Structural Optimization 15, 221–229 © Springer-Verlag 1998

The genetic algorithm applied to stiffness maximization of laminated plates: review and comparison

E. Potgieter and N. Stander
Structural and Multidisciplinary Optimization Research Group, Department of Mechanical and Aeronautical Engineering, University of Pretoria, South Africa

Abstract The design of laminated structures is highly tailorable owing to the large number of available design variables, thereby requiring an optimization method for effective design. Furthermore, in practice, the design problem translates to a discrete global optimization problem which requires a robust optimization method such as the genetic algorithm. In this paper, the genetic algorithm, based on the real variable coding, is applied to the strain energy minimization of rectangular laminated composite plates. The results for both a point load and uniformly distributed load compare well with those achieved using trajectory methods for continuous global optimization.

1 Introduction

The design of laminated plates is highly tailorable due to the large selection of variables introduced by the constitution of the laminated composite. These include, different types of fiber and matrix combinations, layer thickness, layer number and fiber orientation. Due to the design complexity, designers have resorted to optimization methods to achieve best designs (Haftka and Gürdal 1992).

As a complicating factor, studies of the laminated composite design functions have revealed the presence of multiple optima due to harmonic functions included in the formulation (Haftka and Gürdal 1992). To find the global optimum design, a number of optimization approaches have been developed in recent years. These methods can be broadly divided into two categories namely, deterministic and stochastic methods. A summary of the methods is presented by Arora *et al.* (1995) who emphasize that it is necessary to select an optimization method according to the characteristics of the specific problem and the results desired. This could include the nature of the design variables (discrete or continuous or a combination of both), the nature of the objective function (could be nondifferentiable), or the nature of the desired result (all or several of the local minima could be required). A stochastic random multistart global optimization method used in the present paper for comparison is based on a trajectory approach and has been applied to minimize the strain energy for laminated plates by Kam and Snyman (1991) and shells modelled in finite elements by Groenwold *et al.* (1996).

A factor to be considered is that some of the variables such as the ply thicknesses and orientation angles may be discrete in which case the design problem translates to a discrete programming problem. Well-known methods used to solve discrete programming problems include the *genetic algorithm* (GA) (De Jong 1975), a stochastic search method and implicit enumeration procedure, and the *simulated annealing* method (Kirkpatrick *et al.* 1983), which is a random method. An advantage of these methods above gradient-based methods is that they do not require continuity or differentiability of the objective function.

Genetic algorithms date back about two decades to the research of De Jong (1975) and Holland (1975) in the area of genetic and adaptive systems. Since then, the method they proposed has been used in a variety of fields [summarized by Goldberg (1989)] such as biology, computer science and social sciences. More recently, genetic algorithms were introduced in engineering design. These algorithms present alternative methods for optimization with the advantages of being able to solve nonlinear and nonconvex design problems, as is discussed by Hajela and Shieh (1990). The development has greatly extended discrete engineering design capabilities, in particular for the design of laminated composite materials, see e.g. Haftka and Gürdal (1992), Hajela and Shih (1989), Le Riche and Haftka (1993), Nagendra *et al.* (1993) and Soremekun *et al.* (1996). As most of these references relate to aircraft design, a typical problem addressed is that of laminated panel buckling. Following the work of Galante (1996), Groenwold *et al* (1997) applied the GA to discrete truss sizing by applying it in the region of the continuous optimum.

The current discussion deals with the implementation of the GA to solve the discrete programming laminated plate problem. The optimization problem is formulated as the minimization of the bending strain energy in the plate while being subjected to either a central point or uniformly distributed load. The results are compared to the results obtained using the above-mentioned trajectory method for continuous global optimization as presented by Kam and Snyman (1991) and Groenwold *et al.* (1996).

