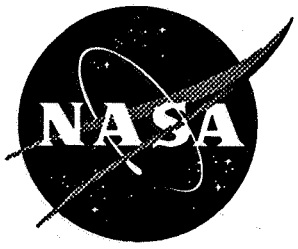
NASA Technical Memorandum 110247



Integrating a Genetic Algorithm Into a Knowledge-Based System for Ordering Complex Design Processes

James L. Rogers
Langley Research Center, Hampton, Virginia

Collin M. McCulley and Christina L. Bloebaum
State University of New York at Buffalo, Buffalo, New York

April 1996

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23681-0001

INTEGRATING A GENETIC ALGORITHM INTO A KNOWLEDGE-BASED SYSTEM FOR ORDERING COMPLEX DESIGN PROCESSES

James L. Rogers
NASA Langley Research Center

Collin M. McCulley
Christina L. Bloebaum
State University of New York at Buffalo

Abstract. The design cycle associated with large engineering systems requires an initial decomposition of the complex system into design processes which are coupled through the transference of output data. Some of these design processes may be grouped into iterative subcycles. In analyzing or optimizing such a coupled system, it is essential to be able to determine the best ordering of the processes within these subcycles to reduce design cycle time and cost. Many decomposition approaches assume the capability is available to determine what design processes and couplings exist and what order of execution will be imposed during the design cycle. Unfortunately, this is often a complex problem and beyond the capabilities of a human design manager. A new feature, a genetic algorithm, has been added to DeMAID (Design Manager's Aid for Intelligent Decomposition) to allow the design manager to rapidly examine many different combinations of ordering processes in an iterative subcycle and to optimize the ordering based on cost, time, and iteration requirements. Two sample test cases are presented to show the effects of optimizing the ordering with a genetic algorithm.

1. Introduction

Many engineering systems are large and multidisciplinary and require a complex design cycle. Before a design cycle begins, the possible couplings among the design processes must be determined. After these possible couplings have been defined, a design cycle can be decomposed to identify its multilevel structure. The Design Manager's Aid for Intelligent Decomposition (DeMAID) is a knowledge-based software tool for ordering the sequence of design processes and for identifying a possible multilevel structure for a design cycle (Rogers 1989). The DeMAID software displays the processes in a design structure matrix format (DSM) in which an element on the diagonal is any process that requires input and generates an output (Steward 1981). Off-diagonal elements indicate a coupling between two processes. The primary advantage of the DSM over display tools such as Program Evaluation and Review Technique (PERT) or process flowcharts is the ability to group and display the iterative subcycles that are commonly found in the design cycle. After the iterative subcycles have been determined, their processes must be ordered in a manner that will produce a design in the least time and at minimum cost. The original DeMAID software employs a knowledge base to handle this task; however, the knowledge-based approach only examines a limited number of orderings, which provides the user a starting point from which to interactively search for the optimum sequence. This paper introduces a genetic algorithm (GA) capability that has been added to DeMAID. This GA examines a large number of orderings of processes in each iterative subcycle and optimizes the orderings based on cost, time, and iteration requirements.

2. Design Structure Matrix

The DSM is used to display the sequence of processes (Steward 1981). A sample DSM is shown in Figure 1. In the DSM, the processes are shown as numbered boxes on the diagonal. Output from a process is shown as a horizontal line that exits a process box, and input is shown as a vertical line that enters a process box. The off-diagonal squares that connect the horizontal and vertical lines represent couplings between two processes. Couplings in the upper triangle portion of the DSM represent feedforward data; couplings in the lower triangle part of the matrix represent feedback data. A feedback implies an iterative process in which an initial guess must be made. The knowledge base within DeMAID which is written with the C Language Integrated Production System (CLIPS, Giarratano and Riley 1989) orders the processes to eliminate as many feedbacks as possible. However, in many cases, not all of the feedbacks can be eliminated. If any feedbacks remain, DeMAID groups the processes into iterative subcycles called circuits. In Figure 1, processes 1-3, 5-19, 21-25, and 26-29 are grouped into circuits.

