AIAA JOURNAL
Vol. 43, No. 10, October 2005

Parallel Multispecies Genetic Algorithm for Physics and Parameter Estimation in Structural Dynamics

David C. Zimmerman* and Soren S. F. Jorgensen†
University of Houston, Houston, Texas 77204-4006

The distinction between model structure verification and model updating is logically clear. In the former, one adjusts the physics to better match experimental data; in the latter the physical parameters of the model are adjusted. The current state of the practice is to iterate back and forth between both problems until reasonable agreement with experimental data is obtained. In this work, the selection of model physics from a predefined set and the simultaneous updating of the corresponding physical parameters are investigated in an unified computational framework. A newly developed parallel (computational efficiency) multispecies (physics models) genetic algorithm is proposed and evaluated. The genetic algorithm "evolves" both model physics and parameters to improve the correlation of analytical models and corresponding test data. Two new genetic operators are developed. The first is a crossover function, which enables the sharing of the genes between two species. The second is a transmutation operator, which allows a species member to transform into a different species. The proposed algorithms are demonstrated and evaluated using three different "numerical" experiments.

Nomenclature

B_i	=	ith continuous design variable
$J(\boldsymbol{r})$	=	fitness function as a function of parameter vector $\boldsymbol{r}$
R_{op}	=	rotation matrix
$\boldsymbol{r}$	=	parameter vector (decision variables)
$V(\boldsymbol{r})$	=	analytical mode-shape matrix as a function of parameter vector $\boldsymbol{r}$
V_{exp}	=	experimental mode-shape matrix
ρ_{trans}	=	probability of transmutation
$\omega_i(\boldsymbol{r})$	=	ith analytical natural frequency as a function of parameter vector $\boldsymbol{r}$
ω_{i_exp}	=	ith experimental natural frequency

I. Introduction

THERE is a broad array of literature over the past three decades concerning computational methods for model updating of structures using dynamic testing data. In the late 1960s and early 1970s, researchers began investigating least-squares[1] and Bayesian[2] parameter estimation methods. In particular, the paper by Collins et al.[2] might be the first paper where statistical parameter estimation was applied directly to the structural model updating problem to determine a set of bending and shear rigidity parameters for the Saturn–Apollo launch vehicle. In the mid-1980s through the present, researchers have investigated a wide variety of optimization schemes, including gradient-descent-based methods (Refs. 3–7 are representative works), genetic algorithms (GAs) (Refs. 8–12 are representative works), and simulated annealing,[13] to adjust physical parameters of a model to better predict experimental observations.

Of particular interest is the use of GAs as applied to model updating. GAs are a form of directed random search.[14] The form of direction is based on Darwin's "survival of the fittest" theories. In GAs, a finite number of candidate solutions or designs are randomly or heuristically generated to create an initial population of designs. This initial population is then allowed to evolve over generations to produce new and hopefully better designs. The basic conjecture behind GAs is that evolution is the best compromise between determinism and chance. GAs have the capability to solve continuous, discrete, and continuous/discrete optimization problems.

Maben and Zimmerman[8] investigated parent selection methods and child insertion options in the model updating problem. Larson and Zimmerman[9] then investigated the effects of noise on parameter estimation as applied to structural damage detection. Fulcher et al.[10] investigated several different parameter estimation techniques on a large-scale model [136,000 degrees of freedom (DOFs)], test data (17 modes, 85 triaxial accelerometers) and parameter set (35). They concluded that "of the optimization techniques used, the most successful was the genetic algorithm." Carlin and Garcia[11] investigated the effects of GA population size, number of bits used in coding, and different coding strategies on damage detection performance. Finally, Zimmerman et al.[12] developed and investigated the effect of an adaptive mutation operator on model updating performance.

In this paper, we establish a GA computational framework for the integrated updating of both the model structure and model parameters suitable for implementation on both serial and distributed computing platforms. The genetic-algorithm framework was adopted because of the anticipated nature of the solution spaces typically generated by these problems: 1) severe nonlinearity; 2) order-of-magnitude differences in parameter values; 3) mixed discrete and continuous design variables; 4) discontinuous, nonsmooth, and/or noisy objective functions; and 5) multimodal design spaces. Although many different implementations of parallel GAs have been investigated,[15] what is unique in this study is that multiple species (model structure) coexist and mate. One can rightfully ask why not just take each proposed model structure, perform classical model updating on each, and then finally review each updated model and choose the "best"? This approach would be clearly suited for distributed processing. The answer to be shown lies in the great computational gains the GA offers in quickly ferreting out unsuitable model forms without sacrificing the gains of distributed computing.

II. Basic Genetic Algorithm

GAs are radically different from the more traditional design optimization techniques. GAs work with a coding of the design variables, as opposed to working with the design variables directly. The search is conducted from a population of designs (i.e., from a large number of points in the design space), unlike the traditional algorithms that search from a single design point. The GA requires only objective function information, as opposed to gradient or other auxiliary information. Finally, the GA is based on probabilistic transition rules, as opposed to deterministic rules.

Received 9 June 2004; revision received 21 February 2005; accepted for publication 29 May 2005.

*Professor, Department of Mechanical Engineering, N207 Engineering Building 1.

†Graduate Student, Department of Mechanical Engineering, N207 Engineering Building 1.

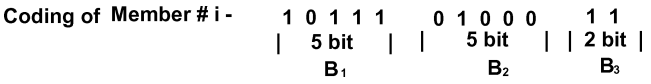

Fig. 1 Coding of a three-design-variable problem.

There are five main operations in a basic GA: coding, evaluation, selection, crossover, and mutation. *Coding* is the process by which the stated optimization problem is transformed into a genetic design space. It must be stressed that the coding for a given problem is not unique, and that significant gains in computational efficiency can be achieved by using a "good" coding scheme. Although it is difficult to place specifics on what a good coding scheme is, a coding scheme that provides for an unique one-to-one mapping of the design space and follows guidelines as suggested by interpretation of the schema convergence theorem[14] typically leads to a more efficient GA search. Once the coding scheme is selected, each continuous design variable is usually coded as a q-bit binary number. Discrete variables would each be assigned an unique binary string. A continuous design variable B_i is approximated by 2^q discrete numbers between lower and upper bounds for the design variable by

$$B_i = B_{i\,\min} + [\text{binary\#}/(2^q - 1)](B_{i\,\max} - B_{i\,\min}) \tag{1}$$

where $B_{i\,\min}$ and $B_{i\,\max}$ are the lower and upper bounds on the ith continuous design variable and binary# is an integer number between zero and $2^q - 1$. The continuous to discrete coding is like that of an analog-to-digital converter used in data-acquisition systems. A population member is obtained by concatenating all design variables to obtain a single string of ones and zeros. Thus, a population member contains all information to completely specify the total design. For example, consider a design that has two continuous variables B_1 and B_2 represented by 5-bit numbers and a discrete design variable B_3, which can take on four different values. An example of a possible coding for a particular member is shown in Fig. 1. A population is defined to be a grouping of n_{pop} members, where n_{pop} is the number of members in the population. The population can be viewed as a matrix of ones and zeros, where each row uniquely defines a single member.

