AIAA-2000-4484

A PARALLEL APPROACH TO OPTIMUM ACTUATOR SELECTION WITH A GENETIC ALGORITHM

James L. Rogers*
NASA Langley Research Center
Hampton, VA

ABSTRACT

Recent discoveries in smart technologies have created a variety of aerodynamic actuators which have great potential to enable entirely new approaches to aerospace vehicle flight control. For a revolutionary concept such as a seamless aircraft with no moving control surfaces, there is a large set of candidate locations for placing actuators, resulting in a substantially larger number of combinations to examine in order to find an optimum placement satisfying the mission requirements. The placement of actuators on a wing determines the control effectiveness of the airplane. One approach to placement maximizes the moments about the pitch, roll, and yaw axes, while minimizing the coupling. Genetic algorithms have been instrumental in achieving good solutions to discrete optimization problems, such as the actuator placement problem As a proof of concept, a genetic algorithm has been developed to find the minimum number of actuators required to provide uncoupled pitch, roll, and yaw control for a simplified, untapered, unswept wing model. To find the optimum placement by searching all possible combinations would require 1,100 hours. Formulating the problem as a multi-objective problem and modifying it to take advantage of the parallel processing capabilities of a multi-processor computer, reduces the optimization time to 22 hours.

INTRODUCTION

Conventional control devices like flaps and ailerons have gaps between the wing and the control surface that contribute to leakage and protuberance drag. This can be a source of aerodynamic noise and increased observability. Recent discoveries in material science have been used to create a variety of aerodynamic actuators which have great potential to enable entirely new approaches to aerospace vehicle flight control. Recent research has examined the feasibility of applying active structures technology to modify and control aircraft aerodynamics.[1] One option, synthetic jet actuators, potentially allows a seamless aircraft with no moving control surfaces, but rather hundreds of small ports (Figure 1) capable of aerodynamically morphing the shape of

---

* Senior Computer Scientist, ASCAC, MDOB

Copyright @2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the Copyright Owner.

the wing as needed. The cost and complexity of such a vehicle is obviously affected by the number and location of these ports. Given this possibility, tools and techniques must be developed to optimally select and distribute such devices over the aircraft surface.

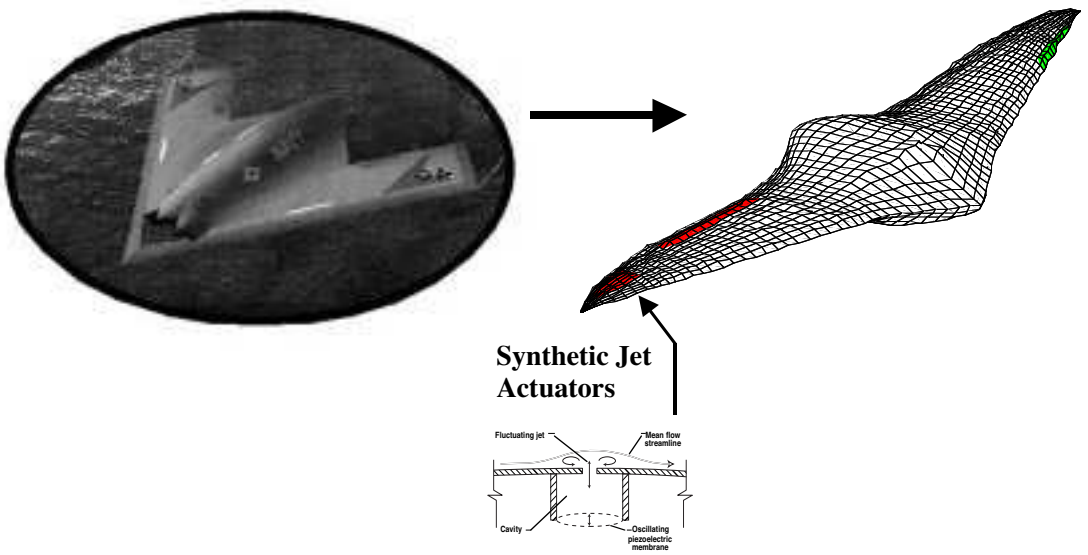

Figure 1. Seamless aircraft with synthetic jet actuators for control.

