aircraft Design Problem

Figure 9 shows the arrangement of analyses in an example aircraft design problem. Here the goal is to select values of the design variable that maximize range with a specified gross weight. The figure shows the analysis grouped into three disciplinary units, aerodynamics, structures, and performance. The problem stated in this way is not directly executable in parallel. Dependent disciplines must wait for their inputs variables to be updated before they may execute. The issue is further complicated when there are feedbacks as between structures and aerodynamics due to aeroelastic effects. In such a case there is an iterative loop that may take several iterations to converge before subsequent disciplines (in this case, performance) may be executed.

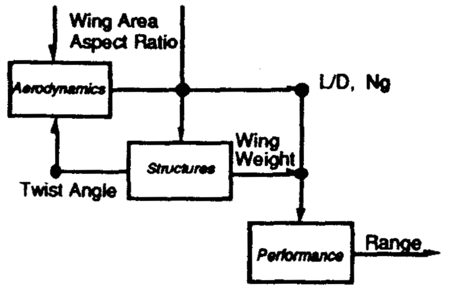

Figure 9. Example multidisciplinary aircraft design problem: analysis structure.

Simple Application of Collaborative Optimization

As a first example of collaborative optimization, consider minimization of Rosenbrock's valley function:

$$\min: \; J(x_1,x_2) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Solution of this two-variable, unconstrained, quartic problem with various optimization techniques is presented in Ref. 1. In the present analysis, all optimization is performed with the sequential quadratic programming algorithm, NPSOL. From a starting point of [0.0, 0.0], using analytic gradients, NPSOL reached the the solution in 13 iterations. For demonstration purposes, assume that we are dealing with a more complex, multidisciplinary optimization problem than the Rosenbrock valley in which the design components are computed by different groups. Although this computationally-distributed problem may be reassembled and solved by a single optimizer, one of the significant advantages of collaborative optimization is that this integration is not necessary. For example, suppose computation of the Rosenbrock objective function is decomposed among two groups as,

$$\begin{aligned} &\min: && J = J_1 + J_2 \\ &\text{where:} && J_1\,(x_1,x_2) = 100\,(x_2 - x_1^2)^2 \\ &\text{and:} && J_2(x_1) = (1 - x_1)^2 \end{aligned}$$

Let one analysis group be responsible for computing J_1 and another J_2. Since the computation is distributed, collaboration is required to ensure that the groups have a consistent description of the design space. Among numerous strategies which achieve the necessary coordination, two example approaches are illustrated in figure 8. In both cases, the system-level optimizer is used to orchestrate the overall optimization process through selection of system-level target variables. A system-level target is needed for each variable which is used by more than one analysis group (e.g., y_1 which is computed in analysis group 1 and input to analysis group 2). The subspace optimizers have as design variables all inputs required by their analysis group. Note that for analysis group 1, this includes the local variable x_2 which is not used in analysis group 2. The goal of each subspace optimizer is to minimize the discrepancy between the local versions of each system-level target variable and the target variable itself. Treatment of this discrepancy minimization is the source of the differences between the two collaborative strategies. In the first approach (left side of figure 8), a summed square discrepancy is minimized and the collaborative framework does not introduce any additional constraints to the analysis group. Note that this subspace objective function is at least quadratic but is generally more nonlinear. In the approach depicted on the right, a constraint is added for each system-level target needed by the analysis group (Ref. 10). If the system-level target is represented locally as an input, this added constraint is linear; otherwise, the added constraint is nonlinear. In this formulation, an extra design variable (x_3) is required and the objective is linear.

Using each of these formulations, collaborative solutions to the Rosenbrock valley function were obtained. Results of the simulations are highlighted in Table 2. In the first formulation of these problems, the system-level objective gradient and constraint Jacobian information is obtained through finite-differencing of the optimized subproblems. Furthermore, each subspace optimization is initiated without the benefit of information from previous runs.