2 Preliminary examples

To emphasize the importance of the selection of a suitable optimization method the following two-dimensional examples will be solved by using the GA and the BFGS (Broyden-Fletcher-Goldfarb-Shanno) (Press *et al* 1988) variable metric unconstrained optimization methods, respectively. Three

functions with diverse characteristics are chosen for the investigation and are listed in Table 1 and depicted in Figs. 1, 2 and 3. Function f_1 is minimized while functions f_2 and f_3 are maximized. The symbols for the GA are defined as follows: n is the population size, Inc is the increment size between two adjacent discrete values, CI is the number of convergence iterations and MI is the maximum number of iterations that the algorithm may execute. The symbols in Table 2 are defined as follows: $(x_{\text{opt}}, y_{\text{opt}})$ is the optimal solution to the function, f_{opt} is the optimal function value, FE is the number of function evaluations executed to obtain the optimum, DE is the number of function gradient evaluations executed, $ITER$ is the number of times the algorithm was executed and $MAX\ FE$ is the maximum number of combinations of discrete values in the design space for the GA.

Table 1. Functions f_1, f_2 and f_3

Function	Design space limits
$f_1(x,y) = x^2 + y^2$	$-5.12 \leq x, y \leq 5.12$
$f_2(x,y) = \text{integer}(x) + \text{integer}(y)$	$-5.12 \leq x, y \leq 5.12$
$f_3(x,y) = 21.5 + x\sin(4\pi x) + y\sin(20\pi y)$	$-3.0 \leq x \leq 12.1$
	$4.1 \leq y \leq 5.8$

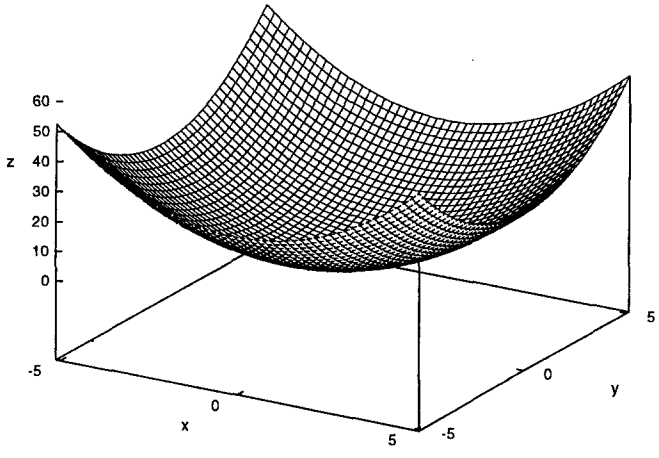

Fig. 1. Function f_1

The first example is represented by a continuous, unimodal function. The GA data is assumed to be $n = 20$, $Inc = 0.001024$, $CI = 100$ and $MI = 1000$. For the BFGS method the starting vector is chosen to be $(4, 5.12)$. The results obtained by the two optimization methods for function f_1 are listed in Table 2. Function f_1 has a minimum of 0.0 at $(0, 0)$.

Considering function f_2 depicted in Fig. 2, it is evident that gradient calculations are not possible and it is thus imperative to use a method based on response values only. This