The placement of the actuators on a wing determines the control effectiveness of the airplane. For the wings-leveler autopilot, optimal placement means maximizing the moments about the pitch, roll, and yaw axes while minimizing the couplings among the moments.[2] For a typical wing, there is a large set of candidate locations for placing actuators. The larger the set, the larger the number of possible combinations to examine in order to find an optimum subset to satisfy the mission requirements and mission constraints.

In the present work, PMARC, a low-fidelity aerodynamic code for modeling complex three-dimensional geometries, is used to evaluate the pitch, roll, and yaw moments of a wing model.[3] The variable input for PMARC consists of an array containing the actuator placement locations. PMARC converts the moments about the three axes into non-dimensional coefficients. The moments include a length, therefore they must be divided by a quantity with a dimension of length as well as by the dynamic pressure and wing area. The length quantity is the mean aerodynamic chord for the pitching moment and the wing semispan for the rolling and yawing moments. PMARC then returns these values for use in later computations.

Before attempting to solve the more complex problem with the seamless aircraft, a simplified, untapered, unswept wing based on the NACA 0015 airfoil with 16 potential locations for actuators is used as the model (Figure 2).

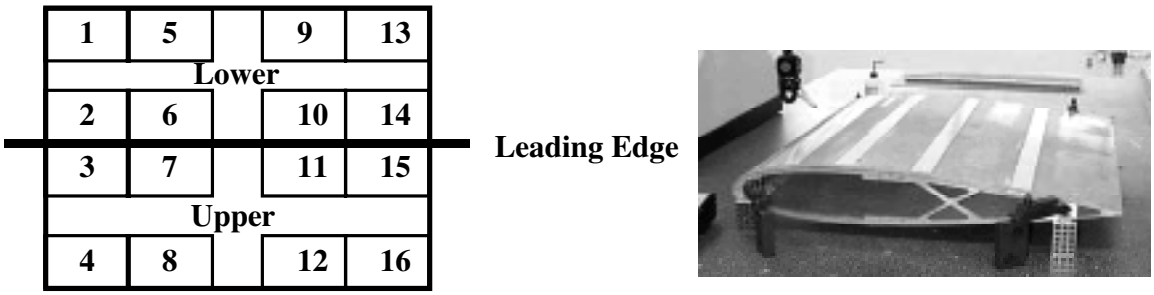

Figure 2. Wing model with 16 potential locations for actuators (unwrapped analysis model left, model right)

PMARC requires about two minutes for the initial, preprocessing call (extra time for setting up some matrices) and one minute for each subsequent call on a single processor Sun UltraSPARC™‡ computer. At this rate, the time required to evaluate all possible combinations for placing the 16 actuators is about 1,100 hours of computer time. If the number of actuators doubles then this time increases to the millions of hours. Clearly, new tools and techniques are needed to reduce this time. The application of a tool like a genetic algorithm (GA) appears to be an excellent choice for reducing the optimization time. GA's have been instrumental in achieving good solutions to discrete optimization problems, such as the actuator placement problem, that have not been satisfactorily solved by other methods.[4] The discrete nature of the actuator placement problem has been recognized previously, and the GA approach has been successfully applied to solve the placement problem for interior noise control.[5] The approach in this study differs in that the fitness of a population member is determined by calling PMARC to evaluate a multiobjective fitness function with several constraints.

A GA is very amenable to parallel processing.[6] After initial testing on the single processor computer, the code was ported to an SGI Origin 2000™ multi-processor computer to further reduce the time required to find the optimal set of actuators.

The results from this project demonstrate the effectiveness of applying a GA, on both single processor and in parallel on multi-processor computers, to optimize the selection and placement of actuators for an aircraft design problem.