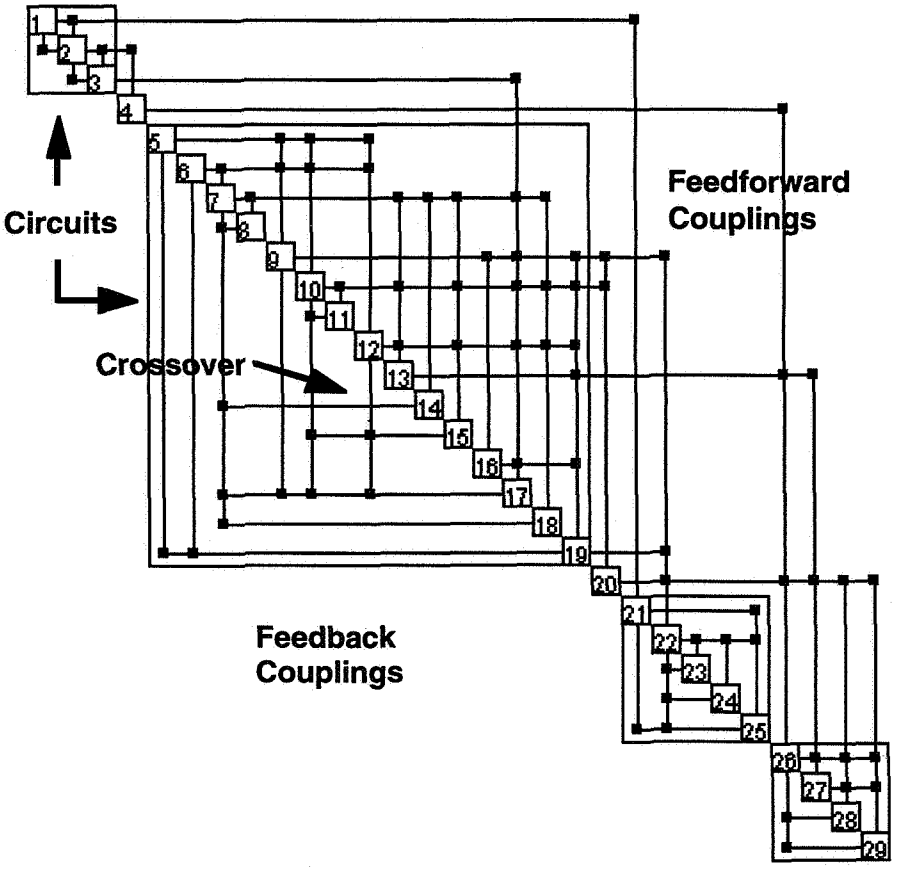


Figure 1. A design structure matrix.

The DeMAID software also identifies crossovers. Crossover, in this context, occurs when feedback from one process crosses that of another process without an exchange of data through the intersection (no off-diagonal square). Crossovers are only defined in terms of feedbacks. For example, in Figure 1 a crossover occurs when the feedback from process 14 to process 7 crosses the feedback from process 17 to process 12. Crossovers

should be avoided if possible because they obscure when to end one iterative loop and begin another. The DSM shown in Figure 1 contains 20 feedbacks and 3 crossovers.

In the original version of DeMAID, a knowledge base was used to minimize feedbacks and group processes into circuits. Crossovers were identified but were not minimized. No time factors, cost factors, or iteration factors (i.e. the number of iterations required for convergence) were applied. After the circuits were identified, DeMAID attempted to minimize the feedbacks within a circuit. In most cases, although more than one ordering could produce the minimum amount of feedbacks, only one ordering was identified.

A large circuit such as the one shown in Figure 1 that contains processes 5-19 can be very expensive to converge because the iterative loops defined by the feedbacks are nested, which require numerous executions of potentially expensive processes. Thus, a new technique is needed that rapidly examines many different orderings of processes within a circuit and selects the best ordering based on cost, time and iteration requirements. The GA capability that has been added to DeMAID meets this need.