Evaluation is the process of assigning a fitness measure to each member of the current population. The fitness measure is typically chosen to be related to the objective function that is to be minimized or maximized. No gradient or auxiliary information is used; only the value of the fitness function is needed. Therefore, GAs are less likely to become "trapped" at a local minima or maxima than traditional "hill climbing" algorithms. Additionally, because no gradient information is required the design space is allowed to be discontinuous.

Selection is the operation of choosing members of the current generation to produce the prodigy of the next generation. Selection is biased toward the most fit members of the population. Therefore, designs that are better as viewed from the fitness function, and therefore the objective function, are more likely to be chosen as parents.

Crossover is the process in which design information is transferred to the prodigy from the parents. Crossover amounts to a swapping of various strings of 1s and 0s between the two parents to obtain two children.

The final operation is that of *mutation*. Mutation is a low-probability random operation that can perturb the design represented by the prodigy. The mutation operator is used to retain design information over the entire domain of the design space during the evolutionary process.

In the non steady-state genetic algorithm (NSSGA), the initial coded population is subject to evaluation. An integer number of the most fit parents is then automatically added to the "next-generation" population. In the purest form of the NSSGA, the integer number is taken to be zero. However, this opens the possibility that the most fit individual at a given generation could be less fit than that of a previous generation (i.e., the best-found design to date could conceivably not be in the current population). To eliminate this chance, the integer number is typically chosen to be nonzero. The initial population is then processed with the selection operator to create a "parent pool." Two parents are then randomly selected from this pool to produce two prodigy. The mutation operator is then applied to the prodigy, and the resulting prodigy is added to the next generation. This process is repeated until the number of produced prodigy in combination with the number of most fit parents equals the set population size. This new next generation is then viewed as the current generation, and the entire process is repeated for either a fixed number of generations or until some set convergence criteria is met.

Parent 1:	1100011100000101
Parent 2:	0011100110010101
Mask:	1001000100010010
Child 1:	1010100110000101
Child 2:	0101011100010101

Fig. 2 Standard uniform crossover.

III. Multispecies Genetic Algorithm

In GAs, the prodigy is created during crossover by randomly combining the decision variables. There exist several methods for recombining the members in the population. Uniform crossover, the method currently implemented, is depicted in Fig. 2. The two parents are linked to the two offspring via a randomly created mask. The mask is a binary string with the same length as the parents. Each bit in the mask decides from which parent the offspring should inherit its properties. In Fig. 2, a 1 in the mask corresponds to "child 1" inheriting the bit of "parent 1," whereas a 0 in the mask corresponds to "child 1" inheriting the bit of "parent 2." The second child is created in the same way but with the parents switched.

Nature has a vast diversity of species, each adapted to live in their habitat. Although the species differ, they share some common genetic information. Mating is even possible between some species. It is this property that the multispecies genetic algorithm (MSGA) will try to mimic. By sharing information between the genes that are common between the species, we hope to create a more effective GA. This calls for a new way of performing crossover. In Fig. 2, each parent is of the same length with the same genes, thereby making it easy to perform uniform crossover. However, this will not be the case in the MSGA, where some species might have more genes than others. One solution to this problem is to perform uniform crossover on the subset of genes that are common between the species. Genes that are not common are unaltered and are directly passed to the offspring. Figure 3 illustrates this crossover between two different species SP1 and SP2. The genes in the species are symbolized by the number-filled array, where each number is a distinct gene. The crossover operator loops through the array of genes in SP1 and searches for the same gene in SP2. When a match is found, the gene is selected for crossover. The genes that are specific to only one species are directly passed to the offspring, as exemplified by gene 5 in SP2.

It is important to maintain species diversity in early generations. In a manner analogous to mutation, a new genetic operator that transmutes species into each other has been investigated. The low-probability transmutation operator addresses this by transmuting one member of an overrepresented species into a member of a species with a lower representation. The member that is transmuted keeps the genes that are common between the species, discards any genes that are specific to itself, and appends the genes that are specific to the other species. This is illustrated in Fig. 4, whereas in Fig. 3 the number-filled array represents distinct genes in the species.

The transmutation operator is called directly after crossover. Each species has its own specific transmutation rate defined as

$$\rho_{\text{trans}} = \gamma \frac{\text{number of members of species in population}}{\text{population size}} \tag{2}$$

where γ is a number between 0 and 1.

IV. Parallel Multispecies Genetic Algorithm

A fast growing configuration of high-performance computers is clustering. This consists of single-instruction/multiple-data

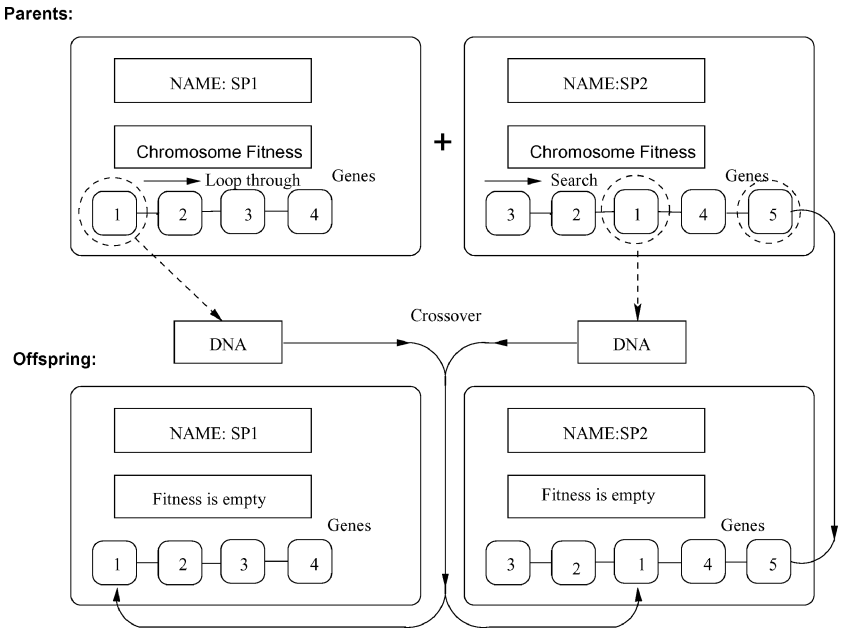

Fig. 3 Crossover in multispecies genetic algorithm.

