AIAA JOURNAL
Vol. 31, No. 5, May 1993

Optimization of Laminate Stacking Sequence for Buckling Load Maximization by Genetic Algorithm

Rodolphe Le Riche* and Raphael T. Haftka†
Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061

The use of a genetic algorithm to optimize the stacking sequence of a composite laminate for buckling load maximization is studied. Various genetic parameters including the population size, the probability of mutation, and the probability of crossover are optimized by numerical experiments. A new genetic operator—permutation—is proposed and shown to be effective in reducing the cost of the genetic search. Results are obtained for a graphite-epoxy plate, first when only the buckling load is considered, and then when constraints on ply contiguity and strain failure are added. The influence on the genetic search of the penalty parameter enforcing the contiguity constraint is studied. The advantage of the genetic algorithm in producing several near-optimal designs is discussed.

Introduction

THE design of composite laminates is often formulated as a continuous optimization problem with ply thicknesses and ply orientations used as design variables.[1] However, for many practical problems ply thicknesses are fixed and ply orientations are limited to a small set of angles such as 0, 90, and ±45 deg. Designing the laminate then becomes a stacking-sequence optimization problem that can be formulated as an integer programming problem.

The laminate stacking-sequence design problem for buckling optimization has been formulated first with the number of plies as the design variables, which leads to a nonlinear integer programming problem.[2] The use of ply identity design variables permits a linear integer programming formulation.[3] However, when strength constraints are also considered, the problem becomes nonlinear again and has been solved as a sequence of linearized integer programming subproblems.[4] The branch-and-bound algorithm was used to solve the integer programming problem in Refs. 2–4.

In the present work, the use of a genetic algorithm was explored. Early implementation of the genetic search method is credited to Rechenberg,[5] although Holland's work[6] has formed the basis of most contemporary developments. Genetic algorithms are probabilistic optimization methods that work on a population of designs by recombining the most desirable features of existing designs. Following the concept of Darwin's theory of evolution, a selection is performed that favors the best-fitted members of the population and assures that they transmit their attractive features to the newly created designs. In the process, new design options are created and tested. Genetic algorithms do not use any gradient-based information and are insensitive to the complexity of the design space. In the last decade, genetic algorithms have proven their ability to deal with a large class of problems. In structural optimization, the genetic approach appears especially promising in cases of large, nonconvex, integer programming problems.[7–9]

Received Jan. 30, 1992; presented as Paper 92-2314 at the AIAA/ASME/ASCE/AHS/ASC 33rd Structures, Structural Dynamics, and Materials Conference, Dallas, TX, April 13–15, 1992; revision received Aug. 19, 1992; accepted for publication Sept. 16, 1992. Copyright © 1992 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

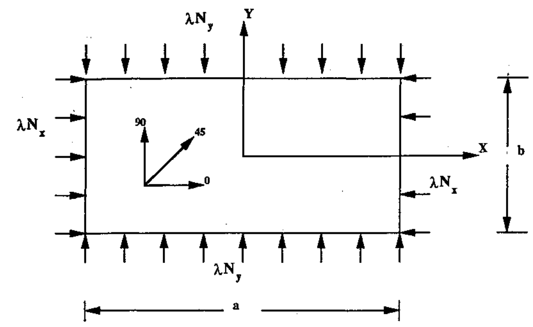
*Graduate Research Assistant, Department of Aerospace and Ocean Engineering.

†Christopher Kraft Professor of Aerospace and Ocean Engineering. Associate Fellow AIAA.

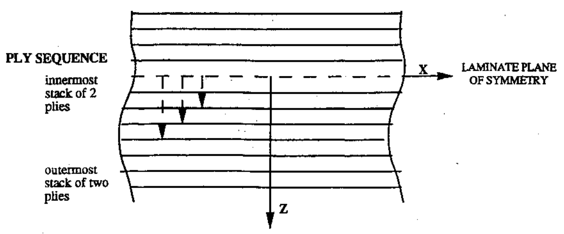
The objective of this work is to devise a genetic algorithm for composite laminate stacking-sequence design. The genetic parameters determine how much effort is devoted to the exploration of the design space vs how much effort is used to exploit previous solutions. Random exploration of the design space is a robust search method, but it is prohibitively expensive. On the other hand, a genetic algorithm that overstresses the exploitation aspect of the search would converge prematurely to a nonoptimal solution. In the first part of the work, we describe tuning the genetic algorithm for optimum values of various parameters. We also introduce a genetic operator specific to this problem, called permutation, and show its effectiveness. We then present results and discuss some peculiarities of the stacking-sequence design problem.

Analysis and Optimization

The simply supported plate, shown in Fig. 1, is loaded in the x and y directions by λN_x and λN_y, respectively, with λ being an amplitude parameter. The laminate is composed of N plies. Its longitudinal and lateral dimensions are a and b, respectively. It is further assumed to be symmetric, balanced, and made of 0, 90, and ±45 deg plies, each of thickness t. To



a) **Laminate plate geometry and applied loading**


PLY SEQUENCE
innermost stack of 2 plies
outermost stack of two plies
X
LAMINATE PLANE OF SYMMETRY
Z

b) **Ply sequence location**

Fig. 1 Laminated plate geometry and loading.

reduce the number of design variables and at the same time take into account the balanced condition, the laminate is considered to be made up of stacks of two plies of the same orientation. Within a 45-deg stack, a + 45-deg ply is always associated with a − 45-deg ply, so that the balanced condition is guaranteed. As a result, only $N/4$ ply orientations are required to describe the entire laminate.