THE GENETIC ALGORITHM

This GA uses a direct representation of the order as a coding of a wing with n actuator locations. Each member of the population consists of a string of numbers where each position (1 through n) in the string contains a one or a zero, designating whether the

‡ The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

actuator exists or not. For example, the member string

[0 1 1 0 1 0 0 1 0 1 0 0 1 0 1 1]

represents a wing with 16 possible actuator locations where actuators exist at locations 2, 3, 5, 8, 10, 13, 15, and 16. An initial population of members is randomly produced and evaluated. Successive populations are produced by the GA operations of selection, crossover, and mutation.

The fitness of a member is determined by counting the number of actuators in the string. The fitness is penalized if any constraints placed on the problem are violated. The penalty is determined by calling PMARC. The input for PMARC remains constant except for the array containing the actuator placements. The pitch, roll, and yaw moments (output from PMARC) are used in determining the penalty, if any, of a member. PMARC is called to evaluate each member of the population, and there may be many generations of populations.

The selection operation 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 for the individual members. Members with better fitness levels tend to survive and are placed in the mating pool. Selection is accomplished by the tournament approach where two members are randomly selected from the parent pool and compared according to their fitness; the member with the best fitness is included in the mating pool.

The crossover operation is the recombination of traits of the surviving members that have been placed in the mating pool, in the hope of producing a child with better fitness levels than its parents. This GA applies the single-point crossover technique as opposed to the uniform crossover method used by Simpson and Hansen.[7] Single-point crossover is accomplished by randomly selecting two parent members from the mating pool and randomly selecting a crossover point (Figure 3). To create members for the next generation population, each location in the first parent member before and including the crossover point is copied to the first child, and each location after the crossover point is copied to the second child. Then the opposite locations from the second parent are copied to each child.

Parent 1 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
Parent 2 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Randomly select crossover point of 4

Child 1 [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
Child 2 [0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1]

Figure 3. Single point crossover example.

To prevent the crossing of like members, called incest prevention in GA terminology, each string is given a unique identifier based on the power of two and the number of ones in the string.[8] The identifiers of each of the two members being crossed over are compared and if they have the same identifier, then the crossover does not take place.

The mutation operation prevents the search of the design space from becoming too narrow. After the production of a child

population, mutation randomizes small parts of the resulting members, with a very low probability that any given member location will be affected. Mutation is accomplished by polling each location in the member. A random number generator, along with a user-defined mutation parameter (default is .01 which means mutation occurs 1 time out of every 100 polls), is used to determine if that location is to be mutated. Only the default rate is used for this project. If mutation occurs and there is a one in the location, it is made a zero (Figure 4), and vice versa.

String before mutation
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

Randomly select mutation point of 6

String after mutation
[1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1]

Figure 4. Mutation example.

MULTI-OBJECTIVE FUNCTION

The problem statement for the application of a GA to actuator placement is, "Given 16 actuator locations, find the minimum number of actuators required to provide uncoupled pitch, roll, and yaw moments." The problem is divided into three distinct subproblems: uncoupled pitch, uncoupled roll, and uncoupled yaw. Each subproblem consists of a separate population and a separate fitness function creating a multi-objective function problem. The fitness function for each subproblem finds the minimum number of actuators required for that particular uncoupled maneuver. There are five constraints for each subproblem and each penalizes the fitness function when a violation occurs. For example, the constraints for the pitch subproblem are:

(1) penalize if the absolute value of the roll moment is greater than .001
(2) penalize if the absolute value of the yaw moment is greater than .001
(3) penalize if the number of actuators is less than 2
(4) penalize if the PMARC code does not converge
(5) penalize if the absolute value of the pitch moment is less than .001

Constraints (1) and (2) ensure that the maneuver is uncoupled. Constraint (3) allows engineers to specify a minimum number of actuators for safety reasons. Constraint (4) takes advantage of convergence information from the PMARC convergence parameter. Constraint (5) ensures that there is a minimum moment for each subproblem.

