Pergamon

Composites Engineering, Vol. 5, No. 2, pp. 143–161, 1995

0961-9526/95 $9.50 + .00

0961-9526(94) E00087-5

IMPROVED GENETIC ALGORITHM FOR MINIMUM THICKNESS COMPOSITE LAMINATE DESIGN

R. Le Riche and R. T. Haftka†

Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, U.S.A.

(*Received* 28 *July* 1994; *final version accepted* 26 *September* 1994)

Abstract—The use of a genetic algorithm for the minimum thickness design of composite laminated plates is explored. A previously developed genetic algorithm for laminate design is thoroughly revised and improved, by incorporating knowledge of the physics of the problem into the genetic algorithm. Constraints are accounted for by combining fixed and progressive penalty functions. Improved selection, mutation, and permutation operators are proposed. The use of an operator called scaling mutation that projects designs toward the feasible domain is investigated. The improvements in the genetic algorithm are shown to reduce the average price of a genetic search by more than 50%.

INTRODUCTION

Because of their high strength to weight ratio and the possibility of fine tuning their behavior, notably by selecting fiber orientations, the optimization of composite laminated plates has received growing attention in the last two decades (e.g. Adali and Duffi, 1990; Le Riche and Haftka, 1993; Miki, 1979; Nagendra *et al.*, 1992; Schmit and Farshi, 1979; Shin *et al.*, 1989). When designing composite laminated plates, typical design variables are the fiber orientations and the number of layers. Manufacturing constraints often limit the choice of fiber orientations to 0°, ±45° and 90°, and fix the basic ply thickness, making the minimum weight design of a laminate a discrete optimization problem.

The design of a composite laminate for maximum buckling load can be formulated as a linear problem, and then the Branch and Bound method can be used to find the optimum design, which is a global minimum (Haftka and Walsh, 1992). In more general cases, however, such as when strain constraints are added, the problem is nonlinear. Furthermore, because composite laminates are characterized by a number of parameters which is usually smaller than the number of design variables, different sets of design variables can produce similar results, i.e. there are many optimal or near-optimal designs. For nonlinear problems, linearization in combination with the Branch and Bound algorithm may yield sub-optimal designs because of the existence of many different locally optimal designs (Nagendra *et al.*, 1992; Shin *et al.*, 1989). For this reason, genetic algorithms, which are typically more robust at finding global or near global optima, have been recently explored for designing composite laminated plates (Ball *et al.*, 1993; Callahan and Weeks, 1992; Kogiso *et al.*, 1994a,b; Le Riche and Haftka, 1993).

Genetic algorithms (GAs) are search techniques based on a simulation of the Darwinian concept of survival of the fittest and natural reproduction genetics operating on a population of designs (e.g. Goldberg, 1989; Holland, 1975; Whitley, 1993). As optimization algorithms, they are global in scope and can work on discrete as well as continuous design problems. They also have the advantage of generating a number of optimal and near-optimal designs during a single search. Early applications of genetic algorithms to structural optimization are due to Goldberg and Santani (1987) and Hajela (1990). Genetic algorithms have since then been applied to numerous structural optimization problems (e.g. Furuya and Haftka, 1993; Nagendra *et al.*, 1993; Powell and Skolnick, 1993; Shoenauer and Xanthakis, 1993; Watabe and Okino, 1993).

In Le Riche and Haftka (1993) a genetic algorithm was exhaustively tested for maximizing the failure load of a fixed thickness laminate by changing its stacking sequence.

† Presently at: Dept. of Aerospace Engineering, Mechanics and Engineering Science, University of Florida, Gainesville, FL 32611, U.S.A.

Kogiso *et al.* (1994b) applied the same basic algorithm to the dual problem of minimizing the thickness with failure constraints. The minimum thickness problem was found to be more costly to solve than the failure load maximization. The objective of the present paper is to explore various techniques for improving the efficiency of the genetic algorithm. This is done by first studying the penalty functions that account for strength and stability constraints and then by improving the genetic operators.

PROBLEM DESCRIPTION

The simply supported laminate shown in Fig. 1 is loaded in the x and y directions by N_x and N_y, respectively. The laminate is composed of N plies, each of thickness t, and is balanced and symmetric. Its longitudinal and lateral dimensions are a and b, respectively.

The laminate buckles into m and n half-waves in the x and y directions, respectively, when the loads reach the values $\lambda_b(m, n)N_x$ and $\lambda_b(m, n)N_y$. The buckling load factor $\lambda_b(m, n)$ is given in terms of the flexural stiffness coefficients D_{ij} as,

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

The smallest value of λ_b over possible combinations of m and n is the critical buckling load factor λ_{cb}. If λ_{cb} is larger than unity the laminate can sustain the actual applied loads N_x and N_y without buckling.

The strength of the plate is predicted here by the maximum strain criterion. That is, the plate is assumed to fail if any of the ply strains exceeds its allowable value. 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 from the classical lamination theory for symmetric and balanced laminates,

$$N_x = A_{11}\varepsilon_x + A_{12}\varepsilon_y$$

$$N_y = A_{12}\varepsilon_x + A_{22}\varepsilon_y$$

and

$$\varepsilon_1^i = \cos^2\theta_i\varepsilon_x + \sin^2\theta_i\varepsilon_y$$

$$\varepsilon_2^i = \sin^2\theta_i\varepsilon_x + \cos^2\theta_i\varepsilon_y$$

$$\gamma_{12}^i = \sin^2\theta_i(\varepsilon_y - \varepsilon_x)$$

where A_{ij} are the coefficients of the extensional stiffness matrix and θ_i is the fiber orientation in the ith layer. The critical strength failure load factor λ_{cs} is defined as

$$\lambda_{cs} = \min_i \left[\min\left(\frac{\varepsilon_1^{ua}}{f|\varepsilon_1^i|}, \frac{\varepsilon_2^{ua}}{f|\varepsilon_2^i|}, \frac{\gamma_{12}^{ua}}{f|\gamma_{12}^i|} \right) \right]$$

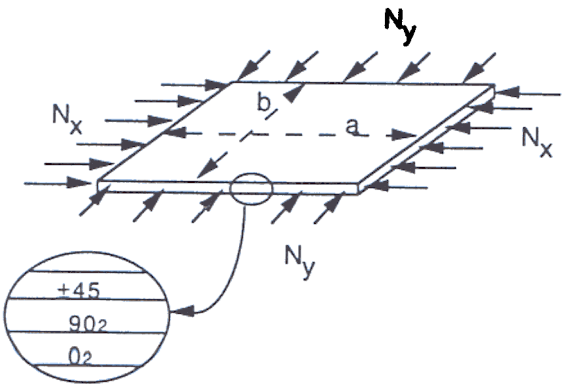

Fig. Simply supported laminated plate subjected to normal in-plane loads.

where ε_i^{ua} are the ultimate allowable strains, and a safety factor $f = 1.5$ is used. If λ_{cs} is smaller than unity, the plate is assumed to fail. The critical load factor λ_{cr} is,

$$\lambda_{cr} = \min(\lambda_{cb}, \lambda_{cs}).$$

To alleviate matrix cracking problems, we also require that there are no more then four contiguous plies with the same fiber orientation.

The purpose of the optimization is to find the thinnest symmetric and balanced laminate satisfying the 4-ply contiguity constraint that will not fail because of buckling or excessive strains. The symmetry constraint is readily accommodated by optimizing only one half of the laminate, the other half being obtained by symmetry. The balance constraint is enforced by associating $+45°$ and $-45°$ plies into one $\pm 45°$ stack. Stacks of 0_2°, $\pm 45°$ and 90_2° are our basic building blocks.

Genetic algorithms solve unconstrained optimization problems, so that constraints need to be either incorporated into the objective function via penalty functions or removed by artful definition of the design variables. The following sections describe the genetic algorithm and then the penalty formulations.

A BASIC GENETIC ALGORITHM FOR LAMINATE DESIGN

The basic genetic algorithm developed by Kogiso *et al.* (1994b), Le Riche and Haftka (1993) and Nagendra *et al.* (1993) is summarized as the baseline for the present work. 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 string. The lengths of all the design strings have to be the same. Because the designs can have different number of plies, empty plies are added as "padding" to make all designs have the same nominal number of plies. The coding used here is the standard stacking sequence notation with the additional symbol E for an empty ply. Because our basic unit is a stack of two plies, laminates can be composed only of 0_2, ± 45, 90_2 and E_2 stacks. A genetic search changes the population of strings by mimicking evolution.