The laminate buckles into m and n half-waves in the x and y directions, respectively, when the load amplitude reaches a value λ_b, which is given in terms of flexural stiffnesses D_{ij} and loads N_x and N_y as

$$\frac{\lambda_b(m,\, n)}{\pi^2} = \frac{[D_{11}(m/a)^4 + 2(D_{12} + 2D_{66})(m/a)^2 + D_{22}(n/b)^4]}{(m/a)^2N_x + (n/b)^2N_y} \tag{1}$$

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

The optimization problem is to maximize the critical buckling load by changing the ply orientations. We also apply strain constraints, and we limit the number of contiguous plies of the same orientation to four to alleviate matrix cracking problems. The strain failure constraint requires all strains to remain below their allowable limits. In our case γ_{xy} is zero, and the principal strains in the ith layer are related to the loads on the plate by the relations

$$\begin{aligned} \lambda N_x &= A_{11}\epsilon_x + A_{12}\epsilon_y \\ \lambda N_y &= A_{12}\epsilon_x + A_{22}\epsilon_y \end{aligned} \tag{2}$$

and

$$\begin{aligned} \epsilon_1^i &= \cos^2\theta_i\epsilon_x + \sin^2\theta_i\epsilon_y \\ \epsilon_2^i &= \sin^2\theta_i\epsilon_x + \cos^2\theta_i\epsilon_y \\ \gamma_{12}^i &= \sin^2\theta_i(\epsilon_y - \epsilon_x) \end{aligned} \tag{3}$$

where the various A_{ij} are the coefficients of the extensional stiffness matrix. The ultimate allowable strains are $\epsilon_1^{ua} = 0.008$, $\epsilon_2^{ua} = 0.029$, and $\gamma_{12}^{ua} = 0.015$. A safety factor equal to 1.5 was used to calculate the strain allowables. The strain failure load λ_{cs} is the smallest load factor λ such that one of the principal strains in one of the layers is above its allowable value.

The constraint limiting the number of contiguous plies of the same orientation to four is implemented by a penalty parameter p. Its value defines the percentage of reduction of the objective function for violation of the contiguity constraint. The objective function λ^* is given as

$$\lambda^* = (1-p)\min(\lambda_{cs},\, \lambda_{cb}) \tag{4}$$

Results were obtained for a graphite-epoxy plate [$E_1 = 18.50E6$ psi (127.59 GPa), $E_2 = 1.89E6$ psi (13.03 GPa), $G_{12} = 0.93E6$ psi (6.41 GPa), $t = 0.005$ in. (0.0127 cm), and $\nu_{12} = 0.3$].

Genetic Algorithm

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. In our case, each string represents an alternative stacking sequence, where 1, 2, 3 stands for the three possible stacks 0_2, ± 45, 90_2 deg, respectively. For example, the laminate $(90_2,\ \pm 45_2,\ 90_2,\ 0_2,\ \pm 45_2,\ 0_2)_s$ is encoded as 3 2 2 3 1 2 2 1. The rightmost 1 corresponds to the layer closest to the laminate plane of symmetry. The leftmost 3 describes the outermost layer. A genetic search method changes the population of strings by mimicking genetic evolution. The individual strings are mated to create children. Each individual has a fitness value that determines its probability of being chosen as a future parent. Here, the fitness was chosen (after some experimentation) to be the relative rank in terms of objective function in the population. The fitness assigned to the ith best individual of m designs is then equal to $[2(m + 1 - i)]/(m^2 + m)$, so that the sum of all fitnesses is equal to one.

The genetic algorithm begins with the random generation of a population of design alternatives. It is processed then, by means of genetic operators, to the creation of a new population, which combines the most desirable characteristics of the old population, and then the old population is replaced by the new one. Herein the best design is always kept unchanged, which is an "elitist plan" version of the genetic algorithm.[10] The process is repeated until a stopping criterion, implemented in the form of a maximum number of generations without improvement in the best design, is satisfied.

The operators applied successively to the parent generation to create a new generation are selection, crossover, mutation, and permutation, each with a given probability of being used. The selection process is biased so that high-performance designs have a high probability of transmitting their features to the next generations. This is implemented by allocating each individual a portion of the segment [0, 1] equal to its fitness. A random number is then created between 0 and 1 that designates the selected individual.