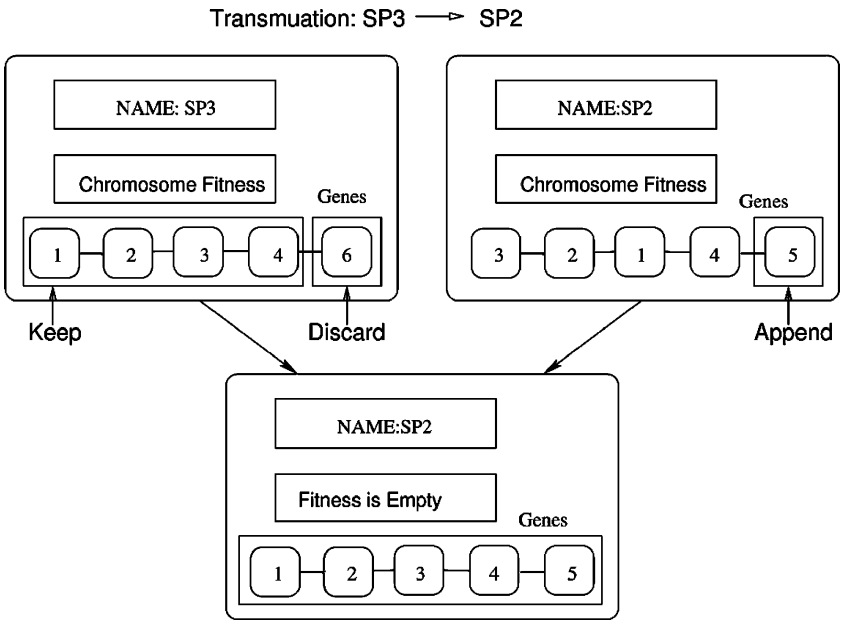

Fig. 4 Transmutation operator.

machines (such as most PCs) that are connected together in a network, thereby creating a form of multiple-instruction/multiple-data machines. This has become popular because relatively inexpensive off-the-shelf components can be used and are easily scalable. The clustered machines are popularly known as Beowulf clusters. The Beowulf cluster that is used in implementing the parallel multispecies genetic algorithm (PMSGA) consists of 16 nodes each with dual processors and 512MB of RAM.

Genetic algorithms are well suited for parallelization. In Debian's Sid Linux distribution[‡] there is even a ready-made library PGAPack[§] available. Parallel GAs are often implemented using an island style.[16] Each island resides on a computational node and contains a subpopulation that is allowed to develop by itself. The members of any population can migrate between the islands and thereby share their genes. An overview of different methods of implementing parallel GAs is provided in Ref. 15. The method chosen is dependent on the problem, and in this work a global single-population master–slave implementation is used. This is most suitable because most of the computation time will occur in the evaluation of the fitness function, as opposed to actual genetic operations.

There exist several different methods of developing parallel code. OpenMP[¶] uses pragmas that are interpreted by the compiler and autogenerates code that can be executed in parallel. The same approach is used in High Performance FORTRAN.[**] The method used in this work utilizes the standard message-passing-interface (MPI) library.[††,‡‡] MPI provides a highly flexible way of writing parallel programs necessary in this work because of the use of three different programming languages and an external program. The flowchart of one iteration of the PMSGA is shown in Fig. 5. Here the MSGA is

[‡]Data available online at http://www.debian.org.

[§]Data available online at http://www-fp.mcs.anl.gov/CCST/research/report_pre1988/comp_bio/stalk/pgapack.html.

[¶]Data available online at http://www.openmp.org.

[**]Data available online at http://www.crpc.rice.edu/HPFF.

[††]Data available online at http://www.mpi-forum.org.

[‡‡]Data available online at http://www-unix.mcs.anl.gov/mpi/mpich.

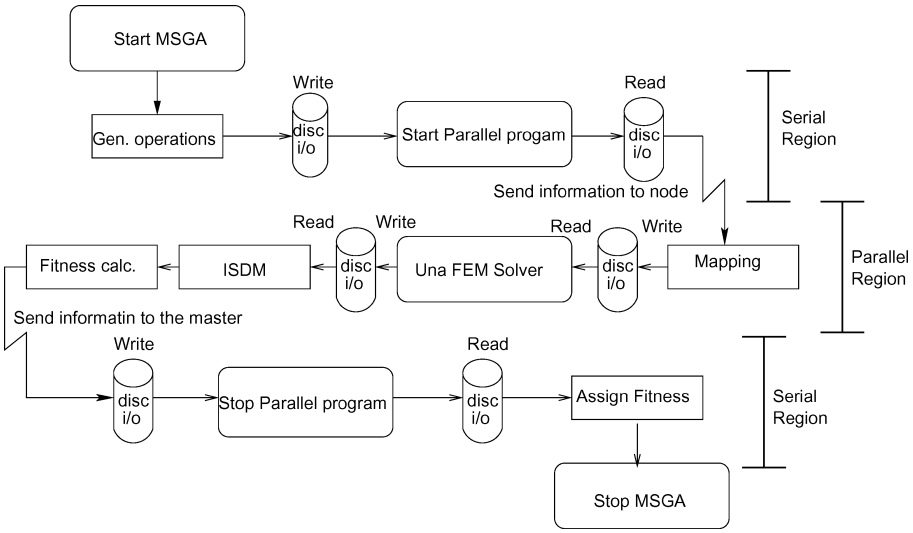

Fig. 5 Parallel multispecies genetic-algorithm flowchart.

implemented in the MATLAB®[§§] environment running on the master node, and the fitness evaluation is run on the slave nodes. When the genetic operations of one generation are completed, the member information is saved to the disk. The master program then calls the parallel program written in C++ with a shell command. This program reads in the members and distributes them to the slave nodes of the cluster. At each node, a member is mapped into a job-deck of the UNA finite element method (FEM) solver.[17] The UNA solver essentially calculates the new mass and stiffness matrices associated with the member and writes it to the local node disk. The "approximate" natural frequencies and mode shapes associated with the member are then calculated locally by using the iterative structural-dynamics modification theory.[18] The fitness function is then evaluated, and the member fitness is returned to the master node where the fitness information of all members is written to disk. The parallel program terminates, and the master program reads in the result and assigns the fitness values to the members, thereby completing one iteration.

The disk input/output in the parallel portion of the code is necessary because the UNA FEM code requires virtual memory and swap files. To minimize the communication on the network, all files needed by the FEM code reside on each local node. The reason that only one processor is used at each local node (recall that each local node does have two processors) is that disk I/O dominates the FEM solution. Therefore, disk I/O is a bottleneck when two processors are used.

