Structural Optimization 7, 207–218 © Springer Verlag 1994

Technical Papers

Genetic algorithms with local improvement for composite laminate design*

N. Kogiso

Department of Aerospace Engineering, Graduate School, University of Osaka Prefecture, 1-1 Gakuen-cho Sakai, 593 Japan

L.T. Watson

Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA

Z. Gürdal

Department of Engineering Science and Mechanics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA

R.T. Haftka

Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA

Abstract This paper describes the application of a genetic algorithm to the stacking sequence optimization of a laminated composite plate for buckling load maximization. Two approaches for reducing the number of analyses required by the genetic algorithm are described. First, a binary tree is used to store designs, affording an efficient way to retrieve them and thereby avoid repeated analyses of designs that appeared in previous generations. Second, a local improvement scheme based on approximations in terms of lamination parameters is introduced. Two lamination parameters are sufficient to define the flexural stiffness and hence the buckling load of a balanced, symmetrically laminated plate. Results were obtained for rectangular graphite-epoxy plates under biaxial in-plane loading. The proposed improvements are shown to reduce significantly the number of analyses required for the genetic optimization.

1 Introduction

The design of composite laminates is often formulated as a continuous optimization problem with ply thicknesses and ply orientation angles used as design variables (e.g. Schmit and Farshi 1977). However, for many practical problems, ply thicknesses are fixed, and ply orientation angles are limited to a small set of angles such as $0°$, $90°$, and $\pm 45°$. Designing the laminate then becomes a stacking sequence optimization problem which can be formulated using integer programming.

The laminate stacking sequence design problem with frequency constraints has been formulated by Mesquita and Kamat (1977) with the numbers of plies as the design variables, leading to a nonlinear integer programming problem. More recently, Haftka and Walsh (1992) showed that the use of ply identity design variables linearizes the integer programming formulation of the stacking sequence buckling maximization design problem. However, when strength constraints are included, the problem becomes nonlinear again. The buckling maximization of laminates with strain constraints has been solved by Nagendra *et al.* (1992) using a sequential linear integer programming technique. The branch and bound algorithm was used to solve integer programming problems in the papers by Mesquita and Kamat (1977), Haftka and Walsh (1992), and Nagendra *et al.* (1992). More recently, Le Riche and Haftka (1993) solved this problem by genetic algorithms (GA), which can handle nonlinear integer problems without the need for linearization.

An early implementation of genetic search methods is credited to Rechenberg (1965), although the work by Holland (1975) has provided the theoretical basis of most contemporary developments. Genetic algorithms are stochastic optimization methods (e.g. Metropolis *et al.* 1953; Rinnoy Kan and Timmer 1984, 1985; Byrd *et al.* 1986; Eskow and Schnabel 1988) that work on a population of designs by recombining the most desirable features of existing designs. Following the evolutionary concept of survival of the fittest, selection of mates favours the fittest members (designs) of the population, and offspring (newly created designs) are created by splicing together features (genes) of the parent designs. Additionally, genetic mutation is used to create new design features. Genetic algorithms do not use any gradient information, and thus are particularly suited for problems (such as discrete optimization) where derivatives are not available. In the last decade, genetic algorithms have proven their ability to deal with a large class of combinatorial problems. In structural optimization applications, GAs have appeared only recently (Le Riche and Haftka 1993; Hajela 1990; Rao *et al.* 1990; Hajela and Lin 1992).

Despite their numerous advantages, a serious drawback of GAs is their high computational cost. Genetic algorithms usually require a large number of analyses, sometimes in the

*Presented at the ASME Winter Annual Meeting "Structures and Controls Optimization", pp. 13-28. Printed with permission from ASME.

range of thousands or even millions. Therefore, improvements in both the efficiency of the analysis and the execution of the GA are needed in order to make the genetic optimization affordable. The objective of the present work is to reduce the cost of the genetic search for the optimization of composite panels. We propose the use of a binary tree data structure to store the results of all new analyses performed during optimization, and retrieve the information for designs that appeared in previous iterations without the need for reanalysis. We also propose an approximation of the buckling load based on two lamination parameters. After evaluating exactly the buckling load for each design created by the genetic operators, new design strings are created by trying all possible exchanges of the locations of pairs of plies in the laminate. Then the buckling loads for these new designs are estimated using the approximation, and the best of these designs replaces the nominal design. By searching for a local optimum in a small neighbourhood of the nominal design, we try to improve the performance of the combinatorial optimization problems. This type of local improvement (searching for a local optimum in a small neighbourhood) was used for combinatorial problems by Tovey (1985, 1986). We apply this idea to improve the performance of genetic optimization for composite panel design.

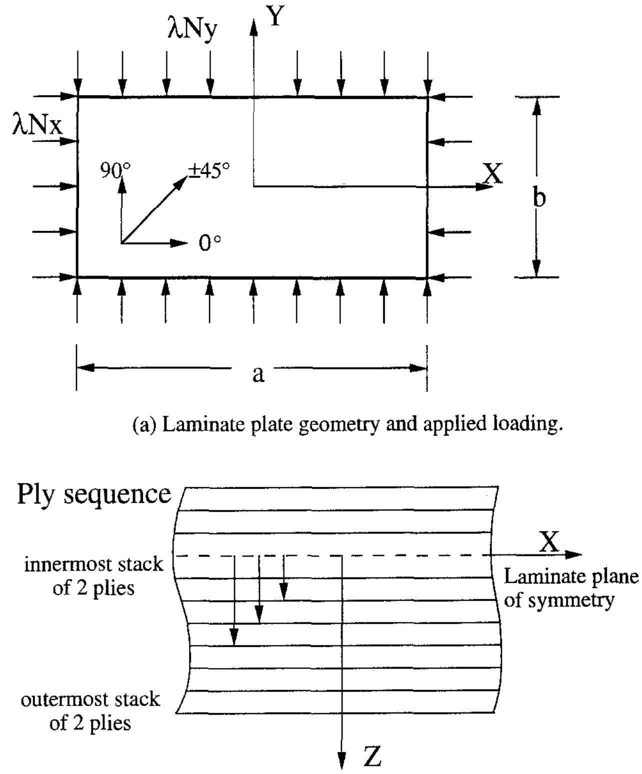
The efficiency of the proposed local improvement for the genetic search, as well as the use of the binary tree to retrieve previously analysed designs, are investigated for buckling load maximization of a rectangular 48-ply unstiffened laminated composite plate subjected to biaxial inplane loads. The analysis cost associated with this problem is low enough that thousands of genetic optimizations can be carried out for the purposes of averaging out the randomness in the performance of a single genetic optimization, and tuning the performance of the algorithm by adjusting the values of various control parameters.

1.1. Analysis and lamination parameters

The optimization problem is to maximize the buckling load of a simply supported plate by changing the laminate stacking sequence. Additionally, strain constraints are applied, and the number of contiguous plies of the same orientation is limited to four to alleviate matrix cracking problems. The analyses of the plate are needed to calculate the buckling load and the strain constraints.

The simply supported plate, shown in Fig. 1, has longitudinal and lateral dimensions of a and b, respectively, and is loaded in the x and y directions by λN_x, λN_y, respectively, where λ is a load amplitude parameter that we would like to maximize. The laminate is composed of N plies and assumed to be a symmetric, balanced laminate, made up of 0°, 90°, and $\pm 45^\circ$ plies of thickness t each. To reduce the number of design variables and enforce the balanced condition, the laminate is constrained to be made up of stacks of two 0° plies, two 90° plies, or a $+45^\circ$ and -45° pair of plies. These stacks are denoted by 0_2°, 90_2°, and $\pm 45^\circ$. Taking into account the symmetry, only $N/4$ ply orientations are required to define the entire laminate.

For a simply supported plate under biaxial compression loading, the plate buckles when the load amplitude parameter λ reaches a critical value λ_{cb} given as



(a) Laminate plate geometry and applied loading.

(b) Ply sequence location.