3. Coupling Strengths

In the original version of DeMAID, a coupling either existed or not. The strength of the coupling could not be quantified. In the latest version of DeMAID, seven levels are used to quantify coupling strengths They are: extremely weak, very weak, weak, nominal, strong, very strong, and extremely strong. These strengths can be supplied by the user or they can be determined through sensitivity analysis (Bloebaum 1992; Rogers and Bloebaum 1994) and quantified according to rules in the knowledge base. The rules for quantifying are based on a statistical analysis of the normalized sensitivities. Recommendations are made as to which processes and couplings might be removed (or temporarily suspended) from the problem without a loss of solution accuracy.

The rules for removing or retaining processes are listed here. All processes with at least one coupling of nominal strength or greater are retained. Processes with only extremely weak coupling strengths are recommended for removal. Other recommendations depend on the relationships among the processes. For example, in figure 1, if the maximum coupling strength of process 19 is very weak, then in order to be retained, one of the processes to which it is coupled (process 5, 6, or 22) must have an extremely strong coupling strength. Otherwise, process 19 is recommended for removal. Similar rules exist for removing or retaining couplings.

The DeMAID software also has the capability to display the DSM with color codings for coupling strengths. To eliminate the use of black boxes to represent couplings in the off-diagonal elements, a color scheme can be used (i.e. extremely weak, red; very weak, pink; weak, yellow; nominal, green; strong, light blue; very strong, blue; and extremely strong; black). The user can interactively move processes along the diagonal to place the weaker couplings which require fewer iterations for convergence into the feedback positions.

After the complexity of the problem has been reduced by removing processes and/or couplings, another examination can be made of the remaining circuits. An iteration factor is identified that relates the coupling strengths to the number of iterations required for convergence. The default values are shown in Table 1. The user can override these default values if necessary. If coupling strengths are not available, the assumed number of iterations for computational purposes is 1.

TABLE 1. Relation of coupling strengths to iterations required for convergence.

Coupling Strength	Default Iterations
Extremely weak	2
Very weak	3
Weak	4
Nominal	5
Strong	6
Very strong	7
Extremely strong	8

4. Cost and Time Requirement Calculation

Rules were added to the DeMAID knowledge base to determine the total cost and time required for a given design process. The DSM in Figure 2 is a circuit taken from a larger design project. Each process has been assigned a cost and a time (units depend on the user). The numbers in the left-hand column correspond to the original process numbers assigned by the user. The sequence of processes has been reordered by DeMAID. This circuit contains eight feedbacks and no crossovers. Coupling strengths were not used to estimate the required number of iterations for convergence for this problem; thus each iteration factor is 1.

Numerous nested iterative processes are evident within this circuit. The DeMAID software sums the time and cost of each process contained in a feedback loop and multiplies those sums by the iteration factor for the feedback. For example, the costs and times for processes 9-18 would be summed and multiplied by the iteration factor (1 in this case) for the feedback coupling from process 18 to process 9. The same would be accomplished for processes 2-19 using the iteration factor (again 1) for the feedback from process 19 to process 2. This computation continues until the contributions from all eight feedbacks have been summed. The drawback to this capability is that it only examines one ordering and makes no attempt to optimize the ordering based on cost and/or time. Thus, a decision was made to complement the knowledge base approach in DeMAID with a GA. This GA examines a large number of orderings of processes in each iterative subcycle and optimizes the ordering based on cost, time, and iteration requirements.

###	Time	Cost
11	30	10
18	40	20
21	10	20
22	20	30
20	20	10
19	30	10
1	50	10
23	30	40
17	50	30
7	30	40
8	40	30
2	40	20
6	20	50
14	20	40
13	10	30
12	20	20
3	30	30
15	30	50
16	40	40
5	10	50
4	20	40
10	40	10
9	50	20

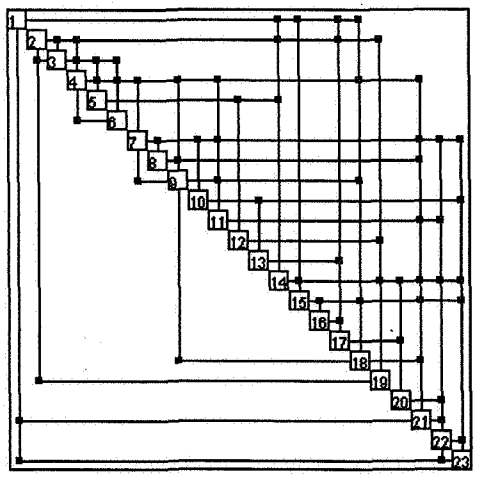


Figure 2. A design structure matrix minimized for feedbacks and crossovers.