The ultimate goal is to find a set of actuators which can provide 3-axis control. To this end, there is also a composite fitness function determined from the members of the three subproblems which have satisfied constraints. The strings of the three individual maneuvers are combined with an "OR" function to form a composite configuration that indicates the actuators needed to perform each of the three uncoupled maneuvers. The composite fitness is the sum of the actuators in the composite configuration (Figure 5).

Pitch 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 1
Fitness 4
Roll 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0
Fitness 4
Yaw 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1
Fitness 4

Composite 0 0 0 1 0 1 1 1 1 1 1 0 1 0 0 1
Fitness 9

Figure 5. Composite configuration and fitness.

Examining the composite fitness one member at a time proved to be inefficient. For example, in one pass through the population, the two members selected for pitch and yaw each had an excellent fitness of 4, however, the one selected for roll had a poor fitness of 10. This resulted in a composite fitness of 16. Another pass found members with the following fitnesses: pitch = 10, roll = 4, yaw = 10, and composite = 18. If the pitch and yaw strings from the first pass were combined with the roll string from the second pass, then the composite fitness would be 9. Thus, arrays were added to save members of each subproblem that do not violate any constraints, and all possible good combinations of strings from the subproblems can now be evaluated to find the best combination for composite fitness. This is an expanded form of the elitist method which places only the best string from the previous population into the mating pool.[9]

Another change was made so that calls to PMARC would not be wasted. In the original program for example, if a call was made to PMARC for the pitch subproblem, no check was made to see if the results would be valid for either of the other subproblems. A change was made to the program so that regardless of the subproblem, the data from PMARC is now checked to see if its results do not violate constraints in any of the subproblems.

APPLICATION ON A SINGLE PROCESSOR COMPUTER

The population size for this application is 100 and there are different populations for pitch, roll, and yaw. The GA iterates through the 100 members of each subproblem, and evaluates the fitness of each individual member (300 function evaluations per generation).

Because absolute values are used in the subproblem fitness evaluations, the GA does not distinguish between pitch up or pitch down, roll left or roll right, and yaw left or yaw right. In this application, after 13 generations (about 65 hours of computer time) the GA finds pitch up (Figure. 6a), roll right (Figure. 6b), and yaw right (Figure. 6c).

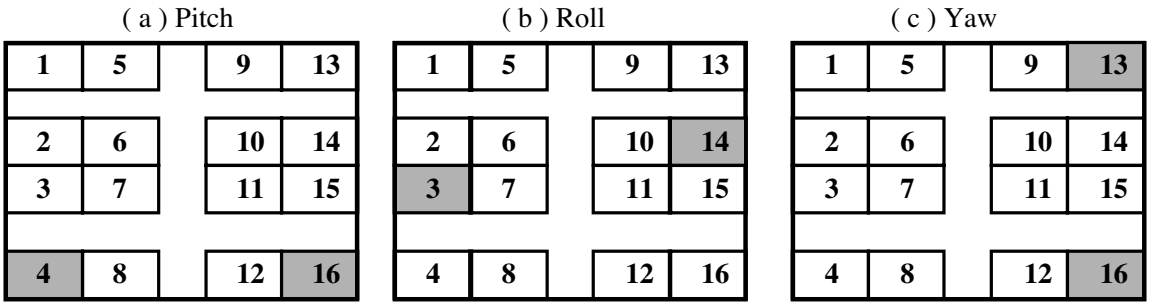

Figure 6. Actuator placement for uncoupled maneuvers.

The composite configuration places actuators at locations 3, 4, 13, 14, and 16 for the three uncoupled maneuvers. The wing model is symmetric, and this information is used to determine a composite configuration for all six uncoupled maneuvers (Figure 7).

.