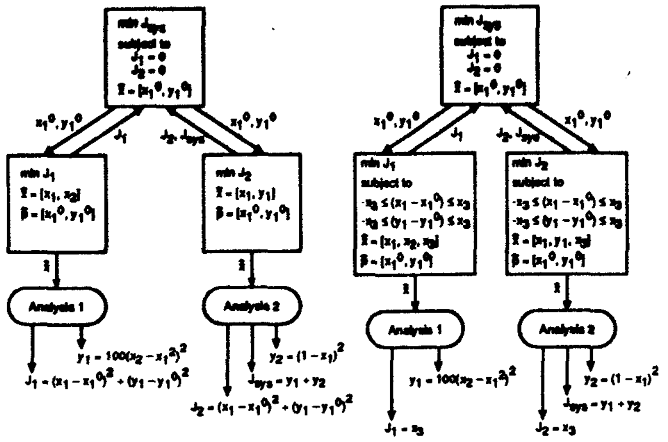

Figure 8. Two collaborative optimization approaches for Rosenbrock's valley function

Collaborative Framework Characteristics			
System Objective Gradient	System Constraint Jacobian	Subspace Optimization Restart Scheme	Total Iterations
	Solution Strategy 1		
Finite-difference	Finite-difference	Fixed $x_{initial}$ Hessian reset	1556 (1262)
Finite-difference	Post-optimality	Fixed $x_{initial}$ Hessian reset	755 (94)
Finite-difference	Post-optimality	$x_{initial}(j) = x^*(j-1)$ Hessian reset	390
Finite-difference	Post-optimality	$x_{initial}(j) = x^*(j-1)$ Hessian retained	385
J^0-analytic	Post-optimality Hessian reset	$x_{initial}(j) = x^*(j-1)$	516
	Solution Strategy 2		
Finite-difference	Finite-difference	Fixed $x_{initial}$ Hessian reset	345 (248)
Finite-difference	Post-optimality	Fixed $x_{initial}$ Hessian reset	336 (48)
Finite-difference	Post-optimality	$x_{initial}(j) = x^*(j-1)$ Hessian reset	208
Finite-difference	Post-optimality	$x_{initial}(j) = x^*(j-1)$ Hessian retained	141
J^0-analytic	Post-optimality Hessian reset	$x_{initial}(j) = x^*(j-1)$	260

Table 2: Performance of two collaborative optimization solution strategies for Rosenbrock's valley function

Obtaining the system-level objective gradient and constraint Jacobian by finite-differencing optimized subproblems requires numerous subproblem optimizations (each with numerous subspace iterations) for every system-level iteration. Furthermore, these subspace optimizations must be tightly converged to to yield accurate derivatives. To further minimize numerical error, additional calculations were performed to select appropriate finite-difference intervals. These extra iterations are shown in parenthesis in Table 1. Without the proper choice of finite-difference interval, the convergence of the collaborative strategies was not robust to changes in the starting point.

To reduce both computational expense and numerical error, optimal sensitivity information from the converged subproblem may be used to provide system-level derivatives. (Ref. 11) This is possible since the system-level targets are treated as parameters in the subproblems and as design variables in the system optimization. Hence, the main problem Jacobian is equivalent to the subproblem dJ^*/dp. The required information is generally available at the solution of each subspace optimization problem (with little to no added cost) through the following equation. (Refs.11-13.)

$$\frac{dJ^*}{dp} = \frac{\partial J}{\partial p} + \lambda_i^{T^*} \frac{\partial c_i}{\partial p}$$

Here λ^* represents the Lagrange multiplier vector at the solution. Because of the special structure of the collaborative optimization solution strategies, this equation becomes,

$$\frac{dJ^*}{dp} = \frac{\partial J}{\partial p} = -2(x - x^0) \qquad \text{for solution strategy 1}$$

$$\frac{dJ^*}{dp} = \lambda_i^{T^*} \frac{\partial c_i}{\partial p} = \lambda_i^T \begin{cases} 0 \\ -1 \end{cases} \qquad \text{for solution strategy 2}$$

Note that in either of the proposed collaborative formulations, calculation of the required partial derivatives is trivial. However, although many optimizers provide an estimate of λ^* as part of the termination process, accurate estimates are only ensured when the problem has converged tightly (Refs. 12-13). For this reason, solution strategy 1 may be preferred. As shown in Table 1, using this post-optimality information results in significant computational savings by reducing both the finite-difference interval computations and the number of calls to each subspace optimizer. Note that in this case, finite-differencing is still used to estimate the system-level objective gradient.

