JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 33, No. 3, May–June 1996

Combining Genetic and Deterministic Algorithms for Locating Actuators on Space Structures

Hiroshi Furuya*
Tokyo Institute of Technology, Nagatsuta, Yokohama 226, Japan
and
Raphael T. Haftka†
University of Florida, Gainesville, Florida 32611-6250

Genetic algorithms are a powerful tool for the solution of combinatorial problems such as the actuator placement problem. However, these algorithms require a large number of analyses with the attendant high computational costs. Therefore, it is useful to tune the operators and parameters of the algorithm on problems with inexpensive analyses that are similar to computationally more expensive problems. An easy-to-calculate measure of actuator effectiveness is employed to evaluate several genetic algorithms for a problem of placing actuators at 8 of 1507 possible locations. Even with the best of the algorithms and with optimum mutation rates, tens of thousands of analyses are required for obtaining near-optimum locations. A hybrid procedure is proposed that first estimates near-optimum locations with a deterministic algorithm and then seeds these locations in the initial population of a genetic algorithm. A simulated annealing technique is also applied as a mutation operator for the genetic algorithm. The hybrid procedure reduces the cost of the genetic optimization by an order of magnitude.

Nomenclature

a^{ij}	= nondimensional fraction of elastic energy of mode i carried by member j
c	= nondimensional cooling rate for annealing procedure
e_{av}	= average of effectiveness index for jth member
e_j	= effectiveness index for jth member
f	= nondimensional value of objective function
f_0	= nondimensional value of objective function before mutation
I	= set of controlled-vibration-mode numbers
J	= set of actuator locations
m	= number of controlled modes
n	= number of actuators
n_s	= number of strings
p_i	= probability for ith ranked design to be selected as parent
s	= number of annealing iterations
T	= nondimensional temperature for annealing procedure
T_0	= nondimensional initial temperature for annealing procedure
w_i	= weight for effectiveness index
β	= nondimensional minimum energy fraction over controlled modes
β_i	= nondimensional fraction of energy associated with the ith vibration mode

Introduction

THE location of active or passive damping elements on a large space structure is significant for effective vibration suppression. Many studies for the problem of selecting locations of actuators have been performed. Padula and Sandridge[1] solved the problem by the branch-and-bound method for placing active members on a controls–structures integration (CSI) evolutionary model with 1507 members by taking advantage of the linearity of their formulation. More general, nonlinear actuator placement problems naturally lend themselves to random search algorithms. Kincaid and Bloebaum[2] solved the CSI placement problem using tabu search. Rao et al.[3] solved a similar problem with a binary-coded genetic algorithm (GA), Chen et al.[4] employed simulated annealing, and Onoda and Hanawa[5] employed a combination of simulated annealing and GAs. Ponslet et al.[6] used GAs with both standard and Gray-code binary strings for the related problem of placing masses on structures for the purpose of changing their natural frequencies.

GAs are probabilistic search algorithms based on the mechanics of natural selection and natural genetics. The design is coded as a string of numbers (or genes), which is the counterpart of a chromosome in a biological system. GAs work with a population of designs and employ three operators: 1) selection, 2) crossover, and 3) mutation. In the selection process individual strings are selected for reproduction according to their objective (fitness) function values. Crossover is a process that splices together the genes of parent designs selected for reproduction to form child designs. Mutation introduces random perturbations in the child genes, thus allowing the incorporation of genes that are not included in the parent population. The three operators are used to replace a parent population with a child population, and this procedure (one generation) is repeated to evolve the population towards the optimum design.

One of the factors that hinder the wide acceptance of GAs for engineering applications is the large number of analyses required for genetic optimization. For example, we have explored several GAs for placing actuators for vibration control on large space structures.[7] We found that even when parameters such as population size and mutation rates were tuned, we still required hundreds of thousands of analyses to obtain near-optimal designs.

The problem of large computational cost can be alleviated by using simple criteria to rule out locations that are obviously inappropriate. Thus, we have shown that by using effectiveness indices, we can reduce the number of candidate locations for the actuators and the number of analyses required by the genetic optimization about a factor of five.[8] The objective of this paper is to pursue this strategy of helping the GA further. Specifically, we propose a deterministic heuristic search technique to produce initial configurations that can then be optimized by GAs.