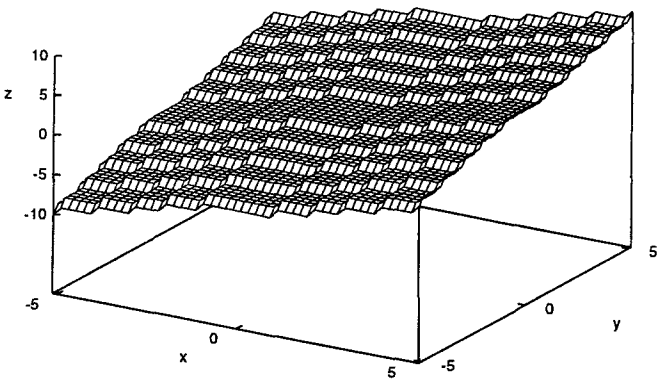

Fig. 2. Function f_2

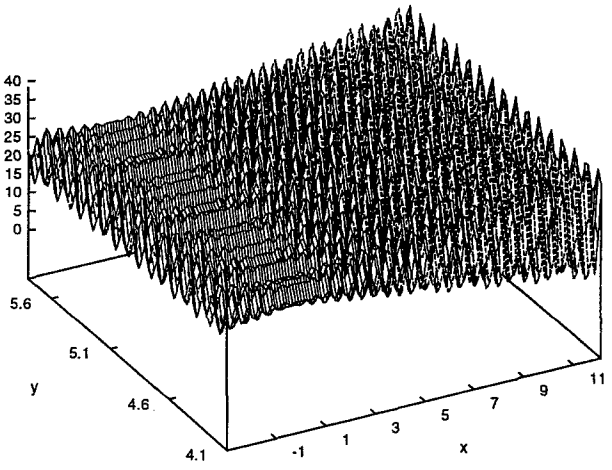

Fig. 3. Function f_3

function was maximized using the GA with its parameters set to: $n = 20$, $Inc = 0.001024$, $CI = 100$ and $MI = 1000$. The results are listed in Table 2.

By considering function f_3 depicted in Fig. 3 it is evident that the function is highly multimodal with a large number of local maxima/minima. A gradient-based method such as a trajectory method (see e.g. Arora 1995) can be used to optimize this function. On the other hand, a GA could be used but then only an approximate solution in the vicinity of the global optimum will be found since it uses discrete values. It is thus conceivable that the GA could be used to find a starting vector near the global optimum for the gradient-based method. This concept was followed with the GA parameters set to: $n = 20$, $Inc_x = 0.0151$, $Inc_y = 0.0017$, $CI = 100$ and $MI = 1000$. The starting vector for the BFGS method was found to be $(11.6319, 5.7252)$. The final results are listed in Table 2.

Table 2 shows that the gradient-based solution is more accurate since it uses continuous functions. The total number of function evaluations for the hybrid method (f_3) was 1920+ 67 = 1987, which by comparison with that of function f_1 for the GA is very small.

In conclusion, the GA performance seems to favour the highly multimodal functions, therefore confirming the results

Table 2. Optimization results for functions f_1, f_2 and f_3

Function	Algorithm	x_{opt}	y_{opt}	f_{opt}	*FE*	*DE*	*ITER*	*MAX FE*
f_1	GA	−0.002048	−0.031744	0.001	4420	–	221	1×10^8
	BFGS	0.0	0.0	0.0	31	2	2	–
f_2	GA	5.12	5.03706	10	160	–	8	1×10^8
	BFGS	–	–	–	–	–	–	–
f_3	GA	11.6319	5.7252	38.8129	1920	–	96	1×10^6
	BFGS	11.625556	5.72504	38.8503	+67	3	3	–

Table 3. Refined solutions for the central point load of GA with $N = 25$

	$b/a = 1$			$b/a = 2$		
NL	Design	$\bar{w}$	FE	Design	$\bar{w}$	FE
4	$[-45/45]_S$	1.915	180	$[-10/50]_S$	4.757	40
8	$[45/-45_3]_S$	1.513	2820	$[-25/30/35/30]_S$	4.304	2080
16	$[45/-45_2/45/-45/$ $45/20/15]_S$	1.505	2520	$[30/-30/25/-25/$ $-35/-25/-5/-25]_S$	4.285	2960

of Le Riche and Haftka (1993). Due to its discrete nature, it is however not as accurate as the gradient-based methods.

3 Genetic algorithm

The GA is based on the natural phenomena and processes that occur among individuals of populations in species in their strive to progress or survive in their natural environment or objective. This method selects the better designs according to the feedback obtained from the objective function evaluations.

The natural phenomena simulated by the GA are implemented as genetic operators which process a set of design alternatives (or population) during each consecutive generation. The operators consist of three standard processes namely reproduction (natural objective selection), mating (crossover) and mutation (Goldberg 1985; and Michalewicz 1992). The reproduction process chooses some of the design vectors (chromosomes) according to their objective function performance as future candidates to produce offspring. The mating process exchanges design variable (gene) information between two of the selected design vectors and thus constructs two new design vectors. The mutation process alters the design vectors by randomly substituting some of the variables in the vector and thus provides a mechanism which not only explores the design space but also inherits some of the information of the previous design vectors.

