NASA TM- 86329

NASA Technical Memorandum 86329

NASA-TM-86329 19850016496

AN EXPERT SYSTEM FOR CHOOSING THE BEST COMBINATION OF OPTIONS IN A GENERAL-PURPOSE PROGRAM FOR AUTOMATED DESIGN SYNTHESIS

James L. Rogers, Jr., and Jean-Francois M. Barthelemy

MARCH 1985

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

AN EXPERT SYSTEM FOR CHOOSING THE BEST COMBINATION OF OPTIONS IN A GENERAL-PURPOSE PROGRAM FOR AUTOMATED DESIGN SYNTHESIS

by

James L. Rogers, Jr.
NASA Langley Research Center
Hampton, Virginia

and

Jean-Francois M. Barthelemy
Virginia Polytechnic Institute and State University
Blacksburg, Virginia

ABSTRACT

An expert system to aid users of the Automated Design Synthesis (ADS) general-purpose optimization program has been developed. ADS has approximately 100 combinations of strategy, optimizer, and search options from which to choose. This expert system aids the user in choosing the best combination of options for solving a particular problem. The system is written in LISP, contains about 200 rules, and executes on DEC-VAX and IBM PC/XT computers.

INTRODUCTION

An expert system to aid a user of the Automated Design Synthesis (ADS) general-purpose optimization program has recently been developed. Because ADS has three levels of options (strategies, optimizers, and one-dimensional searches), the user has approximately 100 combinations from which to choose. This could easily overwhelm a novice user of the program. According to the developer, a principal difficulty with a program of such broad capability as this is the development of a concise set of guidelines identifying the best choice of a combination of strategy, optimizer, and one-dimensional search options for a given problem. Because of this and the anticipated high usage of ADS throughout industry, a decision was made to develop an expert system for ADS (EXADS).

In general, an expert system consists of two major components, the knowledge base and the inference engine. The knowledge base for EXADS was developed from contributions from the author of ADS, a literature search, and discussions with optimization experts; and currently contains 98 different hypotheses and approximately 200 rules. It is divided into three distinct sets of rules and hypotheses. One set is for constrained problems, a second set is for unconstrained problems, and a third is for constrained problems being treated as unconstrained problems. The inference engine asks questions about, makes decisions from, and determines the consequences implied by the knowledge built into the knowledge base. It is written in LISP and currently executes on DEC-VAX and IBM PC/XT computers.

The purpose of this paper is to discuss how this expert system came into being and to describe its major components. Also discussed are the problems encountered during the testing and verification of the system and solutions to those problems.

THE ORIGIN OF THE EXPERT SYSTEM FOR ADS

The ADS (Automated Design Synthesis, ref. 1) computer program, developed under a NASA grant, is a general-purpose, numerical optimization program containing a wide variety of algorithms. ADS requires a three-level decision to select an algorithm for solving a general optimization problem. These levels are strategy, optimizer, and one-dimensional search. ADS allows the user to have great flexibility in solving a problem by providing eight strategy options (table 1), five optimizer options (table 2), and eight one-dimensional search options (table 3). Table 4 shows the large number of possible combinations of options available.

One difficulty with a program like ADS which provides so many options, is choosing the best possible combination of options to solve a given problem. This choice requires knowledge of the problem to be solved and experience in optimization. Typically, an engineer has sufficient knowledge of the problem to be solved in his or her discipline, but lacks the necessary experience in optimization to make a proper

N85-24807 #

choice among the several options available at each of the three levels in ADS. The development of an expert system to aid an engineer in this selection is one solution.

It is generally recognized that the development of an expert system rests on satisfying the following prerequisites (ref. 2):

(1) There must be at least one human expert acknowledged to perform the task well using special knowledge, judgment, and experience. Within the Interdisciplinary Research Office (IRO) at NASA's Langley Research Center, we have access to a renowned expert in the optimization field as well as several other regular, in-house users of optimization techniques. In addition, several optimization experts are currently working under grants and contracts for the IRO, including the author of ADS.

(2) The expert must be able to explain the special knowledge and experience, and the methods used to apply them to a particular problem. Many of the available experts also teach optimization techniques and are accustomed to explaining difficult concepts to novices.

(3) The task must have a well-bounded domain. The domain for this task is bounded by the number of possible combinations available for ADS.

(4) The problem does not require common sense and should take an expert from a few minutes to a few hours to solve. Common sense will not be very helpful to an engineer in selecting the best combination of options. Experience with ADS has shown that a choice can almost always be made within the required time frame.