V. Fitness Function Formulation

The objective function investigated for model refinement involves a comparison of updated and experimental modal parameters. A comparison function can be defined as

$$J(\boldsymbol{r}) = \sum_{i=1}^{p} \frac{|\omega_i(\boldsymbol{r}) - \omega_{i\_\exp}|}{\omega_{i\_\exp}} + \eta \quad \begin{array}{l} \min \|V(\boldsymbol{r}) - V_{\exp} R_{\text{op}}\|_f \\ \text{wrt} \\ R_{\text{op}} \\ \text{subject to} \quad R_{\text{op}}^T R_{\text{op}} = I \end{array} \tag{3}$$

where ω_i is the ith natural frequency and V is the mode-shape matrix. The modal properties with the subscript exp correspond to the experimentally measured modal parameters. The modal properties associated with physical parameter vector $\boldsymbol{r}$ are associated with the finite element modal parameters. The variable η is a weight factor that can be changed to emphasize agreement between natural frequencies or mode shapes. The first summation of Eq. (3) provides for the minimization between measured and updated analytic natural frequencies. The second term of Eq. (3) represents a newly developed approach for comparing analytical and experimental mode shapes.[12] This comparison approach was motivated by the common occurrence of having closely spaced modes of vibration. Previous formulations, which dealt with a direct comparison of an individual analytical/experimental mode-shape pair, require the use of a frequency tracking algorithm. In addition, it was often found that for closely spaced modes the identified experimental mode shapes were in fact linear combinations of the analytical mode shapes, making a direct comparison difficult. Rather than dealing with a direct comparison, the second term of Eq. (3) compares the Frobenius f norm difference of the subspaces spanned by the analytical and experimental mode shapes. The unitary matrix R_{op} is a rotation matrix that rotates the experimental mode shapes into the analytical mode shapes. The calculation of the rotation matrix R_{op} is computationally efficient. The main calculation is the singular value decomposition of a square matrix whose dimension is equal to the number of measured modes of vibration. Because GAs are cast in a maximization mode, the fitness function will be taken as the negative of Eq. (3).

VI. Numerical Experiments

A. Parallel Performance Study

The most important measure of performance in a parallel program is speedup. Speedup is defined as

$$\text{speedup} = \frac{\text{execution time for the serial program}}{\text{execution time for the parallel program}} \tag{4}$$

Another important measure of performance is the concept of scaling. The scaling properties of a program show how much will be gained by increasing the number of processors.

To measure the speedup of the PMSGA, the program is run on 16 processors, where different function evaluation problem sizes are simulated by a wait function with variable execution times. With 16 processors, the maximum value of speedup defined by Eq. (4) would be 16. The measured speedup and scaling performance is shown in Figs. 6a and 6b. From Fig. 6a, it is clear that there is nothing to gain using the PMSGA on a small problem. This is not very surprising because there will be a significant overhead in communication between the nodes. The speedup increases rapidly as the problem size grows. For a problem size of 200 s, the speedup is close to the theoretical limit of 16. For the scaling study, the population size is set to 16 members, and the problem size is 180 s. The scaling properties are quite close to the theoretical limit.

[§§]Data available online at http://www.mathworks.com/products/prod.verview.shtml.

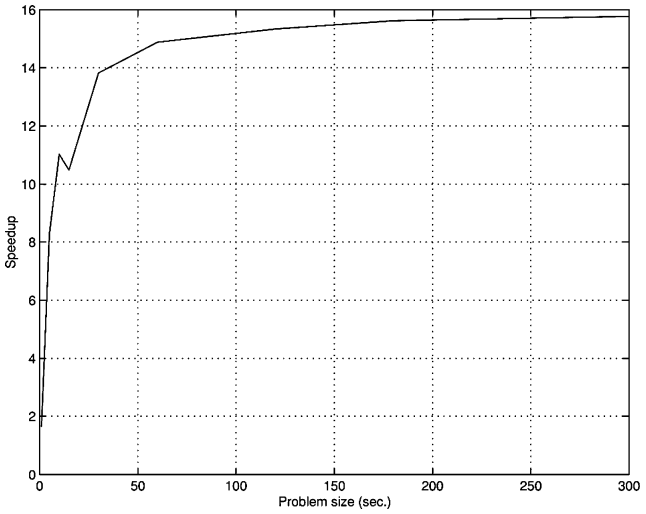

Fig. 6a PSMGA speedup.

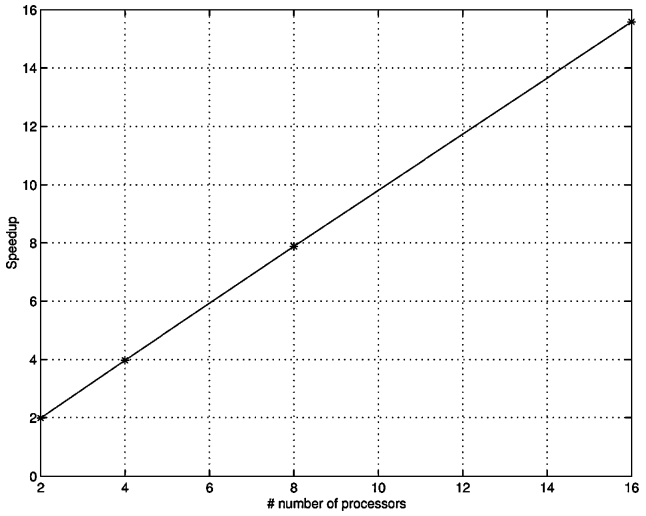

Fig. 6b PSMGA scaling.

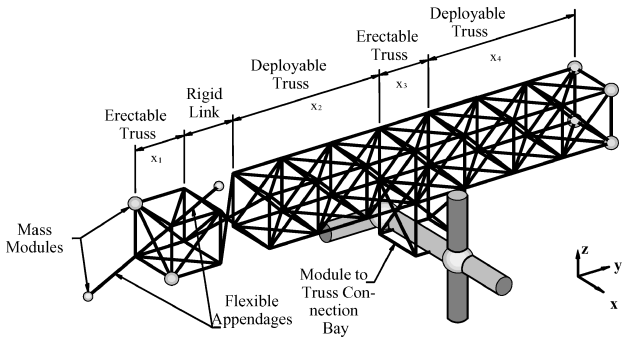

Fig. 7 MiNDE test structure.

In conclusion, the parallel program achieves a speedup close to the theoretical maximum value. The program also responds well to scaling, as long as the number of members in the population exceeds or equals the number of processors used.