3.1 Coding of design variables

The three processes of the GA operate at the unit (gene) level of a string of units (chromosome) and in order to represent a specific design variable value, which could be a real number or an integer, as a string of units, the value has to be transformed to another alphabet. Alternatively, if the design vector is long and the design space small, the transformation is not necessary since the design vector will comply with the schema theory discussed by Goldberg (1985) and Michalewicz (1992). This coding method is referred to as the real variable coding.

3.1.1 Linear discrete coding. The phrase *linear* refers to a constant increment between consecutive values and the *discrete coding* to the discrete nature of the values. The values could be decimal integers, fractions or a combination of both. Decimal integers are coded as binary strings by using a division by 2 and a multiplication of the *rest* by 2 to either yield a 0 or a 1. Decimal fractions are coded with a linear mapping transformation. It is important that all the possible discrete values are represented by a unique character or value in the coded set.

The maximum number of values representable by using an alphabet k of cardinality (number of characters) x and with string length ℓ is x^ℓ and thus for different alphabets is $n_c = {x_1}^{\ell 1} \cdot {x_2}^{\ell 2} \cdot {x_3}^{\ell 3} \cdots {x_m}^{\ell m}$ where x_1 to x_m are m alphabets with m different cardinalities x_i. The most basic representation is where the discrete set is the coding alphabet. This representation exactly matches the number of values in the discrete set whereas other representations sometimes do not.

If the design space for a specific variable is a continuous interval in which the increment size between discrete values can be selected, then it is convenient to introduce the resolution factor $\pi = (U_{\max} - U_{\min})(n_c - 1)$, where $U_{\max}$ and $U_{\min}$ are the upper and lower limits on the interval, respectively. It is used in the decoding of the coded values (after optimization) to the real values using the linear mapping as follows: real value $= \pi \cdot$ decimal (binary string) $+U_{\min}$. For example, if $U_{\min} = 3$ and $U_{\max} = 78$ and $\ell = 4$ and if a binary coding is used, $\pi = (78 - 3)/(2^4 - 1) = 5$, and if it is assumed that the binary string [1010] was obtained, then the real value is computed to be $5 \cdot$ decimal $(1010) + 3 = 53$. Another possible example is, say that $U_{\min} = 1.74$ and $U_{\max} = 2.25$ and $\ell = 8$ then $\pi = 0.002$ and assume that the binary string [11011100] is an optimum, then the real value $= 0.002 \cdot$ decimal $(11011100) + 1.74 = 2.18$.

In order to comply with the predictions of the schema theory, it is better to choose a coding of low cardinality but as was pointed out above it is not always possible to map the exact number of values in the discrete set (having no redundant coded strings). Examples of coding methods different from the binary coding implementations are discussed by Le Riche and Haftka (1993), Nagendra *et al.* (1993), Soremekun *et al.* (1996), Kogiso *et al.* (1994) and Ponslet *et al* (1993).

3.1.2 Random discrete coding. The phrase *random* refers to the random nature of the values inside the discrete set. There exists no relation between consecutive values in the discrete set. If the discrete set is small, it is easy to use a low cardinality coding which would represent the set of random discrete values, for example as $0 = [00]$, $+45 = [01]$, $-45 = [10]$ and $90 = [11]$ (Le Riche and Haftka 1993; Soremekun *et al* 1996).

3.1.3 Multivariate coding. If the objective function is a multivariate function, as for example the strain energy in a laminated plate with respect to its fiber orientations, then the design vector can be represented by a concatenation of the different or same coding in the fiber orientation of each layer. For example, if the design vector is a three-dimensional vector with decimal values as $A = [25, 5, 10]$, then if the same coding is used for each variable with $\ell = 5$, the binary coded concatenated vector will be $A = [11001, 00101, 01010]$ and for processing purposes the vector (chromosome) becomes $A = [110010010101010]$.

For the present discussion it will be assumed that all the laminae are of the same material and have the same number of possible discrete fiber orientations. Thus one discrete set will be used for all the laminae. Since laminated problems sometimes incorporate a large number of layers it is convenient to use the real variable coding method. A typical 6 layer design vector for a symmetric layup is thus represented as $A = [80, -25, 40, 40, -25, 80]$ as equivalent to $[80/-25/40]_S$.