Computational expense can be reduced further by starting the subspace optimizers from their previous solutions and with knowledge of the previous Hessian. While starting the subspace optimization runs from the previous solutions is effective in both solution strategies, use of the previous Hessian information is of greater benefit for solution strategy 2. This performance gain results because knowledge of the previous solution's active constraint set is of significance for strategy 2.

In another approach, an extra system-level design variable is added (J^0) which also serves as the system-level objective function (see Fig. 3). This concept which may be adapted to either solution strategy results in a linear system-level objective and completely eliminates the finite-difference requirements between the system and subspace levels. The analysis group which computes the actual objective function (group 2 in this case) is now also responsible for matching the target system objective. This added responsibility causes more difficulty for the subspace analysis (as evident in the total number of function evaluations). Furthermore, an increased number of system-level iterations is required. An alternate means of eliminating these finite-difference requirements is to rely on additional post-optimality information in the subspace analysis to estimate the change in actual objective function with respect to a change in the parameters as,

$$\frac{dJ_{sys}^*}{dp} = \left(\frac{\partial J_{sys}}{\partial x_i}\right)\left(\frac{\partial x_i^*}{\partial p}\right)$$

<u>Application to an Aircraft Design Problem</u>

Figure 9 shows the arrangement of analyses in an example aircraft design problem. Here the goal is to select values of the design variable that maximize range with a specified gross weight. The figure shows the analysis grouped into three disciplinary units, aerodynamics, structures, and performance. The problem stated in this way is not directly executable in parallel. Dependent disciplines must wait for their inputs variables to be updated before they may execute. The issue is further complicated when there are feedbacks as between structures and aerodynamics due to aeroelastic effects. In such a case there is an iterative loop that may take several iterations to converge before subsequent disciplines (in this case, performance) may be executed.

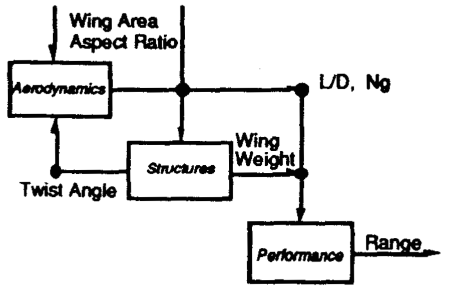

Figure 9. Example multidisciplinary aircraft design problem: analysis structure.

Posed as a collaborative optimization problem, the design task is decomposed as shown in figure 10. Each of the disciplinary units becomes an independent subproblem. Each sub-problem consists of two parts: a subspace optimizer and an analysis routine. The optimizer modifies the input values for the sub-problem analysis and uses the analysis output values to form the constraints and objective. The optimizer also accepts a set of target values for these variables. The goal of the subspace optimization is to adjust the local design variables to minimize the difference between local variable values (both inputs to, and results from, the sub-problem analysis) and the target values passed to the subspace optimizer. To designate these target values the CO methodology adds a system-level optimizer. This optimizer specifies the target values of the design parameters and passes them to each of the sub-problems. The system-level optimizer's goal is to adjust the parameter values so that the objective function (in this case, range) is maximized while the system-level constraints are satisfied.

In this problem there are three system-level constraints. Note from the figure that these constraints are simply the values of the objective function of each of the sub-problems. Thus, the scheme allows the subproblems to temporarily disagree with each other but the equality constraints ($J_1 = J_2 = J_3 = 0$) at the system-level require that in the end, they all agree. So at the end of the optimization all local variable values will be the same as the target values (those with the "o" subscript) designated by the system-level (i.e. $R_0 = R$).

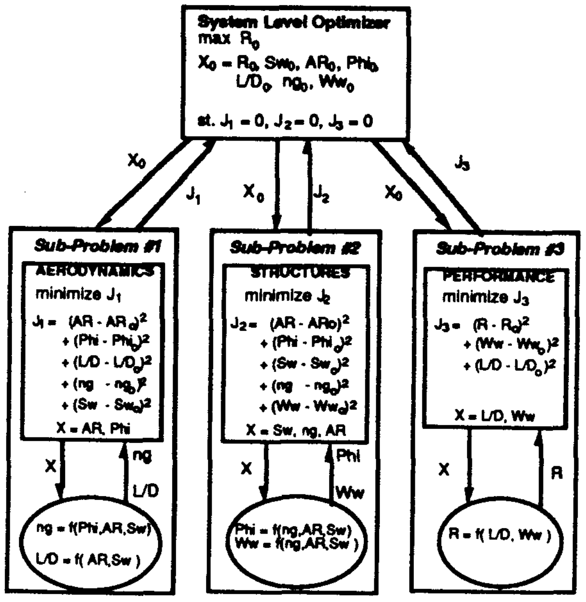

Figure 10. Collaborative form of aircraft design problem.