We first describe the actuator location problem in the next section. The following section introduces an effectiveness index that can be used to eliminate clearly ineffective locations. Next a procedure that tries to imitate an intuitive designer is introduced and

Presented as Paper 95-1400 at the AIAA 36th Structures, Structural Dynamics, and Materials Conference and Adaptive Structures Forum, New Orleans, LA, April 10–14, 1995; received July 28, 1995; revision received Jan. 8, 1996; accepted for publication Jan. 11, 1996. Copyright © 1996 by Hiroshi Furuya and Raphael T. Haftka. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

*Associate Professor, Department of Built Environment. Member AIAA.

†Professor, Department of Aerospace Engineering, Mechanics and Engineering Science. Associate Fellow AIAA.

called critical-mode search. Next we review the various GAs used in the study, as well as how we can combine them with the critical-mode procedure. The next section presents some results comparing the performance of the different algorithms. Finally, we offer some concluding remarks.

CSI Evolutionary Model

Padula and Sandridge[1] considered the placement of actuators on a large space structure that has been developed at NASA Langley Research Center as the phase 1 CSI evolutionary model (Fig. 1). The structure has been developed for testing problems of interaction between position control and structural vibrations. To damp out structural vibrations, the structure contains active members with piezoelectric actuators. The problem is to select locations of actuators from 1507 candidate locations. With eight actuators, the number of candidate sets of locations is

$$ {}_{1507}C_8 = 6.48 \times 10^{20} \tag{1}$$

The objective of the actuator placement problem is to select a set J of locations for actuators that improve modal damping ratios for a selected set I of vibration modes. This task requires that we select members that store a significant portion of the elastic energy associated with each of the desired modes. The optimization problem seeks to maximize the lowest (over $i \in I$) total fraction of elastic energy carried by the set of selected members.

Assuming that the effect of active members on the elastic energy is small, Padula and Sandridge[1] calculated the total fraction of energy β_i associated with the ith mode as

$$\beta_i = \sum_{j \in J} a^{ij} \tag{2}$$

We have used values of energy fractions that we obtained from Padula and Sandridge[1] for 10 modes of the structure and that were originally calculated by NASTRAN.

Padula and Sandridge's[1] objective function β was the minimum total fraction over a set I of controlled modes

$$\beta = \min_{i \in I} \beta_i \tag{3}$$

so that the optimization problem is to find the set of members J that maximizes β,

$$\max_{J} \min_{i \in I} \sum_{j \in J} a^{ij} \tag{4}$$

Effectiveness Index

The actuator placement problem needs to balance the contributions of actuators for the various controlled modes. Some locations are good for only one or two modes, while others contribute to the damping of many modes. However, there are many locations that contribute very little to the damping of any of the controlled modes. Such locations can be eliminated from consideration, thus reducing the size of the optimization problem. One measure of effectiveness of the jth location is the sum e_j of the contributions to all the controlled modes,

$$e_j = \sum_{i \in I} a^{ij} \tag{5}$$

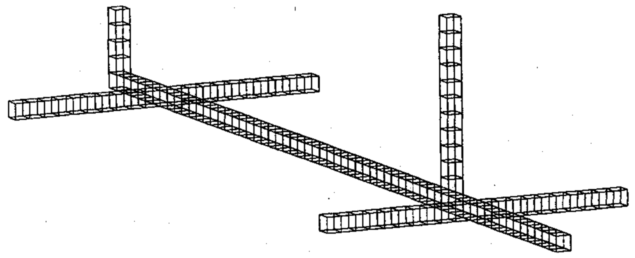

Fig. 1 NASA Langley Research Center CSI phase 1 evolutionary model.

It is easy to check that for a design J with an objective function β, the sum of the effectiveness indices e_j for the n locations in J satisfies

$$\sum_{j \in J} e_j \geq m\beta \tag{6}$$

From Eq. (6), the average effectiveness e_{av} of a location satisfies

$$e_{\text{av}} \geq m\beta/n \tag{7}$$

Therefore, we can eliminate from consideration locations with effectiveness indices that are much lower than $m\beta/n$. We have found that the effectiveness index can be further refined by weighting the contributions of the different modes. So instead of Eq. (5), we use

$$e_j = \sum_{i \in I} w_i a^{ij} \tag{8}$$

where w_i is a weight assigned to the ith mode.

Critical-Mode Search Procedure

Effectiveness indices help the GA by reducing the set of locations that need to be considered. In this work, we go one step further and use a deterministic heuristic procedure that works on a similar logic to that of the effectiveness indices for creating a good initial population for the GA.