3.2 Reproduction process

The reproduction process resembles an evaluation and selection process in natural systems. The implementation of this process consists of five events namely the function evaluation of each individual design in the population, the evaluation of the total function value, the evaluation of the probability of selection for each design, the evaluation of the cumulative probability for each design (which all have to add up to unity) and the random selection itself.

The cumulative probability divides the space $[0, 1]$ into the relative contributions of the probability of selection or "fitness" of each design and therefore causes an above average design to occupy an above average size fraction of the space. A design with an above average size fraction is more likely to be chosen because the random number generator is more likely to generate numbers within that larger fraction of space. The cumulative space is thus analogous to a "roulette wheel". The selection step is made by spinning the roulette wheel n times and generating a random number which is then traced to the cumulative space where the corresponding design is located. The selected designs are copied to a new population of size n.

It is thus conceivable that the number of above average designs will grow exponentially, average designs will stay the same and below average designs will die off.

3.3 Crossover process

The crossover process resembles the mating process which consists of the selection of suitable designs from the present population and the transfer of gene information between the selected mates.

The crossover designs are selected proportional to the prescribed probability of crossover pc. A random number is generated n times and if it is smaller than or equal to pc, the design is selected, else the design is ignored. This process ends up with approximately $pc \cdot n$ selected designs which are then rounded to the nearest even number if necessary. Variations of this implementation can be found in the work of Ponslet *et al.* (1993) and Arora *et al.* (1994).

The selected designs are randomly shuffled in order to avoid alike designs mating with themselves, which may lead to a premature convergence, and to enhance the exploration of the design space. These designs are grouped into mating pairs so that each design can only mate once. The crossover is done by first generating a random number within the range of the length of the design string. The number is used to indicate the position of the cross-site in both mating strings. All the gene values from the cross-site on one side of the string are interchanged with those of the other string in the same positions. For example, if A_1 and A_2 are the parent strings and $\ell = 4$, then there are $4 - 1 = 3$ possible cross-sites and, if it is assumed that a randomly selected cross-site is at 2, then

$$A_1 = 1 \quad 0 \quad | \quad 0 \quad 1, \quad A_2 = 0 \quad 1 \quad | \quad 1 \quad 0,$$

and if the parents are crossed at the indicated cross-site, the offspring become A_1' and A_2' as

$$A_1' = 1 \quad 0 \quad | \quad 1 \quad 0, \qquad A_2' = 0 \quad 1 \quad | \quad 0 \quad 1.$$

For the present coding this process might be executed for a 6 layer symmetric layup as

$$A_1 = 80 \quad -25 \mid 40, \quad A_2 = -55 \quad -10 \mid -10,$$

to yield

$$A_1' = 80\ -25\ |\ -10\,,\ A_2' = -55\ -10\ |\ 40\,.$$

By examining the crossover process of the two coding methods it is evident that if the binary vector is crossed over inside the variable string, the decoded value after crossover is not going to be the same as before the crossover. In the real variable coding this phenomenon is not possible since the real value stays unaltered during crossover. This means that the binary crossover (and general lower order cardinality codings) generates new possible strings (or real values) during crossover and thus enhances the design space exploration. A method to deal with this phenomenon in real variable codings is discussed by Ponslet *et al.* (1993). This method uses a weighted average between the values at the cross-site which resembles the possible change in values at the cross-site in the binary coding.

3.4 Mutation process

The purpose of the mutation process is to explore the design space randomly in a global manner, but at a rate that does not cause a large scale loss of evolved data structures. This is done to prevent premature convergence. In the process all the gene values in the population have an equal chance of being changed which has the effect of a random change in a particular design vector. The process thus consists of a random number being generated for each gene value (or character) and if this number is smaller than a prescribed probability of mutation pm, the value is changed randomly to another possible value, else it is ignored. Variations on this implementation can be found in the work of Huang and Arora (1995), Ponslet *et al.* (1993) and Arora *et al.* (1994). The expected number of genes (bits) or positions to be changed is approximately $\ell \cdot pm \cdot n$.

