ELSEVIER

Comput. Methods Appl. Mech. Engrg. 186 (2000) 357–372

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Permutation genetic algorithm for stacking sequence design of composite laminates

Boyang Liu [a], Raphael T. Haftka [a,*], Mehmet A. Akgün [a,1], Akira Todoroki [b]

[a] *Department of Aerospace Engineering, Mechanics and Engineering Science, University of Florida, Gainesville, FL 32611-6250, USA*

[b] *Department of Mechano-Aerospace Engineering Tokyo Institute of Technology, 2-12-1 Ohokayama, Meguro-Ku, Tokyo 152, Japan*

Received 30 March 1999

Abstract

Stacking sequence design of a composite laminate with a given set of plies is a combinatorial problem of seeking an optimal permutation. Permutation genetic algorithms optimizing the stacking sequence of a composite laminate for maximum buckling load are studied. A new permutation GA named gene–rank GA is developed and compared with an existing Partially Mapped Permutation GA, originally developed for solving the travelling salesman problem. The two permutation GAs are also compared with a standard non-permutation GA. It is demonstrated through examples that the permutation GAs are more efficient for stacking sequence optimization than a standard GA. Repair strategies for standard GA and the two permutation GAs for dealing with constraints are also developed. It is shown that using repair can significantly reduce computation cost for both standard GA and permutation GA. © 2000 Elsevier Science S.A. All rights reserved.

1. Introduction

Due to their high strength to weight ratio and the possibility of tailoring their stiffness by selecting fiber orientations, the optimization of composite laminated plates has received growing attention in the last decades [1–5]. The design of composite laminates is often formulated as a continuous optimization problem with ply thickness and ply orientation angles used as design variables [4]. However, for many practical problems, ply thickness is fixed, and ply orientation angles are limited to a small set of angles such as 0°, ±45°, and 90°. Thus the design problem becomes a combinatorial problem of choosing the fiber direction from a permissible set for each ply.

Genetic algorithms (GA) have been used extensively to solve this combinatorial problem [1,5]). Genetic algorithms are well suited for stacking sequence optimization, and because of their random nature, they easily produce alternative optima in repeated runs. This latter property is particularly important in stacking sequence optimization, because widely different stacking sequences can have very similar performance [6].

Stacking sequence design of composite panels is a local design problem that is often strongly coupled to the overall design of a structure. In wing structural optimization, the overall wing structural design, imposes constraints on individual panel designs. The optimization of the overall wing structure often specifies

[*] Corresponding author.

E-mail addresses: boyang@ufl.edu (B. Liu), haftka@ufl.edu (R.T. Haftka), akgun@tresca.aero.ufl.edu (M.A. Akgün), atodorok@mes.titech.ac.jp (A. Todoroki).

[1] Visiting, at the time of the work, from Aeronautical Engineering Department, Middle East Technical University, 06531 Ankara, Turkey. A version of the paper was presented as AIAA Paper 89-1830, Proceedings of the 39th AIAA/ASME/ASCE/AHS Structures, Structure Dynamics and Material Conference, Long Beach, CA, April 20–23,1998.

the number of 0°, ±45°, and 90° plies and in-plate loads of each panel. The stacking sequence design is then limited to permutations of given plies, but not to changes in the number of plies of each orientation.

It is possible to solve this problem by using a conventional GA with additional constraints imposed on the design. However, permutation GAs, developed mostly for solving scheduling problems [7], handle more efficiently the search for an optimal permutation, because they reduce the dimensionality of the design space. Permutation GAs mostly developed for the travelling salesman problem, which seeks to minimize travel cost for a given list of towns, and is insensitive to where the sequence starts, so that cyclical permutations do not matter. In stacking sequence design, in contrast, a cyclical permutation will move the outermost ply into the innermost position, and thus greatly influence the bending properties of the laminate.

Aside from the use of permutation GAs, number-of-ply constraints may be handled by repair strategies. Such repair strategies may also be useful for dealing with another constraint common to a laminate design – a limit on the number of contiguous plies of the same orientation.

The first objective of this work is to devise a permutation GA that is better suited to stacking sequence design. This permutation algorithm is compared to a standard permutation GA, Partially Mapped GA [8], as well as to a standard genetic algorithm. The new algorithm shares some properties with Bean's Random Keys algorithm [9], and therefore the two algorithms are compared. The second objective is to devise repair strategies for standard GA and permutation GA based on a Baldwinian repair strategy introduced in [24]. The algorithms are compared for maximization of the buckling load of a laminate with specified number of 0°, ±45°, and 90° plies.

Genetic algorithms are random in nature, and therefore comparing the efficiencies of alternative algorithms requires averaging many runs. For this reason, we selected a simply supported unstiffened panel, for which closed form solutions are available. We can thus perform the millions of analyses required for a thorough comparison of the efficiency of the various genetic algorithms. The efficiency of the algorithms is then reported in terms of number of analyses required for high reliability in finding the optimum design. Computation times are not given because they are dominated by GA operations, while in more realistic problems they will be dominated by structural analyses.

In the rest of the paper, we start by describing the physical model of the composite laminates, and a standard formulation of optimization of a composite laminate. A new permutation GA, which we call gene–rank crossover GA, suited for stacking sequence optimization is developed, and the standard GA and a permutation GA based on partially mapped crossover are reviewed and implemented. The computational efficiency of the three GAs are then compared under various load cases. Effect of a contiguity constraint on performance, which limits the number of identical adjacent ply orientations to four, is investigated. Two repair strategies, chromosome repair and laminate repair, for permutations violating the contiguity constraint are discussed.

2. Analysis and optimization

The paper deals with the optimization of symmetric and balanced stacking sequences of composite wing panels. Usually, a panel is to be designed for given in-plane loading and specified total number of of 0°, ±45°, and 90° plies. The loading and the specified number of plies come from the overall wing level optimization. Here the panel is designed to maximize the buckling load subject to a constraint on the number of contiguous plies of the same orientation.

An unstiffened, simply supported, laminated panel with dimensions a and b (Fig. 1) is subjected to normal loads per unit length N_x and N_y, and a shear load per unit length N_{xy}. It is made of a symmetric and balanced graphite–epoxy laminate composed of 0°, ±45°, and 90° plies.

Because of symmetry, there is no extensional–flexural coupling, so that the pre-buckling deformations are purely in-plane. The balance condition requires that for every ply with a positive fiber orientation angle, there is a corresponding ply with the negative fiber orientation angle. This implies that there is no normal-shear extensional couplings. In addition, the laminate is assumed specially orthotropic (i.e. there will be no bending-torsion coupling). This is a common assumption in the analysis of balanced symmetric laminates for which the bending-torsion coupling terms are usually very small and can be neglected.