A flow chart of the genetic algorithm is given in Fig. 2. The GA starts with the generation of an initial random population of designs (three designs on Fig. 2). In a second step, the objective function values of the designs are evaluated. Each design is assigned a fitness value depending on the objective function values of all the designs in the population. It is then processed, by so called genetic operators, to create a new population, which probabilistically combines the most desirable characteristics of the old population. The selection operator selects parent strings based on their fitness and passes them on to the operators performing the genetic recombination. The probability of a design of being chosen as parent increases with its fitness. The old population is then replaced by the new one, with one exception. The best of the previous generation is cloned into the next generation, which corresponds to an "elitist" version of the genetic algorithm. This process is repeated until convergence. The genetic operators are described in more detail hereafter.

Selection scheme

The first step in creating a new generation is the selection of sets of parent designs for mating. Each individual has a fitness value that determines its probability of being chosen as a future parent. In the present work, the fitness was chosen to be the relative rank in terms of the objective function Φ 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. Selection is implemented by allocating to each design a portion of the segment [0, 1] equal in length to its fitness. A random number is then created between 0 and 1 that designates the selected individual.

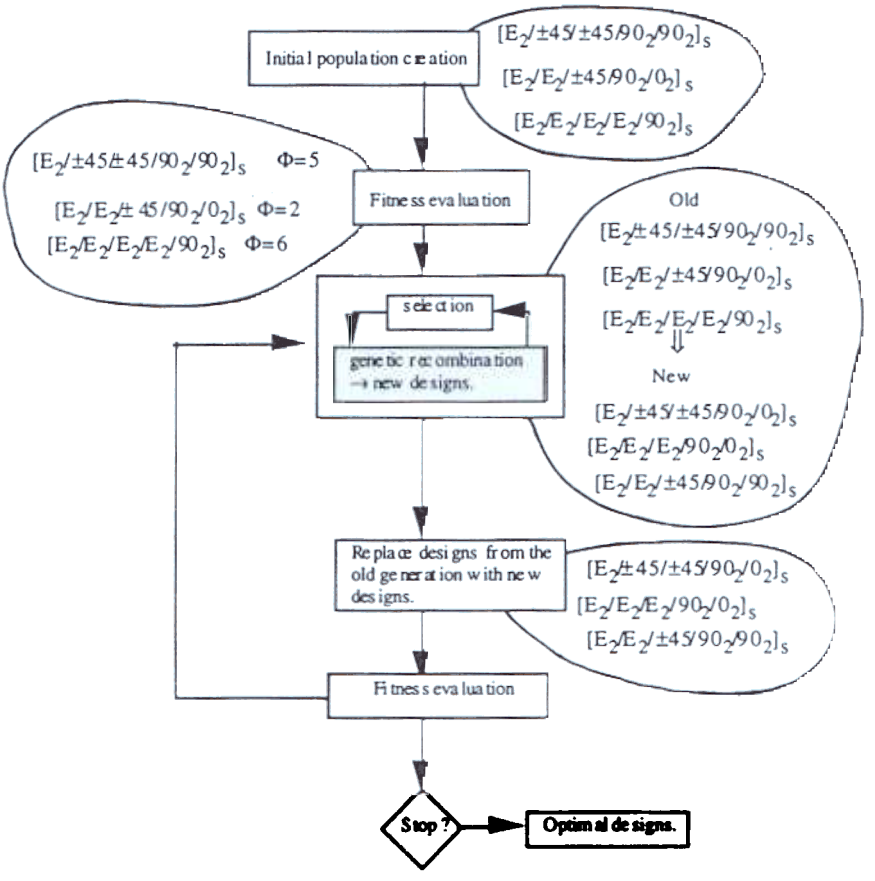

Fig. 2. Genetic algorithm flow chart.

Crossover

Crossover is the operation used to trade characteristics of parent designs and create offsprings by exchanging parts of the parents' strings. The crossover used in the basic genetic algorithm is a two-point crossover, where two break points are chosen randomly. One offspring can be created by splicing the parents' three substrings, i.e. the first substring comes from the first parent, the second from the second parent, and the third from the first parent (see Fig. 3). Crossover is given a specific probability of occurring, usually quite large. The present work uses a probability of applying crossover equal to one, based on the results presented by Le Riche and Haftka (1993).

Selection and crossover are the two pivots of the generic search in the sense that they suffice to simulate evolution on a population of designs, although not in a robust fashion.

Mutation

The design generated by crossover is next subjected to the mutation operator. Mutation performs random changes in an individual's string with a low probability (0.01 in this paper). Its purpose is to bring in completely new bit combinations, thereby protecting

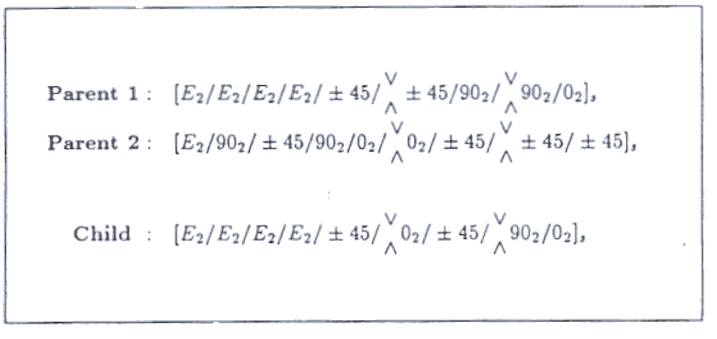

Fig. 3. Traditional two-point crossover

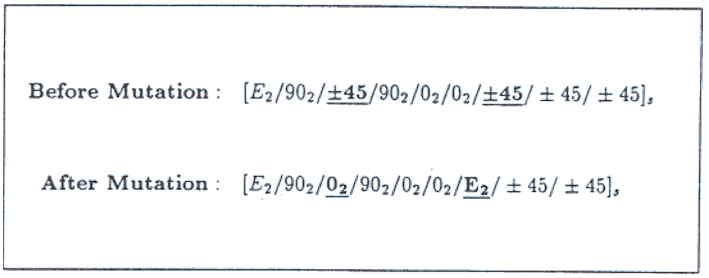

Fig. 4. Mutation operator.

the population against becoming uniform. Each bit in the string has "probability of mutation" chances of changing. Figure 4 gives an example where two bits are mutated.

Permutation

Permutation, a new operator devised for laminate design by Le Riche and Haftka (1993), is also a stochastic operator. Permutation shuffles the location of layers without changing the composition of the laminate. Therefore, the flexural properties of the laminate are modified while its in-plane characteristics are preserved. First, two points in the string are selected, creating three substrings in the chromosome. The two stacks at the two extremities of the central substring are then flipped about the center point of the substring. Figure 5 illustrates the working of permutation. In accord with results presented by Le Riche and Haftka (1993) and Kogiso *et al.* (1994b) permutation is applied with a probability of 1.

Objective function formulation: handling of constraints

The minimum weight design problem treated here has five constraints: buckling, strength failure, symmetry, balance, and the 4-ply contiguity constraint.

Constraints in GAs are handled in the following three ways: data structuring, repair operators and penalty functions. Data structuring refers to defining the design variables so that the optimizer always produces feasible designs. Here, for example, we model only half of the laminate, which guarantees the symmetry; we use stacks of two plies, always pairing $+45$ and -45, which guarantees the balance conditions. Data structuring is sometimes associated with the use of specialized operators in order to hide the existence of constraints (Michalewicz and Janikow, 1991).

The second way of handling constraints in GAs is through the use of repair operators. A repair operator can be thought of as a projection from an infeasible point in the design space to a feasible point. Repair operators are usually applied probabilistically so as to increase the probability of finding a feasible design at the end of the search. Orvosh and Davis (1993) recommend using repair operators on 5% of the points sampled for combinatorial problems. Repair operators, like data structuring, are very case specific. A repair operator called "scaling mutation" will be described later in this paper.

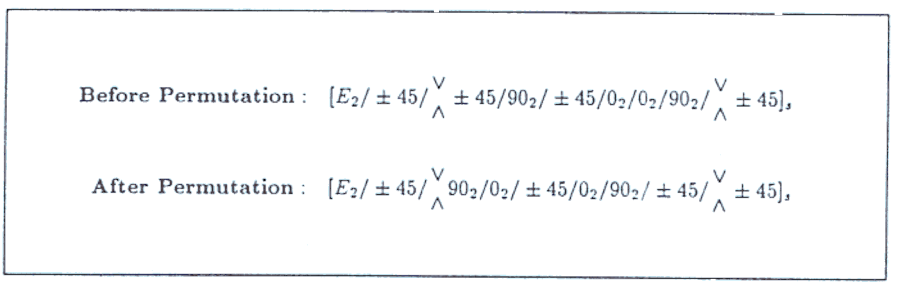

Fig. 5. Permutation operator.