For example, assume that the system-level optimizer chooses an initial set of design parameters, X_0. Each of the sub-levels will then attempt to match this set. However, if the super-level asks for a physically impossible combination, then the sub-problem will only be able to return a non-zero value of its objective function J_1. This non-zero value is then recognized by the system-level and a more feasible choice of target parameters (X_0) is selected. This next set allows the sub-problem to obtain a minimum objective function (J_1) lower than with the previous set. It is through such a mechanism that the sub-problems influence the progress of the design optimization.

Via a similar mechanism, constraints that exist at the sub-problem level may influence the target (system-level) variable values, and, through this, the value used in another discipline. In this way, constraints that affect one discipline may implicitly affect another without the need for an explicit transfer of information concerning the constraint to a foreign discipline.

One of the primary advantages of this arrangement is that the system is now executable in parallel. In fact, this architecture minimizes the interdiscipinary communication requirements, making it ideal for network implementations. This example problem has been run both on a single workstation and on a system of three networked computers.

Results

Figure 11 shows the optimization history of the design variables. Range was computed by the performance analysis using the Breguet range equation. Wing weight was computed using statistically-based equations for typical weights of transport aircraft wings given maximum load factor, aircraft weights, and wing geometry. The twist angle was also calculated by the structures discipline. It represents the maximum structural twist of a wing under certain aerodynamic loads. These twists are computed for a given maximum gust loading which is calculated in the aerodynamics discipline. The twist itself feeds back into the aerodynamics discipline and changes aerodynamic loading on the wing. The aerodynamics discipline uses simple relations to produce the aerodynamic loading and the lift-to-drag ratio (used in calculating range).

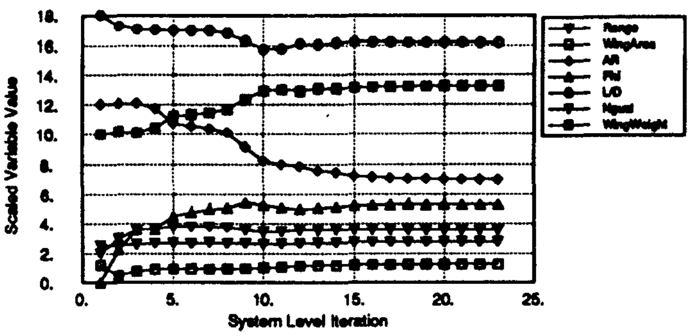

Figure 11: History of Super-Level Variable Changes

The figure shows that the variables nearly reach their converged values within 10 iterations of the starting point. Within each of these iterations there are many sub-level iterations as the sub-levels attempt to match the system-level target variable values. To illustrate this process the variation in the parameter aspect ratio is shown in Figure 12a and b.