5. Genetic Algorithm

The use of GA's has been instrumental in achieving good solutions to discrete optimization problems that have not been satisfactorily addressed by other methods (Goldberg 1989). Because of the discrete nature of the sequencing problem, this solution technique has proved useful in solving this problem (Syswerda, 1990). A population of design points that are coded as finite-length, finite-alphabet strings is searched by the GA. Successive populations are produced primarily by the operations of selection, crossover, and mutation. The selection operator determines those members of the population that survive to participate in the production of members of the next population. Selection is based on the value of the fitness function, or the fitness of the individual members, such that members with greater fitness levels tend to survive. Crossover is the recombination of traits of the selected members, called the mating pool, in the hope of producing a child with better fitness levels than its parents. Crossover is accomplished by swapping parts of the string into which these design points have been coded. The final operation, mutation, prevents the search of the space from becoming too narrow. After the production of a child population, this operator randomizes small parts of the resulting strings, with a very low probability that any given string position will be affected.

Frequently, a binary coding is used with the GA; the values of the design variables are coded as binary numbers and then concatenated. While this approach works well with numerical problems, it is not efficient for the sequencing problem (Altus et al 1995; McCulley and Bloebaum 1994). The GA portion of DeMAID uses a direct representation of the order as a coding of an $\underline{n}$-process system, with each integer 1 through $\underline{n}$ used only once. For example, the string

[5 3 4 2 1]

represents the five-process DSM shown in Figure 3, in which the order from the top left corner of the DSM to the bottom right corner is 5, 3, 4, 2, and 1.

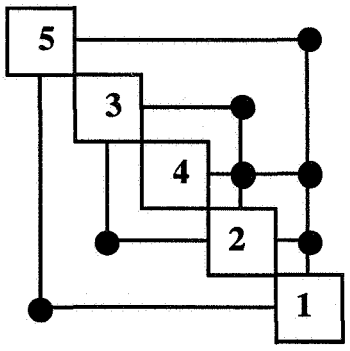


Figure 3. Five-process design structure matrix.

Selection, which only requires the use of the fitness function, is unaltered by this choice of coding. However, special operators for crossover and mutation must be used because these operators operate directly on the strings. The concern is that the result after a GA crossover or mutation operation must be a valid order (i.e. no repeated or missing processes). Valid orders cannot always be guaranteed with arbitrary switching of string information between or within strings.

Selection is accomplished by the *tournament selection* operator. To fill the mating pool, two strings are randomly selected from the parent pool and compared; the one with greater fitness is included in the mating pool. Crossover is accomplished by *position-based* (Syswerda, 1990) crossover as shown in Figure 4. Several processes (i.e. 1, 4, 5, and 6) are chosen from the first parent and placed in the same positions in the child string. Then, the processes (i.e. 2, 3, and 7) that were not taken from the first parent are taken from the second parent to fill the holes in the child string in the order in which they appear in the second parent. The result is a complete string with one and only one copy of each process number.

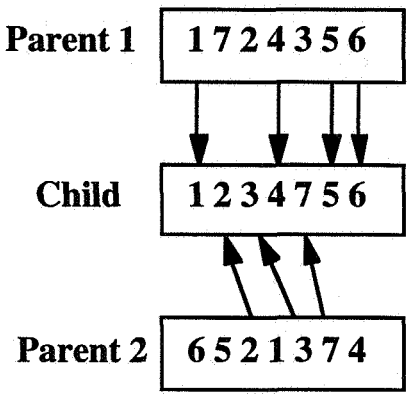


Figure 4. Position-based crossover.

Mutation is accomplished through the *order-based* (Syswerda, 1990) mutation operator, as shown in Figure 5. Each string position is polled; if a given string position (i.e. position 2) is selected to undergo mutation, then its content is swapped with a randomly selected position (i.e. position 4) in the same string.