Fig. 1. Laminated plate geometry and loading

$$\frac{\lambda_{cb}(m,n)}{\pi^2} = \frac{D_{11}\left(\frac{m}{a}\right)^4 + 2(D_{12}+2D_{66})\left(\frac{m}{a}\right)^2\left(\frac{n}{b}\right)^2 + D_{22}\left(\frac{n}{b}\right)^4}{\left(\frac{m}{a}\right)^2 N_x + \left(\frac{n}{b}\right)^2 N_y}, \tag{1}$$

where m and n are the number of half waves in the x and y directions, respectively, that minimize λ_{cb}. The D_{ij}s are the flexural stiffnesses, which depend on the lamination sequence. For a laminate made out of a single fibrous material, the flexural stiffnesses can be expressed in terms of only two lamination sequence parameters and material constants (e.g. Miki and Sugiyama 1991),

$$D_{11} = \frac{h^3}{12}(U_1 + U_2 W_1^* + U_3 W_2^*),$$

$$D_{22} = \frac{h^3}{12}(U_1 - U_2 W_1^* + U_3 W_2^*),$$

$$D_{12} = \frac{h^3}{12}(U_4 - U_3 W_2^*), \quad D_{66} = \frac{h^3}{12}(U_5 - U_3 W_2^*), \tag{2}$$

where U_i $(i = 1, \ldots, 5)$ are the material constants, h is the total plate thickness, and W_1^* and W_2^* are the bending lamination parameters

$$W_1^* = \frac{12}{h^3}\int_{-h/2}^{h/2} z^2 \cos 2\theta \, dz, \quad W_2^* = \frac{12}{h^3}\int_{-h/2}^{h/2} z^2 \cos 4\theta \, dz, \tag{3}$$

defined in terms of the laminate ply orientation angles θ. The flexural stiffnesses D_{16} and D_{26} are assumed to be negligible.

The strain failure constraint requires all strains to remain below their allowable limits. In our case γ_{xy} is zero, and the laminate strains are related to the loads on the plate by the relations

$$\lambda N_x = A_{11}\varepsilon_x + A_{12}\varepsilon_y\,, \quad \lambda N_y = A_{12}\varepsilon_x + A_{22}\varepsilon_y\,. \qquad (4)$$

The strains in the i-th layer are obtained from the laminate strains by transformation

$$\varepsilon_{1_i} = \cos^2\theta_i\varepsilon_x + \sin^2\theta_i\varepsilon_y\,, \quad \varepsilon_{2_i} = \sin^2\theta_i\varepsilon_x + \cos^2\theta_i\varepsilon_y\,,$$
$$\gamma_{12_i} = \sin 2\theta_i(\varepsilon_y - \varepsilon_x)\,, \qquad (5)$$

where the A_{ij}s are the in-plane stiffnesses, and θ_i is the ply orientation angle of the i-th ply. These stiffnesses can also be expressed in terms of in-plane lamination parameters and the material constants as

$$A_{11} = h(U_1 + U_2V_1^* + U_3V_2^*)\,,$$
$$A_{22} = h(U_1 - U_2V_1^* + U_3V_2^*)\,, \quad A_{12} = h(U_4 - U_3V_2^*)\,, \qquad (6)$$

where the in-plane lamination parameters are defined as

$$V_1^* = \frac{1}{h}\int_{-h/2}^{h/2} \cos 2\theta \, dz\,, \quad V_2^* = \frac{1}{h}\int_{-h/2}^{h/2} \cos 4\theta \, dz\,. \qquad (7)$$

The strain failure load λ_{cs} is the largest load factor λ such that all the principal strains in every layer are less than or equal to the strain allowable values.

The ply contiguity constraint is implemented by penalizing the objective function, which is defined as the smallest of the load factors λ_{cs} and λ_{cb}, for constraint violations. This is achieved by redefining the objective function λ^* as

$$\lambda^* = p^n \min(\lambda_{cs}, \lambda_{cb})\,, \qquad (8)$$

where n is the number of contiguous plies in excess of the constraint value of four and p is a penalty parameter with a value of less than 1. The value of p in this study is 0.9.

2 Genetic algorithm

For a genetic algorithm, each design is coded as a finite string of digits. Here, a 0_2° stack is assigned the digit 1, a $\pm 45^\circ$ stack the digit 2, and a 90_2° stack the digit 3. For example, the laminate $[90_2^\circ/\pm 45_2^\circ/90_2^\circ/0_2^\circ/\pm 45_2^\circ/0_2^\circ]_S$ is encoded as 1 2 2 1 3 2 2 3 . The leftmost 1 corresponds to the layer closest to the laminate plane of symmetry. The rightmost 3 describes the outermost layer.

Figure 2 shows the pseudocode for the algorithm. The genetic search begins with the random generation of a population of design alternatives. Each individual has a fitness value based on its objective function that determines its probability for selection as a parent. Parents exchange parts of their genes (strings) in a process called crossover to create offspring (new design strings). Additional genetic operators, namely mutation and permutation, are applied to the child designs which then replace the parent generation.

Here, the best design is always carried to the next generation, which is an "elitist plan" version of the genetic algorithm. The optimization process is repeated until some specified number of generations provide no improvement in the best design. Following the generation of a new population, a local improvement procedure is implemented. The local improvement procedure replaces each design generated by the genetic operators by the estimated best of the neighbour designs. The procedure is described in detail in Section 2.2.

```
Procedure Genetic algorithm
begin
    initialize population;
    do I = 1, population size
    evaluate objective function;
    enddo
    rank designs;
    while number of consecutive generations without
    improvement in the best design less than a speci-
    fied number do
    begin
        do I = 1, population size
                select parents;
                create children by crossover;
                perform mutations;
                perform permutations;
        enddo
        one of new designs replaced by the best
        design of the previous generation; ("elitist
        plan" version)
        do I = 1, population size
                evaluate objective function;
                local improvement;
        enddo
        rank designs;
    end
end
```

```
Procedure Local improvement
begin
    search for 5 nearest neighbours in the binary tree;
    construct a least squares approximation to buck-
    ling load;
    while two point interchange not finished do
    begin
        perform two point interchange of stacks;
        compute buckling load approximation;
        adjust objective function for strain failure
        load and contiguous ply constraint;
    end
    replace nominal design by the best interchanged
    design;
end
```

Fig. 2. Pseudocode for genetic algorithm with local improvement

2.1 Efficient retrieval of designs by a binary tree

During the evolution process, populations often contain designs that have also appeared in previous generations. If the calculation of the objective function is expensive, it is worthwhile to keep track of designs to avoid duplicate calculation. The binary tree data structure (Kernighan and Ritchie 1988, pp. 139-143) provides a way to store previous designs which permits efficient search for duplicate designs. The cost of searching a tree grows logarithmically with the size of the tree, and is therefore practical even for very large trees. The tree is used to store all data pertinent to the design: the design string, in-plane lamination parameters, bending lamination parameters, strain failure load, buckling load, and objective function.

```
Procedure Evaluation of objective function using binary tree
begin
    search for the given design in the binary tree;
    if found;
        get objective function value from the binary
        tree;
    else
        search for design having identical in-plane
        lamination parameters;
        if found;
            get strain failure load from the binary
            tree;
        else
        perform in-plane strain analysis;
        endif
        perform buckling analysis;
        adjust objective function for contiguous ply
        constraint;
        add design to the binary tree;
    endif
end
```

Fig. 3. Calculation of objective function with the aid of the binary tree

Figure 3 shows the pseudocode for calculation of the objective function with the aid of the binary tree. After a new generation of design strings is created by the genetic operations, the binary tree is searched for each new design. If the design is found, the objective function value is obtained from the tree without analysis. Otherwise, the tree is searched for designs with identical in-plane lamination parameters and hence identical in-plane strains. If a design with identical in-plane lamination parameters is found, then the strain failure load is obtained from the tree. Otherwise, the strain failure value is obtained by exact analysis. Then the buckling load is calculated, and finally, the objective function value is adjusted for the ply contiguity constraint. This new design and its concomitant data are then inserted in the tree.