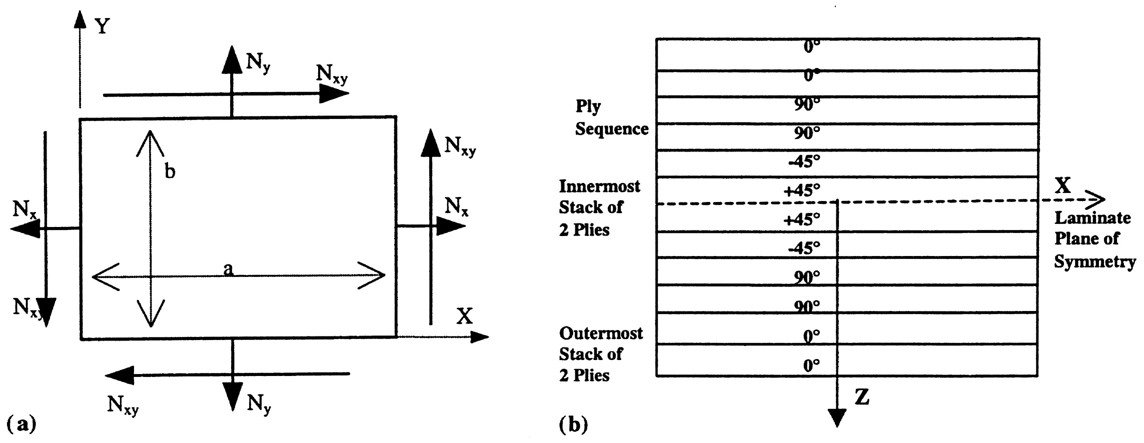
Y
N_y
N_xy
b
N_x
a
X
(a)
Ply Sequence
Innermost Stack of 2 Plies
Outermost Stack of 2 Plies
0°
90°
-45°
+45°
X
Laminate Plane of Symmetry
Z
(b)

Fig. 1. Laminate plate geometry and loading: (a) laminate plate geometry and applied loading; (b) ply sequence location.

3. Buckling analysis

Under biaxial loading, the laminate can buckle into m and n half waves in the x and y directions, respectively, when the load amplitude (a factor multiplying the applied loads) reaches a value $\lambda_{\mathrm{n}}^{(m,n)}$, which is given in terms of flexural stiffness D_{ij} and loads N_x and N_y

$$\frac{\lambda_{\mathrm{n}}^{(m,n)}}{\pi^2} = \frac{D_{11}(m/a)^4 + 2(D_{12} + 2_{66})(m/a)^2(n/b)^2 + D_{22}(n/b)^4}{(m/a)^2 N_x + N_s + (n/b)^2 N_y}. \tag{1}$$

The pair (m, n) that yields the smallest value of $\lambda_{\mathrm{n}}^{(m,n)}$, which is the critical buckling load λ_{cb}, varies with the loading case, total number of plies considered, material, and the plate aspect ratio.

A second mechanism is buckling due to shear loading. Modeling of this buckling mode for a finite plate is computationally expensive. Instead, the plate is assumed to have an infinite length, and analytical solutions available for a plate of infinite length in the x direction are used as approximations [10]. The critical shear buckling load factor λ_{s} is given in Whitney as a function of the variable Γ.

$$\Gamma = \frac{\sqrt{D_{11}D_{22}}}{D_{12} + 2D_{66}} \tag{2}$$

and, values of β_1 are given in Table 1.

Table 1
Coefficient β_1 for shear buckling load factor [10]

Γ	β_1
0.0	11.71
0.2	11.80
0.5	12.20
1.0	13.17
2.0	10.80
3.0	9.95
5.0	9.25
10.0	8.70
20.0	8.40
40.0	8.25
∞	8.13

$$
\begin{aligned}
\lambda_s &= \frac{4\beta_1 (D_{11} D_{22}^3)^{1/4}}{b^2 N_{xy}}, \quad \text{for } 1 \leqslant \Gamma \leqslant \infty, \\
&= \frac{4\beta_1 \sqrt{D_{22}(D_{12} + 2D_{66})}}{b^2 N_{xy}}, \quad \text{for } 0 \leqslant \Gamma \leqslant 1,
\end{aligned}
\tag{3}
$$

when normal and shear loads are applied simultaneously to the panel their interaction is approximated by the following interaction equation [11]

$$
\frac{1}{\lambda_c^{(m,n)}} = \frac{1}{\lambda_n^{(m,n)}} + \frac{1}{\lambda_s^2}, \tag{4}
$$

where $\lambda_n^{(m,n)}$ and λ_s are the critical load amplitudes under normal and shear loads, respectively. The combined buckling load factor $\lambda_c^{(m,n)}$ is always more critical than the normal buckling load factor $\lambda_n^{(m,n)}$.

To prevent buckling, $\lambda_c^{(m,n)}$and λ_s have to be greater than one. Shear buckling occurs independently of the sign of the shear load. So buckling load λ is taken to be the minimum of the load factors

$$
\lambda = \min\left\{ |\lambda_s|, \lambda_c^{(m,n)} \right\}. \tag{5}
$$

Additionally, to reduce problems with matrix cracking, we do not allow more than four contiguous plies with the same orientation. This is referred to as the contiguity constraint.

4. Statement of optimization

For maximizing the buckling load of composite laminates for given total number of 0°, ±45°, and 90° plies, and ply contiguity constraints, the optimization problem can be stated as follows:

- *Given* three ply orientation choices (0°, ±45°, and 90°), applied in-plane normal and shear loading N_x, N_y, and N_{xy}, and the total number of 0°, ±45°, and 90° plies.
- *Optimize* a symmetric and balanced stacking sequence in order to maximize the buckling load λ (that is the panel will buckle under loads λN_x, λN_y, and λN_{xy}).
- *Subject to* the constraints that there be no more than four contiguous plies of the same orientation, and the number of 0°, ±45°, and 90° plies be equal to given total number of 0°, ±45°, and 90° plies.

Results were obtained for a 24-inch square graphite–epoxy plate with the following properties: elastic moduli in fiber and transverse directions $\mathbf{E}_1 = 18.5 \times 10^6$ psi (127.59 GPa) and $\mathbf{E}_2 = 1.89 \times 10^6$ psi (13.03 GPa); shear modulus $\mathbf{G}_{12} = 0.93 \times 10^6$ psi (6.4 GPa); unit ply thickness $\mathbf{t}_{\text{ply}} = 0.005$ in (0.0127 cm); and Poisson's ratio $\nu_{12} = 0.3$.

5. Genetic algorithms

A genetic algorithm is a guided random search technique that works on a population of designs. Each individual in the population represents a design, i.e. a stacking sequence, coded in the form of a bit string. The genetic algorithm begins with the random generation of a population of design alternatives. Designs are processed by means of genetic operators to create a new population, which combines the desirable characteristics of the old population, and then the old population is replaced by the new one. Herein the best design of each generation is always copied into the next generation, which is called an elitist strategy. The process is repeated for a fixed number of generations or for a fixed number of analyses without improvement in the best design.