The deterministic procedure tries to imitate a human designer trying to find a good set of actuator locations from those selected by an effectiveness index. Starting with a trial design, the human designer may be expected to first identify the critical mode. The critical mode is the mode that has the lowest percentage of its elastic energy covered by the set of members where actuators are located. Then, the designer can be expected to look for a location that has high elastic energy in that mode and use this location to replace one of the locations in the current set. We call this procedure the critical-mode search procedure.

The critical-mode procedure is used both in a deterministic mode and in a randomized mode. For the deterministic mode, the initial design consisted of the set of locations with the highest effectiveness indices. A list of locations that are the best for each vibration mode is then created. In this list the first member is the location that is best for this mode, the second member is second best, etc. When a critical mode is identified, the design procedure looks up this database and picks the best location that has not already been used. This deterministic procedure will produce a single design. This design was not found to be the optimum for the cases we studied, and so the procedure was randomized in the hope of finding better designs. First, the initial design was chosen at random, and second, the list of locations used for each mode was initially accessed at some random point in the middle of the list. The procedure can be summarized by the following steps:

1) Randomly select a starting location from the list of the best locations for each mode. This location will be the first one to be used if that mode becomes critical for a current design.

2) Select a random starting design. This and the previous random step allow us to generate multiple solutions with the heuristic procedure.

3) Identify the most critical mode for the current design.

4) Select the next available location in the list of good locations for this mode.

5) Check the improvement in the objective function obtained by exchanging the new location with each one of the locations in the current design. If the best exchange improves the objective, execute it.

6) Check the number of iterations. If it is above the limit, stop. Otherwise, if an exchange was executed in the previous step, go to step 3. If no exchange was executed, go to step 4.

This procedure is repeated several times to create a set of initial designs for the GA.

Description of GA Variants

In this work several implementations of GAs were tested with initial designs generated by the critical-mode search procedure. In view of our previous work,[7] we have used integer coding rather than the more prevalent binary coding for these variants. That is, the genetic strings consist of strings of n integers, each varying between 1 and 1507. The variants differ in the crossover operators used to recombine parent designs into child designs. All variants used an elitist strategy. The results in this work are limited to $n = 8$, so that each chromosome is composed of eight integers denoting the locations of the eight actuators.

Fitness and Selection

We used a rank-based fitness equal to $n_s + 1 - i$, where n_s is the number of strings (or population size) and i is the rank of the string in the population in terms of its objective function β. The probability for the ith ranked design to be selected as a parent is

$$p_i = \frac{2(n_s + 1 - i)}{n_s(n_s + 1)} \tag{9}$$

The selection process is executed twice to select two parents, and then the crossover process, described in the next section, is used to generate one child. The child design is discarded if it contains duplicate actuator positions. The process is repeated until $n_s - 1$ children have been generated. At this point, the best member of the parent generation is added to the child generation, and the parent generation is replaced.

Crossover

We implemented three crossover operators. The first is an averaging crossover (also called intermediate recombination[9]), which can generate genes not present in the parent generation but which is confined to generating child designs that are well related to the parent designs. This crossover operator allows the crossover point to fall in the middle of a gene. That is, the crossover operator generates a random number r between 0 and n. If we let $[r]$ denote the integer part of r, then a child is generated by taking the first $[r]$ genes from the first parent and the last $n - 1 - [r]$ genes from the second parent. In the $[r] + 1$st gene the child has the actuator location q, which is the following weighted average of the actuator locations q_1 and q_2 of the first and second parents, respectively:

$$q = (r - [r])q_1 + ([r] + 1 - r)q_2 \tag{10}$$

The second operator considered in this study is uniform crossover. For each gene, an integer 1 or 2 is randomly selected. When the number is equal to 1, we select the corresponding gene from the first parent; otherwise, we select it from the other parent. Compared to single-point crossover, the uniform crossover is more disruptive of schema formation, but it also tends to reduce the hitchhiking phenomenon, where bad genes get a free ride because they happen to be adjacent to good genes in fit individuals.