2.2 Local improvement

Local improvement is used to improve the performance of combinatorial optimization algorithms by searching for a local optimum in a small neighbourhood of the nominal design. The present work considers as neighbours all the designs obtained by interchanging two stacks in the laminate. In this study, a two point interchange operator is introduced in order to produce neighbouring designs in which two digits of the design string are interchanged. An example of a stack interchange is

$$\text{nominal design: } 1\,2\,3\,\underline{1}\,3\,\underline{2}\,2\,1\,2\,2\,3\,1\,,$$

$$\text{perturbed design: } 1\,2\,3\,\underline{2}\,3\,\underline{1}\,2\,1\,2\,2\,3\,1\,. \qquad (9)$$

The number of all possible different laminates obtained by interchanges is less than or equal to $n(n-1)/2$, where n is the string length.

An example of the distribution of the perturbed designs in the bending lamination parameter space is shown in Fig. 4. The central point corresponds to the nominal design, and the three branches correspond to the three possible exchanges ($1 \leftrightarrow 2, 1 \leftrightarrow 3, 2 \leftrightarrow 3$). The distance from the nominal point depends on the locations of the exchanged stacks.

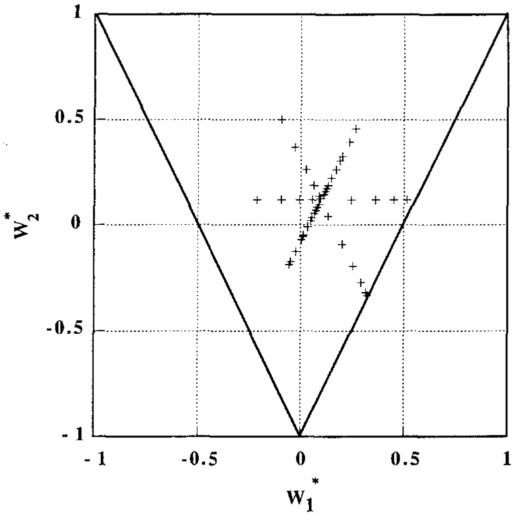


Fig. 4. Example of distribution of perturbed designs, nominal design $[90^\circ_2/(\pm45^\circ/0^\circ_2)_4/\pm45^\circ_2/90^\circ_2]_S$, $W_1^* = 0.09838$, $W_2^* = 0.11806$

The interchange operation does not change the in-plane stiffnesses and the strain failure load. Only the buckling load is influenced by this operation. To reduce the cost of evaluating the objective function for all the possible interchanges, the buckling load at neighbouring designs is estimated by a linear least squares approximation based on the bending lamination parameters,

$$\lambda = \lambda_0 + A\Delta W_1^* + B\Delta W_2^* \,, \qquad (10)$$

where λ_0 is the buckling load of the nominal design, and ΔW_1^* and ΔW_2^* are the changes in the bending lamination parameters (W_1^*, W_2^*) from the nominal design. The coefficients A and B are determined as follows: first, the binary tree is searched for the five nearest neighbours of the nominal design in the Euclidean (W_1^*, W_2^*) plane. Then the coefficients are determined by the least squares fit of the form (10) to the five nearest neighbours.

The approximate buckling load is then used to evaluate the objective function for all the perturbed designs, and the best one is used to replace the nominal one. An example of the effectiveness of this local improvement is shown in Fig. 5. This data was obtained from the 10-th generation of a single genetic optimization run when the population size was set to eight. Only five designs are shown, because the other three designs were repetitions of these designs. The first column of each pair is the objective function value of the nominal design. The second column represents the exact value of the objective function for the best design which replaces that nominal design. The third column represents the approximate value of the best design used by the local improvement procedure to select the best design. Design 2 is not replaced, because this nominal design is better than all of the interchanged designs.

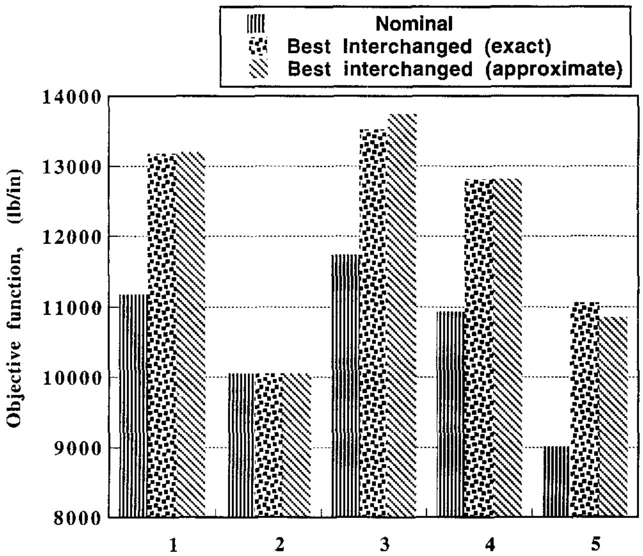


Fig. 5. Example of the effect of local improvement during the 10-th generation

The accuracy of the approximation depends on the distribution of the nearest neighbours. The nominal design of Fig. 4 together with the five nearest neighbours are shown in an expanded view of the bending lamination parameter space in Fig. 6. The accuracy of the approximations for all the perturbed designs of Fig. 4 is shown in Fig. 7. The horizontal axis in Fig. 7 is the distance from the nominal design in the bending lamination plane. The vertical axis is the buckling load normalized by the buckling load of the nominal design $[90^\circ_2/(\pm45^\circ/0^\circ_2)_4/\pm45^\circ_2/90^\circ_2]_S$. When the circles (approximate analysis) overlap the diamonds (exact analysis), the approximation works well. The approximation fails completely when it predicts an increase in buckling load while the true buckling load is actually less than the nominal value. Figure 7 indicates that such failures are likely to occur only when the change in the buckling load from the nominal design is small (in fact no such failure is observed in Fig. 7). Thus, it is unlikely that the design selected as the best among the perturbed designs on the basis of the approximation is worse than the nominal design. Note that the design which has the highest buckling load is not always selected as the best perturbed design, because the selection also considers the contiguous ply constraint.

The distribution of the accuracy of all points from one genetic optimization run is shown in Fig. 8. The horizontal axis is the normalized improved value by approximation $(\lambda_{ap} - \lambda_{nom})/\lambda_{nom}$, and the vertical axis is that by the exact analysis $(\lambda_{ex} - \lambda_{nom})/\lambda_{nom}$. When the approximation works well, points lie close to the 45° line where λ_{ap} is equal to λ_{ex}. When the point lies above the line, the approximation underestimates the buckling load, otherwise, it overestimates the load. If the point is in the first or third quadrants, the approximation predicts correctly whether the interchange increases or decreases the buckling load. When the point is in the second or fourth quadrants, the approximation does not even capture the sense of the effect of the interchange. As can be seen from the figure, very few points lie in the bad quadrants, and these have small normalized values. So, these points are not likely to be selected as the best among the perturbed designs.