Finally, constraints can be incorporated into the objective function through penalty functions. The use of penalty functions is the most general technique for constrained genetic optimization. For continuous optimization, the quadratic penalty function is the most popular, but this popularity is due to differentiability requirements which are not relevant in genetic optimization. Conventional wisdom in GAs is that the penalty should be proportional to the distance to the feasible domain, so as to discriminate between designs having different levels of infeasibility (Richardson *et al.*, 1989). It has also been noticed that the penalty should be kept as low as possible, just above the limit below which infeasible designs become optimal (Le Riche and Haftka, 1993; Richardson *et al.*, 1989). The flaw of heavily penalizing infeasible designs is that it limits the exploration of the design space to feasible regions, precluding short cuts through infeasible regions. Large penalties promote convergence to local minima in the feasible space, a phenomenon often referred to as premature convergence.

For the present laminate design problem, the objective is to minimize the number of plies N in the laminate. The strength, stability and contiguity constraints are incorporated into the objective function through penalty strategies. The contiguity constraint is enforced by multiplying the objective by $P_c^{n_c}$, where P_c is the penalty parameter for the ply contiguity constraints, and n_c is the number of stacks of two plies in excess of the constraint value of two stacks (i.e. 4 plies) in one half of the laminate. For example, the objective function corresponding to $(0_6/90_2)_s$ is multiplied by P_c; the objective function of $(90_6/0_4)_s$ is multiplied by P_c^2 (the first P_c accounts for the extra 0_2 stacks at the laminate midplane and the other P_c accounts for the extra 90_2 stack).

The treatment of the failure and stability constraints is more complicated for two reasons. First, because of the discrete nature of the problem, there may be several feasible designs of the same minimum thickness. Of these designs, we define the optimum to be the design with the largest failure load. Therefore, as in Kogiso *et al.* (1994a,b), the objective function is linearly reduced in proportion to the failure margin for designs that satisfy or almost satisfy the stability and failure constraints. For such designs the objective function Φ_0 is defined as

$$\text{if } \lambda_{cr} \geq (1 - \delta), \qquad \Phi_0 = P_c^{n_c}\{N + \varepsilon[(1 - \delta) - \lambda_{cr}]\}.$$

δ is a small tolerance parameter added for more flexibility in the definition of feasibility, and ε is a small parameter for rewarding constraint margin. When the stability or failure constraints are violated, it is possible to estimate the additional thickness needed to achieve feasibility from simple scaling rule. Based on the scaling rule, the objective function Φ_0 for designs that violate the stability or failure constraints is defined as

$$\text{if } \lambda_{cr} < (1 - \delta), \qquad \Phi_0 = P_c^{n_c}\left\{\frac{N}{\lambda_{cr}^{P_l}}\right\}$$

where P_l is an exponent used in the scaling law.

Three assumptions were made in Kogiso *et al.* (1994b) to estimate the values of ε and P_l:

> Scaling rule: the strain failure loads are approximately proportional to the thickness of the laminate. The buckling loads are approximately proportional to the cube of the laminate thickness.
> The optimum design is located at the boundary between feasible and infeasible domains, i.e. $\lambda_{cr} \simeq 1$.
> When estimating parameters for the failure constraints, the contiguity constraints can be neglected ($n_c = 0$).

Too large a value for ε would favor thick designs having a large feasibility margin over the optimum design. The maximum value of ε was estimated in Kogiso *et al.* (1994b) by comparing the objective function of a hypothetical optimum design having N plies and $\lambda_{cr} = (1 - \delta)$ with a heavier feasible design of cN plies ($c > 1$). Neglecting the discreteness of the number of plies, we assume that c can vary continuously. Then, under the above assumptions, the requirement that a design thicker than the optimal should

have a higher objective becomes,

$$\begin{aligned} N &< cN + \varepsilon[(1-\delta) - c^3], && \text{if buckling is critical,} \\ N &< cN + \varepsilon[(1-\delta) - c], && \text{if strains are critical.} \end{aligned} \tag{1}$$

Buckling failure yields the tighter bound on ε,

$$\varepsilon < \frac{N(c-1)}{c^3 - (1-\delta)} \tag{2}$$

For $\delta < 0.01$, the upper bound on ε increases with c until a point $1 < c_{max} < 1.1$ and then decreases. To be on the safe side, we calculate the upper bound on ε for $c = 2$ and $\delta = 0$, what yields $\varepsilon < N/7$. For the cases considered hereafter, $N > 44$ so eqn (2) will be satisfied for $\varepsilon \leq 6$.

A small penalty parameter P_l could lead to designs with large constraint violation having lower objectives than designs with small violations. So, bounds on the penalty parameter P_l can be obtained by comparing the hypothetical optimum design with an infeasible design that has cN plies (for $c < 1$) and a critical failure load λ_l. From assumptions 2 and 3, the penalty parameters need to be chosen so that,

$$N < \frac{cN}{\lambda_l^{P_l}}. \tag{3}$$

According to the scaling rule, $\lambda_l = c$ if the strains are critical and $\lambda_l = c^3$ if buckling is critical. The bounds found by Kogiso *et al.* (1994b) were $P_l > 1$ if strains are critical and $P_l > 1/3$ if buckling is critical.

The main problem of the previous objective function formulation is that for some load cases, the critical load of infeasible designs that are one stack lighter than the optimum feasible designs is very close to unity. Then P_l needs to be very large for the objective function Φ_0 of feasible optimum designs to be smaller than the objective function of good infeasible designs. But a large P_l was seen as being detrimental to the efficiency of the genetic search (Le Riche and Haftka, 1993; Richardson *et al.*, 1989).

The objective function and operators described so far are those used by Kogiso *et al.* (1994a,b), Le Riche and Haftka (1993) and Nagendra *et al.* (1993). We will now consider modifications for better efficiency in minimum weight laminate design.

IMPROVEMENTS TO THE GENETIC ALGORITHM

Modified objective function formulation

The previously outlined problem of needing large values of P_l is avoided in the present paper by adding a small constant penalty S for violating the strength or stability constraints. That is, the modified objective function Φ is defined as

$$\begin{aligned} &\text{if } \lambda_{cr} \geq (1-\delta), && \Phi = P_c^{n_c}\{N + \varepsilon[(1-\delta) - \lambda_{cr}]\}, \\ &\text{if } \lambda_{cr} < (1-\delta), && \Phi = P_c^{n_c}\left\{\frac{N}{\lambda_{cr}^{P_l}}\right\} + S. \end{aligned} \tag{4}$$

S and $(1/\lambda_{cr}^{P_l})$ are both penalties, but their action differs in that $(1/\lambda_{cr}^{P_l})$ is a function of the distance to feasibility while S is not. If $S > 0$, P_l can be chosen smaller than the minimum value of 1 derived in Kogiso *et al.* (1994b).

Figure 6 shows how the objective function Φ varies as a function of the most critical constraint for $N = 44$, 48 and 52, $P_l = 0.5$ and $P_l = 2$, $S = 1$, $\delta = 0$, and $\varepsilon = 6$. As an example, we show on the figure the best 44, 48 and 52 ply designs obtained for load case 1 (see Results for complete description). The critical failure load factors λ_{cr} for these designs are $\lambda_{cr} = 0.879$, 1.040 and 1.201, respectively, and are marked on the curves by points A, B and C, respectively. For $S = 1$ and $P_l = 2$ the optimal design (point B) has a substantial lower objective function than either the infeasible design (A) or the heavier feasible design (C). When $P_l = 0.5$, the advantage of B over A becomes more marginal. Figure 7 gives a magnified representation of Φ such as shown on Fig. 6 for $P_l = 0.5$

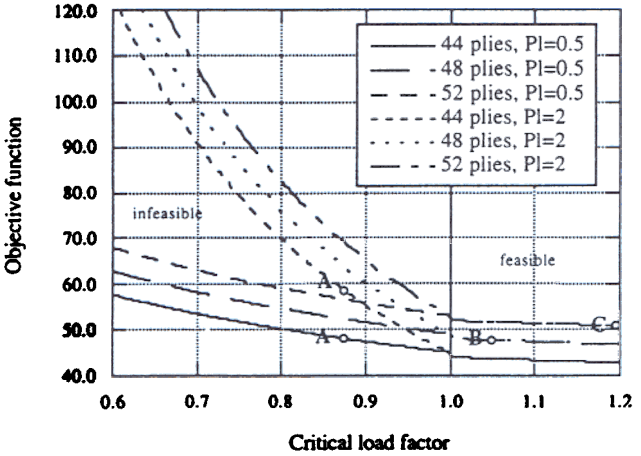

Fig. 6. Objective function Φ, $S = 1.0$, $n_c = 0$. Points A, B and C correspond to best designs obtained for load case 1.

and $S = 1$, with the additional case $P_l = 0.5$ and $S = 0$. The setting $P_l = 0.5$, $S = 0$ makes the infeasible design (point A) have a lower objective function than the optimum (point B). Figures 6 and 7 together illustrate the use of S to allow us to reduce the required value of P_l.