Finally we considered an operator that replaces crossover by random selection of n positions from the $2n$ positions of the two parents. This operator appears to be similar to Radcliffe's random assortment recombination operator RAR_2 (e.g., Ref. 10). One way to visualize the random selection operator is that we throw the two sets of genes from the parents in a bag and take out n of them to form the child design. This operator is even more disruptive of both schema formation and hitchhiking. In fact, random selection provides 12,870 (${}_{16}C_8$) combinations to generate a child strings, compared to 256 (2^8) for uniform crossover.

Mutation with Simulated Annealing

We began by applying a conventional mutation operator with a constant mutation probability to generate a child generation that includes new genes that do not appear in parents' strings. If a random value we generate for each gene is less than the mutation probability, we replace the gene with a randomly selected new gene, and if not, we keep it. However, though the mutation operation introduces new genetic material and helps avoid local optima, it can disrupt progress towards the optimum design, especially in the later stages of the search.

We therefore applied a simulated annealing technique to mutation operations for the children generated by the crossover operation. At the early stages of the annealing procedure, the mutation probability is relatively high, and it decreases gradually with increasing number of generations. Thus, the procedure is able to avoid the disruptive effect of mutation in the later stages of the search.

Three parameters define the simulated annealing procedure: an initial temperature T_0 (used in the first generation), a cooling rate c, and a number of simulated annealing iterations s. The procedure employs the following steps:

1) Evaluate the objective function f for all designs obtained by crossover. Recall the last temperature T used in the previous generation.

2) Randomly select one child generated by crossover.

3) Randomly select one gene of the child, and replace it with a randomly selected gene.

4) Calculate the value f of the objective function, and compare with the value before mutation, f_0.

5) If $f > f_0$, replace the original child with the new design. If $f < f_0$, replace the original child with the new design with probability $\exp[(f - f_0)/T)]$.

6) $T \leftarrow c \times T$.

7) If the number of iterations is less than s, go back to step 2.

For the present study we used an initial temperature $T_0 = 0.1$, a cooling rate $c = 0.99$, and a number $s = 10$ of simulated-annealing iterations.

Performance

Because GAs are random-search algorithms, the performance of a single run is partly a matter of luck. Therefore, the performance of the various algorithms was averaged over a number of runs, with the mean and standard deviation calculated for the objective function. We expected that the performance of the variants would depend strongly on the mutation rate and the population size. Therefore, for several population sizes, the mutation rate was varied in the results presented in the next section.

Various measures of performance may be used to compare the various algorithms, including number of analyses, operation counts, and CPU time. In the present application, the calculation of the objective function is very inexpensive, because the a^{ij} are assumed to be independent of the actuator locations. Since we are interested in evaluating the algorithms for more realistic cases, where the objective function is expensive to evaluate, we use the number of analyses as our measure of performance. A single analysis is considered to be the evaluation of the objective function following a change in the configuration of actuators. For the critical-mode procedure and for the present objective, this evaluation can be made cheaper because one actuator is changed at a time. However, we assume that this still calls for a full analysis.

Results

To provide a point of reference we first summarize the results obtained by GAs alone. First we show the results we obtained without effectiveness indices in Table 1.[7] Table 1 reflects results that were obtained by running each algorithm 40 times for 10 mutation rates. In each run the best objective in the final population was recorded. The average value and standard deviation over 40 runs of the best objective are shown in Table 1 along with the best maximum, defined as the best objective function obtained in any of the 40 optimizations for the given mutation rate. The results are presented for the mutation rate that was the best out of 10 mutation rates that were tested. To get reasonably close to the optimum design of 1.525% energy in the worst mode, we need a population size of 500 and 500 generations, for a total of 250,000 analyses! We also note that for this case, the random-selection crossover was the best of the three variants. Finally, note that each entry in the last column of Table 1 corresponds to 400 optimizations (40 for 10 mutation rates), or 100 million analyses. This large number of analyses was needed to get reasonably accurate statistics needed for comparing the various algorithms.