B. Structure Model and Problem Definition

In this example, the Middeck Nondestructive Experiment (MiNDE) test structure model shown in Fig. 7 is used. This structure was designed by McDonnell Douglas to simulate typical complex space structure dynamics. The MiNDE structure was a derivative of the Middeck 0-Gravity Dynamics Experiment structure.[19] The structure consists of two one-bay erectable truss sections, two three-bay deployable truss sections, an articulating joint, a cluster of mass modules, two flexible appendages, and a module to truss interface assembly. The first four flexible modes of vibration are the first bending modes of the two flexible appendages in each plane of motion. These modes are on average 30% lower in frequency than the first global mode of vibration, which occurs at 17 Hz. The finite element model of this structure has 522 DOF and 221 elements.

The PMSGA will be configured in all studies in the same way. The value of Young's modulus of the truss members for four distinct regions of the MiNDE truss are used as the physical parameters to be estimated. These are depicted with the symbols x_1–x_4 in Fig. 7. A value of $x_i = 1.00$ for the decision variables corresponds to the nominal model Young's modulus, and for estimation purposes they will be allowed to vary between 0.25 and 1.75. The weight η in Eq. (3) is adjusted such that the contribution of the frequency and mode-shape error terms between the nominal model(s) and the experimental values will be equal. The experimental data are obtained by a full solution of the generalized eigenvalue problem with some predefined fixed values of the parameters. Finally, the PMSGA will be run for a total of 25 generations.

C. Parameter Estimation

The experimental data are obtained by solving the eigenvalue problem where the four variables have all been set to 1.5. The experiment will be performed with four different species; MiNDE1, MiNDE2, MiNDE3, and MiNDE4. The first species, MiNDE1, has only x_1 as a decision variable, MiNDE2 will have x_1 and x_2 as decision variables, and so on. The initial population is created with eight members of each species, making the total population size 32. In this experiment, there is no difference in fitness between the nominal species because $x_i = 1.0$ corresponds to exactly the same model. The fitness for a nominal species is $-J = -14.0318$. MiNDE4 is the species that can obtain the maximum fitness in the population with a theoretical value of $-J = 0$.

The performance of the PMSGA will be examined by repeating the experiment 20 times with different initial populations. As a reference, each species will be optimized in a single species population with 16 members, and these experiments will be repeated 10 times. Figure 8 shows a typical result from an optimization with the four-species population. The average fitness of the population is below the fitness of the nominal species in the first generation, whereas after about five generations the average fitness has increased above the nominal fitness. The maximum fitness and the distribution of the different species are shown in Figs. 9a and 9b, respectively. From these figures, it is seen that MiNDE1 and MiNDE2 do not survive past the eighth generation. These typical results therefore serve as a

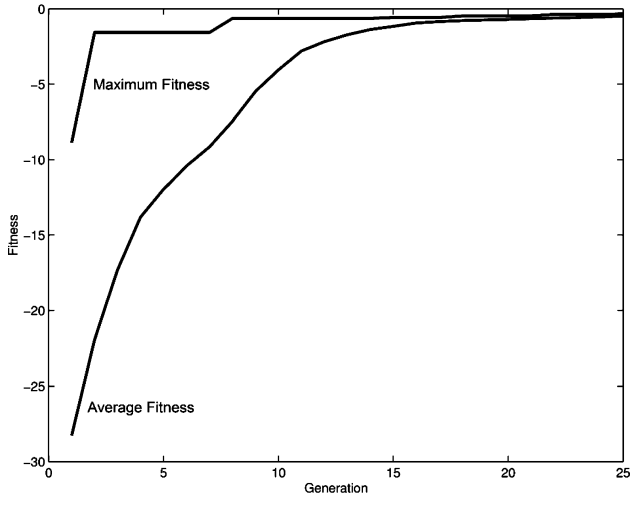

Fig. 8 Maximum and average fitness of the population.

Table 1 Fitness and decision variables for four-species population after 25 generations

Species	Fitness		x_1	x_2	x_3	x_4
MiNDE1	Max	−11.8860	1.7089	N/A	N/A	N/A
	Average	−11.9270	1.7063	N/A	N/A	N/A
MiNDE2	Max	−4.8390	1.7483	1.7326	N/A	N/A
	Average	−9.9720	1.6855	1.5239	N/A	N/A
MiNDE3	Max	−0.4249	1.4974	1.7022	1.5326	N/A
	Average	−0.5246	1.5439	1.6399	1.5272	N/A
MiNDE4	Max	−0.0980	1.4944	1.5845	1.4945	1.4317
	Average	−0.9458	1.4986	1.5040	1.5034	1.3996

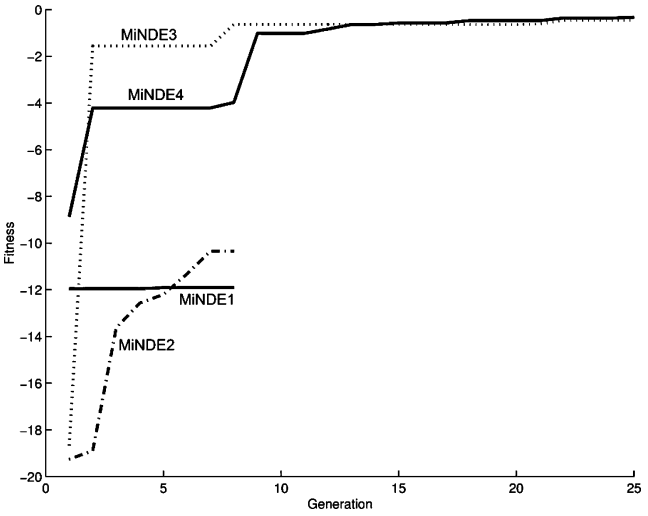

Fig. 9a Maximum fitness of species.

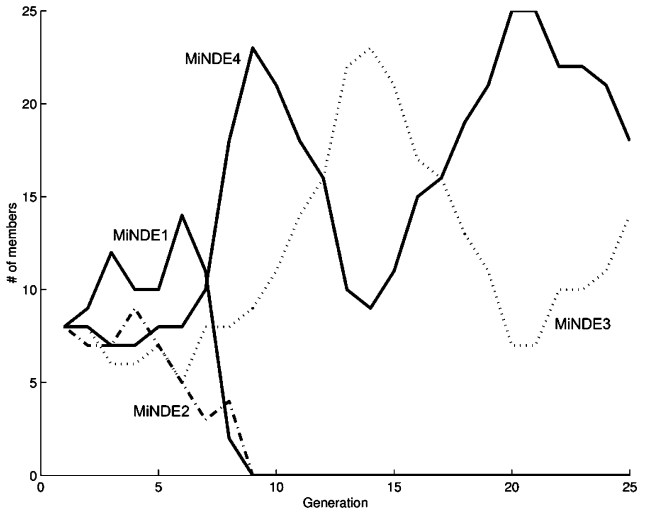

Fig. 9b Distribution of members.

validation of the claim that the GA can quickly ferret out unsuitable model forms.