3.5 Range of crossover and mutation probabilities

Ranges for the crossover and mutation parameters of a binary coding are reported by Hajela (1990) to be 0.6 to 0.8 and 0.01 to 0.02 respectively for most problems. For stacking sequence design, Le Riche and Haftka (1993) use a crossover value of 1 and mutation rates up to 0.1 for a 1,2,3 coding alphabet. It is expected that the optimal mutation rate for the real variable coding will be larger than that of the binary coding because of the fact that a larger rate of mutation is needed for the real variable coding to ensure that all the characters in the coding alphabet enter the population information pool some time during the evolution process. For example, if a binary coding ($\{0,1\}$) is used, the probability of occurrence for each character at a random call is $1/2 \times 100 = 50\%$ and thus will $2 \times 2 = 4$ random calls yield a 100% probability of occurrence of the coding alphabet, but if a real variable coding is used with 37 unique characters, the probability of occurrence for each character at a random call will be $1/37 \times 100 = 2.7\%$ and thus will $37 \times 37 = 1369$ random calls yield a 100% probability of occurrence of the coding alphabet. In general, mutation rates are expected to be small and crossover rates to be relatively large.

4 Implementation of the GA

Currently there exists a number of different implementations as discussed by Huang and Arora (1995), Goldberg (1989), Michalewicz (1992), Kogiso *et al.* (1994), Ponslet *et al.* (1993) and Arora *et al.* (1994). The differences include additional operators, different elitist methods and small variations in operator execution principles. The current implementation is based on that of Michalewicz (1992) and an algorithm is presented in the Appendix.

4.1 Customizing the algorithm

The raw algorithm consisting of the three operators alone is not suitable for general use. It needs to be customized for its specific purpose. Some additional factors that must be considered include the maximization or minimization of the problem and the convergence criterion.

4.1.1 The cost function. Since the selection process in the reproduction phase works relative to the largest function value, it is necessary to make a distinction between the maximization and minimization as follows.

Maximization: It is sufficient to substitute the function value for the fitness value. Thus the fitness of the design string A_i is $f_i = f(A_i)$.

Minimization: It is possible to invert the function values as $f_i' = -f_i + B + C$, where B is the largest positive value from the set of function values and C is an offset value which in this case is taken as the average of the function values. Thus the fitness of design A_i becomes $f_i' = -f(A_i) + B + C$ and the relative probabilities for the calculation of the cumulative probabilities become $p_i' = (-f_i + B + C)/(-F + n(B + C))$. Where n is the population size, $F = \sum_{i=1}^{n} f_i$ and f_i is the function value of design A_i.

4.1.2 Convergence criteria. In general the GA will converge to an optimum through its operations alone but will not remain there since, in subsequent generations, the data structures are disrupted by the genetic operators. It is thus necessary to implement a convergence criterion based on the rate of improvement of the best design. The criterion that is used in this exercise is based on the one proposed by Le Riche and Haftka (1993) which terminates if there is no further improvement in the optimum within a prescribed number of generations.

It is also convenient to keep the best design found and to insert it from time to time in order to preserve the best data structures.

Constraints can be implemented in the form of penalty functions as discussed by Michalewicz (1992), which in effect biases the roulette wheel towards the feasible designs.

Table 4. Solutions for the central point load by Groenwold *et al.* (1996) with $N = 25$

	$b/a = 1$		$b/a = 2$	
NL	Design	$\bar{w}$	Design	$\bar{w}$
4	$[45/-45]_S$	1.915	$[-8.42/55.49]_S$	4.750
8	$[-45/45_3]_S$	1.513	$[26.54/-32.83_2/-32.81]_S$	4.298
16	$[45/-45/45/-45_5]_S$	1.501	$[30.27/-29.35_2/30.26/-29.36/30.35/29.76/-29.83]_S$	4.278

Table 5. GA results for the uniformly distributed load with $N = 7$

	$b/a = 1$			$b/a = 2$		
NL	Design	$\bar{w}$	FE	Design	$\bar{w}$	FE
4	$[-45/45]_S$	0.599	20	$[0/0]_S$	0.865	160
8	$[45/-45_3]_S$	0.506	2000	$[0_4]_S$	0.865	1360
16	$[45/-45_2/45/-40/50/-45/25]_S$	0.504	9120	$[0_4/-5/0_2/-20]_S$	0.866	5560

5 Problem statement for laminated plate design

For the present discussion the design criterion was chosen to be the transverse deflection of the plate. It was decided to *minimize* this deflection due to a central point and uniformly distributed load. Two approaches can be followed, (a) the deflection at a point could be minimized or (b) the strain energy absorbed in the plate could be minimized. In this case the objective function was chosen to be the strain energy.