Considering bounds on S, we enforce $S > 0$ so as not to penalize feasible designs. Because of balance and symmetry constraints, the best feasible design is two stacks (4 plies) heavier than the best infeasible design. Neglecting the ply contiguity constraint, for the most pessimistic scenario where λ_{cr} of the best feasible and infeasible tend towards one, S can be chosen so as to guarantee the optimality of the feasible design,

$$N + S \geq N + 4. \tag{5}$$

So a value of $S = 4$ will guarantee optimality. More refined bounds on S can be calculated on-line once P_l is set using eqn (3). Various strategies have been tested for adapting the value of the step size S in the course of the run, but they did not yield any clear improvement of the GA, and so they will not be discussed any further here.

Modified selection procedure

A test is addded to the selection procedure so as to never mate two identical designs, which would probably create a clone. This test is aimed at preserving population diversity.

Crossover for varying thickness laminates

We consider two questions for tailoring the crossover operator to the laminate design problem. First, how many break points are optimal; and second, should we accord special treatment to empty plies because they control the weight of the laminate. We compared

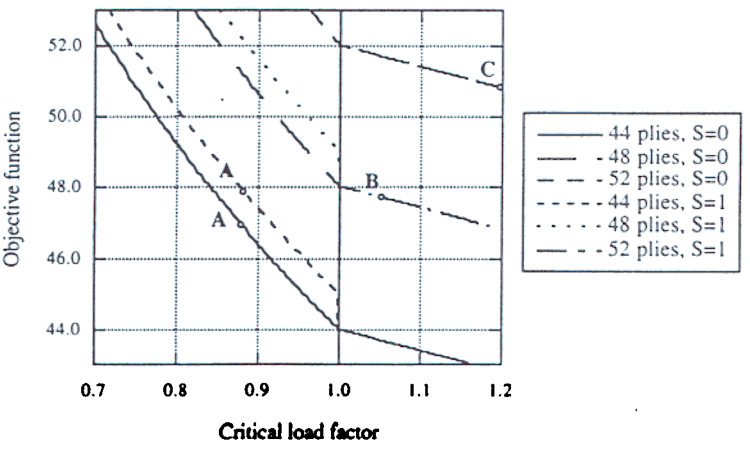

Fig. 7. Objective function Φ, $P_l = 0.5$, $n_c = 0$. Points A, B and C correspond to best designs obtained for load case 1.

the performance of one-point, two-point and uniform crossovers for laminate design. Uniform crossover (Syswerda, 1989) is a recombination process where each bit of the offspring has 50% probability of coming from one parent or the other. For L-digit strings, uniform crossover has a maximum of $L - 1$ breaking points, a minimum of 0 breaking points, and an average of $(L - 1)/2$ breaking points. The two-point crossover is a standard among practitioners of genetic algorithms because, on average, it minimizes the probability that crossover separates k ($2 \leq k \leq L$) beneficially interacting digits in the string (Spears and De Jong, 1991). This effect of crossover to separate bits carried by one of the parents is called "disruption". However, empirical evidence has shown that two-point crossover does not always perform the best (cf. Eschelman *et al.*, 1989; Syswerda, 1989). In Spears and De Jong (1991) it is conjectured that disruption is not always a drawback, but can make the offspring more different from their parents, which might be crucial in avoiding premature convergence, especially for small population sizes. Moreover, two-point crossover minimizes disruption *on the average*, i.e. when sets of digits far apart on the strings as well as digits close to each other are considered. In our case, since the string is the direct representation of the laminate cross-section, layers that are close interact more (in terms of bending and contiguity) than layers that are far from each other. One point crossover is less disruptive to adjacent layers.

When two laminates of different thicknesses are mated, the locations of the break points determine the thickness of the child design. Also, with a single-point crossover when the break point falls in the empty part of both parent laminates (in the middle of the *E*s on the left of the string), the first parent design is cloned. We considered three crossover strategies that have different effects on the thickness of the child design. The first strategy corresponds to the crossover used in Kogiso *et al.* (1994b) and Nagendra *et al.* (1993) where the breaking points can fall anywhere in the parent strings. The second strategy called "thin crossover" resricts the location of the break points to the full part of the thinner parent laminate (not in the *E*s). For such a crossover, the thickness of the offspring is equal to the thickness of one of its parents, but no averaging of the thickness is possible. The third strategy called "thick crossover" restricts the break points to the full part of the thicker parent laminate. In that case, some averaging of the parents' thicknesses might occur. A fourth strategy, called "averaging crossover", was tried which cuts and splices the parents' strings so that the offspring will always have a thickness which is near the average thickness of the parents. Averaging crossover did not work well, probably because it sometimes moves plies closer or further from the laminate midplane during recombination, thus changing substantially the bending properties. This strategy is not discussed further.

The aforementioned features were combined in the seven crossover operators described in Table 1. They will be evaluated later. Figure 8 shows examples of two-point crossovers (one-point crossovers can be easily deduced from the former). To facilitate tracking of where offspring's bits come from, a new notation was substituted to the usual laminate notation. The *E*s symbols are kept for empty plies, but I_{PI}s replace the fiber orientations, where I is the digit position number and $_{PI}$ designates the parent.

New mutation

The traditional mutation operator previously described, appears to have a biased and uneven effect depending on the string length (i.e. maximal laminate thickness) and

Table 1. Descriptions of crossover operators

X1	One-point crossover, the break point can be chosen anywhere in the string
X1-thin	One-point crossover, the break point is chosen in the full part of the thinner laminate
X1-thick	One-point crossover, the break point is chosen in the full part of the thicker laminate
X2	Two-point crossover, both break points are chosen anywhere in the parent string
X2-thin	Two-point crossover, both break points are chosen in the full part of the thinner parent string
X2-thick	Two-point crossover, both break points are chosen in the full part of the thicker parent string
UX	Uniform crossover

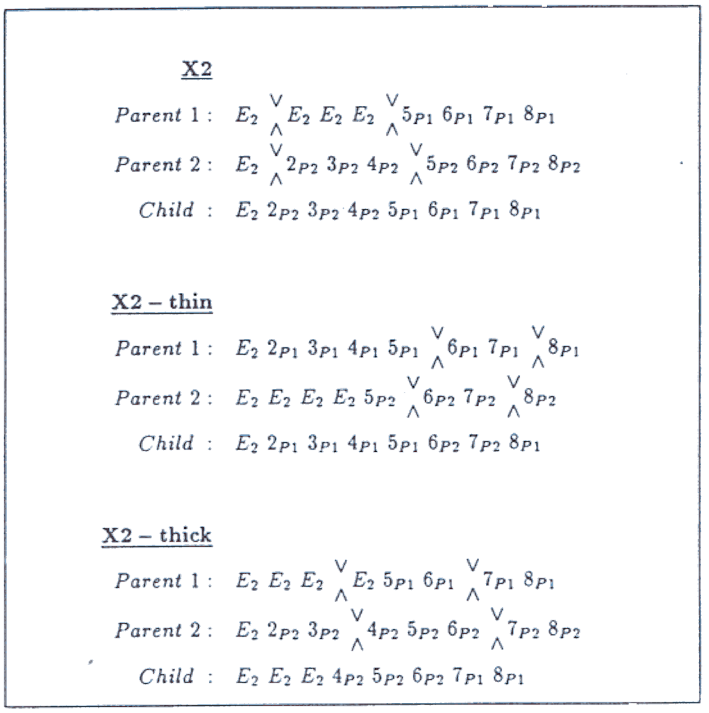

Fig. 8. Examples of crossover operators.

number of digits denoting full stacks (i.e. actual laminate thickness). To illustrate this, let us calculate the probabilities of adding at least two stacks, deleting at least two stacks and changing the fiber orientation of at least two stacks per laminate (remember that because of symmetry one digit in the string actually represents two stacks). For the example problems presented in this report, optimal designs have 12 full stacks and a string length of 15. A mutation hitting a full stack has 67% chance of changing the ply orientation and 33% chance of having this stack deleted. If the probability of mutating a stack is set to 0.01, a design with 12 full stacks has 0.67 $(1 - 0.99^{12}) \simeq 0.0750$ probability of having at least two stacks changed, 0.0375 probability of having two stacks deleted and $(1 - 0.99^3) \simeq 0.0300$ probability of having two stacks added. If the number of full stacks decreases, the probability of stack deletion per string decreases, the probability of stack addition per string increases, and the probability of change of fiber orientation per string decreases. The oppposite holds when the number of full stacks increases.