Table 1 shows the result from the repeated experiments with the four-species population. It is clear in Fig. 9 that not all species survive to the final generation. The values shown in Table 1 are therefore the maximum values of the different species over all generations. The values obtained from repeating the optimization of the single-species population are shown in Table 2. The discrepancy between the maximum fitness of the species in Tables 1 and 2 is not very large, showing the PMSGA produces results that are essentially equivalent to running four independent optimizations. However, there is a notably large difference in the average fitness of MiNDE4. This is because MiNDE4 does not always become the optimal species. The success rate of this experiment is 90%, that

Table 2 Fitness and decision variables for single-species optimization after 25 generations

Species	Fitness		x_1	x_2	x_3	x_4
MiNDE1	Max	−11.8837	1.7087	N/A	N/A	N/A
	Average	−11.8932	1.7076	N/A	N/A	N/A
MiNDE2	Max	−4.7114	1.7500	1.7465	N/A	N/A
	Average	−4.8170	1.7492	1.7257	N/A	N/A
MiNDE3	Max	−0.4023	1.5093	1.6790	1.5340	N/A
	Average	−0.4874	1.5222	1.6727	1.5242	N/A
MiNDE4	Max	−0.0360	1.5050	1.5088	1.4999	1.4766
	Average	−0.1867	1.4958	1.4979	1.5167	1.4887

Table 3 Fitness and decision variables for two-species optimization after 25 generations

Species	Fitness		x_3
MiNDE:B	Max	-3.4543×10^{-4}	1.5000
	Average	−0.0346	1.5021
MiNDE:R	Max	−13.7564	1.1430
	Average	−13.8135	1.1454

Table 4 Fitness and decision variables for two-species optimization after 25 generations

Species	Fitness		x_3
MiNDE:B	Max	-7.9873×10^{-4}	1.4999
	Average	−0.0074	1.4991
MiNDE:R	Max	−13.7556	1.1454
	Average	−13.7568	1.1449

is, in 18 cases of the 20 performed experiments MiNDE4 is the optimal species. In the other two cases MiNDE3 is identified as the optimal species. This occurs because there is very little difference in achievable fitness between MiNDE3 and MiNDE4 as seen in the final maximum fitness as shown in Fig. 9a. If the average fitness for MiNDE4 is calculated only when the algorithm is successful, a comparable value of −0.2486 is obtained.

D. Model Structure Verification

In this numerical study, the PMSGA's capability of model structure verification will be examined. Two different finite element models of the MiNDE structure will be used. The models differ for the erectable truss section in the middle of the structure. In MiNDE:B, the section is modeled with bar elements, whereas in MiNDE:R rod elements are used. The parameter to be optimized is Young's modulus of the material used in the section. The experimental data will be generated by using MiNDE:B with a value of $x_3 = 1.5$. The nominal fitness for MiNDE:B ($x_3 = 1.0$) is $-JB = -6.8767$ and for MiNDE:R ($x_3 = 1.0$) is $-JR = -15.2243$. The theoretical upper limit for the fitness is as before $-J = 0$. The two-species population will have eight members of each species in the initial population, and the performance will be compared again with a single-species optimization with 16 members.

Figures 10 and 11a and 11b show typical results from the two-species optimization. From Fig. 10, it is seen that the result is close to the theoretical limit of the fitness already in the early generations. This is not surprising because the species have only one decision variable. Table 3 shows the maximum and average values of the repeated experiments with the two-species population. The correct model MiNDE:B is identified with the optimal parameter value of $x_3 = 1.5$. Table 4 shows the values obtained from optimizing the single-species population. The success rate for this experiment is 100%. Comparing the tables, it is seen that MiNDE:R was optimized before it became extinct, despite only surviving for only four generations.

E. Model Structure Verification and Parameter Estimation

In this numerical study, two different finite element models of the MiNDE structure are used as described in Sec. VI.D. The PMSGA will be configured with four species: MiNDE2:B, MiNDE4:B,

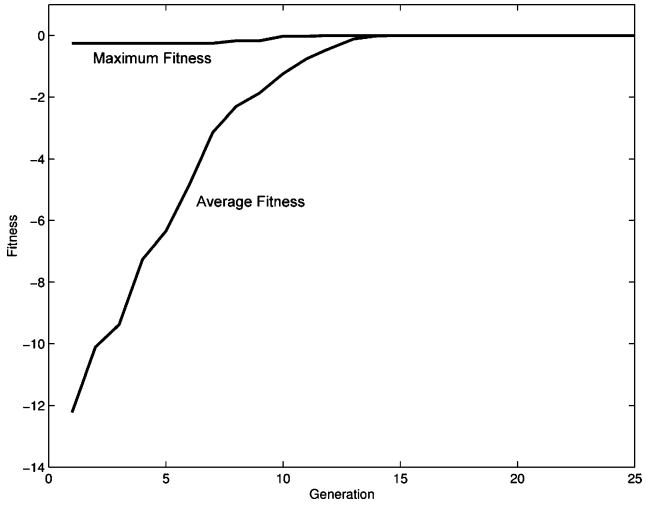

Fig. 10 Maximum and average fitness of the population.

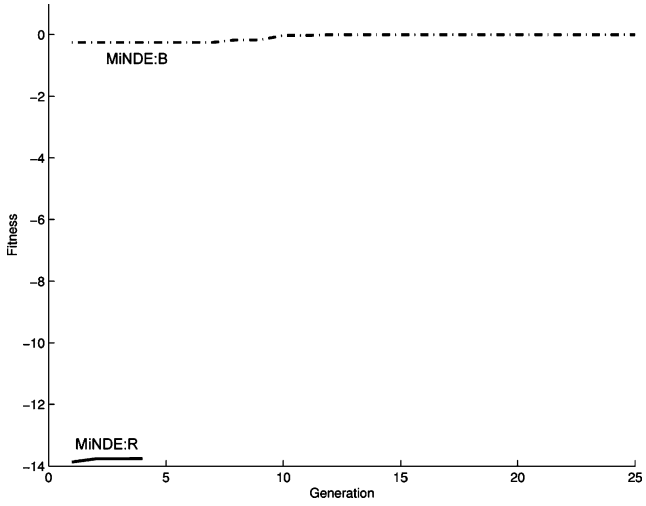

Fig. 11a Maximum fitness of species.