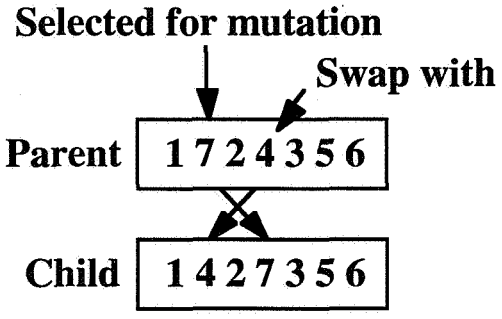


Figure 5. Order-based mutation.

In addition to minimizing feedbacks and crossovers, the fitness function for the GA in DeMAID can be used to determine the minimum cost and time required for convergence of each circuit. The GA sums the time and cost of each process contained in a feedback loop and multiplies those sums by the iteration factor for the feedback to obtain the total cost and time to converge a circuit. The user-definable weights determine the relative importance of each of the major components of the fitness function. The fitness function is:

fitness=1.0/((wf*f+wc*c+wtime*time+wcost*cost)**4)

where f is the number of feedbacks, c is the number of crossovers, time is the total time required to converge the circuit, cost is the total cost to converge the circuit; and wf, wc, wtime, and wcost are user-definable weights. For the simple tournament selection, the relative scale of this fitness function is not important. Only the relation of the values (i.e. whether one fitness function is larger than the other) matters.

Each circuit is passed to the GA to optimize individually. A window (Figure 6) is displayed for each circuit. The window indicates the default values for the GA. The GA begins with a randomly generated initial population of a size determined by the user and proceeds from generation to generation by applying the three previously described operations.

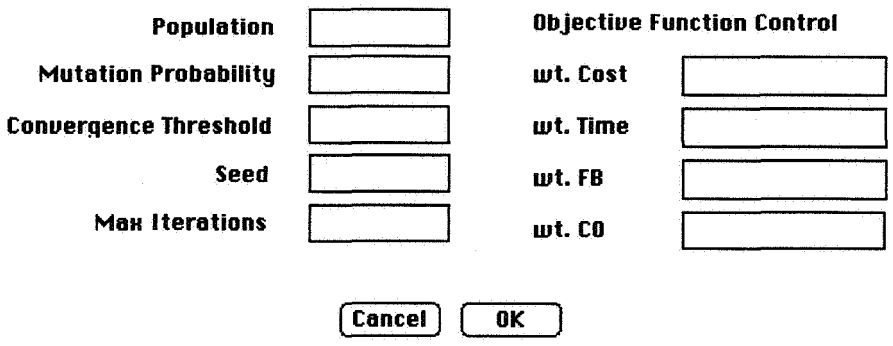


Figure 6. Window for setting genetic algorithm parameters.

The following parameters, shown in Figure 6, are available with their defaults in parentheses:

- Population (100) - population size
- Mutation Probability (1.0) - mutation probability in percent, default is 1%
- Convergence Threshold (0.9) - a converged population is one for which the average fitness is at least convThresh of the best fitness, with the best fitness seen so far (default is 90%)
- Seed (3818969) - seed for random number generator
- Max Iterations (500) - maximum number of iterations to find the best sequence
- wt. Cost (1.0) - cost weight
- wt. Time (1.0) - time weight
- wt. FB (1.0) - feedback weight
- wt. CO (1.0) - crossover weight

Convergence is achieved when the average fitness of a population rises above some user-defined percentage (convergence threshold) of the best fitness for that population. At that point, the member of the population with the best fitness is chosen as the optimal. After the GA has completed ordering all the circuits, a new DSM can be displayed to demonstrate the changes.

6. Sample Cases

The two examples below indicate the savings that can be obtained by reordering the sequence of modules. In the figures, each process is assigned a cost and a time (units depend on the user). The numbers in the left-hand column correspond to the original process numbers assigned by the user. The sequence of processes has been reordered by DeMAID. Each table displays the modules coupled by feedbacks (iterative loops) for the corresponding DSM with the number of iterations for the feedback coupling along with the total time and cost to converge each iterative loop.

The DSM in Figure 7 is a circuit taken from a conceptual design project. This circuit contains 24 feedbacks and 16 crossovers. Coupling strengths are used to estimate the number of iterations required for convergence.