Thus the effect of the mutation operator will vary depending on the thickness of the design and the maximum thickness that can be coded. Moreover, it is not possible to tune independently the probabilities of adding plies, deleting plies and changing fiber orientations.

To correct this, three separate mutation probabilities are used: a probability of adding plies per string (PADD), a probability of deleting plies per string (PDEL), and a probability of changing the fiber orientation per digit (PCFO). The new mutation adds or deletes two stacks somewhere in the laminate if the corresponding probabilistic test is passed. For change of fiber orientation, a test is implemented for each digit coding two full stacks, and if it is passed, the fiber orientation is changed.

New permutation

The original algorithm (Le Riche and Haftka, 1993) has been developed for a class of relatively easy problems for which only one load case was considered in the optimization problem. However, for more difficult problems where many loadings are considered when designing a laminate, we believe that the old version of permutation shuffles the digits too much. Accordingly, we propose and test a new permutation operator inducing less changes in the string. This operator randomly selects two stacks in the full part of the laminate and swaps them (cf. Fig. 9).

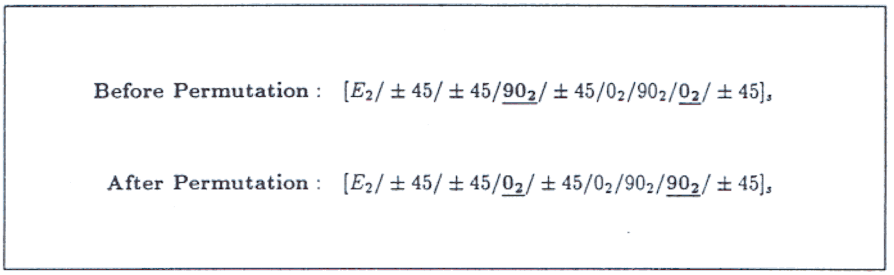

Fig. 9. New permutation operator.

Scaling mutation

When a design is infeasible, it makes sense to bias mutations toward adding rather than deleting stacks. Furthermore, using the scaling rule, we can estimate the thickness deficiency and use it to determine the magnitude of the bias. Similarly, for a feasible design, we can bias mutations to favor deleting stacks. The strain load factor (λ_{cs}) varies proportionally to the laminate thickness, and the critical buckling load factor (λ_{cb}) varies proportionally to the cube of the laminate thickness. These results are exact if each ply thickness is scaled up or down by the same ratio. When the thickness is changed by adding or deleting a stack, the scaling rules provide only rough approximations. However, they help to direct the population of designs towards optimal weight regions of the design space at a very low calculation cost. Hence scaling can be seen as a "repair operator" (Michalewicz and Janikow, 1991; Orvosh and Davis, 1993; Underbrink and Williams, 1994), where the buckling and strain constraints are being repaired. When performing scaling, the two following equations are solved for h_b^{new} and h_{st}^{new},

$$= \lambda_{cb}^{\text{new}} = \lambda_{cb}\left(\frac{h_b^{\text{new}}}{h}\right)^3 \tag{6}$$

$$1 = \lambda_{cs}^{\text{new}} = \lambda_{cs}\left(\frac{h_{st}^{\text{new}}}{h}\right), \tag{7}$$

where h is the present laminate thickness, h_b^{new} and h_{st}^{new} are the desirable laminate thicknesses after scaling due to buckling and strains, respectively. The laminate thickness after scaling is $h_{\text{new}} = \max(h_b^{\text{new}}, h_{st}^{\text{new}})$. If $h_{\text{new}} < h$, the number of plies that would yield laminate thickness the closest to h_{new} is deleted. If $h_{\text{new}} > h$, $(h_{\text{new}} - h)$ is truncated to the inferior multiple of the stack thickness, and the corresponding number of stacks is added. The consequence of this last truncation is that when a design is almost feasible, scaling is not applied. In that case indeed, it is likely that feasibility can be achieved without increase in thickness, by finding a better stacking sequence. To minimize the influence of the fiber orientations, the plies are added or deleted at the laminate midplane. The orientation of the added plies is random except that no contiguity constraint violation is introduced in the process. Figure 10 illustrates how scaling works.

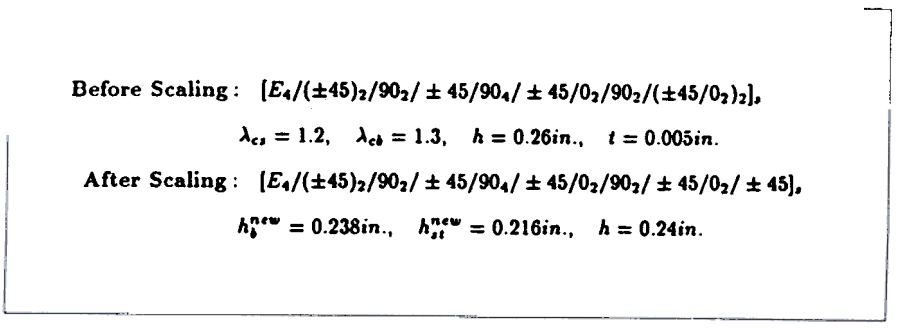

Fig. 10. Scaling mutation.

Scaling is performed after fitness evaluation and before the selection-recombination process (cf. Fig. 2) in order to have all the needed data available (λs and weights). For the designs that have been changed during scaling, an approximate analysis is performed to update their objective function value Φ. In the approximate analysis, the weight and the number of contiguous plies in excess of four are known exactly, and it is assumed that the most critical constraint after scaling has a value equal to unity (which would be true if we were scaling all the layers by the same amount, and if h was not limited to an integer multiple of the ply thickness).

RESULTS

Genetic algorithm performances will be discussed in terms of "price of the search" and "reliability". The reliability of the algorithm is the probability it has of finding a practical optimum. A practical optimum is defined here as a optimal weight design with λ_{cr} within 0.1% of λ_{cr} of the global optimum. The price of the search is the number of analyses necessary to reach 80% reliability, i.e. to have 80% probability of finding a practical optimum.

We consider a graphite–epoxy plate with longitudinal and lateral dimensions $a = 20$ in and $b = 5$ in, respectively. The material properties are: $E_1 = 18.50 \times 10^6$ psi; $E_2 = 1.89 \times 10^6$ psi; $G_{12} = 0.93 \times 10^6$ psi; $\nu_{12} = 0.3$; $t = 0.005$ in (t is the basic ply thickness); $\varepsilon_1^{ua} = 0.008$; $\varepsilon_2^{ua} = 0.029$; $\gamma_{12}^{ua} = 0.015$. The maximum thickness for a laminate is assumed to be 64 plies (i.e. string length = 64/4 = 16).

Four different load cases are considered: load case 1, $N_x = 13{,}000$ lb/in and $N_y = 1625$ lb/in; load case 2, $N_x = 12{,}500$ lb/in and $N_y = 3125$ lb/in; load case 3, $N_x = 9800$ lb/in and $N_y = 4900$ lb/in; and a multiple load case where the loadings [$N_x = 12{,}000$ lb/in, $N_y = 1500$ lb/in], [$N_x = 10{,}800$ lb/in, $N_y = 2700$ lb/in] and [$N_x = 9000$ lb/in, $N_y = 4500$ lb/in] are considered simultaneously, i.e. the optimal design should be able to withstand each of these loadings. For each case and each tuning of the genetic algorithm 200 independent searches are performed, each of them stopped after 6000 analyses. The multiple load case was not studied in Le Riche and Haftka (1993), and this prevents direct comparisons on the price of the searches between Le Riche and Haftka (1993) and this work. All the load cases considered here were first treated by Kogiso *et al.* (1994b). Table 2 summarizes the main characteristics of the optimal designs for each of the load cases considered here.