Table 1 Mean values and standard deviations of best objective, and maximum values (in percent) for GA variants[a]

	Total number of analyses (number of strings)					
	50,000 (100)[b]		100,000 (200)[c]		250,000 (500)[d]	
Crossover type	Mutation rate	Best objective (max. value)	Mutation rate	Best objective (max. value)	Mutation rate	Best objective (max. value)
Averaging crossover	0.06	1.428 ± 0.054 (1.516)	0.07	1.436 ± 0.042 (1.495)	0.04	1.461 ± 0.045 (1.524)
Random selection	0.02	1.445 ± 0.031 (1.496)	0.03	1.481 ± 0.019 (1.496)	0.02	1.492 ± 0.006 (1.496)
Uniform crossover	0.10	1.432 ± 0.058 (1.510)	0.09	1.454 ± 0.043 (1.497)	0.08	1.476 ± 0.027 (1.504)

[a]Number of generations, 500; number of optimizations, 40.
[b]Best maximum value of 1.521 obtained with averaging single-point crossover with a mutation rate of 0.03, uniform crossover with a mutation rate of 0.03.
[c]Best maximum value of 1.523 obtained with uniform crossover with a mutation rate of 0.08.
[d]Best maximum value of 1.525 obtained with averaging single-point crossover with a mutation rate of 0.02.

Table 2 Mean values and standard deviations of best objective, and maximum values (in percent) for GA variants with number of locations reduced by using location effectiveness indices[a]

	$w_i = 1$[b]		$w_1 = 5,\ w_i = 1\ (i \neq 1)$[c]		$w_i = (11 - i)/55$[d]	
Crossover type	Mutation rate	Best objective (max. value)	Mutation rate	Best objective (max. value)	Mutation rate	Best objective (max. value)
Averaging crossover	0.07	1.457 ± 0.049 (1.519)	0.07	1.471 ± 0.042 (1.524)	0.07	1.467 ± 0.028 (1.525)
Random selection	0.02	1.463 ± 0.021 (1.497)	0.02	1.475 ± 0.026 (1.497)	0.02	1.481 ± 0.018 (1.497)
Uniform crossover	0.08	1.458 ± 0.029 (1.504)	0.09	1.486 ± 0.030 (1.525)	0.10	1.474 ± 0.033 (1.524)

[a]Total number of analyses, 50,000; number of strings, 100; number of generations, 500; number of optimizations, 40.
[b]537 locations with an effectiveness index larger than 0.3%.
[c]356 locations with effectiveness index larger than 1.5%.
[d]265 locations with effectiveness index larger than 0.075%.

Table 3 Mean values and standard deviations of best objective, and maximum values (in percent) for GA with initial designs from critical-mode search procedure[a]

	10 strings[b], 2000 generations		40 strings[c], 500 generations	
Crossover type	Mutation rate	Best objective (max. value)	Mutation rate	Best objective (max. value)
Averaging crossover	0.03	1.520 ± 0.002 (1.521)	0.03	1.521 ± 0.002 (1.525)
Random selection	0.05	1.519 ± 0.003 (1.521)	0.02	1.516 ± 0.004 (1.521)
Uniform crossover	0.05	1.519 ± 0.003 (1.521)	0.04	1.519 ± 0.003 (1.521)

[a]Total number of analyses, 24,000; number of optimizations, 40.
[b]10 strings from critical-mode procedure are used as initial strings.
[c]10 strings from critical-mode procedure are used as initial strings, and 30 strings are randomly generated.

Next we show the results obtained by using various weighting procedures with the effectiveness indices from Ref. 8. In that work it was found that the original algorithm had trouble with the first mode, and often selected an actuator that was good for no other mode than the first. Table 2 summarizes the results, which show that by using the effectiveness indices, the population size can be reduced to 100 with comparable results to those obtained previously with a population size of 500. Still, the total number of required analyses is 50,000. The results also indicated that a weighting emphasizing the first mode or the first few modes was superior to no weighting.

We tried the deterministic critical-mode algorithm by itself. Based on the results in Table 2, we used the weighting index with $w_1 = 10$. First we considered the completely deterministic search where the locations with the highest effectiveness indices were used for the initial design, and the lists of effective locations for each mode were accessed first at the top. We obtained a design with an objective function of 1.341%.

Next we tried the randomized critical-mode selection procedure by itself. It appeared to be comparable to the GA alone. For example, with 50 iterations, the average objective was 1.25%, with a standard deviation of 0.1%. Because 50 iterations represent only 400 analyses (8 analyses per iteration to check all the possible exchanges of 8 actuators), it is reasonable to run many cases and take the best of these. When we ran the procedure 40 times, the best design was 1.446%; with 400 times it was 1.468%. Increasing the number of iterations to 100 increased the best objective over 400 trials to 1.497%. However, at this point we were again up to 320,000 analyses. Clearly, the critical-mode procedure could not outperform the GA.