###	Time	Cost
19	30	30
16	40	20
5	10	50
21	10	50
13	10	50
1	50	10
18	40	20
17	50	10
20	20	40
12	20	40
2	40	20
3	30	30
15	30	30
14	20	40
11	30	30
6	20	40
7	30	30
8	40	20
9	50	10
10	40	20
4	20	40
22	20	40

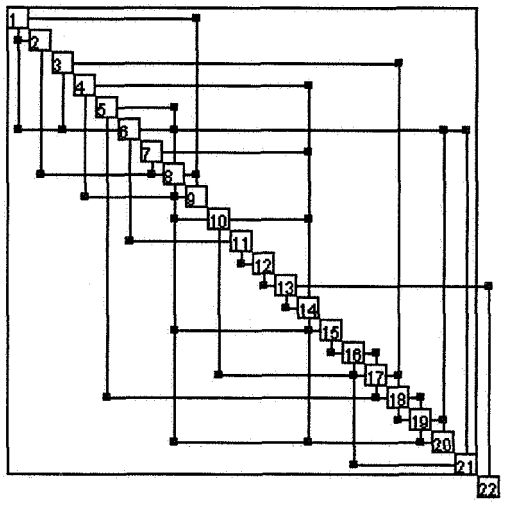


Figure 7. A design structure matrix for example 1

The DSM in Figure 8 contains the same set of processes with the same times, costs, and coupling strengths that are shown in Figure 7. However, the sequence of processes has been reordered and optimized by the GA and are different from those in Figure 7 as shown by the numbers in the left-hand column. This DSM contains eight feedbacks and no crossovers.

###	Time	Cost
2	40	20
1	50	10
5	10	50
4	20	40
6	20	40
7	30	30
8	40	20
9	50	10
10	40	20
13	10	50
3	30	30
11	30	30
12	20	40
17	50	10
16	40	20
19	30	30
20	20	40
21	10	50
18	40	20
15	30	30
14	20	40
22	20	40

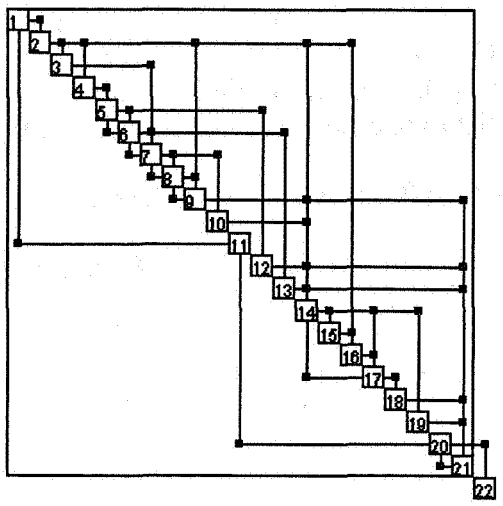


Figure 8. Reordering of the design structure matrix for example 1

Table 2 contains the data corresponding to Figure 7. The total design cycle for this DSM requires 21,340 time units and 19,640 cost units for completion.

TABLE 2. Time and cost for iterations in unordered design cycle for example 1.

To module	From module	Iterations	Time	Cost
1	2	8	560	400
1	6	4	600	840
2	8	8	1680	1680
3	6	2	160	320
4	9	7	1260	1260
5	18	6	2580	2460
6	11	8	1760	1120
7	8	6	540	180
8	9	2	140	100
8	10	8	720	720
8	15	4	960	960
8	20	7	2940	2520
10	17	8	1760	2080
11	12	5	350	250
12	13	3	180	180
13	14	6	300	420
14	15	8	400	560
14	20	4	920	760
15	16	6	300	420
16	17	7	350	490
16	21	8	1600	1280
17	18	8	560	400
18	19	6	540	180
19	20	2	180	60

Table 3 contains the data corresponding to Figure 8. The number of processes contained in the iterative loops has been reduced by reordering the sequence with the modified GA. With the same summing method described before, the total cost to complete the design cycle with this optimized ordering sequence is reduced from 19,640 to 3,950 units and the total time is reduced from 21,340 to 4,570 units.

TABLE 3. Time and cost for iteration in ordered design cycle for example 1.