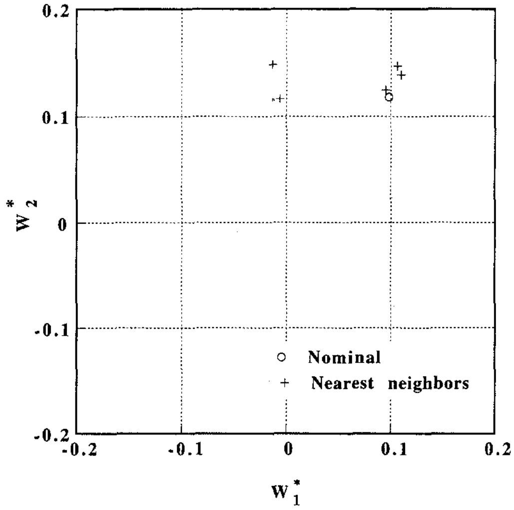


Fig. 6. Example of the distribution of nearest neighbours

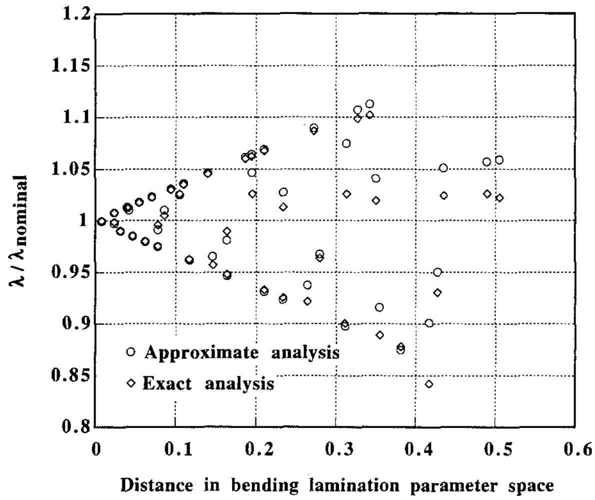


Fig. 7. Exact versus approximate normalized buckling loads. Nominal design and perturbed designs shown in Fig. 4

2.3 Variation of failure and buckling loads in the lamination parameter planes

The objective function for the genetic algorithm combines the buckling and strength failure loads which are determined uniquely by the bending and in-plane lamination parameters, respectively. The variation of the failure load, as calculated from (4)-(7), as a function of V_1^* is shown in Fig. 9 for a fixed value of $V_2^* = -0.5$. It can be seen that there is a singularity at the boundary where $V_2^* = -2V_1^* - 1$. Note that because the laminate is limited to 0°, ± 45° and 90° plies, the lamination parameters V_1^* and V_2^* are confined to a triangular region (similar to the one shown in Fig. 4 for W_1^* and W_2^*). The boundary $V_1^* = -0.25$ represents laminates which do not have any 0° plies. Points near this boundary have few

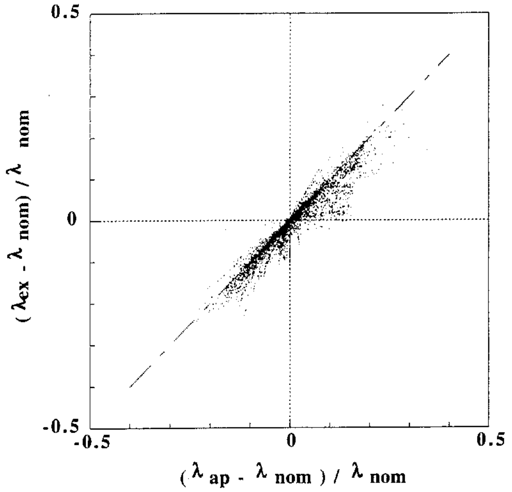


Fig. 8. Distribution of accuracy of all designs from a single run

0° plies, and the strain in these plies is critical, reducing the failure load. When the last 0° ply is eliminated, the failure load increases suddenly because failure of 0° plies need not be considered. Such singularities are known to cause difficulties for continuous optimization algorithms because, unless the ply that causes the singularity is absent, the algorithm would tend to increase that ply's thickness rather than eliminate the ply. Genetic algorithms can circumvent singularities because they permit temporary degradation in performance. Thus a design with two zero stacks can change into one with only a single stack by mutation or crossover, although this reduces the failure load. A subsequent mutation or crossover can then eliminate the remaining stack.

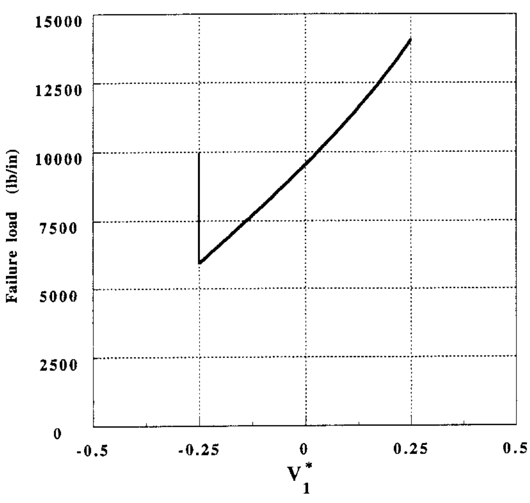


Fig. 9. Variation of the failure load in the lamination parameter space at $V_2^* = -0.5$ for load case 3

The distribution of the buckling load with respect to the bending lamination parameters is shown in Fig. 10. The contour plot for each buckling mode is obtained from (1)-(3), and the contours are given by

$$W_2^* = \Big[\pi^2 h^3 U_2(a^4n^4 - b^4m^4)W_1^* + 12\lambda a^2b^2(b^2m^2N_x + a^2n^2N_y) - \pi^2h^3\Big\{U_1(a^4n^4 + b^4m^4) + 2a^2b^2m^2n^2(U_4 + 2U_5)\Big\}\Big] / \pi^2h^3(a^4n^4 - 6a^2b^2m^2n^2 + b^4m^4)\,. \quad (11)$$

The contours in Fig. 10 are piece-wise linear with each segment corresponding to combinations of the wave numbers m and n that minimize the buckling load.

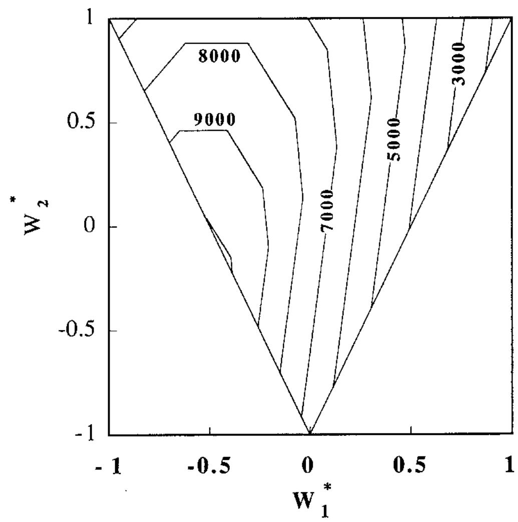


Fig. 10. Variation of the buckling load for load case 3

3 Results

Results were obtained for a 48-ply graphite-epoxy plate with the following material properties: $E_1 = 18.50E6$ psi (127.59 GPa); $E_2 = 1.89E6$ psi (13.03 GPa); $G_{12} = 0.93E6$ psi (6.41 GPa); $\nu_{12} = 0.3$; $t = 0.005$ in (0.127 mm). The ultimate allowable strains are $\varepsilon_1^{ua} = 0.008$, $\varepsilon_2^{ua} = 0.029$ and $\gamma_{12}^{ua} = 0.015$. These allowable strains were reduced by a safety factor of 1.5. The plate has longitudinal and lateral dimensions of $a = 20\,$in (0.508 m) and $b = 5\,$in (0.127 m), respectively. Because of symmetry and the use of 2-ply stacks, the 48-ply laminate is described by a 12-gene string. The genetic algorithm was applied to three load cases with $N_y/N_x = 0.125, 0.25$, and 0.5, called load cases 1, 2, and 3, respectively, where N_x is set to 1.0 lb/in (175 N/m).

In this problem there are many near optimal designs. For this reason, designs that are within a tenth of a percent of the global optimum are accepted as optimal and are called *practical optima* here. To evaluate the efficiency of the algorithm we define a *normalized price*, which is the average number of evaluations (also called *price*) of the objective function divided by the probability of reaching a practical optimum (called *practical reliability*).

Average prices and practical reliabilities were calculated by performing one hundred genetic optimizations for each of the three load cases. The algorithm was considered satisfactory only if the practical reliability was at least 0.8. This means that a single genetic optimization run has at least an 80 percent chance of finding a design within 0.1 percent of the global optimum. The requirement of 0.8 practical reliability was used to determine the stopping criterion. We start 100 optimization runs with the stopping criterion set to 10 generations without improvement. We then increase the stopping criterion until a practical reliability of 0.8 is achieved.