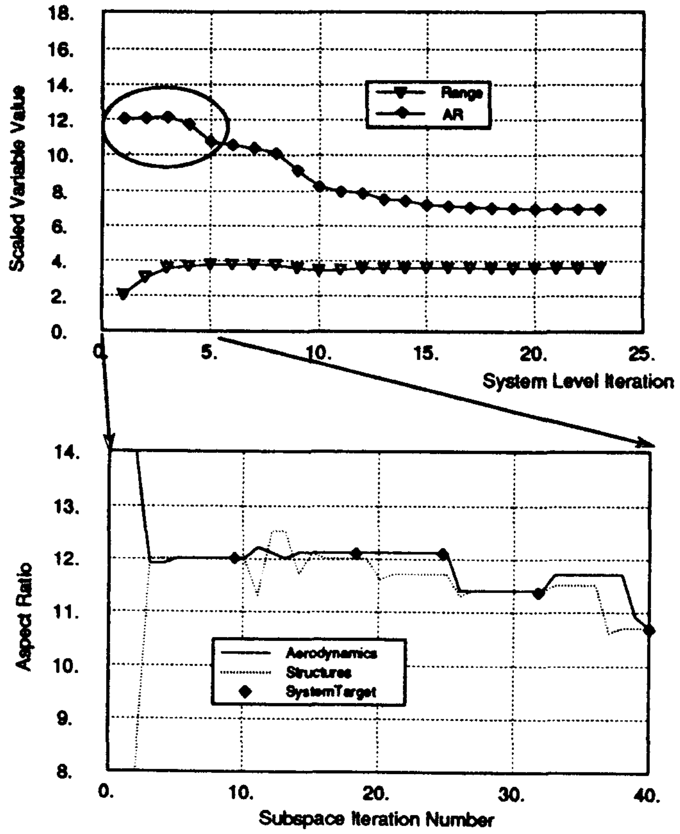

Figure 12a&b : Changes in Aspect Ratio at System and Subsystem Levels

Figure 12a shows the variation of aspect ratio and range versus system-level iterations. Figure 12b shows subspace results during the first five system-level iterations. Included on the plot are the target values for aspect ratio designated by the system-level optimizer along with the value of aspect ratio used in the local sub-problems of structures and aerodynamics. Note that initially these groups start out with their own guesses for aspect ratio (the structures group guesses a low 8.0, the aerodynamicists hope it will be 14.0). They receive the target value from the system-level optimizer (12.0) and immediately try to match it. For the next two system-level iterations the target value of aspect ratio is not changed by the system optimizer. At the third iteration, though, thetarget value for aspect ratio is reduced to about 11.5. One can see that the sub-problems immediately try to match this value.

Note that the sub-problems occasionally do not exactly match the target value as may be seen after system-level iteration #2;. This is because, as was described earlier in this section, the sub-problems are trying to achieve the best match for the entire set of design variables not simply the one shown here. So in this instance, the sub-problem found that allowing a slight deviation in the local value of aspect ratio allowed a greater degree of compatibility between all the variable values used in that sub-problem and their corresponding target values.

As described in the previous section, the gradient of the objective function and the gradient of the constraintsat the system level may be computed analytically based on optimal sensitivity results. In fact, in this problem it was found that these gradients need to be specified analytically for computational stability. Obtaining gradients via finite-differencing can lead to spurious gradient values and difficulties for the system level optimizer. These same problems result when the tolerances of the sub-problems are not set properly and the sub-problems do not reach a satisfactory solution.

A Tool for Decomposition Planning

Motivation: Structure of the Decomposition

Whether the analysis is decomposed using compatibility constraints, or the design problem is decomposed using collaborative optimization, the structure of the decomposition determines the efficiency of the decomposed system. For example, consider two subroutines with computation times A and B, with n inputs to A, and m intermediate values computed by A which are inputs to B. To compute gradients of quantities computed by B with respect to the inputs of A, the computational time using finite differences is $n(A+B)$. If we use decomposition, the computation time for all the gradients becomes $nA + mB$. When n is much larger than m (contraction), decomposition will generally reduce computation time. If, however, m is larger than n, decomposition may increase computation time. Such ideas were considered in the decomposition of the aircraft design problem in figure 4, but only in a qualitative fashion. Moreover, the ad hoc procedure was quite time-consuming. To exploit contraction, avoid expansion, and assign analyses to subproblems efficiently, an automatic tool is desirable. Such a program is described here. It uses a genetic algorithm to find a decomposition that minimizes the estimated computational time of a gradient-based optimization of the resulting decomposed system.

Given a list of analyses and the global variables which are inputs and outputs to each, the program creates a dependence matrix of integers. The element $Dep(i,j)$ corresponds to the number of outputs from routine i which are inputs to routine j. If the routines are executed sequentially, entries in Dep which are below the main diagonal are feedbacks, and entries above the main diagonal are feedforwards. As the orders of the routines are changed, the structures of the dependence matrix changes. By including information about where in the ordering there are "breaks" between subproblems, various objective functions can be evaluated from the dependence matrix.