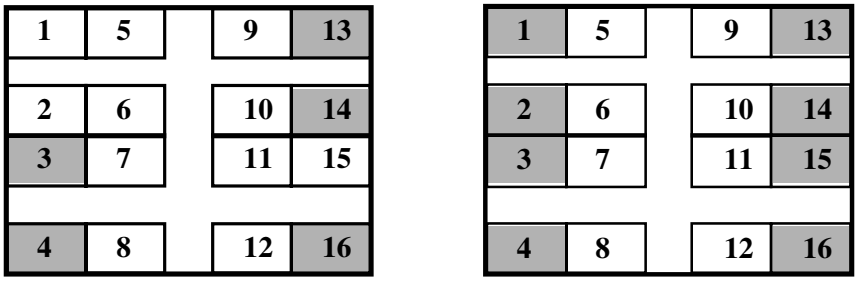

Figure 7. Composite actuator placement for 3 and 6 uncoupled maneuvers

These results show that the GA is a useful tool for reducing the time required to determine the optimum placement of actuators for a simplified, untapered, unswept wing based on the NACA 0015 airfoil. This proof-of-concept was needed before attempting to solve a more complex problem such as the seamless aircraft shown in Figure 1. However, before moving to the complex model, the program was ported to a multi-processor computer in a further attempt to reduce the time.

THE PARALLEL GENETIC ALGORITHM

The program was then modified to execute in parallel on four processors of an SGI Origin 2000™ multi-processor computer. Each of the three subproblems executed on a different processor and one processor was devoted as a master processor sending data (an array containing the actuator locations) to and receiving data (pitch, roll, and yaw moments) from the subproblems. The code for the GA operations (selection, crossover, and mutation) remained unchanged.

APPLICATION ON A MULTI-PROCESSOR COMPUTER

After a call is made to each subproblem processor for initialization purposes, the subproblem processors are only called to execute PMARC in parallel. For this

application, parallelism potentially results in a decrease in optimization time by a factor of three to about 22 hours for 13 generations. However, this ideal speedup was not achieved due to the limitation of a maximum of 8 wall clock hours for single program execution placed on this particular parallel system. This resulted in the execution of only one generation at a time; and the program had to be restarted for the next generation. Finally, because of the randomness of the GA, no direct comparison could be made to the time required to find the optimum placement on the single processor computer.

In this application, the parallel GA finds pitch down (Figure. 8a), roll right (Figure. 8b), and yaw right (Figure. 8c). Although the configuration is slightly different than the one for the single processor, the composite fitness is the same.

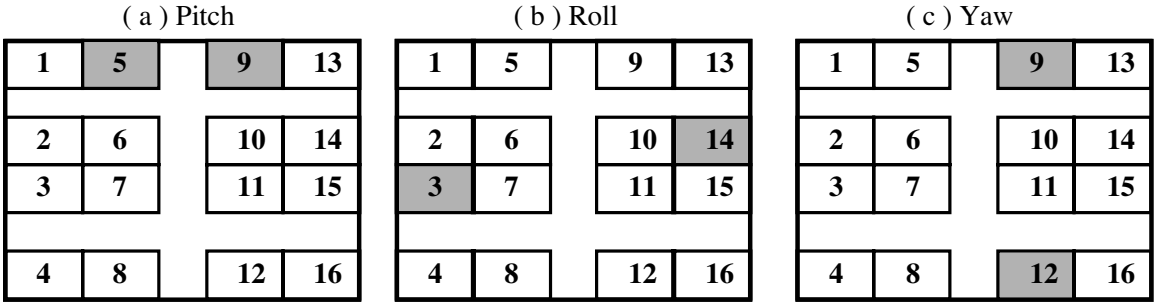

Figure 8. Actuator placement for three uncoupled maneuvers.

The composite configuration places actuators at locations 3, 5, 9, 12, and 14 for the three uncoupled maneuvers. The wing model is symmetric, and this information is used to determine a composite configuration for all six uncoupled maneuvers (Figure 9)

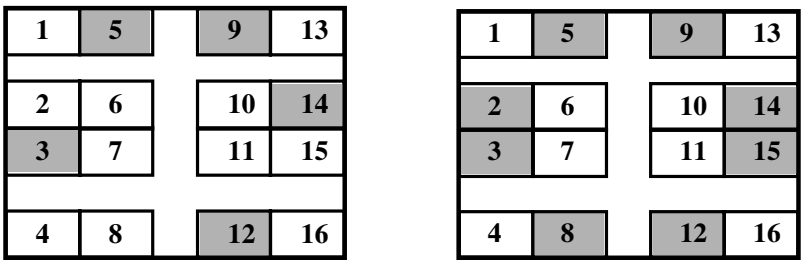

Figure 9. Composite actuator placement for 3 and 6 uncoupled maneuvers.