Among the single load cases, load case 1 was found to be easy for the genetic algorithm because strain constraints are by far the most critical. There are many practical optima that are different combinations of the same pool of fiber orientations. Load case 1 has many more practical optima than any other load case considered here (cf. Table 2). Load case 2 is more difficult since it has only three practical optima. Although load case 3 has 13 practical optima, it was found to be misleading for the genetic algorithm (Kogiso *et al.*, 1994b): practical optimal designs are composed of 45° and 90° plies exclusively. However, in the course of the search, the algorithm has difficulties removing the 0° plies because the strain constraint is critical in these plies. As the number of 0° plies in the

Table 2. Summary of optimal designs

Load case	Optimal design	Failure mode	Number of practical optima
	$[\pm 45_3/0_4/\pm 45/0_4/90_2/0_2]_s$	Strain	>13*
2	$[\pm 45_2/90_2/\pm 45_3/0_2/\pm 45/0_4/\pm 45/0_2]_s$	Strain	3
3	$[90_2/\pm 45_2/(90_2/\pm 45)_2/\pm 45_3]_s$	Buckling	13
Mult.	$[90_4/\pm 45_3/0_4/\pm 45/0_4/90_2/0_2]_s$	Buckling and strain†	4

† Many constraints active.

* Although we did not try to enumerate all of the practical optima for load case 1, it was easy to find 14 of them, which establishes that load case 1 has the most practical optima.

laminate is reduced, the strains in the remaining 0° plies increase. However, once the last 0° ply has been removed, the strain constraint is no longer critical, 48 ply feasible designs that fail in buckling can be found. This kind of problem is known as a singular optimum and as being "GA-deceptive", leading to convergence to non-optimal designs. The multiple load problem is also a difficult test case because four failure modes (strain failure due to the first load set and buckling failure due to the three load sets) occur in the optimal region of the design space. Many active constraints around the optimum design means that many of the design points surrounding the optimum will be infeasible, therefore penalized, which lowers their probability of being selected for reproduction. However, some of these infeasible designs might have given a quick access to the optimum. Typically, the presence of many active constraints around the optimum reduces the number of ways in which designs can be recombined to produce the optimum, thus slowing down the search. The GA exhibits difficulties of a different nature for load case 3 and for the multiple load case. As a result, changes made to the GA often had opposite effects on the performance of the multiple load case and of load case 3. The GA needs to be able to make radical changes to the designs to allow removal of several 0° plies and solve load case 3. On the other hand, the large number of constraints in the multiple load case makes too much random exploration of the design space a rather inefficient strategy compared to a more careful juxtaposition of building blocks.

The rest of the paper describes the effect of each of the proposed modifications on the genetic search. A new GA composed of all the modifications will be used as reference and compared to versions of the GA where the modifications are replaced one at a time by their old counterparts. The new GA is made of the new selection, X1-thick crossover (probability = 1), new mutation (probabilities of adding and deleting a stack = 0.05, probability of changing the fiber orientation = 0.01), new permutation (probability = 1). Other parameters are: population size = 8, $P_l = 0.5$, $S = 1$, $P_c = \sqrt{10/9}$, $\delta = 0.005$. The old GA, such as used by Kogiso *et al.* (1994a,b) and Le Riche and Haftka (1993) is composed of the old selection, X2 crossover (probability = 1), the old mutation (probability = 0.01), the old permutation (probability = 1), has $P_l = 2$, $S = 0$, and has otherwise the same settings as the new GA. The old and new versions of the algorithm are compared in Fig. 11 in terms of the reliability as a function of the number of analyses. The plot results from 200 independent runs made on each one of the four load cases described above.

The intersections of the curves with the 80% reliability line represent the price of the associated searches. The new GA costs 1450 analyses, compared to 3300 for the old GA, which represents a 56% decrease in the price of the search. Note, however, that with the old GA, the reliability is 0.6 at 1500 analyses, so making two runs of 1500 analyses has a reliability of $(1 - 0.4^2) = 0.84$. This is better than one run of 3000 analyses which has a reliability of 0.78.

Even though 200 tests are performed for each experiment, non-negligible deviations in the price of the searches have been noticed. The 200 runs have been repeated five

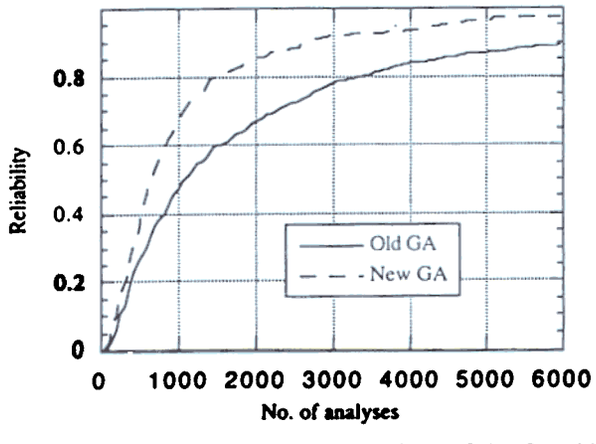

Fig. 11. Comparison of the old and new versions of the GA, 200 runs

times. Assuming a normal distribution of the price of the search, we calculated that there is a 90% chance that the price of the search is 440 ± 14 for load case 1, 1180 ± 71 for load case 2, 1490 ± 106 for load case 3, 3250 ± 252 for the multiple load case, and 1450 ± 135 on average. Note that the price of the search is more subject to variations on difficult problem cases (third and multiple load case) than on easy cases (first and second load cases). We will now detail the different improvements made to the old GA.

Effect of the penalty parameters

We discuss in this section the effect of the penalty parameters P_l and S on the price of the search of the new GA. Table 3 shows the price of the search and the reliability after 6000 analyses for the different load cases, and the average when $P_l = 0.5$ for $S = 0$, 0.5, 1, 2 and 4. Tables 4 and 5 show the effect of changing P_l for $S = 0$ and $S = 1$, respectively.

On average, the trend shown in the tables follows the rule that the minimum penalty is the most efficient (as long as it remains high enough to force convergence to the feasible domain) (Le Riche and Haftka, 1993; Richardson *et al.*, 1989). There seems to be an advantage in using a combination of fixed penalty and penalty proportional to the distance to the contraints boundary on the third and multiple load cases. This is an experimental argument, in favor of the use of some fixed penalty. Two other arguments were previously described: some fixed penalty makes it possible to lower the pressure for feasibility in the infeasible domain (i.e. lower P_l), allowing a more careful sampling of the infeasible domain while the step guarantees convergence to feasible designs. The use of the fixed step penalty S removes the need for infinitely large P_l for load cases where light infeasible designs can be found very close to the constraint boundary.

Table 3. Effect of S on the price of the search/reliability after 6000 analyses, $P_l = 0.5$

	Load case 1	Load case 2	Load case 3	Multiple load case	Average
$S = 0$		1070/0.99	1390/0.98	2760/0.95	
$S = 0.5$		1220/0.995	1490/0.95	2970/0.93	
$S = 1$	440/1.00	1180/1.00	1490/0.94	3250/1.00	1450/0.98
$S = 2$	450/1.00	1270/1.00	1700/0.98	4670/0.85	1640/0.96
$S = 4$	510/1.00	2200/0.99	4150/0.85	4670/0.85	3340/0.93

The GA did not converge in the feasible domain.

Table 4. Effect of P_l on the price of the search/reliability after 6000 analyses, $S = 0$

	Load case 1	Load case 2	Load case 3	Multiple load case	Average
$P_l = 0.5$	—	1070/0.99	1390/0.98	2760/0.95	
$P_l = 0.65$	360/1.00	1100/1.00	1540/0.98	3420/0.96	1590/0.98
$P_l = 0.85$	380/1.00	1080/1.00	1520/0.97	4660/0.85	1750/0.96
$P_l = 1.0$	410/1.00	1170/1.00	1480/0.99	5220/0.85	1700/0.96
$P_l = 2.0$	440/1.00	1550/0.99	3050/0.86	5010/0.91	1950/0.94
$P_l = 4.0$				*/0.76	

*The GA did not achieve the required 80% reliability.
—The GA did not converge in the feasible domain.

Table 5. Effect of P_l on the price of the search/reliability after 6000 analyses, $S = 1$

	Load case 1	Load case 2	Load case 3	Multiple load case	Average
$P_l = 0.3$				—	
$P_l = 0.5$	440/1.00	1180/1.00	1490/0.94	3250/1.00	1450/0.98
$P_l = 1.0$	420/1.00	1250/1.00	2760/0.91	3300/0.92	1680/0.96
$P_l = 2.0$	480/1.00	1870/0.99	3650/0.88	4950/0.88	2420/0.93

The GA did not converge in the feasible domain.

Table 6. Effect of selection on the price (top entry) and reliability after 6000 analyses (bottom entry)

	Load case 1	Load case 2	Load case 3	Multiple load case	Average
Old selection	480	980	3050	3300	1600
	1.00	1.00	0.91	0.97	0.97
New selection	440	1180	1490	3250	1450
(no duplicate parents)	1.00	1.00	0.94	1.00	0.98

Selection

Table 6 compares the price of genetic searches with the new (no duplicate parents) and old selection schemes for each load case and the average, using the reference GA previously described. For all cases except for load case 2, the new selection procedure was better. The gain in performance is explained by enhanced genetic diversity due to the prohibition on two identical parents. On average, the new selection procedure is responsible for a drop in price of the search from 1600 to 1450 analyses, a saving of 9%.