(5) The problem should be nontrivial but tractable, with promising avenues for incremental expansion. Within the subject domain, the problem is nontrivial and tractable. It can be expanded to accomodate new combinations of options as strategies, optimizers, and one-dimensional searches are added to ADS.

Since the present application appeared to meet the required prerequisites, it was decided to proceed with building an expert system for ADS. The first step was to begin acquiring the expert knowledge for the knowledge base.

KNOWLEDGE ACQUISITION

The acquisition of the expert knowledge for the knowledge base portion of the expert system proved to be difficult as discussed in reference 3. Initially, the first author, acting as knowledge engineer, met with the in-house expert and the second author to discuss what rules they would follow in choosing from a small subset of the possible combinations of options in ADS. These rules were then coded into production rules similar to those described in reference 4. Concurrently, a questionnaire was sent to engineers, knowledgeable in optimization, to solicit their input. This did not prove to be beneficial because no usable knowledge was returned. This could be attributed to the fact that many of the engineers surveyed use only one combination, and use it as a "black-box." The main benefit from the questionnaire was that it reinforced our belief that an expert system would be very beneficial to most engineers using ADS. The remainder of the knowledge that currently resides in the knowledge base came from two primary sources. First, the second author performed a literature search to identify rules for each strategy, optimizer, and one-dimensional search option in ADS. Second, the developer of ADS included a set of rules in his documentation (ref. 1). The collection of rules from these sources provided sufficient rules to begin developing the expert system.

THE EXPERT SYSTEM FOR ADS

EXADS, the expert system developed for ADS, like most expert systems, consists of two major components, an inference engine and a knowledge base. These two components are discussed in detail in this section.

The Inference Engine

The inference engine described in reference 4 was used to help us get started with this project. This engine is written in LISP. Users respond to a question with either a "yes" or a "no." This proved to be very helpful in learning some of the basics about building expert systems with production (if-then) rules. However, the inference engine lacked some useful capabilities, such as confidence levels and dealing with uncertainty. About the time a search was begun to find or develop a new inference engine, one, which appeared to meet our needs, was delivered to another organization at Langley. This engine, called AESOP (An Expert System engine Operative with Probabilities), was developed under a NASA grant. AESOP is a rule-based inference engine written in LISP which can make decisions about a particular situation using user-supplied hypotheses (potential solutions), rules (guidelines to finding the correct solution), and answers to the questions drawn from the rules. It is a backward chaining problem solver, i.e. works from hypotheses to facts.

One of the important features of AESOP is that questions do not have to be answered with only a "yes" or a "no". A confidence level ranging from 0 (no) to 10 (yes) may be given instead, depending upon how certain the user is about a particular piece of information. The user can also respond with a "maybe" (5), "probably" or "likely" (7), "not-likely" (3), or "don't know" (dk). If the user responds "dk", AESOP checks the knowledge base to determine if rules exist that deal with this uncertainty. The "dk" capability is very powerful, allowing several levels of rules to exist with each level containing more specific rules to help decide the appropriate response to a higher level question. The rules must be structured top-down so that the rules for resolving a "dk" response are located below the original rules because AESOP will not jump backwards through the rules to resolve a "dk" response. If no rules exist for resolving a "dk" response, the default is "no." However, the user is given an opportunity to override the default.

AESOP has an "explanation" feature and a "help" command. When any hypothesis reaches a confidence level of 90% or more, it is deemed confirmed as the best choice and displayed to the screen. If all rules have been exhausted and no hypothesis has been confirmed, then the status of all hypotheses with a confidence level greater than 10% are displayed on the screen. The user can choose a combination of options based on the confidence levels or examine the explanation of the hypotheses that appear promising and determine why they failed to reach the 90% level. The "help" command displays the choices currently available to a user.

Other reasons for choosing AESOP are its availability and its portability. The program is in the public domain and available from COSMIC, NASA's software dissemination center at the University of Georgia. It executes on both the DEC-VAX (Franz LISP) and the IBM PC/XT (IQ-LISP) at Langley, and should be portable to any computer running a version of LISP with little or no modification.