Early implementation of the genetic search method is credited to Bremermann [12] and Rechenberg [13], although Holland's [14] work has formed the basis of most contemporary developments. Early applications of genetic algorithms to structural optimization are due to Goldberg and Samtani [15] and Hajela [16]. Genetic algorithms have since then been applied to numerous structural optimization problem [3,17–20].

A genetic search changes the population of strings by mimicking evolution. The individual strings are mated to create child designs. Each individual has a fitness value that determines its probability of being chosen as parents. Here the fitness is based on rank in terms of objective function in the population. The fitness assigned to the ith best individual of n designs is then equal to $[2(n+1-i)/(n^2+n)]$, so that the sum of all fitnesses is equal to one.

6. Standard GA

For the standard GA, a laminate is coded using the standard stacking sequence notation. To take advantage of the symmetry of the laminate and its balance, only one quarter of the plies is encoded. This is done by adding the requirement that the laminate is composed of pairs of 0° plies, pairs of 90° plies, or a stack of ±45° plies. For example, the laminate $[0_2/\pm 45/90_2/90_2/\pm 45/0_2]$ is encoded as [0/45/90], the latter being the chromosome for the laminate. The rightmost gene corresponds to the stack closest to the laminate mid-plane. The leftmost position in the chromosome describes the outermost stack of two plies. A two-point crossover is used.

Mutation is applied with a small probability by randomly switching a stack orientation (0°, ±45°, 90°) to one of the other two choices available. Since the total numbers of 0°, ±45°, and 90° plies are fixed, the mutation is biased to promote compliance with this constraint. The mutation is biased so that a 0° stack will mutate only if the number of 0° plies is not equal to the allocated amount. This rule also applies to ±45° and 90° stacks. The mutation operator hence uses the problem information and acts as a partial repair operator. Besides the regular mutation there is also an interchange mutation operator called stack-swap, which allows two stacks to exchange their genes with a given probability.

The objective function for maximizing the failure of the composite laminate is equal to the failure load λ penalized for violations of the given number of plies and the limit of no more than four contiguous plies of the same orientation. We denote the number of 0°, ±45°, and 90° plies in the string by n_0, n_{45}, n_{90}, respectively, and denote the specified total number as n_{0g}, n_{45g}, n_{90g}, respectively. Then the objective function is given as

$$\varphi = r^{\mathrm{Penalty}}\phi, \tag{6}$$

where Penalty is a parameter (set to 2.0) for violation of specified amounts of 90°, 45°, and 0° plies, and

$$r = r_0 r_{45} r_{90}, \tag{7}$$

$$r_0 = \begin{cases} (n_0+1)/(n_{0g}+1) & \text{if } n_0 < n_{0g}, \\ 1 & \text{if } n_0 = n_{0g}, \\ (n_{0g}+1)/(n_0+1) & \text{if } n_0 > n_{0g}. \end{cases} \tag{8}$$

with similar definitions for r_{45} and r_{90}.

$$\phi = \frac{\lambda}{P_{\mathrm{cont}}^{N_{\mathrm{cont}}}}. \tag{9}$$

This form of the penalty function and penalty parameters were selected based on previous studies with similar constraints [21,22]).

P_{cont} is penalty parameter (set to 1.05 here) for violation of the four ply limit on contiguous plies of the same orientation, N_{cont} is total number of same-orientation contiguous plies in excess of four same orientation plies. Note that the contiguity constraint is applied only to 0° and 90° plies. The ±45° plies alternate between 45° and −45° directions, and so do not have any contiguity problem, no matter how many stacks are contiguous.

Using a penalty function to incorporate the limits on the number of plies slows down the progress of the optimization, and this is the reason for using permutation based GAs, which do not need these constraints.

Permutation problems seek the optimal arrangement of a list of items, in our case the given 0°, ±45°, and 90° stacks. Natural coding with the orientation angles 0°, ±45°, and 90° is not well suited for representing permutations since it will tend to generate duplicate or missing allele values. A permutation encoding is represented by a list of distinct integer values, such as $1, 2, 3, \ldots$, coding the orderings of 0°, ±45°, and 90° stacks referenced to a baseline laminate. We selected the baseline laminate to have all the specified 90° stacks on the outside, followed by the ±45° plies and then the 0° stacks. So the baseline laminate looks like $[90/90/\cdots/45/45/\cdots/0/0/\cdots]_s$ and it is coded as $[1/2/\cdots/n_0+1/\cdots/n_0+n_{45}+1/\cdots/n_0+n_{45}+n_{90}]$. A baseline laminate $[90_2/\pm45_2/0_2]_s$, for example, is coded into the permutation [1/2/3], while the laminate $[90_2/0_2/\pm45_2]_s$ is coded [1/3/2] by reference to the baseline laminate.

Permutation coding has the advantage, compared to the traditional coding, that the specified amounts of 0°, 45°, and 90° stacks are always met. However traditional crossover and mutation do not work well for permutation coding because they tend to produce infeasible children from feasible parents. Specific permutation crossovers have been developed for the travelling salesman problem (TSP). In this work we employ the partially mapped crossover, developed by Goldberg and Lingle [8]. We also developed a crossover suited for the design of composite laminates that we call a gene–rank crossover.

Mutation for permutation coding is performed by randomly selecting two genes, and then swapping them with a given probability.

7. Gene–rank crossover

In a composite laminate, the outermost plies, hence leftmost genes, affect flexural stiffnesses more than the inner plies. This is in contrast with TSP, where the chromosome may be viewed as a ring, where the absolute position of a gene does not matter. A chromosome for coding a stacking sequence in contrast may be viewed as a directed linear segment.

Gene–rank crossover is based on imitating the process used to average the rankings that two judges give a group of contestants with plies playing the role of contestants. Each laminate can then be viewed as a ranking of the set of plies, and gene–rank crossover averages the two rankings. For example, consider the simple case with three contestants, A, B, and C. The first judge ranked them as: A – 1, B – 2, C – 3, denoted in shorthand as [A, B, C]. The second judge ranked them as: A – 2, B – 3, C – 1, or [C, A, B]. We associate weights W_1 and W_2 with the two judges, representing their relative influence (with $W_1 + W_2 = 1$). In the implementation of the crossover, W_1 is a uniformly distributed random number in [0., 1.] selected anew for each pair of parents for each generation. The final ranking is then obtained by summing the weighted rank of each individual

A:	$(1)\ (W_1) + (2)(W_2)$
B:	$(2)\ (W_1) + (3)(W_2)$
C:	$(3)\ (W_1) + (1)(W_2)$

For example, with $W_1 = 0.4, W_2 = 0.6$, we get [1.6, 2.6, 1.8] for the weighted averages, corresponding to a composite ranking of [A, C, B].