Crossover allows selected individuals to trade characteristics of their designs by exchanging parts of strings. We used a two-point crossover, where two break points in the string are chosen randomly. Two offsprings are created by swapping the parents' substrings. Crossover is applied with a given probability, usually between 0.7 and 1. If crossover is not applied, the parents are copied into the next generation. The following is an example where a $(0_2,\ \pm 45_2,\ 90_2,\ 0_4,\ 90_2,\ \pm 45)_s$ laminate is mated with a $(\pm 45_2,\ 0_2,\ 90_4,\ 0_2,\ 90_4)_s$ laminate to produce $(0_2,\ \pm 45,\ 0_2,\ 90_4,\ 0_2,\ 90_2,\ \pm 45)_s$ and $(\pm 45_3,\ 90_2,\ 0_4,\ 90_4)_s$ laminates:

Parent 1	1 2 / 2 3 1 1 / 3 2	
Parent 2	2 2 / 1 3 3 1 / 3 3	(5)
Child 1	1 2 / 1 3 3 1 / 3 2	
Child 2	2 2 / 2 3 1 1 / 3 3	

The children thus combine genetic information carried by the parents and previously tested by the selection. Selection and crossover are the two pivots of the genetic search.

Next, one of the two children is randomly picked and endures mutation. Mutation is a stochastic operator, applied with a low probability. It protects from a complete loss of genetic material by changing at random a bit in a string. A first random number decides whether or not mutation will be applied to a string. A second random number selects the bit to be mutated. Since a three-element alphabet was used, a new value of the bit is chosen, with equal probability, out of the two remaining new values:

Before mutation	2 2 2 3 1 $\underline{1}$ 3 3	(6)
After mutation	2 2 2 3 1 2 3 3	

Permutation, a new operator devised for laminate design, is also a stochastic operator. It has been applied with a high probability because it has some advantages over mutation that we will present with more details later. The operator permutates the order of the bits between two randomly assigned points. To contrast permutation with the more commonly used inversion operator, a second string that specifies the position of the plies is used here. Permutation inverts the order of the bits between two randomly assigned points, whereas the meaning of each position, coded on the position string, remains unchanged:

$$\begin{array}{ll} \text{Before permutation} & 2\,/\,3\,1\,2\,2\,2\,/\,1\,1 \\ \text{Position string} & 1\,/\,2\,3\,4\,5\,6\,/\,7\,8 \\ \text{After permutation} & 2\,/\,2\,2\,2\,1\,3\,/\,1\,1 \\ \text{Position string} & 1\,/\,2\,3\,4\,5\,6\,/\,7\,8 \end{array} \tag{7}$$

In this example, the laminate described by the pair of strings before permutation is $(\pm 45, 90_2, 0_2, \pm 45_3, 0_4)_s$ and becomes $(\pm 45_4, 0_2, 90_2, 0_4)_s$ after permutation.

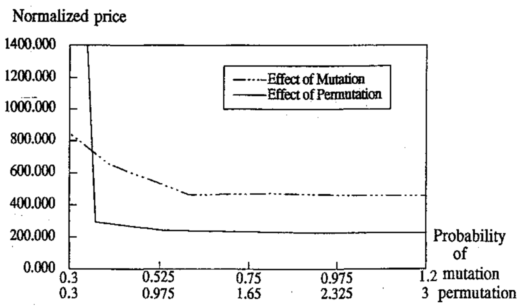
Inversion is a reordering operator that shuffles the bits while it keeps track of the original position (or meaning) of each bit. For example,

$$\begin{array}{ll} \text{Before inversion} & 2\,3\,/\,2\,2\,1\,/\,3\,1\,1 \\ \text{Position string} & 1\,2\,/\,3\,4\,5\,/\,6\,7\,8 \\ \text{After inversion} & 2\,3\,/\,1\,2\,2\,/\,3\,1\,1 \\ \text{Position string} & 1\,2\,/\,5\,4\,3\,/\,6\,7\,8 \end{array} \tag{8}$$

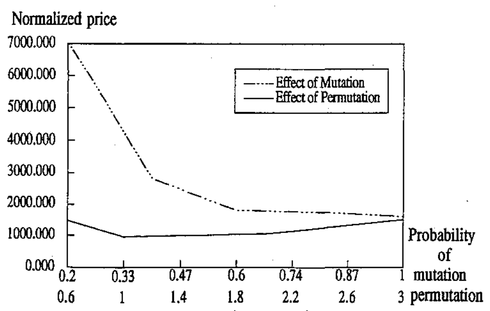
That is, by virtue of the position string, the laminate is $(\pm 45, 90_2, \pm 45_2, 0_2, 90_2, 0_4)_s$ both before and after inversion. Inversion is different from permutation in that it does not change the design but only its string representation. It is used for changing the effect of crossover, allowing physically remote properties to be adjacent on a string so as to reduce their probability of being separated by crossover.

Tuning the Genetic Algorithm

The design space for our problem was found to include a large number of near-optimal designs. For this reason, designs that are within a 10th of a percent of the global optimum were accepted as optimal and are called here practical optima. The genetic algorithm was tuned by numerical experiments. The



a) Buckling-only problem: probability of crossover = 1, population size = 8, search stopped after 10 generations without improvement



b) Buckling with strain and contiguity constraint: probability of crossover = 1, population size = 8, search stopped after 56 generations without improvement

Fig. 2 Comparison between mutation and permutation, 48-ply laminate.