The Knowledge Base

The knowledge base contains: (1) the rules to be used in the decision making; (2) the hypotheses to be investigated; (3) the list of mutually exclusive rules (opposites) to avoid giving essentially the same information twice; and (4) detailed information about specific rules. The general format of the knowledge base is:

```
(setq *rules '(
      ((hypothesis1)
            (or
               (confidence-level1   rule1)
               (confidence-level2   rule2)
                         .
                         .
                         .
               (confidence-levelN   ruleN))
      ((hypothesis2)
            .
            .
            .
      ((hypothesisN)
            (or
               (confidence-level1   rule1)
               (confidence-level2   rule2)
                         .
                         .
                         .
               (confidence-levelN   ruleN))))

(setq *hypotheses '(
     (hypothesis1)
     (hypothesis2)
         .
         .
         .
     (hypothesisN)))

(setq *opposites '(
     ((ruleA) (ruleB))
     ((ruleC) (ruleD))
         .
         .
         .
     ((ruleY) (ruleZ))))

(setq *details '(
     ((ruleA) (details about ruleA))
     ((ruleB) (details about ruleB))
         .
         .
         .
     ((ruleZ) (details about ruleZ))))
```

The confidence level assigned to each rule by the expert works in conjunction with the confidence level the user expresses about how well the rule applies to his or her problem. This combination is divided by 10 before it is stored. For example, if the expert has placed a confidence level of 8 on a rule and the user responds to a question about that rule with a confidence level of 5, then the combined confidence level is (8*5)/10 or 4. The confidence level for the hypothesis is computed by again dividing by 10 resulting in a value between 0 and 1 given as a percentage. When this confidence level exceeds 90%, the hypothesis is deemed confirmed as the best choice. The rules can be expressed in two ways, as "and" or as "or." An example of an "and" rule follows:

```
((strategy 1 optimizer 2 and 1d-search 3
          is best for *)
  (or
    (10 and (* requires a strategy of 1)
            (* requires an optimizer of 2)
            (* requires a 1d-search of 3))))
```

The * serves as a "wild card" for describing the problem on which the user is working and is discussed in more detail below. The 10 shows that the expert is certain that this hypothesis is correct if the facts enclosed in the parentheses are true. The user responds with a confidence level to each of the three questions included in the "and" portion of the rule. For example, the engine would ask: "Does * require a strategy of 1?" The user responds "l" (or likely or 7). If the user responds "y" (yes or 10) to the remaining two questions, the 10 preceding the "and" would only be multiplied by 7 because in an "and" rule the engine chooses the minimum of all responses. After the two divisions by 10 the confidence level for this hypothesis is at 70%.

An "or" rule looks like the following:

```
((* requires a strategy of 4)
  (or
     (8 * requires starting from a feasible
        design space)
     (7 * is a second order problem)
     (6 * has more than 50 design variables)
     (5 analytical gradients are available
        for *)))
```

Each of the "or" rules works in combination with the others. A confidence level is computed for the first rule by multiplying the confidence level of the expert by the confidence level of the user and dividing by 10. The confidence level of the second rule is computed likewise. These two confidence levels are then combined according to the computation

```
new_confidence = confidence1 +
((1-confidence1) * confidence2)
```

This combinatorial process (replacing confidence1 with new_confidence and confidence2 with the confidence level of the next rule) is repeated until all of the rules in the "or" have been used or, after another division by 10, a 90% confidence level has been attained for that hypothesis.

AESOP allows the knowledge engineer to store an initial prompt function in the knowledge base. This function, which is contained on a file loaded into memory by the inference engine, lets the user describe the problem being solved (ex. building-bridges). This description replaces the wildcard (*) in the remainder of the knowledge base.

PROBLEMS ENCOUNTERED AND THEIR SOLUTIONS IN DEVELOPING THE EXPERT SYSTEM

Testing and verification of the knowledge base proceeded along several different lines. First, the authors tried numerous test cases to test the validity of the rules and the associated confidence levels. This step eliminated most of the simple problems and errors. Next five students taking a graduate level optimization course were invited to test the system. These students each had an optimization problem to solve and used EXADS as would a typical optimization novice. They responded to questions with answers based on their particular problem. In two cases, a combination of options for ADS was deemed confirmed as the best choice for that student's problem. In the other three cases, no hypothesis was confirmed, but the students were given 2-4 combinations from which to choose. New problems were discovered and solved at this step also. Finally, the EXADS system will be sent to ADS users for testing and evaluation in the field. It is expected that new and modified rules and confidence levels will be added as a consequence of the evaluation by the ADS users.

The remainder of this section discusses the problems discovered by the authors and the students during the first two levels of testing and evaluation. The solutions to these problems are also discussed because these problem solving experiences may be important to potential developers of expert systems.