Several methods for task-ordering have been developed previously, but not with the objective of scheduling the optimization of a decomposed system. The Design Manager's Aid, DeMaid (Ref. 2), for example, uses a heuristic approach to order tasks into a system of subproblems. One of the results of the DeMaid heuristic is that feedback loops, particularly long loops, are removed. Figure 13 illustrates the ordering of tasks, as described in reference 2, that minimizes a measure of feedback extent, J. The objective function used here is:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{i-1} Dep\ (i,j)\ \ (i-j)$$

and reflects the "total length of feedback" in the system. The solution shown in Figure 13, was obtained by the present scheduling algorithm using this objective. It is nearly identical to the one obtained by DeMaid, which does not employ an explicit objective function. Although the subproblems are in a different order, each consists of the same analyses as in the DeMaid problem, except the second, which is a union of two subproblems from the DeMaid solution.

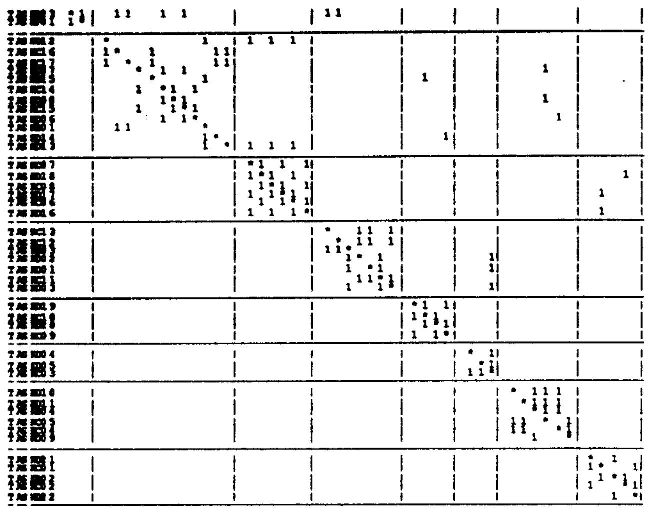

Figure 13. Solution of the DeMaid example problem using extent of feedback as objective function.

Minimizing feedback extent is not the goal for a system that is decomposed using compatibility constraints and optimization. The difference between feedback and feedforward disappears, since the subproblems run in parallel. Therefore the length of an individual feedback is unimportant, and some feedback between the subproblems is not disadvantageous. Conversely, the number of feedforwards between subproblems is important. Thus, a rather different objective is used for the optimal decomposition formulation. The objective used here is an explicit estimate of the computation time for optimization of the decomposed design problem.

Time Estimate for Optimization of Decoupled System

The time estimate assumes the following about the optimization of the system: gradients are estimated by finite-differencing; no second-derivatives are computed (the estimated Hessian is updated); and gradient calculations dominate the computation time. These assumptions are consistent with aircraft sizing problems and with the use of NPSOL, a commonly used optimization package. In addition, it is assumed that the subproblems can be executed in parallel. Finally,we assume that gradients for a subproblem depend on all routines within that subproblem.

The objective function is then:

$J = T_{\text{optimization}} = (N_{\text{line searches}}) * (T_{\text{each line search}})$

$N_{\text{line searches}}$ is proportional to the total number of vari-

ables. Therefore the number of line searches varies as:
$N_{\text{line searches}} = N_{\text{design vars}} + N_{\text{auxiliary design vars}}$

For a typical gradient-based optimization each line search requires that each subroutine be executed once for each independent variable (for gradients) and then an average of twice more for the line search itself. The total number of calls to the routines in each group is then:
$Ncalls_i = 2 + N_{\text{design vars(sub i)}} + N_{\text{auxiliary var (sub i)}}$

where $N_{\text{design vars (sub i)}}$ is the number of design variables that are inputs to routines in the i^{th} subproblem. Since the subproblems are run in parallel, the time for each line search is determined by the slowest subproblem:
$T_{\text{line i}} = \max\{Ncalls_i * \Sigma(\text{execution times of routines in subproblem i})\}$

The full objective function is therefore:
$J = (N_{\text{design vars}} + N_{\text{auxiliary design vars}}) * T_{\text{line i}}$

A Genetic Algorithm for Decomposition
Planning an efficient decomposition is an optimization task in itself. The optimizer seeks to find the correct location for each subroutine in the analysis procedure. Conventional calculus-based optimizers are not effective in this domain, but a number of genetic algorithms have been developed for the solution of planning problems.