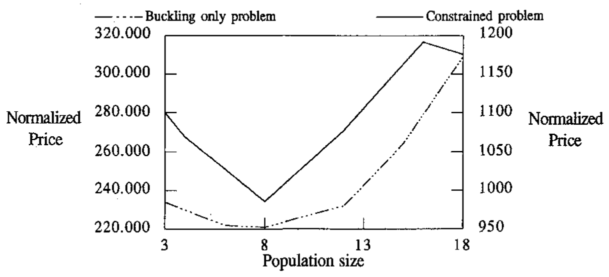


Fig. 3 Effect of population size, 48-ply laminate.

evaluation measure is the *normalized price,* which is the number of evaluations of the objective function (also called price) divided by the probability of reaching a practical optimum (called practical reliability). To compensate for random fluctuations in performance, the selection of the optimal genetic parameters was conducted by performing 200 genetic optimizations for each of 3 load cases (48 plies, $N_y/N_x = 0.125$, 0.25, and 0.5, respectively) to obtain average prices and reliabilities and also standard deviations. Typically, the coefficient of variation (standard deviation over mean value) was about 0.25–0.3 for the price.

The first step in the implementation of the algorithm was to choose a coding. Experiments were conducted to compare a traditional 0, 1 coding (where 00 stands for 0_2, 01 and 10 stand for ± 45, and 11 stands for 90_2) with the 1, 2, 3 alphabet. Four hundred runs were made for three different load cases. The 1, 2, 3 alphabet was 2% cheaper in terms of the normalized price than the 0, 1 coding. The binary coding is biased because the probability of appearance of a ± 45 stack of two plies is twice that of other stacks. In contrast, the 1, 2, 3 alphabet fits well the three possible stacks of plies and was adopted for further experiments.

The population size; probabilities of crossover, mutation, and permutation; the length of the search; and the value of the penalty parameter were optimized. A minimum value of the practical reliability was set at 80%. Many experiments were carried out and only the variations near the optimal set of parameters are discussed here. Note also that probabilities higher than 1 can appear in the mutation, permutation, and inversion operators as a result of the operator being applied more than once to the bit string.

Permutation is a mutation operator, with two additional features. First of all, new values of the bits come from the other positions in the string. In stacking sequence design, a pool of "good bits" usually dominates, independently of the position in the string. Restricting mutation to this pool improves the performance. Permutation can be viewed as a perturbation of the original design. In addition, permutation changes the bits in the center of the string more often than those at the extremities of the string. This feature is appealing for the buckling optimization. The bits on the extreme left side of the string correspond to the outermost plies. Their values have a dominant effect on the buckling load. Therefore, the best values on the left side of the string are easily discovered by the selection process. The difficulty is to find the optimum sequence going from the middle left to the right side of the string. Permutation helps by exploring essentially the center of the string. Figure 2a shows that the use of permutation lowers the normalized price of the optimization by about a factor of 2 compared with mutation and that very high probabilities of permutation are beneficial in the buckling-only case.

Figure 2b shows that extensive exploration of the middle of the laminate makes more sense for the buckling-only problem than when strain failure constraints are added. The genetic algorithm needs 300% permutation for the buckling-only problem, whereas only 100% is sufficient in the constrained case. Although buckling loads depend mainly on the outermost plies through the flexural stiffness matrix $[D]$, strain calculations depend on the extensional stiffness matrix $[A]$ for

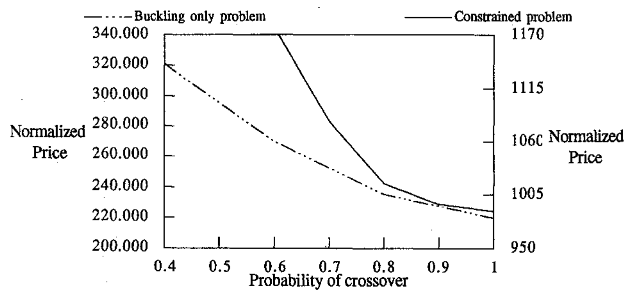


Fig. 4 Effect of probability of crossover, 48-ply laminate.

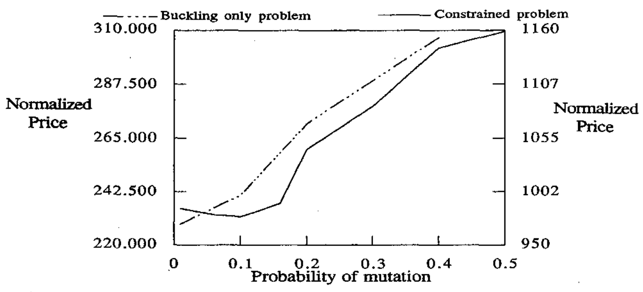


Fig. 5 Effect of probability of mutation, 48-ply laminate.

which each ply of the laminate has the same importance. Compared with mutation, permutation improves drastically the practical reliability without much increase in the price of the search.