The original knowledge base contained 98 hypotheses (the possible combinations for ADS) and about 550 rules in a single file. Many of the "or" rules were combined into "and" rules, thereby reducing the total number of rules to near 200. AESOP works with frames (ref. 4). A frame is a list of properties about an entity, similar to relations and attributes in a relational data base management system. To reduce the excessive amount of time AESOP was taking to create the frames, it was decided to divide the rules into three separate categories depending upon whether the problem to be solved is unconstrained, constrained, or constrained but being treated as unconstrained. The rules corresponding to each category were then written as separate files. The possible combinations do not overlap among these files, although many of the rules do. In addition to reducing the amount of time to generate the frames, the user is nót asked questions that cannot possibly pertain to the problem.

To handle these three categories of problems, the initial prompt function was expanded to ask additional questions about which category is to be used. Once the category is determined, then the appropriate file is loaded from the knowledge base. After initialization of certain variables, the initial prompt function queries the user for a description of the problem to be solved. This description replaces the "*" wildcard in the rules as before. A diagram of EXADS is shown in figure 1. Because the inference engine, the knowledge base, and the initial prompt function are all written in LISP, there is no way to distinguish among the three once all of the files are loaded into memory.

Because of the way the original AESOP system was written, the user was required to answer redundant questions. "Remember" and "recall" features from the inference engine in reference 4 were added to AESOP to let the inference engine store and recall user responses thus minimzing the number of questions a user is asked. In addition, there are a number of rules in the knowledge base which are exact opposites, such as (iterative analysis is available for *) as opposed to (iterative analysis is not available for *). An "opposites" feature was added so that the user would not be asked about both rules. Using this feature, when a user responds to a question and its opposite is in the knowledge base, the corresponding inverse confidence level is given to its opposite. For example, if a user responds "probably" (7) to a question, then its opposite is automatically given a "not-likely" (3) confidence level. The students found that responding to rules stated negatively proved to be a problem. The addition of the "opposites" feature and a slight reordering of the rules and hypotheses seems to have helped here because, except in rare occasions, the user now only sees a positively stated question.

The addition of the "opposites" feature to AESOP led to a problem with the default of "no" for a "dk" (don't know) response. Since "dk" defaults to a "no," its opposite defaults to a "yes" without the user ever seeing the question. Obviously the opposite of "dk" is not "yes." The engine was modified so that the opposite of "dk" is "dk," with both defaulting to "no." A feature to allow the user to override the default was added.

The authors found that the "explanation" feature tended to generate quite a bit of output that really did not help in understanding what rules were being used to find the best combination. All combinations of strategy, optimizer, and one-dimensional search options with their rules were displayed even though there may have been no evidence that that combination was best. To reduce the amount of output, only hypotheses and rules having a confidence level of greater than 10% are now displayed, eliminating the hypotheses and rules with little or no evidence.

Because some users may not be familiar with the optimization terminology used in the rules, a "detail" feature was added. This feature allows users to type "detail" if they do not understand a rule, and details for that rule are displayed on the screen if they are available in the knowledge base.

CONCLUDING REMARKS

An expert system, called EXADS, has been developed to aid a user of the Automated Design Synthesis (ADS) general-purpose optimization program. Because of the general-purpose nature of the program, it is difficult for a non-expert to select the best choice of a combination of strategy, optimizer, and one-dimensional search options from among the many combinations which are available. An expert system for ADS (EXADS) consisting of an inference engine (AESOP), and a knowledge base of approximately 200 rules was developed to aid an engineer in determining the best combination based on the his or her knowledge of the problem and the expert knowledge stored in the knowledge base. After in-house testing and verification, EXADS is to be delivered to ADS users for their evaluation. After their evaluation, the system will be modified to correct any errors, problems, or "holes" in the knowledge base. It will then be expanded periodically to account for any new ADS options.

REFERENCES

1. Vanderplaats, G. N., "ADS - A FORTRAN Program For Automated Design Synthesis - Version 1.00," NASA CR 172460, October 1984.

2. Gevarter, W. B., "An Overview Of Expert Systems," NBSIR 82-2505, May 1982.

3. Hayes-Roth, F., Waterman, D. A., and Lenat, D. B., Building Expert Systems , Addison-Wesley Publishing Company, Inc., 1983.