Crossover

The prices of the search and reliabilities after 6000 analyses associated with the use of different crossover operators are shown on Table 7.

Based on the extensive testing reported in Table 7, we can answer our two questions on how many break points are best, and the treatment of empty plies in crossover.

For the -thick and -thin versions of crossover, the best performance is yielded by the one-point crossovers. When break points are taken anywhere in the parent strings, uniform crossover performs the best and one-point crossover the worst. A possible explanation for this result is that one-point crossover is more likely to degenerate to cloning which occurs when the break point falls in the empty part of the laminates. The better performance of X1-thick is related to its performance on the difficult multiple load case. We conjecture that the multiple load case is effeciently solved when a careful juxtaposition of short substrings occurs. In contrast, the simultaneous suppression of the last 0° plies necessary to reach practical optimum designs in load case 3 benefits from crossover operators which are highly disruptive at a local level. The most effective crossover operator for solving load case 3 is the uniform crossover, followed by two-point crossover and one-point crossover. Figure 12 is a plot of the average number of stacks of two plies in excess of 2 in the best designs as a function of the number of analyses for the multiple load case, which is the case that causes the most difficulty for satisfaction of the 4-ply contiguity constraint. Figure 12 shows that the satisfaction of the contiguity constraints does not seem to play a role in the respective effect of the crossover operators, because the GA always locates good laminates without more than four contiguous plies in less than 10 generations for X1-thick and X-2 thick. X1-thick has an average price of 1450 while X2-thick costs 1580 analyses. About 8% of analyses can be saved by selecting the right number of break points for laminate design.

Table 7. Price of the search/reliability after 6000 analyses for various crossovers (crossovers defined in Table 1)

Crossover	Load case	Load case 2	Load case 3	Multiple load case	Average
X1	530/1.00	1220/1.00	1920/0.97	4250/0.89	1710/0.96
X2	500/1.00	1080/1.00	1780/0.99	3550/0.90	1680/0.97
UX	490/1.00	1440/1.00	880/0.99	4430/0.87	1610/0.96
X1-thin	520/1.00	1230/1.00	1420/0.98	3890/0.98	1470/0.98
X2-thin	550/1.00	1260/1.00	2330/0.95	3760/0.91	2070/0.96
X1-thick	440/1.00	1180/0.98	1490/0.94	3250/1.00	1450/0.98
X2-thick	490/1.00	1190/1.00	1380/0.92	3700/0.89	1580/0.95

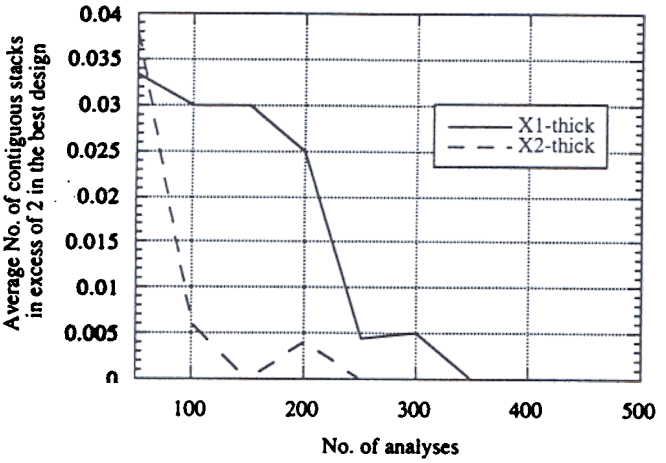

Fig. 12. Average number of contiguous stacks in excess of two in the best designs, multiple load case, 200 runs.

An evaluation of the different strategies for recombining laminate thicknesses can be made either by comparing the effects of X2, X2-thin and X2-thick, or by comparing X1, X1-thin and X1-thick. The -thick versions of crossover, where the break points are taken in the full part of the thicker laminate, create thickness averaging when one break point falls inside the thicker laminate but outside the thinner. With the -thin crossovers, the offspring's thickness will be equal to the weight of one of its parents. It can be seen on Table 7 that the -thick versions yield the best performance. Except for X1-thin on the load case 3 problem, -thin crossovers are not efficient. We conclude that for minimum laminate thickness design, a good crossover has a dual role of recombining both the fiber orientations in the layers and the total number of layers. The crossover operator that exhibits best all the above desirable features is X1-thick.

Mutation

The new mutation operator provides individual control over the probabilities of adding and deleting plies, and of changing fiber orientations. Table 8 reports tests performed with the new and old mutations operators. Various combinations of probabilities

Table 8. Effect of mutation on the price of the search (top entry) and reliability after 6000 analyses (bottom entry)

	Load case	Load case 2	Load case 3	Multiple load case	Average
Old mutation pm = 0.01	530 1.00	980 1.00	1750 0.96	3280 0.92	1490 0.97
New mutation pf = 0.01, pad = 0.01	1025 1.00	1340 0.94	1700 0.95	5460 0.82	1850 0.93
New mutation pf = 0.01, pad = 0.05	440 1.00	1180 1.00	1490 0.94	3250 1.00	1450 0.98
New mutatioon pf = 0.01, pad = 0.1	390 1.00	1740 1.00	1180 1.00	4030 0.88	1860 0.97
New mutation pf = 0.001, pad = 0.05	530 1.00	1060 0.97	2250 0.93	3600 0.93	1880 0.96
New mutation pf = 0.05, pad = 0.05	490 1.00	1690 1.00	1200 1.00	4230 0.86	1820 0.96
New mutation pf = 0.1, pad = 0.05	640 1.00	3630 1.00	1360 1.00	5920 0.81	3010 0.95

pm: probability of applying the old mutation/bit
pf: probability of changing the fiber orientation/bit
pad: probability of adding and deleting a stack/string.

of adding/deleting plies and changing the fiber orientations are tried. The best set had the probability of adding plies and probability of deleting plies equal to 0.05, and the probability of changing the fiber orientation was 0.01. The corresponding probabilities are quite close in the old mutation operator (probability of adding plies = 0.03, probability of deleting plies = 0.0375, probability of changing the fiber orientation = 0.0067). The main difference between new and old mutations is that the new mutation can add plies anywhere in the laminate, whereas the old mutation could only add new plies on the outside (where the Es are). As a result, the new mutation is a more uniformly random operator than the old mutation was. A minor improvement in the price of the search ensues from the new mutation operator (1450 against 1490 analyses).

Permutation

Because we believe that the digit shuffling caused by the old permutation was excessive, we introduced a new permutation that interchanges fewer digits. We also tested whether instead of this new permutation we can improve performance by reducing the probability of using the old permutation. Results presented in Table 9 confirm that the optimal probability of applying the old permutation is at or close to 100%. Permutation is thus a crucial operator. Indeed, while tracking the origin of practical optima for the multiple load case, we found that in 8 cases out of 10, permutation directly created the optimal design.

The new permutation improves performance for all cases, but it particularly helps the multiple load case. Table 9 shows that lowering the probability of permutation does not make the old permutation perform like the new one. It is logical that making few changes of a large magnitude to the designs (i.e. using the old permutation at a low rate) is not equivalent to making many small changes (i.e. intensively using the new permutation). Table 9 also shows that the optimal probability of applying the new permutation is about 1.0.

Scaling mutation

We tried a strategy where most of the additions and deletions of stacks were taken care of by scaling mutation, while reducing the rate of random mutation. We lowered the probabilities of adding and deleting stacks from 5% to 1%, and we applied scaling mutation on 10% of the designs. Table 10 reports what consequences scaling mutation has on the price of the search and reliability after 6000 analyses.

Table 9. Effect of permutation on the price of the search (top entry) and reliability after 6000 analyses (bottom entry)

	Load case 1	Load case 2	Load case 3	Multiple load case	Average
Old permutation	460	1340	3870	4840	2400
pp = 0.5	1.00	1.00	0.86	0.88	0.93
Old permutation	440	1480	1670	4750	2080
pp = 0.8,	1.00	0.99	0.95	0.86	0.95
Old permutation	460	1480	1570	4020	1990
pp = 1.0	1.00	1.00	0.97	0.85	0.96
New permutation	490	1210	5170	3110	1820
pp = 0.5	1.00	1.00	0.82	0.92	0.93
New permutation	440	1080	2190	3100	1500
pp = 0.8	1.00	1.00	0.94	0.92	0.96
New permutation	440	1180	1490	3250	1450
pp = 1.0	1.00	1.00	0.94	1.00	0.98

pp: probability of permutation/bit.