Consider next, for example, the stacking sequence of the baseline laminate $[90_2/90_2/90_2/\pm45/\pm45/0_2]_s$, with its permutation being defined by the chromosome [1/2/3/4/5/6]. If two permutations of the laminate are:

Permutation 1 (Parent 1)	[2/5/4/3/6/1]
Laminate	$[90_2/\pm45/\pm45/90_2/0_2/90_2]_s$
Permutation 2 (Parent 2)	[1/2/4/5/3/6]
Laminate	$[90_2/90_2/\pm45/\pm45/90_2/0_2]_s$

For $W_1 = 0.4634$ and $W_2 = 0.5366$, the average rank of each gene of the child design is shown in the table below. For example, the average rank of gene 1 is equal to $W_1^{*}6 + W_2^{*}1$ since gene 1 is ranked the sixth and the first in the two permutations, respectively (Table 2).

Sorting genes by their average weighted ranks, the permutation of the child is

Permutation (Child)	[2/4/5/1/3/6]
Laminate	$[90_2/\pm45/\pm45/90_2/90_2/0_2]_s$

Table 2
Illustration of weighted averaging for gene–rank algorithm

Gene	Rank-value in permutation 1	Rank-value in permutation 2	Weighted rank-value
1	6	1	3.317
2	1	2	1.53
3	4	5	4.54
4	3	3	3.0
5	2	4	3.07
6	5	6	5.54

Besides the uniformly distributed random weight, W_1, we also experimented with a random variable biased to be close to one or zero, so that one of the parent laminates dominates. However, we did not find a distinct advantage to that variant. The gene–rank GA has some similarities with Bean's random keys algorithm [9]. The random keys algorithm uses a chromosome with numbers in [0, 1.], with their order determining the permutation. For example, the chromosome [.46/.91/.33/.75/0.51] corresponds to the permutation [3/1/5/4/2]. The advantage of this form of coding is that standard crossover and mutation can be used. This coding tends to preserve rank more than the partially mapped crossover discussed next, but it is not as conscious of rank as the Gene Rank algorithm. For example, consider two parents that are both identical with the baseline laminate, so that in permutation coding they will both be coded as [1/2/3/4]. Any gene–rank crossover will produce a child design identical to the parents. On the other hand, with the random keys algorithm, one parent may be coded as [.1/.2/.3/.4], and the other parent may be coded as [.5/.6/.7./.8]. Some of the child designs obtained by crossover are very different. For example, with a cut in the middle of the chromosome, one child design is [.5/.6./.3/.4], which corresponds to a permutation of [3/4/1/2].

8. Partially mapped crossover

The partially mapped crossover, developed by Goldberg and Lingle [8] for the TSP, employs the following four steps:

- define two break points randomly,
- use the middle sub-string between the two cut points from the second parent,
- take genes of the two outer sub-strings from the first parent when they do not conflict with the genes taken from the second parent,
- define the map relationship of genes in conflict, and fill genes in conflict by a map-relationship.

The mechanism of the crossover is illustrated through an example of a laminate with a nominal stacking sequence of 8 stacks corresponding to 32 plies. The stacking sequence of the baseline laminate is $[90_2/90_2/\pm45/\pm45/\pm45/\pm45/0_2/0_2]_s$, its gene code defined as [1/2/3/4/5/6/7/8]. Two permutations of the laminate are listed below:

Permutation 1 (Parent 1)	[3/6/4/2/7/5/8/1]
Laminate	$[\pm45/\pm45/\pm45/\pm45/90_2/0_2/0_2/90_2]_s$
Permutation2 (Parent 2)	[3/7/5/1/6/8/2/4]
Laminate	$[\pm45/0_2/\pm45/90_2/\pm45/0_2/90_2/\pm45]_s$

The random cut-points are 2 and 5, so the segment between two-cut points of the child design is

Child permutation [*/7/5/1/6/*/*/*]

where the asterisk denotes presently unknown. Then we fill positions of the genes, which are not in conflict with these genes,

Child permutation [3/7/5/2/1/6/8/*/*]

Two genes from Parent 1 in position 6 and 8 of the permutation conflict with genes in the middle substring, which come from Parent 2. The conflicting gene in position 6 is 5, and the corresponding gene in Parent 2 was in same position as gene 4 from Parent 1. (Since the gene will not conflict with any genes from same parent, we need to go back to Parent 2 to find the corresponding gene of Parent 2). We check whether the mapped gene 4 conflicts with genes previously filled in the child. We find that it does not conflict with any. So the conflicting gene 5 from parent 1 in position 6 in the child's permutation is 4. Similarly, we find that the conflicting gene 1 from parent 1 in position 8 of the child's permutation mapped gene 2 from parent 2. We fill genes 4 and 2 into position's 5 and 8 of the child's permutation to obtain

Child permutation	[3/7/5/1/6/4/8/2]
Laminate	$[\pm 45/0_2/\pm 45/90_2/\pm 45/\pm 45/0_2/90_2]_s$

9. Comparison of efficiency of three GAs

The efficiency of the three GAs is discussed here in terms of the computational cost – the average of number of analyses required for obtaining a given level of reliability in finding the global optimum. The reliability is calculated here by performing 100 optimization runs each for 4000 analyses and checking how many of the 100 runs reached the optimum at any given point. For example, if 63 runs reached the global optimum after 500 analyses, then the reliability of the algorithm is estimated to be 0.63 after 500 analyses. Of course, this is only an estimate, but it is easy to check that the standard deviation of a value r of the reliability estimated from n runs is

$$\sigma_r = \sqrt{\frac{r(1-r)}{n}}. \tag{10}$$

So that for 100 runs and $r = 0.63$, we obtain a standard deviation of about 0.048.

Because reaching the global optimum is often very time consuming, the requirement is often relaxed, and replaced by a *practical global optimum*, which is defined to be within a specified fraction of the optimum. In the present work, a design was considered to be a practical optimum if the failure load was within 0.5% of the global optimum.

Normally, the loads and number of plies used in the panel optimization come from the overall wing design. Here, in order to generate test cases, we selected some representative load cases, and used continuous optimization to find reasonable required number of plies. For the continuous optimization we used nine ply thicknesses as continuous design variables t_i, $i = 1, \ldots, 9$ and sequential quadratic programming as implemented in the DOT program [23]. The stacking sequence was set as $[90^\circ_{t9}/\pm 45^\circ_{t8}/0^\circ_{t7}/90^\circ_{t6}/\pm 45^\circ_{t5}/0^\circ_{t4}/90^\circ_{t3}/\pm 45^\circ_{t2}/0^\circ_{t1}]_s$. The results are given in Table 3 in terms of number of plies of a given orientation (for ply thickness of 0.005 in.) rather than the detailed stacking sequence. Next, the number of plies was rounded into integers, and the rounded numbers used as the specified set for the genetic algorithms.