The optimization problem for the linear discrete coding method is therefore formulated as follows:

$$\text{minimize } U_b(\theta) = \frac{1}{2}\int_A \kappa^T \tilde{\mathbf{D}} \kappa \, \mathrm{d}A,$$

$$\text{subject to } -90^\circ \leq \theta_k \leq +90^\circ,$$

with

$$\Delta\theta_k = 5^\circ,$$

and

$$t = \sum_{k=1}^{NL} t_k,$$

$U_b(\theta)$ is the bending strain energy in the plate, θ_k is the fiber orientation of lamina k, $\tilde{\mathbf{D}}$ is the bending stiffness matrix of the laminate, κ is the bending curvature matrix, A is the area of the plate and t is the total thickness of the laminate which consists of NL equally thick layers.

The displacements are listed in *normalized* form as presented by Tauchert and Adibhatla (1984) which are calculated as $\bar{w} = (w_o 10^3 E_o b t^3)/(Qa^3)$ for the point load and $\bar{w} = (w_o 10^3 E_o t^3)/(qa^4)$ for the uniformly distributed load. The symbol w_o represents the actual calculated displacement, E_o is a stiffness magnification factor, t is the total thickness of the plate, Q and q are the point and distributed loads, respectively, and a and b are the side lengths in the $x-$ and $y-$ directions of the plate, respectively.

The Rayleigh-Ritz method is employed to construct the structural problem where the displacement function is assumed to be approximated as follows:

$$w_o(x,y) = \sum_{m=1}^{M}\sum_{n=1}^{N} a_{mn} \sin\left(\frac{m\pi x}{a}\right) \sin\left(\frac{n\pi y}{b}\right). \tag{1}$$

The boundary conditions are assumed to be free-free.

The potential energy of the transverse loads on the plate can be expressed by

$$W_z = \int_A q(x,y)\, w_o(x,y)\, \mathrm{d}A + \sum_i Q_i\, w_o(x,y),$$

where $q(x,y)$ is the distributed load and Q_i the i-th point load on the plate and $w_o(x,y)$ the transverse deflection of the neutral plane. The total potential energy of the elastic continuum can be written as

$$\Pi = U_b - W_z = 0.$$

For the Rayleigh-Ritz method, it is necessary that the derivative of the total potential energy with respect to each individual coefficient $a_{k\ell}$ be zero as

$$\frac{\partial \Pi}{\partial a_{k\ell}} = 0, \quad k = 1,\ldots,M;\ \ell = 1,\ldots,N.$$

The derivatives yield a system of equations which are conveniently expressed in matrix form as $\mathbf{K}\,\mathbf{a} = \mathbf{f}$. Where $\mathbf{K}$, $\mathbf{a}$ and $\mathbf{f}$ are the symmetric stiffness, weighting coefficients and force matrices respectively. This system of equations is solved for the displacement weighting coefficients which are then used to calculate the displacement and strain energy associated with the specific problem parameters. The strain energy is expressed as $W_z = \mathbf{a}^T\,\mathbf{f}$.

6 Example problems

The example problems represent 12 plates with 4, 8 and 16 symmetrically stacked layers subjected to either a central point load (CPL) or uniformly distributed load (UDL) and with an aspect ratio (b/a) of 1 and 2.

Analysis parameters used are the number of Ritz harmonics, $N = 7$ and the population size for all the problems, $n = 20$. The GA will start with $\boldsymbol{\theta} = \mathbf{0}$. For all cases the convergence iteration parameter (CI) is set to 100 and the maximum iteration parameter (MI) to 1000. The material data used throughout the calculations is assumed to be $E_{11} = 1.81\times10^{11}$ Pa, $E_{22} = 1.03\times10^{10}$ Pa, $G_{12} = 7.17\times10^{9}$ Pa, $\nu_{12} = 0.28$ and $t_k = 1.0$ mm.