Table 10. Effect of scaling mutation on the price of the search (top entry) and reliability after 6000 analyses (bottom entry)

	Load case	Load case 2	Load case 3	Multiple load case	Average
Scaling	320	990	1570	3110	1310
	1.00	1.00	0.96	0.96	0.97
No scaling	440	1180	1490	3250	1450
	1.00	1.00	0.94	1.00	0.98

Scaling is beneficial to load cases 1 and 2, and in the multiple load case. In every load case, scaling helps to find designs in the optimal weight region early on in the search (before 500 analyses). Load case 1 is the ideal case where there are sufficiently many practical optima for scaling to have some chance of getting close to one. On the difficult third and multiple load cases, locating the optimal weight region does not mean getting very much closer to the optimum. On these cases, scaling yields mainly largely infeasible designs that have little chance of survival. It just wastes some of the genetic memory carried in the population. That explains why no decrease in price of the search is seen on load case 3, and why the improvement noted on the multiple load case is marginal (remember that this load case has a large standard deviation in price).

More generally, scaling is really intended to help the GA locate optimal weight regions, which is not a critical issue for the test problems treated here. The GA without scaling will always converge to 48 plies designs in about 500 analyses. On the other hand, the use of scaling mutation is very promising on cases where finding the optimal weight is a challenge. Such problems might arise when the string is much longer.

CONCLUSIONS

The genetic algorithm devised in Le Riche and Haftka (1993) and Kogiso *et al.* (1994b) has been improved and tested for minimum thickness design of composite plates.

A combination of fixed penalty function and a variable penalty function was proposed for optimizing composite laminates with strength and stability constraints by genetic algorithms. The combination of the two penalty functions guarantees convergence to feasible designs, which the variable penalty functions did not. The combination also yielded a more efficient genetic search, apparently due to a better control of the pressure towards feasibility.

Careful handling of laminate thicknesses as well as fiber orientations made it possible to improve crossover and mutation operators. A less disruptive permutation operator greatly helped performance. The resulting GA, tested over 200 runs on four different load cases, needs on average 1450 analyses to locate a practical optimum with 80% reliability, while the original GA needs 3300 analyses, which represents a 56% reduction in the price of the search. Scaling mutation, an operator that biases the change of thickness of the laminates according to their failure load factor, was found to provide a further 10% decrease in the price of the search.

This work illustrates the sensitivity of the performance of a GA to the implementation of the genetic operators. We demonstrated that problem specific knowledge can be incorporated in the fitness evaluation and genetic operators for improving the search efficiency.

Acknowledgements—This work was supported in part by NASA grants NAG1-168 and NAG1-1669. We thank Catherine Vayssade, Gilbert Touzot and the Université de Technologie de Compiègne for the provision of funding, equipment and fruitful discussions.

REFERENCES

Adali, S. and Duffy, K. J. (1990). Design of antisymmetric hybrid laminates for maximum buckling load: I. Optimal fiber orientation. II. Optimal layer thickness. *Compos. Struct.* 14(1) 49-60, 14(2), 113-124.

Ball, N. R., Sargent, P. M. and Ige, D. O. (1993). Genetic algorithm representations for laminate layups. *Artificial Intell. Engng* 8(2), 99-108.

Callahan, K. J. and Weeks, G. E. (1994). Optimum design of composite laminates using genetic algorithms. *Compos. Engng* 2(3), 149–160.

Eschelman, L., Caruana, R. and Scaffer, D. (1989). Biases in the crossover landscape. *Proc. Third Int. Conf. Genetic Algorithms*. Morgan Kaufmann, Fairfax.

Furuya, H. and Haftka, R. T. (1993). Genetic algorithms for placing actuators on space structures. *Proc. Fifth Int. Conf. Genetic Algorithms*. University of Illinois at Urbana-Champaign, 17-21 July, pp. 536–542. Morgan Kaufmann, Fairfax.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Massachusetts.

Goldberg, D. E. and Santani, T. A. (1987). Engineering optimization by a genetic algorithm. *Proc. 9th Conf. Electronic Computation*, ASCE.

Haftka, R. T. and Walsh, J. L. (1992). Stacking-sequence optimization for buckling of laminated plates by integer programming. *AIAA J.* 30(3), 814–819.

Hajela, P. (1990). Genetic search—an approach to the nonconex optimization problem. *AIAA J.* 26(7), 1205–1210.

Holland, J. H. *Adaption in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI.

Kogiso, N., Watson, L. T., Gürdal, Z. and Haftka, R. T. (1994a). Genetic algorithms with local improvement for composite laminate design. *Struct. Optimization* 7(4), 207–218.

Kogiso, N., Watson, L. T., Gürdal, Z., Haftka, R. T. and Nagendra, S. (1994b) Minimum thickness design of composite laminates subject to buckling strength constraints by genetic algorithm. *Mech. Compo. Mater. Struct.* 1(1), 95–117.

Le Riche, R. and Haftka, R. T. (1993). Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm. *AIAA J.* 31(5), 951–970.

Michalewicz, Z. and Janikow, C. Z. (1991). Handling constraints in genetic algorithms. *Proc. Fourth Int. Conf. Genetic Algorithms*, San Mateo, California, pp. 151–157. Morgan Kaufmann, Fairfax.

Miki, M. (1979). Optimum design of fibrous laminated composite plates subject to axial compression. *Proc. 3rd Japan-US Compos. Mater. Conf.* Tokyo, pp. 1017–1019. Technomic Publishing Co., Lancaster.

Nagendra, S., Haftka, R. T. and Gürdal, Z. (1992). Stacking sequence optimization of simply supported laminates with stability and strain constraints. *AIAA J.* 30(8), 2132–2137.

Nagendra, S., Haftka, R. T. and Gürdal, Z. (1993). Design of blade stiffened composite panel by a genetic algorithm. *Proc 34th Structures, Structural Dynamics and Material Conf.* La Jolla, California, 19–21 April pp. 2418–2436.

Orvosh, D. and Davis, L. (1993). Shall we repair? Genetic algorithms, combinatorial optimization, and feasibility constraints. *Proc. Fifth Int. Conf. Genetic Algorithms*, University of Illinois at Urbana-Champaign, 17-21 July, p. 650. Morgan Kaufmann, Fairfax.

Powell, D. and Skolnick, M. M. (1993). Using genetic algorithms in engineering design optimization with nonlinear constraints. *Proc. Int. Conf. Genetic Algorithms*. University of Illinois at Urbana-Champaign, 17-21 July, pp. 424–431. Morgan Kaufmann, Fairfax.

Richardson, J. T., Palmer, M. R., Liepins, G. and Hilliard, M. (1989). Some guidelines for genetic algorithms with penalty functions. *Proc. Third Int. Conf. Genetic Algorithms*, George Mason University, 4 June, pp. 191–197. Morgan Kaufmann, Fairfax.

Schmit, L. A. and Farshi, B. (1979). Optimum design of laminates composite plates. *Int. J. Numer. Methods Engng* 11(4) 623–640.

Schoenauer, M. and Xanthakis, S. (1993) Constrained GA optimization. *Proc. Fifth Int. Conf. Genetic Algorithms*. University of Illinois at Urbana-Champaign, 17-21 July, pp. 573–580. Morgan Kaufmann, Fairfax.

Shin, Y. S., Haftka, R. T., Watson, L. T. and Plaut, R. T. (1989). Design of laminated plates for maximum buckling load. *J. Compos Mater.* 23, 348–369.

Spears, W. M. and De Jong, K. A. (1991). An analysis of multi-point crossover. *Foundations of Genetic Algorithms*, pp. 301–315. Morgan Kaufmann, San Mateo.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. *Proc. Third Int. Conf. Genetic Algorithms*, pp. 2–9. Morgan Kaufmann, Fairfax.

Underbrink, A. J. Jr and Williams, G. P. W. Jr. Genetic algorithms applied to assembly operations. To appear in *Proc 1994 SPIE Conference KB AI Systems in Aerospace and Industry*, SPIE code number 2244-04.

Watabe, H. and Okino, N. (1993). A study on genetic shape design. *Proc. Fifth Int. Conf. Genetic Algorithms*, University of Illinois at Urbana-Champaign, 17-21 July, pp. 445–450. Morgan Kaufmann, Fairfax.

Whitley, L. D., Editor (1993). *Foundations of Genetic Algorithms 2*. Morgan Kaufmann, San Mateo.