Table 3
Optimum number of stacks of the three orientations for five load cases using sequential quadratic programming[a]

Loading (lb/in.)					Non-rounded optimization results				
Case	N_x	N_y	N_{xy}	n_0	n_{45}	n_{90}	n_{total}	λ_s	λ_c
1	−20000	−2000	1000	9.18	18.32	9.18	36.65	26.56	1.0
2	−15000	−2000	1000	8.41	16.82	8.41	33.63	21.24	1.0
3	−10000	−2000	1000	7.49	14.98	7.49	30.00	15.09	1.0
4	−5000	−2000	1000	6.28	12.52	6.28	25.10	8.87	1.0
5	0	−2000	1000	4.31	8.62	4.31	17.24	2.87	1.0

[a] λ_s and λ_c are the shear buckling and combined buckling load factors, respectively.

Table 4
Comparison of computational efficiency of the three GAs

Loading case	Given number of stacks (rounded from Table 2)			Failure load	Number of analyses required for 80% reliability		
	n_{0g}	n_{45g}	n_{90g}	λ	SGA	GR	PMX
1	9	18	9	0.948	10432	1184	1328
2	8	17	8	0.948	8600	856	1224
3	7	15	7	0.909	5216	776	1024
4	6	12	6	0.870	3304	608	824
5	4	8	4	0.778	1672	408	560

SGA: Standard GA, PMX: partially mapped crossover, GR: gene–rank crossover, refer to Appendix A for selection of the genetic parameters.

The results for the three algorithms were obtained with a population size of 8 and with the probabilities of mutation and crossover set to one. Appendix A discusses choice of three parameters. For the mutation operation, one gene is changed to one of two other alleles available in each child design for the standard GA, and for the permutation GAs, two genes are swapped for each child design. The number of multiple-runs is 100, and the number of generations is 500. Table 4 gives results for the three GAs in terms of number of analyses required for 80% reliability.

From Table 4 we see that for the first three load cases, reliability of the standard GA did not reach 80% for 4000 analyses. The reliability is shown versus number of analyses in Fig. 2. From Fig. 2, it is clear that the two permutation GAs perform much better than the standard GA. The gene–rank crossover generally has the highest reliability except occasionally for low number of generations.

From Table 4, we can see that, as expected, thicker laminates are computationally more expensive to optimize. All the laminates in Table 4 are quasi-isotropic or close to quasi-isotropic. The small number of 0° and 90° plies in such laminates makes contiguity constraint easy to satisfy. To explore the performance for more general and thicker laminates, three new cases, defined in Table 5, were selected.

Table 5
Three thick test laminates

Case	N_x (lb/in.)	N_y (lb/in.)	N_{xy} (lb/in.)	n_0	n_{45}	n_{90}	n_{total}
6	0	−16000	8000	8	16	8	32
7	15980	−14764	10160	9	8	13	30
8	−16657	1963	828	13	7	15	35

Table 6
Comparison of computational efficiency of the three GAs for the three thick laminates

Case no.	Number of analyses required for 80% reliability					
	SGA		GR		PMX	
	With contiguity constraint	Without contiguity constraint	With contiguity constraint	Without contiguity constraint	With contiguity constraint	Without contiguity constraint
6	7984	5112	1328	480	1480	848
7	23544	2176	11840	360	5784	336
8	26320	5024	2216	296	2504	840

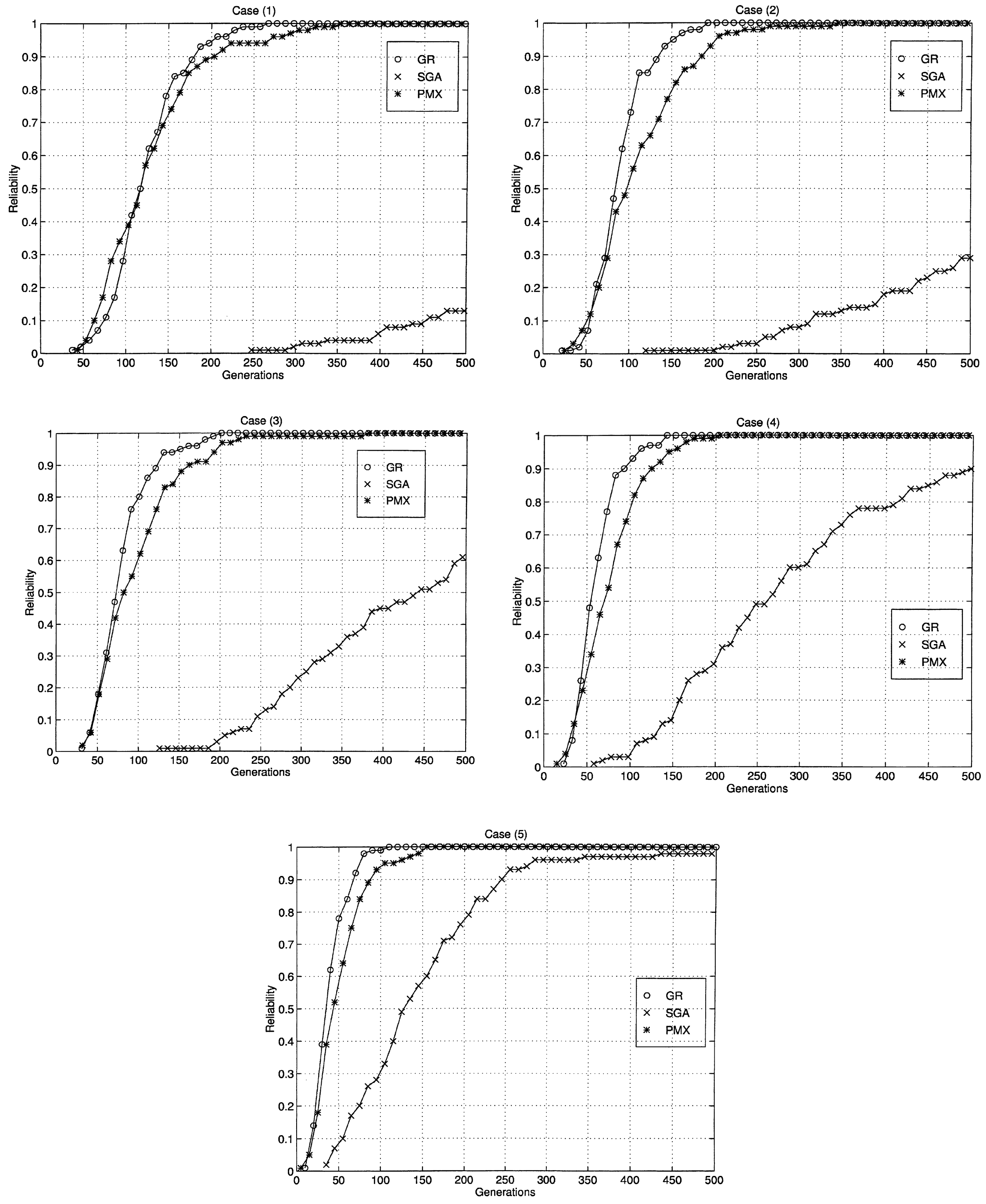

Fig. 2. Reliability versus number of generations for five loading cases.