Table 1. Normalized prices and practical reliabilities without local improvement (probability of permutation = 1.0, population size = 8)

Load case	Stopping criterion	Normalized price	Practical reliability
1	19	350	0.84
	44	530	0.99
2	44	1126	0.71
	63	1250	0.78
3	38	832	0.81
	44	963	0.77
Average	44	836	0.823

3.1 Performance of a genetic algorithm without local improvement

Le Riche and Haftka (1993) investigated the performance of a genetic algorithm without local improvement for the same problem with the same three load cases. They found that good performance was obtained with a population size of 8, probability of mutation of 0.01, probability of crossover of 1.0, probability of permutation of 1.0, and a penalty constant in the objective function of 0.9. The same values are used here. The performance of the algorithm depends strongly on the load case, as shown in Table 1.

Table 2a. Optimal designs for load case 1*

Design variable		Load factor λ	
Stacking sequence	Genetic code	Buckling	Failure
$[\pm45_5/0_4/\pm45/0_4/90_2/0_2]_S$	131121122222	14659.583	13518.661
$[\pm45_5/0_4/90_2/0_4/\pm45/0_2]_S$	121131122222	14610.845	13518.661
$[\pm45_2/90_2/\pm45/(\pm45/0_4)_2/\pm45/0_2]_S$	121121122322	14421.311	13518.661
$[\pm45_4/0_2/\pm45/0_4/\pm45/0_4/90_2]_S$	311211212222	14284.145	13518.661
$[\pm45_4/0_2/\pm45/0_4/90_2/0_4/\pm45]_S$	211311212222	14251.656	13518.661
$[\pm45_3/0_2/\pm45_2/0_4/90_2/0_2/\pm45/0_2]_S$	121311221222	14029.490	13518.661
$[90_2/\pm45_2/(\pm45/0_2)_3/0_2/\pm45/0_2]_S$	121121212223	14013.722	13518.661
$[90_2/(\pm45_2/0_2)_2/\pm45/0_4/\pm45/0_2]_S$	121121221223	13831.525	13518.661
$[\pm45_3/(0_2/\pm45)_2/0_4/\pm45/0_2/90_2]_S$	312112121222	13744.604	13518.661

*There are many other practical optimum designs because the strain failure is critical

The table shows the stopping criterion (number of generations without improvement), the normalized price, and the practical reliability for the three load cases without local improvement. For load case 2, the practical reliability did not reach 0.80 even with a stopping criterion of more than 60 generations, but in the other load cases, it reached 0.80 with a much lower stopping criterion. This behaviour was investigated and found to depend on the nature of the optimum for each load case.

Table 2b. Optimal and near-optimal* designs for load case 2

Design variable		Load factor λ	
Stacking sequence	Genetic code	Buckling	Failure
$[\pm45_2/90_2/\pm45_3/0_2/\pm45/0_4/\pm45//0_2]_S$	121121222322	12743.451	12678.777
$[\pm45/90_2/\pm45_4/(0_2/\pm45/0_2)_2]_S$	121121222232	12725.257	12678.777
$[90_2/\pm45_5/(0_2/\pm45/0_2)_2]_S$	121121222223	12674.853	12678.777
$[\pm45_2/90_2/\pm45_3/0_4/(\pm45/0_2)_2]_S$	121211222322	12622.464	12678.777
$[\pm45/90_2/\pm45_4/0_4/(\pm45/0_2)_2]_S$	121211222232	12617.837	12678.777
$[\pm45_2/90_2/\pm45_3/(0_4/\pm45)_2]_S$	211211222322	12592.217	12678.777
$[\pm45/90_2/\pm45_4/(0_4/\pm45)_2]_S$	211211222232	12590.982	12678.777
$[\pm45_3/90_2/\pm45_2/(0_2/\pm45/0_2)_2]_S$	121121223222	12568.942	12678.777

*Only the top three designs are the practical optima

Table 2c. Practical optimal designs for load case 3

Design variable		Load factor λ	
Stacking sequence	Genetic code	Buckling	Failure
$[90_2/\pm45_2/(90_2/\pm45)_2/\pm45_5]_S$	222222323223	9998.198	10398.136
$[90_2/\pm45_2/(90_2/\pm45)_2/\pm45_4/90_2]_S$	322222323223	9997.614	10187.937
$[(90_2/\pm45_2)_2/(90_2/\pm45)_2/\pm45_2]_S$	222323223223	9997.614	10187.937
$[(90_2/\pm45)_2/\pm45_2/(\pm45/90_2/\pm45)_2]_S$	232232222323	9994.836	10187.937
$[\pm45/90_4/\pm45_2/90_2/\pm45_4/90_2/\pm45]_S$	232222322332	9994.836	10187.937
$[(\pm45/90_2)_2/90_2/\pm45_4/90_2/\pm45_2]_S$	223222233232	9994.836	10187.937
$[90_4/\pm45_7/90_2/\pm45_2]_S$	223222222233	9994.694	10398.136
$[90_4/\pm45_6/(\pm45/90_2)_2]_S$	323222222233	9994.110	10187.937
$[(90_2/\pm45)_2/\pm45_3/(90_2/\pm45)_2/\pm45]_S$	223232222323	9994.110	10187.937
$[\pm45/90_4/(\pm45_2/90_2/\pm45)_2/\pm45]_S$	223222322332	9994.110	10187.937
$[90_4/\pm45_7/90_4/\pm45]_S$	233222222233	9990.606	10187.937
$[\pm45/90_4/\pm45_3/90_4/\pm45_4]_S$	222233222332	9990.606	10187.937
$[90_2/\pm45_3/90_4/\pm45/90_2/\pm45_4]_S$	222232332223	9990.606	10187.937

For the optimum design for load case 1, the failure load is critical while the buckling load is not. The failure load depends only on the ratio of total thicknesses associated with the ply orientation angles, and does not depend on the through-the-thickness location of the plies as long as the contiguous ply constraint is satisfied. Consequently, there are many optimum designs, some of which are shown in Table 2a. Therefore, it is easy to reach a practical optimum design and the price of the optimization is low.

For load case 2, there are only three practical optimum designs as shown in Table 2b. Two of these designs have critical failure load and one has critical buckling load. The failure load and the buckling load are very close at the optimum. Changes in the stacking sequence easily degrade either the buckling load or the failure load, so there are few practical optimum designs, and it is more difficult to find one.

For load case 3, only the buckling load is critical for the practical optimum designs as shown in Table 2c. The optimum designs do not have any zero plies, and there is some freedom to change the ratio of the $\pm 45^\circ$ plies and 90° plies without degrading the buckling load significantly.

3.2 Effect of binary tree

The efficiency of the binary tree for the objective function evaluation without local improvement is shown in Table 3. The second column in the table, the price, is the mean (over 100 runs) of the number of designs considered by the algorithm. Since an elitist strategy is adopted in our genetic algorithm implementation, the best design is passed on from one generation to the next, and the reanalysis of this design is not necessary. The elitist price is the mean of the number of analyses required by taking advantage of this property. Finally, the average number of nodes in the tree shows the mean of the number of different designs per optimization. The standard deviations of these means are also given in this table. It is clear from Table 3 that 20 to 50 percent of the analyses can be avoided by keeping track of previous designs.

Table 3a. The efficiency of the binary tree; population size = 8, stopping criterion = 56 generations without improvement probability of permutation = 0.5

Load case	Price	"Elitist" price*	Avg. no. of nodes in tree	% saving	Practical reliability
1	656.0 ± 11	575.0 ± 10	329.5 ± 6	42.7	0.99
2	937.0 ± 25	820.8 ± 22	467.8 ± 12	43.0	0.63
3	937.0 ± 27	820.8 ± 23	394.9 ± 12	51.9	0.74
Average	843.3 ± 15	738.9 ± 13	397.4 ± 7	46.2	0.786

*Price excluding repeated analyses of best design (passes on to next generation with elitist strategy)

Table 3b. The efficiency of the binary tree; population size = 8, stopping criterion = 56 generations without improvement probability of permutation = 1.0

Load case	Price	"Elitist" price*	Avg. no. of nodes in tree	% saving	Practical reliability
1	646.2 ± 12	566.4 ± 11	453.0 ± 9	20.0	0.99
2	907.0 ± 27	794.7 ± 23	631.9 ± 18	20.5	0.71
3	918.5 ± 27	804.7 ± 23	504.4 ± 16	37.3	0.86
Average	823.9 ± 15	721.9 ± 13	529.7 ± 10	26.6	0.853

*Price excluding repeated analyses of best design (passes on to next generation with elitist strategy)

Table 3 also shows that the number of nodes in the binary tree increases with the permutation rate. This indicates that the permutation operator contributes to the creation of new design strings. Table 3 may indicate that if a binary tree is used to avoid reanalyses of designs, the optimum permutation probability may be lowered.