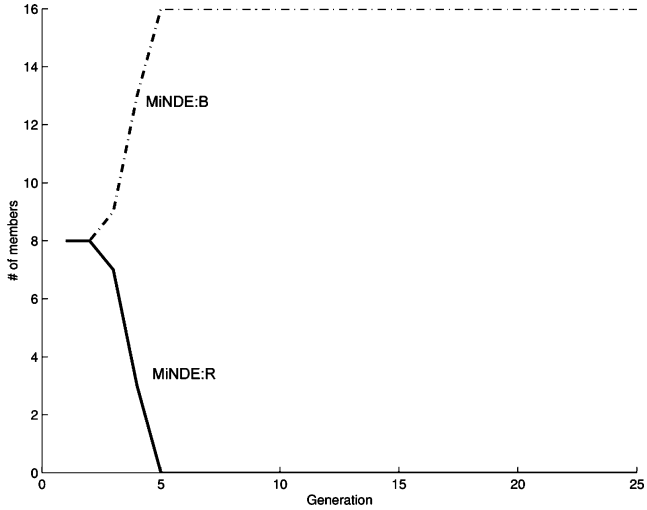

Fig. 11b Distribution of members.

MiNDE2:R, and MiNDE4:R. The digit in the name of the species corresponds to the number of decision variables: two (x_1, x_2) and four (x_1, x_2, x_3, x_4), respectively. The experimental data are generated using MiNDE4:B with all the x_i values set to 1.5.

MiNDE2:B and MiNDE4:B correspond exactly to MiNDE2 and MiNDE4 used in the first numerical study. The nominal fitness of the two species therefore will be the same; $-JB = -14.0318$. The nominal fitness for MiNDE2:R and MiNDE4:R is $-JR = -25.5634$.

Table 5 Fitness and decision variables for four-species population after 25 generations

Species	Fitness		x_1	x_2	x_3	x_4
MiNDE2:B	Max	−4.7959	1.7490	1.7453	N/A	N/A
	Average	−5.7774	1.7244	1.6857	N/A	N/A
MiNDE4:B	Max	−0.0508	1.4920	1.5219	1.5066	1.4938
	Average	−0.2094	1.5000	1.4808	1.5131	1.4751
MiNDE2:R	Max	−14.5394	1.7407	1.6659	N/A	N/A
	Average	−19.7632	1.5652	1.4956	N/A	N/A
MiNDE4:R	Max	−6.2025	1.7158	1.7474	1.7151	1.7089
	Average	−14.3626	1.5987	1.4892	1.3476	1.1489

Table 6 Fitness and decision variables for single-species optimization after 25 generations

Species	Fitness		x_1	x_2	x_3	x_4
MiNDE2:B	Max	−4.7114	1.7500	1.7465	N/A	N/A
	Average	−4.8170	1.7492	1.7257	N/A	N/A
MiNDE4:B	Max	−0.0360	1.5050	1.5088	1.4999	1.4766
	Average	−0.1867	1.4958	1.4979	1.5167	1.4887
MiNDE2:R	Max	−14.3102	1.7500	1.7489	N/A	N/A
	Average	−14.3214	1.7491	1.7472	N/A	N/A
MiNDE4:R	Max	−5.9774	1.7335	1.7374	1.7282	1.7449
	Average	−6.8475	1.7391	1.7266	1.7209	1.7272

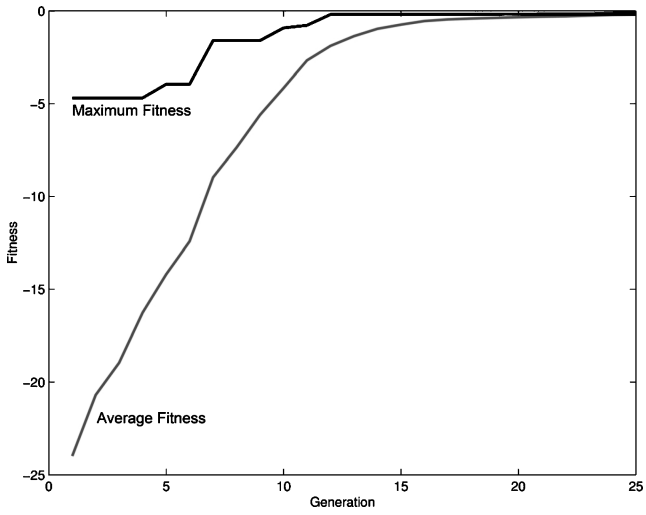

Fig. 12 Maximum and average fitness of the population.

The PMSGA will have eight members of each species in the initial population, and, as before, the experiments will be repeated 20 times. The result will be compared with 10 repeated experiments of the single-species population with 16 members.

The difference in nominal fitness between the models suggests again that MiNDE*:B models the physics of the structure more accurately. Figures 12 and 13a and 13b show typical results from the four-species optimization. It is seen that the rod models quickly become extinct and that from generation 12 the only species that survives is MiNDE4:B. Tables 5 and 6 show the maximum and average fitness of the four species, as well as the average values of the decision variables. The success rate for this experiment is 100%. The discrepancy in nominal fitness is not surprising because the PMSGA is independent of the species it optimizes. The only factor that can contribute to the success rate of the algorithm is the difference in fitness between the members in the population. If the discrepancy in fitness is not noticeable between any species, then most probably they will all survive to the final generation.

Table 5 shows the result from the repeated experiments with the four-species population. The values obtained from repeating the optimization of the single-species population are shown in Table 6. The discrepancy between the maximum fitness of the species in Tables 5 and 6 is not very large for the MiNDEx:B structures, although there

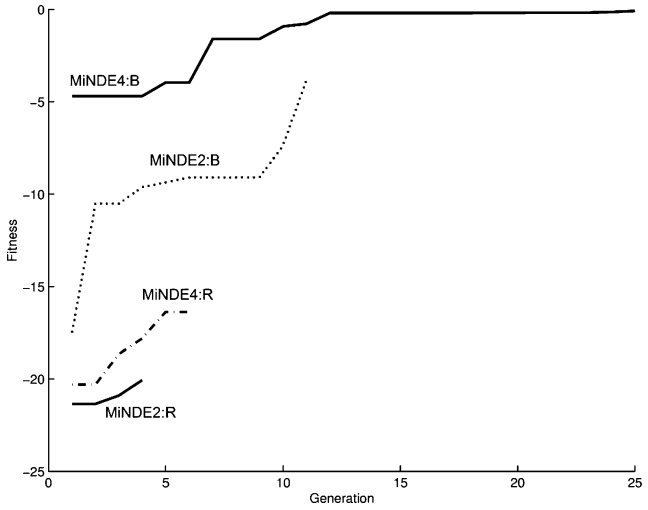

Fig. 13a Maximum fitness of species.