We combined the two procedures by using the randomized critical-mode approach to generate some initial designs for the GA. Using 50 iterations for convergence of the critical-mode procedure, we used it to generate 10 designs, at a cost of 4000 analyses. We then generated additional initial designs at random and used the combined population to start the GA. Results with population sizes of 10 (only critical-mode initial designs) and 40 (30 additional random initial designs) are shown in Table 3. The number of generations was selected so that the total number of analyses with the GAs was 20,000. Together with the critical-mode search, this adds up to a total of 24,000 analyses.

Table 4 Mean values and standard deviations of best objective, and maximum values (in percent) for phase 1 evolutionary model by using critical-mode search procedures[a]

	10 strings[b], 800 generations		10 strings[c], 600 generations	
GA operator	Mutation rate	Best objective (max. value)	Mutation rate	Best objective (max. value)
Averaging crossover	0.04	1.516 ± 0.005 (1.521)	0.06	1.514 ± 0.006 (1.521)
Random selection	0.02	1.513 ± 0.006 (1.521)	0.03	1.513 ± 0.007 (1.521)
Uniform crossover	0.04	1.513 ± 0.007 (1.521)	0.03	1.512 ± 0.006 (1.521)

[a]Total number of analyses, 10,000; number of optimizations, 40.
[b]5 strings from deterministic procedures are used as initial strings.
[c]10 strings from deterministic procedures are used as initial strings.

Table 5 Mean values and standard deviations of best objective, and maximum values (in percent) with annealing schedule for GA with initial designs from critical-mode search procedure[a]

	10 strings[b], 150 generations		10 strings[c], 50 generations	
Crossover type	Mutation rate	Best objective (max. value)	Mutation rate	Best objective (max. value)
Averaging crossover	0.00	1.520 ± 0.002 (1.521)	0.01	1.516 ± 0.004 (1.521)
Random selection	0.00	1.519 ± 0.003 (1.521)	0.00	1.513 ± 0.007 (1.521)
Uniform crossover	0.03	1.520 ± 0.002 (1.525)	0.00	1.517 ± 0.005 (1.521)

[a]Total number of analyses, 5000; number of optimizations, 40. The number of interations for the annealing procedure is 10.
[b]5 strings from critical-mode procedure are used as initial strings.
[c]10 strings from critical-mode procedure are used as initial strings.

Table 6 Mean values and standard deviations of best objective, and maximum values (in percent)[a]

	10 strings			
	Without critical-mode procedure		With critical-mode procedure[b]	
Crossover type	Mutation rate	Best objective (max. value)	Mutation rate	Best objective (max. value)
Basic GA	0.02	8.999 ± 0.0015 (9.243)	0.02	8.812 ± 0.0020 (9.369)
Basic GA + SA[c]	0.00	9.383 ± 0.0015 (9.629)	0.00	9.331 ± 0.0019 (9.608)

[a]Total number of analyses, 20,000; number of actuators, 50; number of optimizations, 40.
[b]10 strings from critical-mode procedure are used as initial strings.
[c]The number of iterations for annealing procedure is 40.

Comparing Table 3 with Table 2, note that even though the number of analyses has been halved, the average result has improved from about 1.48% to 1.52%, much closer to the optimum of 1.525%. Also, the standard deviation of the optimal solutions is less than 0.5%. If we generate all the initial strings by the critical-mode procedure, the expected value is slightly higher, although the standard deviation becomes smaller. Thus, the highest value 1.525% was obtained in the case of 10 deterministic initial strings and 30 randomly generated initial strings.

The gain in the objective function can be traded for reduced computational cost. Table 4 shows what can be accomplished with 10,000 analyses using the combined procedure. We note some deterioration in performance, but compared to Table 2, we still have better performance at reduced cost. Furthermore, the difference between the three variants of the GA, which was reduced by the introduction of the effectiveness index (compare Table 2 with Table 1), became even smaller with the use of the critical-mode starting designs.

Finally we introduced the simulated annealing procedure as a mutation operator. We limited the total number of analyses to 5000. We used 10 iterations for the annealing procedure, and we used 5 and 10 initial designs from the critical-mode approach. The results are shown in Table 5 and indicate improvement in performance over Table 4, even though the number of analyses has been slashed by half. Comparing Table 5 with Table 2, note again substantial improvement in the objective function at one-tenth of the cost.