A large population does not get enough mixing of the strings in a reasonable number of computations of the objective function. A good reliability would only be achieved at a very high price. On the other hand, too small a population misses so much genetic material that the search is driven by costly random mutation and permutation. A rational performance measure has been developed in Ref. 10, whose maximization over the population size yields a theoretical optimum population size. This calculation has been adapted to our case (1, 2, 3 coding, 48 plies in the laminate that gives chromosomes 12 bits long). The computed population size was about 22 members (see the Appendix). This theoretical result does not account for operators like mutation and permutation that create new bit combinations in the population. The theoretical optimum population size is thus overestimated. Figure 3 shows that the minimization of the normalized price yielded an optimum population size of eight individual for both the constrained and unconstrained buckling problems.

The effect of the probability of crossover on the normalized price is shown in Fig. 4. The optimum probability of crossover is found to be 1 in the unconstrained as well as in the constrained buckling maximization. Part of the reason for this uncommonly high probability is the straightforward way of coding a stacking sequence into a bit string. The main flaw of crossover is its tendency to separate interacting bits that are far apart. However, for our problem, interacting plies (e.g., due to contiguous ply constraint) are next to each other in reality and also have representing bits next to each other. Therefore, the tendency of crossover to separate interacting bits that are far apart is not relevant. Moreover, the loss of the best individual has often motivated a reduction in the probability of crossover. This argument does not affect the present algorithm, since it transmits the best individual unaltered.

Mutation is a pure exploration operator, and its effect is theoretically clear: as the mutation rate goes up, price and reliability increase. This was confirmed by the experiments. Nevertheless, a little mutation is an indispensable guarantee against eventual loss of diversity in the population. As shown in Fig. 5, 1% mutation in the buckling-only problem and 10% in the constrained case yielded the best normalized prices. The high probability of mutation in the constrained optimization may be linked to a wider diversity of ply orientations for the optimum laminates.

Experiments made with inversion yielded poorer performance. As for crossover, the explanation relies on the quality of the coding adopted. Since interacting bits are originally as close as possible, there is no need for inversion.

The value of the penalty parameter, enforcing the contiguous ply constraint, was also analyzed through numerical experiments. Figure 6 shows normalized prices. Five loading cases were necessary to find meaningful averaged results since, among the three loading cases previously chosen, one (N_y/

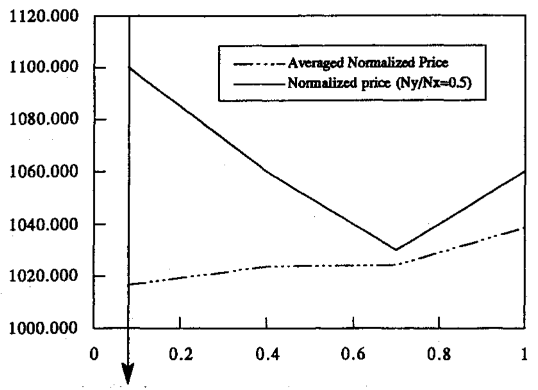
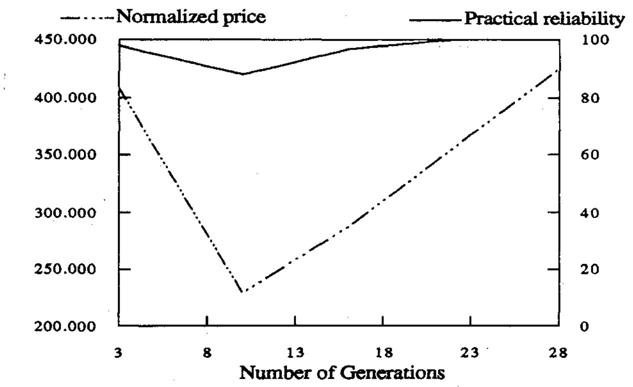
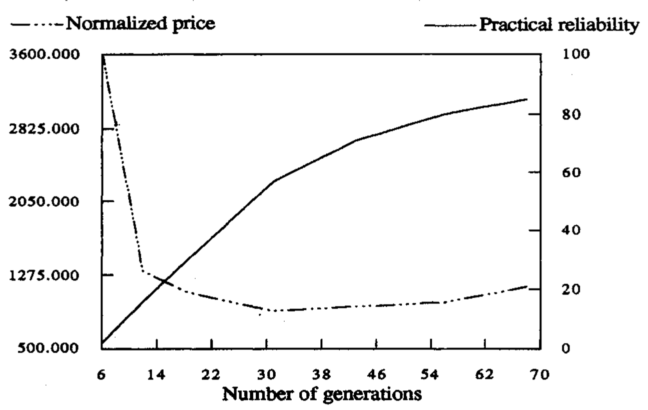


Fig. 6 Effect of penalty parameter for contiguous ply constraint, 48-ply laminate. Buckling with strain and contiguity constraint: probability of crossover = 1, probability of permutation = 1, probability of mutation = 0.06, population size = 8, search stopped after 56 generations without improvement.



a) Buckling-only problem: population size = 8, probability of permutation = 2.25, probability of mutation = 0.01, probability of crossover = 1



b) Buckling with strain and contiguity constraints: population size = 8, probability of permutation = 1, probability of mutation = 0.06, probability of crossover = 1

Fig. 7 Effect of stopping criterion (number of generations without improvement), 48-ply laminate.