Table 4. Normalized price and practical reliability with local improvement (probability of permutation = 1.0, population size = 8).

Load case	Stopping criterion	Normalized price	Practical reliability
1	10	193	0.80
	50	520	1.00
2	22	435	0.81
	50	720	0.96
3	50	1646	0.46
	63	2209	0.45
Average	50	813	0.807

3.3 Effect of local improvement

The performance of the genetic algorithm with the local improvement using the same genetic parameters as in Table 1 is shown in Table 4 for the three load cases. For load cases 1 and 2, local improvement worked very well, especially for the load case 2, where the performance improved by about a factor of three.

For load case 3, however, local improvement made the performance worse (see tables). We investigated the reasons for the poor performance of the algorithm for load case 3. Table 5 shows frequently obtained designs by local improvement for this load case. In comparison with the practical optima found without the local improvement (see Table 2c), these designs all have 0° plies near the midplane. The rest of the stacking sequence is the same as those of the practical optima which do not have any 0° plies.

Upon further investigation the problem was found to be due to the singularity of the optimum for load case 3. As discussed earlier, the failure load exhibits a singularity at the boundary of the lamination diagram (see Fig. 9) where the laminate does not have any 0° plies. This situation is illustrated in Fig. 11 which shows the effect of pushing the 0° plies toward the midplane and replacing them with $\pm 45^\circ$ stacks (indicated by the laminate definition on the horizontal axis) when they reach the midplane. As can be seen in Fig. 11, the buckling load increases monotonically during this operation. However, the failure load decreases as we reduce the number of 0° plies until they are all eliminated, when it jumps up. Recall that this happens because we no longer need to enforce strain constraints in the 0° plies.

Crossover and mutation are two genetic operators that can potentially get rid of the undesirable layers from the laminate. Consider, for example, the following crossover scenario where both parents have 0° plies occupying different positions in the strings:

$$\text{parent1}\ \ 22/11, \quad \text{parent2}\ \ 11/22, \text{child}\ \ 22\,22\,. \tag{12}$$

Local improvement interferes with this mechanism because it will move 0° plies towards the midplane, where they have the least detrimental effect on the buckling load, in all members of the population. For example, due to local improvement parent 1 in the above design is likely to be changed to

Table 5. Frequently obtained designs for load case 3 (with local improvement)

Design variable		Load factor λ	
Stacking sequence	Genetic code	Buckling	Failure
$[90_4/\pm 45_6/(\pm 45/0_2)_2]_S$	121222222233	9910.385	10251.197
$[(90_2/\pm 45)_2/\pm 45_3/90_2/0_4/90_2/0_2]_S$	131132222323	9803.273	10787.530
$[\pm 45/90_2/(90_2/\pm 45_2)_2/0_4/90_2/0_2]_S$	131122322332	9803.273	10787.530
$[(\pm 45/90_2)_4/0_4/\pm 45/0_2]_S$	121132323232	9803.273	10787.530
$[90_2/\pm 45_2/(90_2/\pm 45)_2/\pm 45/0_4/\pm 45/0_2]_S$	121122323223	9803.105	11404.473
$[(\pm 45/90_2)_4/0_4/90_2/0_2]_S$	131132323232	9801.937	10193.772
$[90_4/\pm 45_6/0_4/90_2/0_2]_S$	131122222233	9801.119	11404.473
$[90_2/\pm 45_2/(90_2/\pm 45)_2/\pm 45/0_4/90_2/0_2]_S$	131122323223	9799.017	10787.530
$[\pm 45/90_4/\pm 45_3/90_4/0_4/\pm 45/0_2]_S$	121133222332	9795.513	10787.530
$[90_2/\pm 45_3/90_4/\pm 45/90_2/0_4/\pm 45/0_2]_S$	121132332223	9795.513	10787.530
$[(90_2/\pm 45)_2/\pm 45_2/90_2/\pm 45/0_4/\pm 45/0_2]_S$	121123222323	9792.593	11404.473
$[\pm 45/90_4/\pm 45_3/90_4/0_4/90_2/0_2]_S$	131133222332	9791.425	10193.772
$[90_2/\pm 45_3/90_4/\pm 45/90_2/0_4/90_2/0_2]_S$	131132332223	9791.425	10193.772

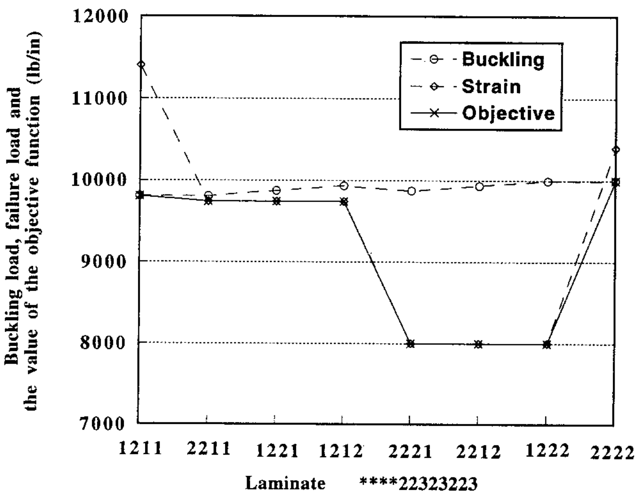


Fig. 11. The effect of reducing the number of zero plies near the optimum laminate for load case 3

$$22\,11 \longrightarrow 12\,12\,, \tag{13}$$

Now the crossover cannot eliminate both 0° plies, because some of the 0° plies occupy the same location in both laminates and, therefore, appear in both children.

To ameliorate this situation, we seeded the initial population with "no-zero-ply" (NZP) designs. With the population sized fixed at 8, we tried two, four, or six initial NZP designs. This corresponds to 25%, 50%, or 75% of the population.

The results obtained by this NZP seeding are shown in Table 6 for load case 3 both with and without the local improvement and for two permutation probabilities. The performance is seen to improve both with and without local improvement as the number of NZP designs increases in the initial population. The effect on the average of all three load cases is shown in Table 7. The practical reliability reached 80% at a very small normalized price.

Table 6. Normalized price near 0.80 practical reliability for load case 3 starting with seeded nonzero degree plies in initial population

No local improvement (permutation probability = 0.50)				
Initial population		Stopping criterion	Practical reliability	Normalized price
Normal	Nonzero			
8	0	56	0.80	1181
6	2	32	0.83	566
4	4	22	0.85	378(147)*
2	6	16	0.80	280(104)*
Local improvement (permutation probability = 0.50)				
8	0	63	0.30	2997
6	2	63	0.73	1161
4	4	16	0.81	284(131)*
2	6	16	0.85	218(121)*
No local improvement (permutation probability = 1.00)				
8	0	38	0.81	832
6	2	25	0.83	496
4	4	13	0.83	249(134)*
2	6	10	0.80	173(97)*
Local improvement (permutation probability = 1.00)				
8	0	63	0.45	2209
6	2	56	0.83	943
4	4	13	0.80	256(144)*
2	6	10	0.81	190(109)*

* Average number of nodes in binary tree

Table 7. Normalized price near 0.80 practical reliability for average of the three load cases starting with seeded nonzero degree plies in initial population

No local improvement (permutation probability = 0.50)				
Initial population		Stopping criterion	Practical reliability	Normalized price
Normal	Nonzero			
8	0	56	0.827	1034
6	2	44	0.810	821
4	4	32	0.820	602(250)*
2	6	38	0.827	734(292)*
Local improvement (permutation probability = 0.50)				
8	0	63	0.767	968
6	2	19	0.810	362
4	4	13	0.820	263(135)*
2	6	16	0.860	287(148)*
No local improvement (permutation probability = 1.00)				
8	0	44	0.823	836
6	2	50	0.857	853
4	4	38	0.833	665(365)*
2	6	38	0.827	737(362)*
Local improvement (permutation probability = 1.00)				
8	0	50	0.807	813
6	2	19	0.827	353
4	4	16	0.817	307(179)*
2	6	16	0.840	305(188)*

* Average number of nodes in binary tree

The comparison in Table 7 shows that while the NZP seeding helps also without local improvement, its effect on the local improvement procedure is dramatic. With half of the initial population seeded NZP, the local improvement procedure reduced the price of the GA by better than a factor of two.