To check the performance of the various algorithms when the number of actuators is large, we compared the basic GA, the GA with critical-mode search, the GA with simulated annealing mutation, and the GA with both critical-mode search and simulated annealing mutation for 50 actuators. In all cases the averaging crossover was used. With 50 actuators, the best design that was found had 9.6% of the energy, which is almost exactly 50/8 times the maximum percentage (1.525%) obtained with 8 actuators. The performance of the four variants is summarized in Table 6. This time the best performance is of the GA with simulated annealing mutation, but without the critical-mode search. This is understandable because with large number of actuators, the critical-mode search may be both unnecessary and more expensive to conduct. Table 6 also reflects the fact that with 50 actuators the design problem is more difficult, so that 10,000 configuration evaluations are not sufficient to get as close to the optimum as they were for the 8-actuator case.

Concluding Remarks

A new deterministic procedure for obtaining optimal locations of actuators for vibration suppression has been developed. The procedure quickly finds near-optimal sets of locations, which can then be seeded into the initial population for a GA procedure. The combined deterministic and genetic procedure achieves better performance at one-fifth the cost of the GA alone. Additional improvements were obtained by replacing the standard mutation operator of the GA with mutation based on simulated annealing. The combined procedure yielded better optimal solutions at one-tenth the computational cost. Furthermore, the new procedure was found to be less sensitive to the choice of GA than the genetic search alone.

Acknowledgments

This research was supported in part by NASA Grants NAG-1-224 and NAG-1-1669.

References

[1]Padula, S. L., and Sandridge, C. A., "Active Strut Placement Using Integer Programming for the CSI Evolutionary Model," *Proceedings of the AIAA/USAF/NASA/OAI 4th Symposium on Multidisciplinary Analysis and Design* (Cleveland, OH), Pt. 2, AIAA, Washington, DC, 1992, pp. 784–793.

[2]Kincaid, R., and Bloebaum, C., "The Damper Placement Problem for the CSI–Phase I Evolutionary Model," *Proceedings of the AIAA 34th Structures, Structural Dynamics and Materials Conference, Adaptive Structures Forum* (La Jolla), AIAA, Washington, DC, 1993, pp. 3086–3095 (AIAA 93-1655).

[3]Rao, S. S., Pan, T. S., and Venkayya, V. B., "Optimal Placement of Actuators in Actively Controlled Structures Using Genetic Algorithms," *AIAA Journal*, Vol. 29, No. 6, 1991, pp. 942, 943.

[4]Chen, G.-S., Bruno, R. J., and Salama, M., "Optimal Placement of Active/Passive Members in Structures Using Simulated Annealing," *AIAA Journal*, Vol. 29, No. 8, 1991, pp. 1327–1334.

[5]Onoda, J., and Hanawa, Y., "Actuator Placement Optimization by Genetic Optimization and Improved Simulated Annealing," *AIAA Journal*, Vol. 31, No. 6, 1993, pp. 1167–1169.

[6]Ponslet, E., Haftka, R. T., and Cudney, H. H., "Optimal Placement of Actuators and Other Peripherals for Large Space Structures in Topology Design of Structures," *Proceedings of NATO Advanced Research Workshop on Topology Design of Structures* (June 1992), edited by M. P. Bendsoe and C. A. Mota Soares, Kluwer, Dordrecht, The Netherlands, 1993, pp. 135–144.

[7]Furuya, H., and Haftka, R. T., "Genetic Algorithms for Placing Actuators on Space Structures," *Proceedings of the 5th International Conference on Genetic Algorithms* (Urbana–Champaign), edited by S. Forrest, Morgan Kaufmann, San Mateo, CA, 1993, pp. 536–542.

[8]Haftka, R. T., and Furuya, H., "Comparison of Genetic Algorithms for Placing Actuators on Space Structures," *19th International Symposium on Space Technology and Science* (Yokohama), Inst. of Space and Astronautical Science, ISTS 94-b-08, 1994, pp. 1–9.

[9]Bäck, T., and Schwefel, H.-P., "An Overview of Evolutionary Algorithms for Parameter Optimization," *Evolutionary Computation*, Vol. 1, No. 1, 1993, pp. 1–23.

[10]Radcliffe, N. J., and George, F. A. W., "A Study in Set Recombination," *Proceedings of the 5th International Conference on Genetic Algorithms* (Urbana–Champaign), edited by S. Forrest, Morgan Kaufmann, San Mateo, CA, 1993, pp. 23–30.

F. Milos
Associate Editor