With the successful implementation of the parallel version of the GA, the time for finding the optimum placement for the simplified, untapered, unswept wing model with 16 actuators has been reduced from 1,100 hours for an exhaustive search to about 22 hours. Because the GA is easily scalable to more complex problems, the tool appears to be an excellent choice to apply to the placement of actuators for the seamless aircraft.

CONCLUSIONS

The purpose of this project is to determine if a genetic algorithm can reduce the amount of computer time required to find the optimum placement of actuators for a simplified, untapered, unswept wing. This wing model has 16 potential locations for actuators The fitness of a member is determined by counting the number of actuators in the string. The fitness is penalized if any constraints placed on the problem are violated. A low-fidelity aerodynamic code is used to determine the penalty, if any. If this program were to be applied to all possible combinations of 16 actuators, it would require about 1,100 hours of computer time. The first application of a genetic algorithm to this model executes on a single processor computer and contains a multiobjective function which finds the minimum number of actuators required to provide uncoupled pitch, roll, and yaw control. The problem is divided into three distinct subproblems, each with its own population, fitness function, and constraints. A composite fitness function finds the minimum number of actuators (at least two) required to accomplish three uncoupled maneuvers. Wing symmetry is used to determine a composite configuration for all six uncoupled maneuvers. This application requires about 65 hours of computer time. The second application of the genetic algorithm executes on a multi-processor computer and uses four processors to find the optimum placement. One processor is used as a master and the other three solve the subproblems. The parallel application results in a further reduction to about 22 hours of computer time to find the optimum placement, which is about 2% of the time required for an exhaustive search of all possible combinations.

Acknowledgement. The author would like to acknowledge Dr. William A. Crossley of Purdue University for his development in the PMARC model and consultation and Mrs. Sharon Padula of NASA Langley Research Center for her assistance in converting the program to run on the multi-processor computer. In addition, NASA Ames Research Center is acknowledged for the use of their SGI Origin 2000™ multi-processor computer under their joint IT/HPCCP program.

REFERENCES

[1]Raney, D. L., Montgomery, R. C., Park, M. A., and Green, L. L. (2000). Flight Control Using Distributed Shape-Change Effector Arrays, AIAA Paper 2000-1560, 41st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, April 2000.

[2]Scott, M. A., Montgomery, R. C., and Weston, R. P. (1998). Subsonic Maneuvering Effectiveness of High Performance Aircraft Which Employ Quasi-Static Shape Change Device. SPIE's 5th Annual International Symposium on Smart Structures and Materials, San Diego, pp.223-233.

[3]Ashby, D. L., Dudley, M. R., and Iguchi, S. K. (1988). Development and Validation of an Advanced Low-Order Panel Method. NASA TM 101024.

[4]Goldberg, D. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Co., New York.

[5]Simpson, M. T., and Hansen, C. H. (1996).

Use of Genetic Algorithms to Optimize Vibration Actuator Placement for Active Control of Harmonic Interior Noise in a Cylinder with Floor Structure. Noise Control Engineering Journal, Vol. 44, No. 4, pp. 169-184.

[6]Gordon, V. S. and Whitley D. (1993). Serial and Parallel Genetic Algorithms as Function Optimizers, ICGA93, pp. 177-183.

[7]Back. T. (1996). Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York.

[8]Eshelman, L. J. (1991). The CHC Adaptive Search Algorithm: How to Have a Safe Search When Engaging in Nontraditional Genetic Recombination. FOGA91, pp. 265-283.

[9]Whitley, D., Soraya, R., Dzurba, J. and Mathias, K. E. (1996). Evaluating Evolutionary Algorithms. Artificial Intelligence 85, pp. 245-276.