Table 8. Normalized price near 0.80 practical reliability for the average of the three load cases starting with 4 NZP and 4 normal designs in the initial population

(a) No local improvement			
Permutation probability	Stopping criterion	Practical reliability	Normalized price
0.0	63	0.283	3278(115)*
0.5	32	0.820	602(250)*
1.0	38	0.833	665(365)*
(b) Local improvement			
0.0	25	0.807	439(106)*
0.5	13	0.820	263(135)*
1.0	16	0.817	307(179)*

* Average number of nodes in binary tree

We also checked the need for permutation when seeding and local improvement are used. The results for 50% NZP seeding and three permutation probabilities are given in Table 8. It is seen that without local improvement permutation is critical for achieving the desired reliability. However, with local improvement it may not be necessary. Furthermore, if the performance of the algorithm is based on the number of different designs (nodes in the binary tree), no permutation could be the most efficient choice.

Finally, we checked the case where all three load cases are applied at once. Because the buckling load depends roughly on $N_x + N_y$, we scaled all three load cases so that $N_x + N_y = 1.125$ (the value for load case 1). This resulted in scaling load case 2 by 0.9 and load case 3 by 0.75.

The application of all three load cases simultaneously is handled by choosing the failure load to be the minimum of the three individual failure loads (each of which is the minimum of the buckling loads and strength failure loads). This procedure leads to a compromise optimum design where several failure modes are critical. Table 9 shows the optimal and near optimal designs for this case. It is seen that while the buckling load for load case 3 is the critical load, the other two buckling loads as well as the strength failure load for load case 1 are near critical and constrain the design. It is also seen that the number of practical optima is only four.

The effect of the multiple constraints and the small set of practical optima had a dramatic effect on the performance of the algorithm without local improvement. To achieve 80 percent reliability, the population size and stopping criterion needed to be increased substantially, as can be seen from Table 10. With local improvement, on the other hand, the effect on performance was rather small, as can be seen from Table 11. Comparing Tables 10 and 11, for this case, local improvement reduced the number of analyses by more than an order of magnitude.

For the present problem, the calculations of the buckling load and the strength failure load involve the use of simple algebraic formulae so that it takes longer to search for the nearest neighbours and construct the approximation than to evaluate the objective function. Also, the binary tree requires a large amount of memory to keep track of all the designs. Therefore, the binary tree is not cost-effective for this problem. Depending on the load cases, local improvement may or may not be worthwhile. However, these procedures have high potential and wide applicability to problems with a very expensive objective function.

Table 9. Global optimum, practical optima, and near-optimal designs for multiple load case ($N_x = 1.0$, $N_y = 0.125$; $N_x = 0.9$, $N_y = 0.225$; $N_x = 0.75$, $N_y = 0.375$)

Design variables		Objective	Failure load/Buckling load		
Stacking sequence	Genetic code	function	Load case 1	Load case 2	Load case 3
$[90_4/\pm45_3/0_4/\pm45/0_4/90_2/0_2]_S$	131121122233	12080.62	13090.97 12925.51	15026.57 12925.51	19309.07 **12080.61**
$[(90_2/\pm45)_2/\pm45/(0_4/90_2)_2/0_2]_S$	131131122323	12079.84	12735.80 12921.83	14485.38 12921.83	18244.96 **12079.84**
$[(90_2/\pm45)_2/\pm45/0_4/90_2/0_4/\pm45/0_2]_S$	121131122323	12074.07	13090.97 12947.57	15026.57 12947.57	19309.07 **12074.07**
$[90_4/\pm45_3/(0_4/\pm45)_2/0_2]_S$	121121122233	12071.19	13363.93 12951.25	15536.32 12951.25	20532.05 **12071.19**
$[(\pm45/90_2)_2/90_2/0_4/\pm45/0_4/90_2/0_2]_S$	131121133232	12059.70	12735.80 12965.95	14485.38 12965.95	18244.96 **12059.70**
$[\pm45/90_4/\pm45/90_2/(0_4/\pm45)_2/0_2]_S$	121121132332	12052.58	13090.97 12793.16	15026.57 12793.16	19309.07 **12052.58**
$[\pm45/90_4/\pm45/90_2/0_4/\pm45/0_4/90_2/0_2]_S$	131121132332	12047.13	12735.80 12767.43	14485.38 12767.43	18244.96 **12047.13**
$[(\pm45/90_2)_2/(90_2/0_4)_2/\pm45/0_2]_S$	121131133232	12047.13	12735.80 12767.43	14485.38 12767.43	18244.96 **12047.13**
$[90_2/\pm45_2/90_4/(0_4/\pm45)_2/0_2]_S$	121121133223	12043.24	13090.97 12749.04	15026.57 12749.04	19309.07 **12043.24**
$[(\pm45/90_2)_2/(90_2/0_4)_2/90_2/0_2]_S$	131131133232	12041.68	12328.82 12741.69	13923.94 12741.69	17276.89 **12041.68**
$[90_4/\pm45_3/0_4/90_2/0_4/\pm45/0_2]_S$	121131122233	12038.57	13090.97 12726.99	15026.57 12726.99	19309.07 **12038.57**
$[90_2/\pm45_2/90_4/0_4/\pm45/0_4/90_2/0_2]_S$	131121133223	12037.79	12735.80 12723.31	14485.38 12723.31	18244.96 **12037.79**
$[\pm45/90_4/\pm45/90_2/0_4/90_2/0_4/\pm45/0_2]_S$	121131132332	12005.08	12735.80 12568.90	14485.38 12568.90	18244.96 **12005.08**
$[(90_2/\pm45)_2/90_2/(0_4/\pm45)_2/0_2]_S$	121121132323	12001.19	13090.97 12550.52	15026.57 12550.52	19309.07 **12001.09**

* Only the top four designs are the practical optima

Table 10. Effect of the population size on the multiple load case without local improvement from 50 optimization runs ($N_x = 1.0$, $N_y = 0.125$; $N_x = 0.9$, $N_y = 0.225$; $N_x = 0.75$, $N_y = 0.375$)

Pop. size	Stopping criterion	Price	Pract. reliability	Norm. price	No. of nodes
(a) Permutation probability = 0.50					
8	500	5592.0 ± 168	0.64	8732.50	2189.2 ± 73
12	334	6260.4 ± 240	0.78	8026.15	2729.6 ± 106
16	250	5902.1 ± 228	0.84	7026.29	2697.6 ± 101
20	200	6050.0 ± 213	0.88	6875.00	2981.6 ± 105
24	167	5804.2 ± 197	0.76	7637.05	2965.1 ± 100
(b) Permutation probability = 1.00					
8	500	5748.0 ± 149	0.78	7369.23	3633.0 ± 95
12	334	6054.5 ± 238	0.70	8649.26	4226.2 ± 163
16	250	6345.9 ± 236	0.76	8349.89	4684.7 ± 174
20	200	6919.6 ± 292	0.84	8237.62	5347.6 ± 214
24	167	6795.4 ± 243	0.68	9993.18	5439.5 ± 188

* Stopping criterion times population size was set to around 4000 for all cases

4 Concluding remarks

We introduced two approaches for reducing the number of analyses required by a genetic algorithm for the stacking sequence optimization of composite plates. The binary tree data structure is effective for avoiding the reanalysis of designs which appeared in previous generations. A local improvement scheme considered all designs that can be obtained by interchanging two stacks of plies in the nominal designs and estimated the buckling load of these designs using a linear approximation based on lamination parameters. This local improvement was found to substantially reduce the cost of the genetic optimization. We also found that to avoid difficulties with singular optima, the initial population needs to be seeded with designs containing no zero degree plies.