$N_x = 0.5$) has an atypical optimal value of the penalty parameter. A penalty of 8% ($p = 0.08$) is the averaged optimum setting of the penalty parameter. This value is close to a minimum value for which the best infeasible designs are at the border of the practical optimum region. At about 8% penalty, designs that violate the contiguity constraint but that have high buckling and strain failure loads are able to compete and thus breed with average-performance feasible designs. Below 8% penalty, infeasible designs result from the genetic search. It should be noted, however, that in some cases, like ($N_y/N_x = 0.5$), a large penalty parameter significantly improves the price. For these loading cases, it is found that all of the practical optima naturally do not violate the contiguity constraint. The strong penalty merely limits the search to the good part of the design space.

The stopping criterion that defines the length of the search has also been optimized. Figure 7 illustrates that the optimum values found for the maximum number of generations without improvement in the best design were 10 for the buckling-only problem and 56 for the constrained case. The stopping criterion was driven by the condition of 80% reliability in the constrained problem.

Multiplicity of Optima

The stacking sequence design problem has the peculiarity of many near-optimal (practical) solutions or even sometimes many global optimum designs. This feature contributes to the atypical setting of the genetic parameters. For instance, a high rate of permutation is beneficial as well as a fast-stopping criterion.

Because the genetic algorithm optimizes a population of designs and relies on probabilistic transitions, many near-optimal stacking sequences can be found. Compared with other optimization methods, the genetic algorithm gives the designer more freedom in the choice of the best plate. Table 1 shows, for example, the global optimum and the next best designs for a load ratio $N_y/N_x = 0.5$. In the cases shown, the failure mode was buckling. The best laminate has 4% margin between strain failure and buckling failure, the second and third best designs have a margin of about 2%. The laminate with ±45-deg plies on the outside may be preferred to one with 90_2-deg plies on the outside for its improved damage tolerance, despite a slightly lower buckling load.

For multimodal functions, the genetic search can converge to many of the optima. This property is particularly useful in stacking sequence design, where multiple global optima often occur. A single optimization by genetic algorithm yields many optima. By keeping track of stacking sequences whose fitnesses are within a 10th of a percent from the best-so-far design, we generate a list of practical optima. The numbers of practical optima found per search that are presented next were average values over 20 trials. For the case presented in Table 2 (buckling-only problem, 64 plies, $a = 20$, $b = 10$, $N_x = 1$, $N_y = 1$), 1.2 practical optima were found on the average during one search, with the stopping criterion of 10 generations without improvement and a normalized price of 309.7. For a slower stopping criterion (50 generations without improvement), we averaged 9.5 practical optima per search, with a normalized price or 695.0. It was observed that most of the practical optima are discovered late in the genetic search. Table 3 presents the constrained optimization problem: 48 plies, $a = 20$, $b = 5$, $N_x = 1$, $N_y = 0.125$. Strain failure is the critical failure mode for all of these laminates. On the average, 11.1 practical optima were found per search, with a normalized price of 545.

Table 1 Optimal and near-optimal designs for buckling, strain, and contiguous ply constraints

	Load factor, λ	
Stacking sequence[a]	Buckling	Failure
$(90_2, \pm45_2, 90_2, \pm45, 90_2, \pm45_6)_s$	9998.19	10,394.81
$[(90_2, \pm45_2)_2, 90_2, \pm45, 90_2, \pm45_3]_s$	9997.60	10,187.93
$(\pm45, 90_4, \pm45, 90_2, \pm45_5, 90_2, \pm45)_s$	9976.58	10,187.93

[a]48 plies, $a = 20$ in., $b = 5$ in., $N_x = 1$ lb, $N_y = 0.5$ lb.

Table 2 Multiple optima for buckling-only problem

	Load Factor	
Stacking sequence[a]	Buckling	Failure
$(90_{10}, \pm45_2, 90_2, \pm45_3, 90_2, \pm45_4)_s$	3973.01	14,205.18
$(\pm45, 90_{10}, \pm45, 90_8, \pm45, 90_8)_s$	3973.01	8,935.74
$(90_4, \pm45_2, 90_{16}, \pm45, 90_6)_s$	3973.01	8,935.74
$(90_2, \pm45, 90_6, \pm45, 90_8, \pm45, 90_{10})_s$	3973.01	8,935.74
$(90_8, \pm45, 90_2, \pm45, 90_2, \pm45, 90_2, \pm45_6)_s$	3973.01	14,205.18

[a]64 plies, $a = 20$ in., $b = 10$ in., $N_x = 1$ lb, $N_y = 1$ lb.

Table 3 Multiple optima for buckling, strain, and contiguous ply constraints

	Load Factor	
Stacking sequence[a]	Buckling	Failure
$(90_2, \pm45_4, 0_4, \pm45, 0_4, \pm45, 0_2)_s$	14,168.12	13,518.66
$(\pm45_3, 0_2, \pm45_2, 0_2, 90_2, 0_4, \pm45, 0_2)_s$	14,134.76	13,518.66
$(90_2, \pm45_3, 0_2, \pm45, 0_2, \pm45, 0_4, \pm45, 0_2)_s$	14,013.71	13,518.66
$(\pm45_2, 0_2, \pm45_2, 90_2, 0_4, \pm45, 0_2, \pm45, 0_2)_s$	13,662.61	13,518.66

[a]48 plies, $a = 20$ in., $b = 5$ in., $N_x = 1$ lb, $N_y = 0.125$ lb.