To module	From module	Iterations	Time	Cost
1	11	5	1700	1600
5	6	7	350	490
6	7	8	560	400
7	8	6	540	180
8	9	2	180	400
11	21	3	960	1020
14	17	2	280	200

The DSM in Figure 2 is a circuit taken from another design project. The sequence of processes has been reordered by DeMAID. This circuit contains 8 feedbacks and no crossovers. Coupling strengths are not available therefore, the number of iterations required for convergence is set to 1.

The DSM in Figure 9 contains the same set of processes with the same times and costs, that are shown in Figure 2. However, the sequence of processes has been reordered and optimized by the GA and are different from those in Figure 2 as shown by the numbers in the left-hand column. This DSM also contains 8 feedbacks and no crossovers.

###	Time	Cost
11	30	10
21	10	20
18	40	20
22	20	30
19	30	10
20	20	10
14	20	40
2	40	20
16	40	40
13	10	30
15	30	50
17	50	30
1	50	10
23	30	40
8	40	30
4	20	40
7	30	40
9	50	20
6	20	50
12	20	20
3	30	30
5	10	50
10	40	10

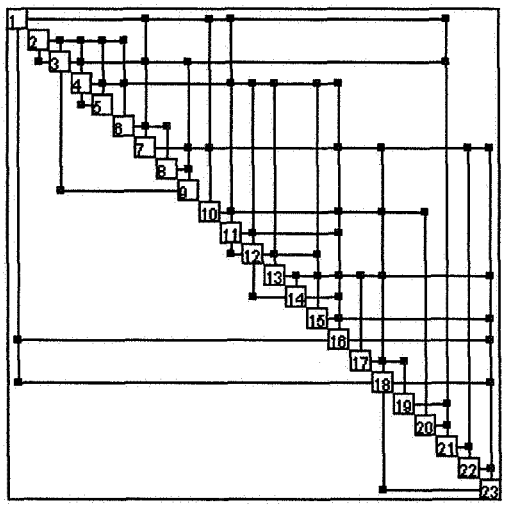


Figure 9. Reordering of the design structure matrix for example 2.

Table 4 contains the data corresponding to Figure 2. The total design cycle for this DSM requires 2,430 time units and 2,330 cost units for completion.

TABLE 4. Time and cost for iteration in unordered design cycle in example 2.

To module	From module	Iterations	Time	Cost
1	21	1	590	620
1	23	1	680	650
2	3	1	50	40
2	19	1	530	520
4	6	1	70	50
7	9	1	130	80
9	18	1	290	340
22	23	1	90	30

Table 5 contains the data corresponding to Figure 9. The number of processes contained in the iterative loops has been reduced by reordering the sequence with the modified GA. The total cost to complete the design cycle with this optimized ordering sequence is reduced from 2,330 to 1,510 cost units and from 2,430 to 1,730 time units.

TABLE 5. Time and cost for iteration in ordered design cycle for example 2.

To module	From module	Iterations	Time	Cost
1	16	1	480	430
1	18	1	560	490
2	3	1	50	40
3	9	1	210	170
4	5	1	50	40
11	12	1	80	80
12	14	1	130	80
18	23	1	170	180

In the above examples, the number of processes contained in the iterative loops has been reduced by reordering the sequence with the modified GA. This reordering requires about 1 minute on a Macintosh Quadra 700. In each case, the total cost and time in the design cycle are substantially reduced by reordering the sequence of the design processes.

7. Concluding Remarks

The Design Manager's Aid for Intelligent Decomposition (DeMAID) is a knowledge-based software tool for ordering the sequence of complex design processes, grouping iterative subcycles, and identifying a possible multilevel structure for a design cycle. The DeMAID software displays the processes in a design structure matrix format in which an element on the diagonal is any process that requires input and generates output. Off-diagonal elements indicate a coupling between two processes. The knowledge base in DeMAID attempts to eliminate all feedbacks in the design cycle. If all feedbacks cannot be eliminated, iterative subcycles are identified. If sensitivity analysis results are available, the DeMAID software can be used to examine the ordering within a subcycle to determine the strengths of the couplings between any two processes. These coupling strengths, when input to the knowledge base, determine those processes and couplings might be removed or temporarily suspended without sacrificing system solution accuracy. In addition, a relation

is formed between the coupling strengths and the number of iterations required to converge the iterative processes that are created by a feedback coupling.