The results summarized in Table 6 show that for the three thicker laminates, the contiguity constraint dominates the search for the optimum. Comparing the three thick laminates above with contiguity constraints and without contiguity constraint, we can easily see that case 7 has the most difficult constraints. This is explained by examining the optimum laminates shown in Table 7. For case 6 and case 8 the outermost plies in the optimum design are $\pm 45°$, so that the contiguity constraint affects only the less important inner plies, while for case 7 it affects the critical outer plies. Fig. 3 shows the reliability versus number of

Table 7
Optimum lay-up for the three thick laminates

Case 6	Without contiguity contiguity	$[(\pm45)_{16}/(90_2)_8/(0_2)_8]_s$
	With contiguity	$[(\pm45)_{16}/(90_2)_2/0_2/90_2/0_2/(90_2)_2/(0_2)_2/(90_2)/(0_2)_2/(90_2)/(0_2)_2/(90_2)]_s$
Case 7	Without contiguity contiguity	$[(90_2)_8/(\pm45)/90_2/(\pm45)_2/(90_2)/(\pm45)_2/(90_2)_2/(\pm45)/90_2/(\pm45)/(0_2)_9/(\pm45)]_s$
	With contiguity	$[(90_2)/(\pm45)/(90_2)_2/(\pm45)/(90_2)/(90_2/\pm45)_3/(90_2)_2/(\pm45)/(0_2)/(90_2)_2/(0_2)_2/(90_2)$ $/(0_2)_2/(\pm45)/(0_2)_2/(\pm45)/(0_2)/(90_2)/(0_2)]_s$
Case 8	Without contiguity contiguity	$[(\pm45)_7/(90_2)_{15}/(0_2)_{13}]_s$
	With contiguity	$[(\pm45)_4/(90_2/\pm45)_2/(90_2)_2/(\pm45)/(90_2)/(0_2/90_2)_4/(0_2)/(90_2/0_2)_2/(0_2)/(90_2)/(0_2)_2/(90_2/0_2)_3]_s$

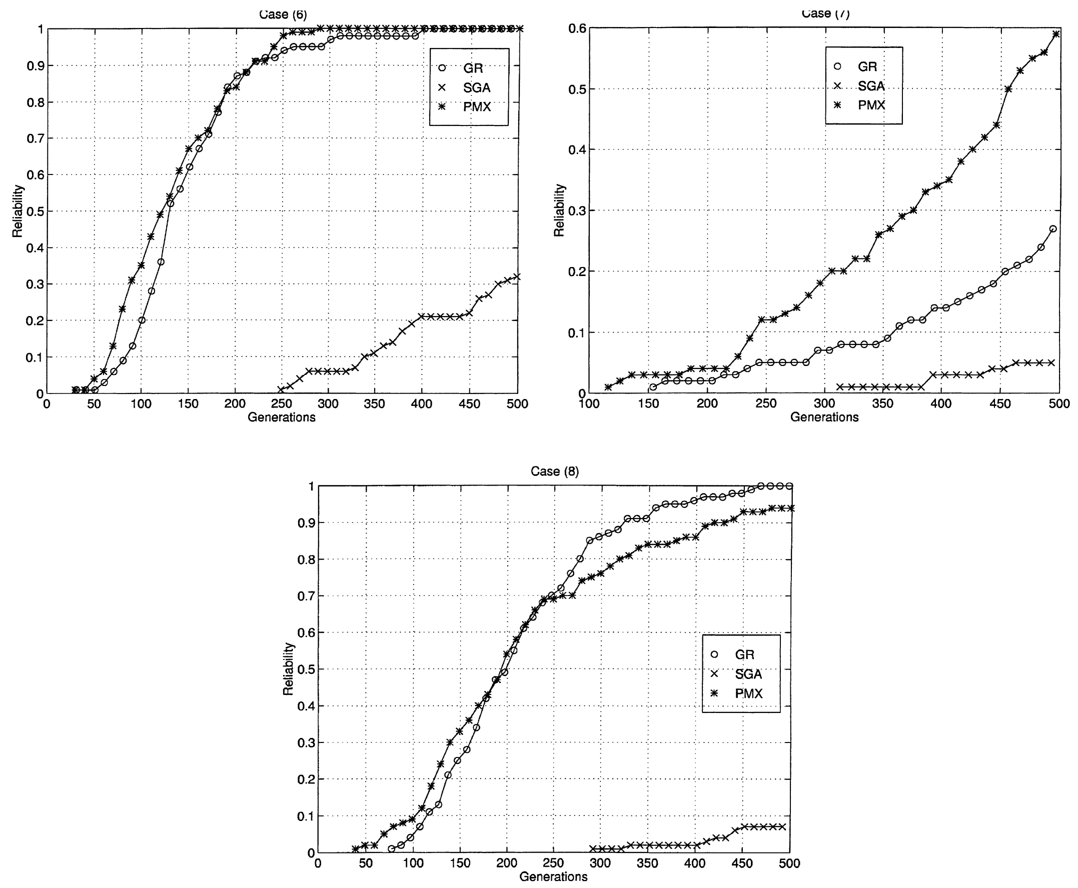

Fig. 3. Reliability versus number of generations for the three thick laminates.

generations of the three thick laminates. We also inspected the various solutions and found that for the cases we optimized here, the optimum design was unique, so that the number of analyses needed for 80% reliability is a good indicator of the efficiency of the algorithm.

10. Baldwinian repair for number of plies and contiguity constraint

The previous results demonstrate the high cost of dealing with constraints via penalty function. An alternate approach is to repair laminates that violate constraints. Todoroki and Haftka [24] introduced a Baldwinian repair strategy, which they called recessive repair, for dealing with contiguity constraints for standard GA. Here the strategy is extended to the permutation GAs. Additionally, a similar repair approach is used for enforcing the required number of plies of given orientations for the standard GA.

The key concept of Baldwinian repair is to repair the stacking sequence without repairing the chromosome. Repairing the chromosome is known as Lamarckian repair. The advantages of Baldwinian repair have been noted before, for example, by Hinton and Nowlan [25]. There may be also an advantage to repairing a small percentage of the chromosomes [26].

The process is explained first for enforcing the required number of plies of given orientation for the standard GA. The decoding of a chromosome proceeds from the outermost plies to the innermost ones, one two-ply stack at a time. As long as the number of decoded stacks of any given orientation does not exceed the prescribed number of stacks, the decoding proceeds normally. However, once the number of stacks of any given orientation reaches the prescribed number, subsequent genes that indicate that orientation will be translated to the next available orientation (in a circular 0/45/90 order). For example consider a laminate with $n_0 = 2, n_{45} = 0,$ and $n_{90} = 1$. When a chromosome [0/90/90] is decoded, the first two genes are decoded normally, but when the third gene is encountered, it cannot be decoded into a 90-stack because the number of decoded 90-genes already reached the target of $n_{90} = 1$, so it is decoded as a zero ply. Similarly, when a chromosome [0/0/0] is decoded, the first two genes are decoded normally. The third gene cannot be decoded into a 0-stack, because the number of required 0-stacks is two. The decoding procedure then tries to see if there are available stacks for a 45-stack, and when it finds that none are available, it puts a 90-stack in the innermost position. It should be noted that the circular order chosen for the orientation used in repair may introduce some bias, and a random selection of the orientation may be a good alternative.