Acknowledgements

This work was supported in part by NASA grants NAG-1-168 and NAG-1-643, Department of Energy Grant DE-FG05-88ER25068, Air Force Office of Scientific Research Grant F49620-92-J-0236, and a scholarship from the Osaka Foundation of International Exchange. The authors wish to thank Dr. S. Nagendra for his helpful suggestions.

Table 11. Effect of the population size on the multiple load case with local improvement ($N_x = 1.0$, $N_y = 0.125$; $N_x = 0.9$, $N_y = 0.225$; $N_x = 0.75$, $N_y = 0.375$)

Pop. size	Stopping criterion	Price	Prac. reliability	Norm. price	No. of nodes
(a) Permutation probability = 0.00					
8	44	595.68 ± 14.7	0.81	735.41	164.75 ± 3.6
12	21	437.76 ± 10.6	0.85	515.01	175.38 ± 3.9
16	10	342.08 ± 8.4	0.82	417.17	183.17 ± 3.9
20	8	370.20 ± 9.0	0.86	430.47	216.06 ± 4.1
24	6	344.40 ± 7.3	0.84	410.00	224.48 ± 3.9
28	5	393.12 ± 9.0	0.83	473.64	263.67 ± 5.3
(b) Permutation probability = 0.25					
8	38	519.36 ± 13.1	0.83	625.73	229.82 ± 5.3
12	15	377.88 ± 9.5	0.82	460.83	211.98 ± 5.0
16	10	368.16 ± 9.9	0.82	448.98	233.78 ± 5.7
20	8	369.40 ± 7.8	0.87	424.60	255.07 ± 4.8
24	7	390.72 ± 9.9	0.84	465.14	280.00 ± 6.2
(c) Permutation probability = 0.50					
8	44	596.32 ± 14.0	0.81	736.20	330.04 ± 7.2
12	15	372.12 ± 9.0	0.80	465.15	248.42 ± 5.8
16	11	379.20 ± 8.4	0.82	462.44	273.92 ± 5.3
20	8	370.40 ± 8.3	0.82	451.71	286.47 ± 6.1
24	8	439.92 ± 10.8	0.85	517.55	339.67 ± 7.8
(d) Permutation probability = 0.75					
8	44	661.52 ± 18.8	0.82	806.73	424.55 ± 11.6
12	30	650.16 ± 17.9	0.85	764.89	454.05 ± 11.7
16	16	507.68 ± 11.4	0.84	604.38	390.36 ± 8.5
20	10	482.00 ± 11.3	0.85	567.06	387.24 ± 8.4
24	9	494.16 ± 11.7	0.84	588.29	410.03 ± 9.1
(e) Permutation probability = 1.00					
8	63	865.76 ± 23.4	0.81	1068.84	599.28 ± 15.6
12	30	672.48 ± 18.4	0.82	820.10	516.91 ± 13.5
16	16	546.88 ± 14.0	0.85	643.39	452.77 ± 11.1
20	13	528.40 ± 11.5	0.80	660.50	448.48 ± 9.1
24	11	558.00 ± 12.0	0.81	688.89	485.13 ± 10.0

References

Byrd, R.H.; Dert, C.; Rinnoy Kan, A.H.G.; Schnabel, R.B. 1986: Concurrent stochastic methods for global optimizations. *Tech. Rep., CU-CS-338-86*, Computer Sci. Dept., Univ. Colorado

Eskow, E.; Schnabel, R.B. 1988: Mathematical modeling of a parallel global optimization algorithm. *Tech. Rep., CU-CS-395-88*, Computer Sci. Dept., Univ. Colorado

Haftka, R.T.; Walsh, J.L. 1992: Stacking-sequence optimization for buckling of laminated plates by integer programming. *AIAA J.* **30**, 814-819

Hajela, P. 1990: Genetic search – an approach to the nonconvex optimization problem. *AIAA J.* **28**, 1205-1210

Hajela, P.; Lin, C.Y. 1992: Genetic search strategies in multicriteria optimal design. *Struct. Optim.* **4**, 99-107

Holland, J.H. 1975: *Adaptation in natural and artificial systems.* Ann Arbor: The University of Michigan Press

Kernighan, B.W.; Ritchie, D.M. 1988: *The C programming language* (second edition). Englewood Cliffs: Prentice Hall

Le Riche, R.; Haftka, R.T. 1993: Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm. *AIAA J.* **31**, 951-956

Mesquita, L.; Kamat, M.P. 1987: Optimization of stiffened laminated composite plates with frequency constraints. *Eng. Opt.* **11**, 77-88

Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. 1953: Equation of state calculations by fast computing machines. *J. Chem. Physics* **21**, 1087-1092

Miki, M.; Sugiyama, Y. 1991: Optimum design of laminated composite plates using lamination parameters. *Proc. AIAA/ ASME/ASCE/AHS/ASC 32nd Structures, Structural Dynamics and Materials Conf.* (held in Baltimore MD), Part 1, pp. 275-283

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

Rao, S.S.; Pan, T.S.; Venkayya, V.B. 1990: Optimal placement of actuators in actively controlled structures using genetic algorithms. *AIAA J.* **28**, 942-943

Rechenberg, J. 1965: Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment*, Liberty translation, 1122, Farnborough, England

Rinnoy Kan, A.H.G.; Timmer, G.T. 1985: A stochastic approach to global optimization. In: Boggs, P.; Byrd, R.; Schnabel, R.B. (eds.) *Numerical optimization 1984*, pp. 245-262. Philadelphia: SIAM

Rinnoy Kan, A.H.G.; Timmer, G.T. 1984: Stochastic methods for global optimization. *American J. Math. Management Sci.* **4**, 7-40

Schmit, L.A.; Farshi, B. 1977: Optimum design of laminated fiber composite plates. *Int. J. Num. Meth. Eng.* **11**, 623-640

Tovey, C.A. 1985: Hill climbing with multiple local optima. *SIAM J. Alg. Disc. Meth.* **6**, 384-393

Tovey, C.A. 1986: Low order polynomial bounds on the expected performance of local improvement algorithms. *Mathematical Programming* **35**, 193-224

Received Aug. 24, 1993
Communicated by J. Sobieski

Preliminary Announcement

First World Congress of Structural and Multidisciplinary Optimization
Congress of the International Society of Structural and Multidisciplinary Optimization (ISSMO) and the German Research Foundation (DFG)
28 May – 2 June, 1995 (change of dates)
Goslar, Lower Saxony, Germany

Deadlines for Abstracts and Registration will be announced soon. Due to substantial support from the German Research Foundation and other sources, registration fees and accommodation charges will be subsidized.

Research papers from all fields of structural or multidisciplinary optimization are invited.

For further information contact one of the joint chairmen.

Joint Chairmen

Prof. G. Rozvany	Prof. N. Olhoff
FB 10	Inst. of Mechanical Engrg.
Universität-GH-Essen	University of Aalborg
Postfach 10 37 64	Pontoppidanstraede 101
D-45117 Essen	DK-9220 Aalborg East
Germany	Denmark
Tel.: ..49 201 183 2795 ..49 201 183 2695	Tel.: ..45 98 15 85 22 ext. 3513
Fax: ..49 201 183 2695	Fax: ..45 98 15 14 11

Co-Chairmen

Prof. R.T. Haftka	Prof. Z. Mroz
Dept. of Aerospace	Inst. of Fundamental
& Ocean Engineering	Technological Research
Virginia Polytechnic	Polish Academy of Sciences
& State University	Swietokrzyska 21
Blacksburg, VA 24061	PL-00-049 Warsaw
USA	Poland

Dr. J. Sobieski
Structural Dynamics Division
MS 242
NASA Langley Research Center
Hampton, VA 23681-0001
USA