Genetic algorithms are designed to mimic evolutionary selection. A population of candidate designs is evaluated at each iteration, and the candidates compete to contribute to the production of new designs. Each individual is represented by a string, which is a coded listing of the values of the design variables. The entire string is analogous to a chromosome, with genes for the different features (or variables). When individuals are selected to be parents for offspring designs, their genetic strings are recombined in a crossover operation, so that the new designs have elements of two earlier designs. A mutation operation also allows modification of elements of the new individual so that it may include new features that were not present in either parent.

The genetic string for the decomposition problem is an integer vector of length $n+m$, where n is the number of analysis subroutines and m is the number of potential break points (allowing $m+1$ independent sub-tasks). Each population member is a permutation of the integers between 1 and $n+m$. For a task with 10 subroutines to be split into 3 sub-tasks, n=10 and m=2. A sample genetic string and the computational system that it represents are shown in Figure 14.

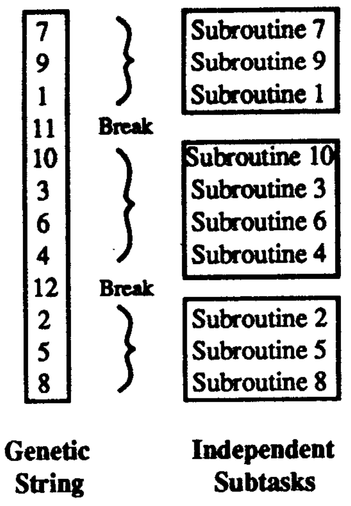

Figure 14. Decoding of genetic string into subroutine order and subproblems.

Simple crossover operators are not appropriate for permutation problems, because they do not guarantee offspring that include exactly one copy of each design variable, and no duplicates. Several crossover schemes have been developed for use in planning problems. Six of these were compared by Starkweather et al. (Ref 14). They found that the best overall performance (on a travelling salesman problem and a warehouse/shipping scheduling problem) was achieved by position-based crossover, originally introduced by Syswerda (Ref. 15). This scheme was adopted for the decomposition problem.

Position-based crossover requires the random selection of several positions in the string. The entries at these positions are passed directly to the offspring by one parent. The remaining positions are filled by variables from the second parent, in the order that they appear in that parent. In Figure 15, b1, d1 and g1 are inherited directly from the first parent, so a, c, e, f must be supplied by the second parent. They appear in the order a2, f2, c2, e2 and fill positions 1, 3, 5, 6 of the offspring.

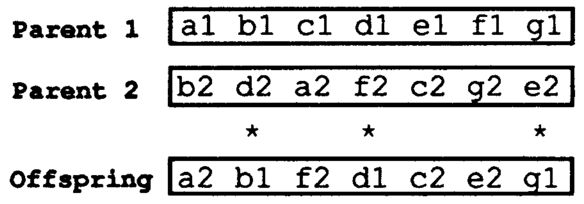

Figure 15. Position-based crossover.

The mutation operator is applied pointwise along the string. When a point is selected to undergo mutation, the variable at that position is swapped with another at a different point in the string. The second point is randomly selected between 1 and a user-specified maximum mutation range. The genetic algorithm uses tournament selection to choose parents to participate in reproduction into the next generation. Each time a parent is needed, two members of the current population are selected at random. Their fitness is compared, and the individual with greater fitness becomes the parent.

The following parameter settings were used in studies reported here: crossover probability is fixed at 0.9; pointwise mutation probability is 0.01; and maximum mutation range is half the total string length, or $(n+m)/2$.

Results

The DeMaid problem, described previously, was solved with this objective, assuming that the overall design problem consisted of as many as 9 subproblems. The optimal decomposition is shown in figure 16. The estimated optimization time for this system is 3.147, compared with DeMaid's ordering which yields a time of 3.325. As is clear from the figure, the extent of feedback is substantially greater for this solution despite its more efficient parallel decomposition.