The repair of the stacking sequence without changing the chromosome allows a sequence of mutations needed to achieve a good design to complete successfully even if the intermediate steps are infeasible designs. For example, consider the evolution of a design defined by [0/0/90] chromosome when the optimum is defined by [0/90/0] chromosome (that is stacking sequences of $[0_4/90_2]_s$ to $[0_2/90_2/0_2]_s$, respectively). Without repair we have to depend on hitting the single permutation that will exchange the second and third genes. With the repair strategy described above, we can also go through the intermediate step of [0/90/90], which is decoded into $[0_2/90_2/0_2]_s$, or through the intermediate step of [0/0/0], which is decoded into $[0_4/90_2]_s$. Then another mutation can transform either intermediate step into the optimum. The last gene in both alternatives acts like a recessive gene, in that it is unexpressed due to the decoding scheme, but it will become expressed following the mutation of another gene.

The repair of violations of contiguity constraints follows the similar approach of repairing only the laminate, and of trying to apply the repair to the innermost plies, which have the least effect on the buckling load. Details may be found in [24].

For the permutation GA, the constraints of number of plies are incorporated into gene coding, and only contiguity constraints may be violated. To repair contiguity violations, it is desirable to interchange the closest couple of genes with different orientation angles since this minimizes the change in bending properties. The following example illustrates the repair operator.

For the laminate

$$[0_2/\mathbf{0_2}/\mathbf{90_2}/90_2/\mathbf{90_2}/\pm 45]_s.$$

Three contiguous 90 stacks violate the contiguity constraint. Two candidate couples of stacks can be swapped: the rightmost 90° with the ±45°, or the leftmost 90° with its neighbouring 0° stack. The first option is selected because the inner plies influence laminate stiffness less than the outer plies.

$$[0_2/0_2/90_2/90_2/\pm\mathbf{45}/\mathbf{90_2}]_s.$$

In order to demonstrate the advantage of recessive repair, it is compared to direct repair of the chromosome in Table 8.

Table 8
Computational cost of laminate repair and chromosome repair

Case no.	Number of analyses required for 80% reliability					
	GR		PMX		SGA	
	Chromosome repair	Laminate repair	Chromosome repair	Laminate repair	Chromosome repair	Laminate repair
1	1048	456	944	792	368	672
2	952	400	808	792	400	536
3	832	352	784	658	384	368
4	680	304	560	496	280	224
5	304	184	272	272	128	80
6	744	416	688	552	416	400
7	480	288	416	336	3936	3512
8	728	352	744	696	56	48

From Table 8, we can see that the Baldwinian repair (laminate only) is more efficient than repairing the chromosome (Lamarckian repair). The advantage is most pronounced for the repair strategy helps the standard GA achieve similar efficiencies to that of the permutation GAs, except for the most difficult case (7). Comparing Table 8 to Table 6, we see that the combined use of permutation and repair is to reduce the cost of the standard GA by one to two orders of magnitude.

11. Summary and concluding remarks

In this paper, maximization of the buckling load of composite laminates via stacking sequence optimization for a given number of 0°, ±45°, and 90° plies and for a given in-plane loading was investigated using genetic algorithms. A new permutation GA, which we called a gene–rank crossover GA, was developed and implemented along with two other GAs, a standard GA and a permutation GA based on partially mapped crossover. Computational efficiency of these GAs were compared under eight load cases in terms of the number of analyses required to reach a certain reliability. Effect of a contiguity constraint on performance, which limits the number of identical adjacent ply orientations to four, was investigated and two repair strategies for dealing with violation of this constraint were implemented.

Stacking sequence design for given number of plies is a combinatorial problem consisting of seeking an optimal permutation. It has been demonstrated that the two genetic algorithms based on permutation are much more efficient and more reliable for solving this problem than standard genetic algorithms. Furthermore, a genetic algorithm developed for stacking sequence design showed an advantage over an algorithm developed originally for the travelling salesman problem. Repair developed for overcoming violation of constraints can significantly reduce the computational cost for both the standard GA and the permutation GAs, and with repair the difference between the standard GA and permutation GA are smaller.

In another study by the authors, the permutation GA and its corresponding chromosome-repair technique were employed in a large number of stacking sequence optimization runs for a range of loads and number of plies. Based on these optima, a cubic polynomial response surface was fitted as a function of in-plane loads and number of 0°, ±45°, and 90° plies. The response surface was then used in a wing box optimization [27]. The study demonstrated the viability of permutation GAs as part of large scale optimization.

The permutation GAs and the repair strategy developed can be easily tailored for application to more complicated structures with more constraints by coding these constraints into gene coding or through repair.

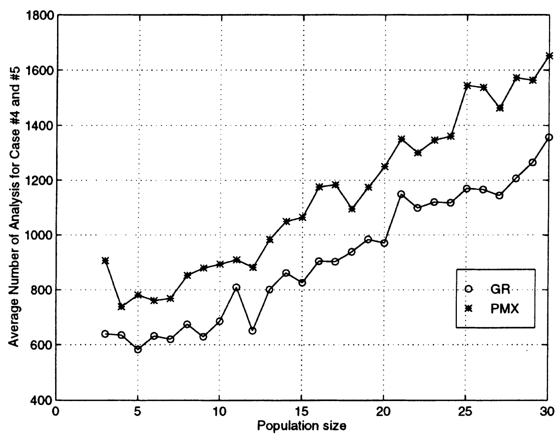

Fig. 4. Effect of poulation size.

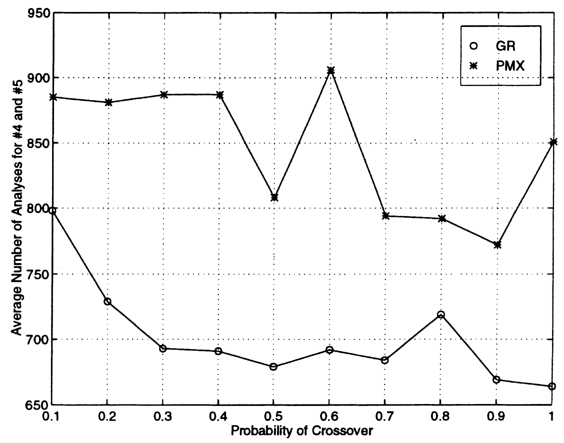

Fig. 5. Reliability versus number of generations for the three thick laminates.