4. Winston, P. H., and Horn, B. K. P., LISP , Addison-Wesley Publishing Company, Inc., 1981.

Table 1: Strategy Options

No.	Strategy To Be Used
0	None. Go directly to optimizer.
1	Sequential unconstrained minimization using the exterior penalty function method.
2	Sequential unconstrained minimization using the linear extended interior penalty function method.
3	Sequential unconstrained minimization using the quadratic extended interior penalty function method.
4	Sequential unconstrained minimization using the cubic extended interior penalty function method.
5	Augmented Lagrange Multiplier method.
6	Sequential linear programming.
7	Method of centers (method of inscribed hyperspheres).
8	Sequential quadratic programming.

Table 2: Optimizer Options

No.	Optimizer To Be Used
0	None. Go directly to the search. This option should only be used for program development.
1	Fletcher-Reeves algorithm for unconstrained minimization.
2	Davidon-Fletcher-Powell (DFP) variable metric method for unconstrained minimization.
3	Broydon-Fletcher-Goldfarb-Shanno (BFGS) variable metric method for unconstrained minimization.
4	Method of feasible directions (MFD) for constrained minimization.
5	Modified method of feasible directions for constrained minimization.

Table 3: One-dimensional Search Options

No.	One-dimensional Search To Be Used
1,5	Find the minimum of an (1) unconstrained or (5) constrained function using the Golden Section method.
2,6	Find the minimum of an (2) unconstrained or (6) constrained function using the Golden Section method followed by polynomial interpolation.
3,7	Find the minimum of an (3) unconstrained or (7) constrained function by first finding bounds and then using polynomial interpolation.
4,8	Find the minimum of an (4) unconstrained or (8) constrained function by polynomial interpolation/extrapolation without first finding bounds on the solution.

Table 4: Program Options

	OPTIMIZER				
STRATEGY	1	2	3	4	5
0	X	X	X	X	X
1	X	X	X	0	0
2	X	X	X	0	0
3	X	X	X	0	0
4	X	X	X	0	0
5	X	X	X	0	0
6	0	0	0	X	X
7	0	0	0	X	X
8	0	0	0	X	X
ONE-D SEARCH					
1	X	X	X	0	0
2	X	X	X	0	0
3	X	X	X	0	0
4	X	X	X	0	0
5	0	0	0	X	X
6	0	0	0	X	X
7	0	0	0	X	X
8	0	0	0	X	X

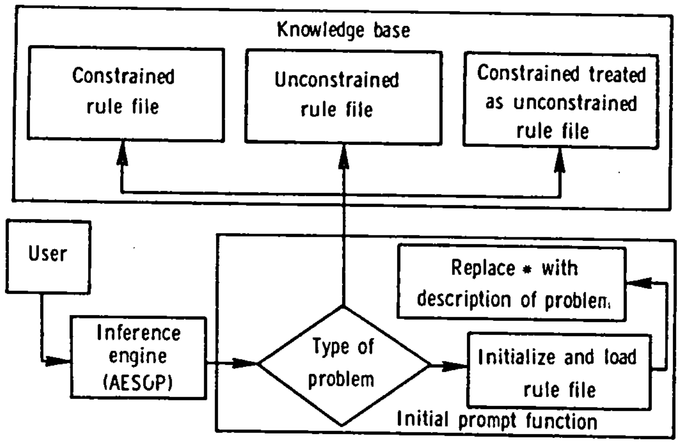

Figure 1: Diagram of EXADS

1. Report No. NASA TM-86329	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle An Expert System for Choosing the Best Combination of Options in a General-Purpose Program for Automated Design Synthesis		5. Report Date March 1985
		6. Performing Organization Code 506-83-43-02
7. Author(s) James L. Rogers, Jr. Jean-Francois M. Barthelemy		8. Performing Organization Report No.
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665		10. Work Unit No.
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		13. Type of Report and Period Covered Technical Memorandum
		14. Sponsoring Agency Code

15. Supplementary Notes

To be presented at 1985 International Computers in Engineering Conference and Exhibition, August 4-8, 1985, Boston, MA

16. Abstract

An expert system has been developed to aid a user of the Automated Design Synthesis (ADS) general-purpose optimization computer program in selecting the best combination of strategy, optimizer, and one-dimensional search options for solving a problem. There are approximately 100 such combinations available in ADS. The knowledge base contains over 200 rules, and is divided into three categories: constrained problems, unconstrained problems, and constrained problems treated as unconstrained problems. The inference engine is written in LISP and is available on DEC-VAX and IBM PC/XT computers.

17. Key Words (Suggested by Author(s)) Expert system, optimization, LISP, knowledge base, inference engine, automated design synthesis	18. Distribution Statement Unclassified - Unlimited Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	7	A02