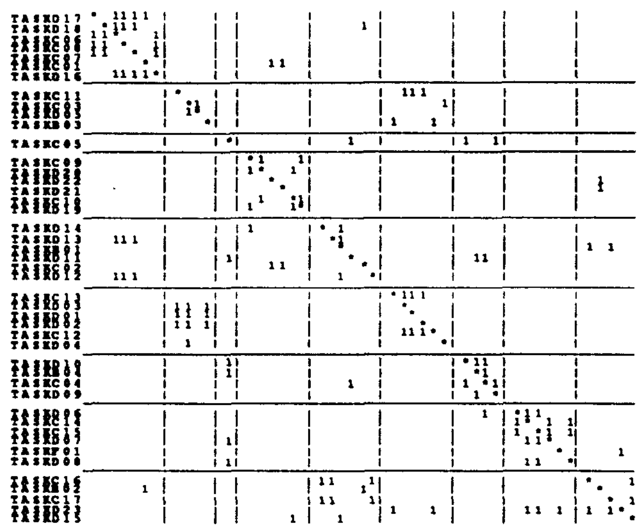

Figure 16. The DeMaid sample problem as ordered for optimization with decomposition using compatibility constraints.

Using this objective function, the optimal decomposition tool has been applied to PASS, yielding a structure very similar to that produced manually and shown in figure 4.

Continuing Work

Other objective functions which will soon be formulated include one for decomposition with compatibility constraints assuming gradient information from automatic differentiation rather than finite-differencing, and one for collaborative optimization. The fact that a new objective for a different type of problem can be easily inserted makes this program a flexible and useful tool which can be adapted to a variety of different formulations.

Conclusions

The present work represents an initial look at new architectures for multidisciplinary optimization applied to the preliminary design of complex systems. Continuing work includes evaluation of a variety of implementation strategies, further development of decomposition and optimization tools compatible with these approaches, and application to larger scale design problems. The work reported here has been supported by NASA Langley Research Center, NASA Ames Research Center, and Boeing Commercial Airplane Group.

References

1. Kroo, I., "An Interactive System for Aircraft Design and Optimization", AIAA Paper #92-1190, Feb. 1992.
2. Rogers, J.L., "DeMaid -- A Design Manager's Aid for Intelligent Decomposition, Users Guide," NASA TM 101575, March 1989.
3. Gage, P., Kroo, I., "Quasi-Procedural Method and Optimization", AIAA/NASA/AirForce Multidisciplinary Optimization Conference, Sept. 1992.
4. Cousin, J., Metcalfe, M., "The BAe Ltd Transport Aircraft Synthesis and Optimization Program," AIAA 90-3295, Sept. 1990.
5. Kroo, I., Takai, M., "A Quasi-Procedural, Knowledge Based System for Aircraft Synthesis", AIAA-88-6502.
6. Gill, P.E., Murray, W., and Wright, M.H., Practical Optimization, Academic Press, Inc., 1981.
7. Haftka, R.T., "On Options for Interdisciplinary Analysis and Design Optimization", *Structural Optimization*, Vol. 4, No. 2, June 1992.
8. Padula, S., Polignone, D., "New Evidence Favoring Multilevel Decomposition and Optimization," Third Symposium on MDO, Sept. 1990.
9. Sobieszczanski-Sobieski, J., "Optimization by Decomposition: A Step from Hierarchic to Non-Hierarchic Systems," NASA CP-3031, Sept. 1989.
10. Sobieszczanski-Sobieski, J., "Two Alternate Ways For Solving the Coordination Problem in Multilevel Optimization," Structural Optimization, Vol. 6, pp. 205-215, 1993.
11. Braun, R.D., Kroo, I.M., and Gage, P.J., "Post-Optimality Analysis in Aerospace Vehicle Design," AIAA 93-3932, Aug. 1993.
12. Hallman, W., "Sensitivity Analysis for Trajectory Optimization Problems," AIAA 90-0471, January 1990.
13. Beltracchi, T.J., and Nguyen, H.N., "Experience with Parameter Sensitivity Analysis in FONSIZE," AIAA 92-4749, Sept. 1992.
14. Starkweather, T., McDaniel, S., Mathias, K., Whitley, D., Whitley, C., "A Comparison of Genetic Sequencing Operators", Proceedings of the 4th International Conference on Genetic Algorithms, Belew, R. & Booker, L., ed. Morgan Kaufmann, San Mateo, 1991.
15. Syswerda, G. "Schedule Optimization Using Genetic Algorithms", in Handbook of Genetic Algorithms. Davis, I., ed. Van Nostrand Reinhold, New York, 1990.