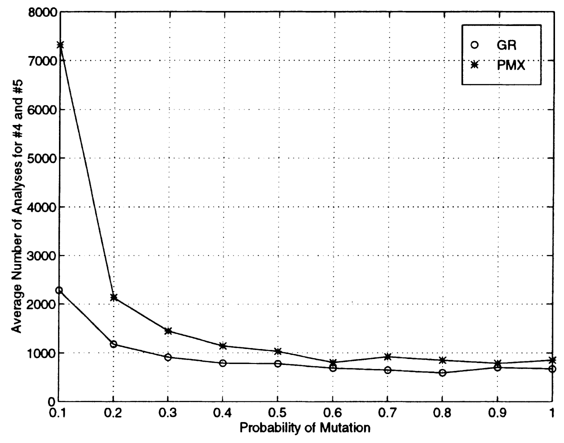

Fig. 6. Reliability versus number of generations for the three thick laminates.

Acknowledgements

This research was supported by Air Force Office of Scientific Research Grant F49620-96-1-0099. The third author acknowledges the fellowship granted by Turkish Scientific and Technical Research Council and the leave of absence granted by the Middle East Technical University.

Appendix A. Tuning genetic parameters

The efficiency of genetic algorithm is often sensitive to parameters, such as population size, probabilities of crossover, mutation, etc. We used numerical experiments to tune these parameters. The evaluation measure is number of analyses required for obtaining optimum averaged over case 4 and case 5. Typically, the coefficients of variation (standard deviation over mean value) were below 5%. Here we only tune three genetic parameters: population size, and probability of crossover, probability of mutation.

Fig. 4 shows the effect of population size. Fig. 5 displays effect of probability of crossover. Fig. 6 shows effect of probability of mutation.

From three figures, we can observe that the optimal population size should be below 10 for GR GA and PMX GA, and that the optimal probabilities of crossover and mutation are around 1.0.

References

[1] R. Le Riche, R.T. Haftka, Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm, AIAA Journal 31(5) (1993) 951–956.

[2] M. Miki, Optimum design of fibrous laminated composite plates subject to axial compression, in: Proceedings of the Third Japan–US Composite Materials Conference, Tokyo, Technomic Publishing Company, Lancaster, 1979, pp. 1017–1019.

[3] S. Nagendra, R.T. Haftka, Z. Gurdal, Design of blade stiffened Composite Panel by a Genetic Algorithm. AIAA-93-1584-CP, in: Proceedings of the 34th Structures, Structural Dynamics and Material Conference, La Jolla, California, 19–21 April, 1993, pp. 2481–2436.

[4] L.A. Schmit, B. Farshi, Optimum design of laminate fiber composite plates, International Journal for Numerical Methods in Engineering 11 (1977) 623–640.

[5] R. Le Riche, R.T. Haftka, Improved genetic algorithm for minimum thickness composite laminate design, Composite Engineering 5 (2) (1995) 143–161.

[6] Y.S. Shin, R.T. Haftka, L.T. Watson, R.H. Plaut, Design of laminated plates for maximum buckling load, Journal of Composite Materials 23 (1989) 348–370.

[7] Z. Michalewicz, Genetic Algorithms + Data Structure = Evolution Programs, Springer, Berlin, 1992.

[8] D.E. Goldberg, R. Lingle Jr., Alleles, loci, and travelling salesman problem, in: J.J. Grefenstette (Ed.), Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985, pp. 154–159.

[9] J.C. Bean, Genetic algorithms and random keys for sequencing and optimization, ORSA Journal on Computing 6(2) (1994) 154–160.

[10] J.M. Whitney, Structural Analysis of Laminated Anisotropic Plates, Technomic Publishing Company, Lancaster, 1985, pp. 119–122.

[11] S.G. Lekhnitskii, Anisotropic Plates, Gordon and Breach, London, 1968 (Trans. S.W. Tsai, T. Cheron).

[12] H.J. Bremermann, Optimization through evolution and recombination, in: M.C. Yovits, G. Tl Jacobi, G.D. Goldman (Eds.), Self-organizing Systems, Spartan Books, Washington, DC, 1962, pp. 93–106.

[13] I. Rechenberg, Cybernetic Solution Path of an Experimental Problem, Royal Aircraft Establishment, Library Translation 1122, Farnborough, England, UK, 1965.

[14] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1995.

[15] D.E. Goldberg, T.A. Samtani, Engineering optimization by a genetic algorithm, in: Proceedings of the Ninth Conference on Electronic Computation, ASCE, 1987, pp. 471–482.

[16] P. Hajela, Genetic search an approach to the nonconvex optimization problem, AIAA Journal 26 (7) (1990) 1205–1210.

[17] H. Furuya, R.T. Haftka, Genetic algorithms for placing actuators on space structures, in: Proceedings of the Fifth International Conference on Genetic Algorithms, University of Illinois at Uabana-Champaign, 17–21 July, Morgan Kaufmann, Fairfax, 1993, pp. 536–542.

[18] D. Powell, M.M. Skolnick, Using genetic algorithms in engineering design optimization with nonlinear constraints, in: Proceedings of the Fifth International Conference on Genetic Algorithms, University of Illinois of Urbana-Champaign, Morgan Kaufmann, Fairfax, 17–21 July, 1993, pp. 424–431.

[19] M. Schoenauer, S. Xanthakis, Constrained GA optimization, in: Proceedings of the Fifth International Conference on Genetic Algorithms, University of Illinois at Urbana-Champaign, Morgan Kaufmann, Fairfax, 1993, 17–21 July, pp. 573–580.

[20] H. Watabe, N. Okino, A study on genetic shape design, in: Proceedings of the Fifth International Conference Genetic Algorithms, University of Illumes at Urbana-Champaign, Morgan Raufmann, Fairfax, 17–21 July, 1993, pp. 445–450.

[21] N. Kogiso, L.T. Watson, Z. Gürdal, R.T. Haftka, S. Nagendra, Design of composite laminates by a genetic algorithm with memory, Mechanics of Composite Materials and Structures 1 (1) (1994) 95–117.
[22] R. Le Riche, R.T. Haftka, Improved genetic algorithm for minimum thickness composite laminate design, Composite Engineering 5 (2) (1995) 143–161.
[23] DOT Users Manual, Version 4.20, Vanderplaats Research and Development, 1995.
[24] A. Todoroki, R.T. Haftka, Stacking sequence optimization by a genetic algorithm with a new recessive gene like repair strategy, Composite Part B 29 (3) (1998) 277–285.
[25] G.E. Hinton, S.J. Nowlan, How learning can guide evolution, Complex Systems 1 (3) (1987) 495–502.
[26] D. Orvosh, L. Davis, Using a genetic algorithm to optimize problems with feasibility constraints, in: Proceedings of the First IEEE Conference On Evolutionary Computation, vol. 2, Orlando, Florida, 27–29 June, 1994, pp. 548–553.
[27] B. Liu, R.T. Haftka, M.A. Akgün, Composite wing structural optimization using genetic algorithms and response surfaces, AIAA Paper 98-4854, Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, 2–4 September, 1998.