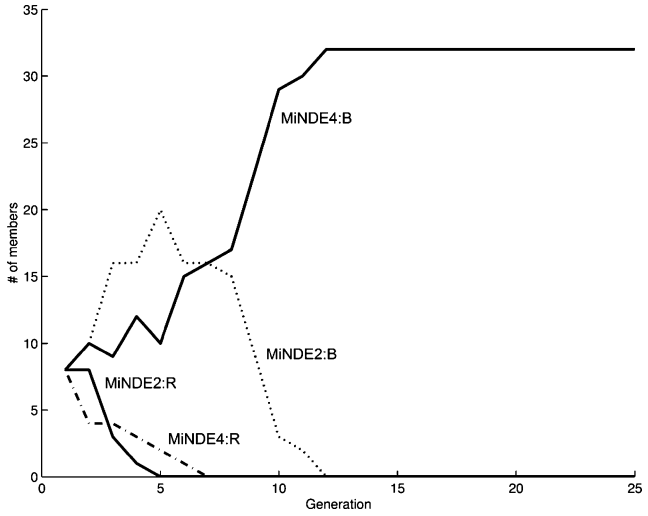

Fig. 13b Distribution of members.

is some difference for the average fitness of the MiNDEx:R structures. The reason for this is that the independent optimization runs were optimized for 25 generations, whereas in the multispecies the species only existed for less than eight generations. However, the most important result is that the maximum fitness, or in other words the optimal solution, between the two approaches is nearly identical, showing that the PMSGA produces results that are essentially equivalent to running four independent optimizations.

VII. Conclusions

In this work, a parallel, multispecies genetic algorithm has been developed and tested. The motivation for developing the multispecies algorithm is to allow both model physics and physical parameters to be updated to better match experimental observations. The parallel evaluation of the members of the genetic-algorithm population greatly reduces the required computation time. The algorithm responds well to scaling, and given even moderate problem size the speedup achieved is close to the theoretical limit. It was demonstrated that the multispecies optimization produces nearly equivalent results to running each species in its own separate optimization.

Acknowledgments

The authors acknowledge the support of the Texas Institute for Intelligent Bio-Nano Materials and Structures for Aerospace Vehicles, funded by NASA Cooperative Agreement NCC-1-02038 and the National Science Foundation (Grant CMS-0084573). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NASA. The authors are most grateful to Zoran Rudic, who provided the UNA finite element program and actually made several modifications to the source program at the request of the authors.

References

[1]Clarke, D. W., "Generalized Least-Squares Estimation of the Parameters of a Dynamic Model," *Proceedings of the IFAC Symposium on Identification of Automatic Control Systems*, Paper 3.17, June 1967.

[2]Collins, J. D., Hart, G. C., Hasselman, T. K., and Kennedy, B., "Statistical Identification of Structures," *AIAA Journal*, Vol. 12, No. 2, 1974, pp. 185–190.

[3]Chen, Y. M., and Lin, Y., "An Iterative Algorithm for Solving Inverse Problems in Structural Dynamics," *International Journal for Numerical Methods in Engineering*, Vol. 19, June 1983, pp. 825–829.

[4]Berger, H., Chaquin, J. P., and Ohayon, R., "Finite Element Model Adjustment Using Experimental Data," *Proceedings of the 2nd International Modal Analysis Conference*, Society for Experimental Mechanics, Bethel, CT, 1984, pp. 638–642.

[5]Hoff, C. J., Bernitsas, M. M., Sandstrom, R. E., and Anderson, W. J., "Inverse Perturbation Method for Structural Redesign with Frequency and Mode Shape Constraints," *AIAA Journal*, Vol. 22, No. 9, 1984, pp. 1304–1309.

[6]Soeiro, F. J., and Hajela, P., "Damage Detection in Composite Materials Using Identification Techniques," *Proceedings of the 31st AIAA Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1990, pp. 950–960.

[7]Hemez, F. M., and Farhat, C., "Correlating Finite Element Models to Modal Tests for Large Flexible Space Structures," *Proceedings of the 1993 CSMA/GAMNI/INRIA National Conference in Structural Design*, 1993, pp. 480–493.

[8]Maben, E., and Zimmerman, D. C., "Advances in Parameter Estimation Techniques Applied to Flexible Structures," NASA CP 3242, June 1994.

[9]Larson, C., and Zimmerman, D. C., "On the Use of Genetic Algorithms in Structural Damage Detection," *Proceedings of the 11th International Modal Analysis Conference*, Society for Experimental Mechanics, Bethel, CT, 1993, pp. 1095–1101.

[10]Fulcher, C. W., Marek, E. L., and Mayes, R. L., "Test/Analysis Correlation and Model Updating of the STARS I & II Missile Systems," *Structural Dynamic Modelling Test, Analysis and Correlation*, Bell and Bain, Glasgow, Scotland, U.K., 1993, pp. 433–446.

[11]Carlin, R. A., and Garcia, E., "Parameter Optimization of a Genetic Algorithm for Structural Damage Detection," *Proceedings of the 14th International Modal Analysis Conference*, Society for Experimental Mechanics, Bethel, CT, 1996, pp. 1292–1298.

[12]Zimmerman, D. C., Yap, C. K., and Hasselman, T., "Evolutionary Approach for Model Correlation," *Mechanical Systems and Signal Processing*, Vol. 13, No. 4, 1999, pp. 609–625.

[13]Ruotolo, R., Surace, C., and Mares, C., "Damage Identification Using Simulated Annealing," *Proceedings of the 15th International Modal Analysis Conference*, Society for Experimental Mechanics, Bethel, CT, 1997, pp. 954–960.

[14]Holland, J. H., *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, MI, 1975, Chaps. 1–6.

[15]Adamidis, P., "Parallel Evolutionary Algorithms: A Review," *Proceedings of the 4th Hellenic-European Conference on Computer Methods and Its Applications*, edited by E. A. Lipitakis, Dept. of Infomatics, Athens Univ. of Economics and Business, Athens, Greece, 1998, Session 12E.

[16]Cantu-Paz, E., "A Survey of Parallel Genetic Algorithms," *Calculateurs Parallelles, Reseaux et Systems Repartis*, Vol. 10, No. 2, 1998, pp. 141–171.

[17]Rudic, Z., *UNA—Computer Program for Static and Dynamic Structural Analysis by Finite Element Method: User and Verification Manual*, Structural Analysis and Research Systems, Inc., 2001.

[18]Yap, K. C., and Zimmerman, D. C., "A Comparative Study of Structural Dynamics Modification and Sensitivity Method Approximation," *Mechanical Systems and Signal Processing*, Vol. 16, No. 4, 2002, pp. 585–597.

[19]Crawley, E. F., Van Schoor, M. C., and Bokhour, E. B., "The Middeck 0-Gravity Dynamics Experiment—Summary Report," NASA CR 4500, Jan. 1993.

B. Balachandran
Associate Editor