Table 4 Effect of size of design space for buckling-only problem: population size = 8, probability of permutation = 2.25, probability of mutation = 0.01, probability of crossover = 1, search stopped after 10 generations without improvement

Number of layers	Size of design space	Normalized price	Design space explored, %	Price	Practical reliability
16	81	172.3	213.0	172.3	100.0
32	6,561	325.4	4.9	273.0	83.9
48	531,441	327.7	0.06	323.4	98.7
64	43,046,721	375.0	0.00087	371.0	98.9

Table 5 Effect of size of design space for buckling with strain and contiguity constraint: population size = 8, probability of permutation = 1, probability of mutation = 0.06, probability of crossover = 1, search stopped after 56 generations without improvement

Number of layers	Size of design space	Normalized price	Design space explored, %	Price	Practical reliability
16	81	1546.7	1900.0	1546.7	100.0
32	6,561	1850.0	28.19	1796.7	97.1
48	531,441	2163.4	0.41	2136.7	98.8
64	43,046,721	3090.0	0.0072	2636.7	85.3

Effect of the Size of the Problem

Like most pseudorandom search methods, a genetic algorithm improves its performance as the size of the problem increases. Tables 4 and 5 describe how price, normalized price, and practical reliability evolve as function of the number of plies. The price and the normalized price go up with the number of layers, but in a proportion that makes the genetic search more and more effective as related to the size of the design space. Credit should be given not only to the intrinsic properties of the genetic algorithm but also to the fact that thick laminates have many global optima. For the buckling-only optimization, the normalized price does not have such a regular progression. The 32-ply case shows low practical reliability, i.e., a high normalized price. Most of the time, performing a short genetic search was found advantageous for the buckling-only problem, but it makes the method more

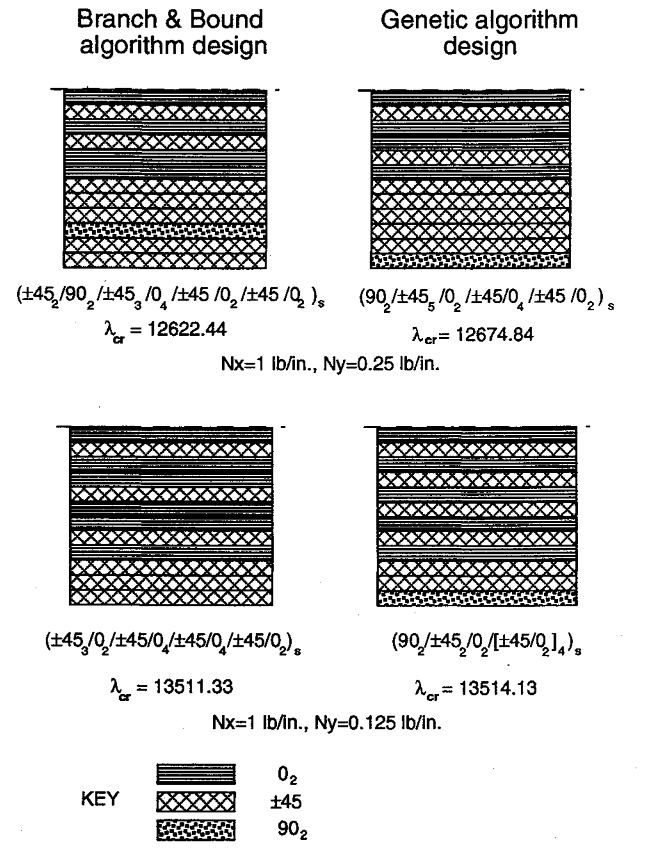

Branch & Bound algorithm design
Genetic algorithm design
$(\pm45_2/90_2/\pm45_3/0_4/\pm45/0_2/\pm45/0_2)_s$
$\lambda_{cr} = 12622.44$
$(90_2/\pm45_5/0_2/\pm45/0_4/\pm45/0_2)_s$
$\lambda_{cr} = 12674.84$
Nx=1 lb/in., Ny=0.25 lb/in.
$(\pm45_3/0_2/\pm45/0_4/\pm45/0_4/\pm45/0_2)_s$
$\lambda_{cr} = 13511.33$
$(90_2/\pm45_2/0_2/[\pm45/0_2]_4)_s$
$\lambda_{cr} = 13514.13$
Nx=1 lb/in., Ny=0.125 lb/in.
KEY
0_2
±45
90_2

Fig. 8 Comparison of optimum designs, 48-ply laminate: buckling with strain and contiguity constraint.

sensitive to premature convergence. Some loading cases are prone to premature convergence, and two of them occurred for the 32-ply case. Note also that all of the numerical experiments for the size of the problem included five load cases compared with three for the previous experiments. This explains the slight discrepancies between Tables 4 and 5 and other results. For the constrained problem, 48 plies, $N_y/N_x = 0.75$ and 1 were two cases difficult to optimize. The normalized prices of these optimizations were about 4000. The averaged normalized price over the five loading cases was 2163.4.