In the original version of DeMAID, the optimal ordering of processes in an iterative subcycle was generated with a knowledge base, and only minimized the number of feedbacks. The primary drawback to the original method was that only a single ordering sequence could be examined at a time. Changes to the sequence were made interactively and then the costs and times were re-evaluated. This process was extremely slow with no guarantee that a reasonable optimum sequence would be found.

To remedy this problem, a genetic algorithm has been added to DeMAID to examine many possible orderings of the design processes in a design cycle. Each process can now have a time and/or cost associated with it. The GA in DeMAID examines the iterative subcycles to determine their time and cost. The GA fitness function is computed by summing the time and cost of each process contained in an iterative loop and multiplying the totals by the number of iterations required for convergence based on the coupling strength of the feedback coupling forming the loop. The GA determines the best ordering of each iterative subcycle by minimizing the total cost and time requirements, in addition to minimizing the number of feedbacks and crossovers for a particular ordering. This modification increases the likelihood that an optimal or near optimal sequence will be found.

Acknowledgments

The authors wish to acknowledge William J. LaMarsh II of the Computer Sciences Corporation for initially suggesting a genetic algorithm approach be incorporated into DeMAID. In addition, the second and third authors would like to acknowledge partial support of this work under NASA Grant NAG - 11599.

References

Altus, S. S.; Kroo, I. M.; and Gage, P. J. :1995, "A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems", ASME Paper 95-141.

Bloebaum, C.L.: 1992, "An Intelligent Decomposition Approach for Coupled Engineering Systems," proceedings of the Fourth AIAA/AF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH.

Giarranto, J. and Riley, G.: 1989, *Expert Systems Principles and Programming,* PWS - Kent Publishing Company, Boston.

Goldberg, D.: 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Co., New York.

McCulley, C. M.; and Bloebaum, C. L.: 1994, "Optimal Sequencing for Complex Engineering Systems Using Genetic Algorithms." Fifth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Panama City, FL.

Rogers, J. L.: 1989, "A Knowledge-Based Tool for Multilevel Decomposition of a Complex Design Problem," NASA TP-2903.

Rogers, J. L.; and Bloebaum, C. L.: 1994, "Ordering Design Tasks Based on Coupling Strengths", AIAA Paper No. 94-4326.

Steward, D. V.: 1981, *Systems Analysis and Management: Structure, Strategy and* Design. Petrocelli Books Inc.

Syswerda, G.: 1990, "Schedule Optimization Using Genetic Algorithms." *Handbook of Genetic Algorithms*, Van Nostran Reinhold, New York.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1996	Technical Memorandum

4. TITLE AND SUBTITLE

Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Processes

5. FUNDING NUMBERS

509-10-11-01

6. AUTHOR(S)

James L. Rogers
Collin McCulley
Christina L. Bloebaum

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

NASA TM-110247

11. SUPPLEMENTARY NOTES

To be presented at the Artificial Intelligence in Design Conference at Stanford University, June 24-27, 1996

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 05

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

The design cycle associated with large engineering systems requires an initial decomposition of the complex system into design processes which are coupled through the transference of output data. Some of these design processes may be grouped into iterative subcycles. In analyzing or optimizing such a coupled system, it is essential to be able to determine the best ordering of the processes within these subcycles to reduce design cycle time and cost. Many decomposition approaches assume the capability is available to determine what design processes and couplings exist and what order of execution will be imposed during the design cycle. Unfortunately, this is often a complex problem and beyond the capabilities of a human design manager. A new feature, a genetic algorithm, has been added to DeMAID (Design Manager's Aid for Intelligent Decomposition) to allow the design manager to rapidly examine many different combinations of ordering processes in an iterative subcycle and to optimize the ordering based on cost, time, and iteration requirements. Two sample test cases are presented to show the effects of optimizing the ordering with a genetic algorithm.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Design, Scheduling, Genetic Algorithm, Sensitivities, Knowledge-Based System, Coupling Strengths, Decomposition	13
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	