Why Stacking-Sequence Design by Genetic Algorithm?

Genetic algorithms require more objective function evaluations for stacking-sequence design than traditional search methods. However, as computers gain in efficiency, this drawback should cease to be as critical. A way of reducing the genetic algorithm's running time is parallel computation. Genetic algorithms can be adapted to parallel machines without major changes in their organization. For example, subpopulations can be reproduced simultaneously on separated processors and then regularly exchange information.

As explained previously, the genetic algorithm yields many practical optima during one search. It also deals well with integer variables and is not sensitive to problem nonlinearities. As a result, the genetic algorithm looks for global optimum, i.e., does not get trapped in local optimum. Figure 8 considers two load cases for 48-ply laminates, $a = 20$, $b = 5$, $N_x = 1$, and $N_y = 0.125$ and 0.25, respectively. In both cases, a genetic search finds laminates with higher failure loads than sequential linearization with the branch-and-bound algorithm does.

Concluding Remarks

The use of a genetic algorithm to optimize the stacking sequence of a laminated plate was presented. Buckling constraints as well as contiguity and strain constraints were considered. The genetic method was adapted to take advantage of the special features of the stacking-sequence design. A permutation operator was implemented and improved the performance of the genetic search. The design space was found to include multiple optima, especially for a large number of plies. The genetic algorithm was shown to be able to produce several of these optima in a single execution.

Appendix: Optimum Population Size for Genetic Algorithm Using 1, 2, 3 Coding

We reproduce Goldberg's derivation of the optimum population size for binary coded genetic algorithm[11] and adapt it to 1, 2, 3 coding. The fundamental tool to describe the processing of a genetic algorithm is called *schema.* A schema describes a subset of strings with similarities at certain string positions (i.e., a hyperplane in the design space). If we consider a string of length 5, the schema @1111 (@ is the "don't care" symbol) matches three strings (11111, 21111, 31111). In a string with l positions, there are 2^l schemata, since each position can either be fixed or nondefined.

The first step of the derivation is to count the number n_s of unique schemata expected in the original population. This number is derived in Ref. 10 to be

$$n_s = \sum_{j=1}^{l} \binom{l}{j} 3^j \left\{ 1 - \left[1 - \left(\frac{1}{3} \right)^j \right]^m \right\} \tag{A1}$$

where m is the population size, and l is the length of the string.

Then Goldberg defines a performance measure for the genetic algorithm. Noticing that a population with only one member (2^l schemata) is "something of a zero point," he defines the number of excess schemata as $n_{es} = n_s - 2^l$ and then maximizes the ratio $(n_s - 2^l)/m$. The key idea is to maximize the number of schemata per individual. The optimization was performed using a safeguarded quadratic interpolation method. The optimum population size found for 48 plies ($l = 12$) is 22.

Acknowledgment

This work was supported by NASA Grant NAG-1-168.

References

[1]Schmit, L. A., and Farshi, B., "Optimum Design of Laminated Fibre Composite Plates," *International Journal for Numerical Methods in Engineering,* Vol. 11, 1977, pp. 623–640.

[2]Mesquita, L., and Kamat, M. P., "Optimization of Stiffened Laminated Composite Plates with Frequency Constraints," *Engineering Optimization,* Vol. 11, 1987, pp. 77–88.

[3]Haftka, R. T., and Walsh, J. L., "Stacking-Sequence Optimization for Buckling of Laminated Plates by Integer Programming," *AIAA Journal,* Vol. 30, No. 3, 1992, pp. 814–819.

[4]Nagendra, S., Haftka, R. T., and Gürdal, Z., "Buckling Optimization of Laminate Stacking Sequence with Strain Constraints," *Proceedings of the Tenth Conference on Electronic Computation* (Indianapolis, IN), 1991, pp. 205–212.

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

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

[7]Hajela, P., "Genetic Search—An Approach to the Nonconvex Optimization Problem," *AIAA Journal,* Vol. 26, No.7, 1990, pp. 1205–1210.

[8]Rao, S. S., Pan, T. S., and Venkayya, V. B., "Optimal Placement of Actuators in Actively Controlled Structures Using Genetic Algorithms," *AIAA Journal,* Vol. 28, No. 6, 1990, pp. 942–943.

[9]Hajela, P., and Lin, C. Y., "Genetic Search Strategies in Multicriterion Optimal Design," *Proceedings of the AIAA/ASME/ASCE/AHS/ASC 32nd Structures, Structural Dynamics, and Materials Conference* (Baltimore, MD), Vol. 2, AIAA, Washington, DC, 1991, pp. 354–363.

[10]De Jong, K. A., "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Ph.D. Dissertation, Dept. of Computer Science, Univ. of Michigan, Ann Arbor, MI, 1975.

[11]Goldberg, D. E., "Optimal Initial Population Size for Binary Coded Genetic Algorithms," Univ. of Alabama, TCGA Rept. 85001, Tuscaloosa, AL, Nov. 1985.