Table 6. Solutions for the uniformly distributed load from Kam and Snyman (1991) with $N = 7$

	$b/a = 1$	
NL	Design	$\bar{w}$
4	$[45/-45]_S$	0.599
8	$[45/-45_3]_S$	0.506
16	$[45/-45/45/-45_5]_S$	0.503

Table 7. FE cost comparison for the linear discrete coding

		FE ($b/a = 1$)		FE ($b/a = 2$)	
NL	MAX FE	CPL	UDL	CPL	UDL
4	1369	180	20	40	160
8	1.87×10^6	2820	2000	2080	1360
16	3.5×10^{12}	2520	9120	2960	5560

The optimization analyses are done by assuming a coarse set of the pc and pm algorithm parameters. For the present discussion the parameter sets are chosen to be $pc \in \{1.0, 0.75, 0.5\}$ and $pm \in \{0.4, 0.25, 0.1\}$. By performing nine analyses, using all the possible combinations, confidence is established in the best solution found since the GA is more likely to find an optimum when more function evaluations are done. The normalized displacement (see Section 5) is used for the cost evaluation and the lowest cost of all the analyses is chosen to be the best. If more than one parameter combination yields the same lowest cost, further analyses at these parameter combinations are executed until the first lowest cost is recorded. The best design is reported in Tables 3 and 5.

The GA solutions were obtained for the linear discrete set with $N = 7$ and the displacement solutions of the resulting optimal designs were subsequently refined with $N = 25$ (Table 3). The number of function evaluations (FE) used by the GA to generate the optimum is also listed in Table 3. For comparison, the results of Groenwold *et al.* (1996) with $N = 25$ are listed in Table 4. A gradient-based continuous global optimization method was used to generate these results. From Tables 3 and 4 it is evident that the GA found optima with normalized displacements within 1% of the optimizer used by Groenwold *et al.* (1996). For a uniformly distributed load, the GA solutions for the linear discrete set with $N = 7$ are listed in Table 5 and, for comparison, the results of Kam and Snyman (1991) using a method similar to that of Groenwold *et al.* (1996) are presented in Table 6. The results are similar, differing only with respect to 16 layers. The cost function is relatively insensitive to these differences.

In conclusion, the GA seems to find the region of the optimum and could thus also be used to obtain starting points for gradient-based optimizers.

An important factor to consider is the number of function evaluations (or cost) the GA takes to generate the optimum which is expressed as cost $= BI \cdot n$, where BI is the generation in which the best optimum was found. The maximum possible number of unique design vectors for the discrete symmetric plate problem can be calculated to be (number of discrete values in variable)$^{NL/2}$. The number of discrete values in the fiber orientation variable for the linear discrete coding is 37. Thus if the number of function evaluations required by the GA is compared to the maximum number of discrete combinations (Table 7), it is evident that in all cases the cost is significantly less than the maximum possible cost, especially where a large number of variables are involved.

7 Conclusions

This paper discussed the principles and implementation of the genetic algorithm (GA) as a global discrete optimizer for the strain energy minimization of a simply supported symmetric laminated plate bending problem. A real variable coding of the discrete values of the fiber orientation variables was used. The GA is compared with a trajectory method for global optimization. The conclusions are summarized as follows.

1. The GA is a robust global discrete optimization method which is easy to implement on a variety of problems and which does not require gradient calculations and line searches.

2. The GA seems to find the region of the optimum relatively fast, especially if the objective function is multimodal. The GA yielded solutions within 1% of the global optima obtained using comparable continuous methods.

3. Conventionally, binary codings of the variables have been used in GA's but this study shows that the real variable coding method could also be used successfully. This method is easier to implement than the binary coding.

4. The genetic algorithm can only use discrete variable values. However, the values do not have to be integers but can be real numbers. If desired, a fine resolution of the discrete variable set can be chosen to enlarge the design space so that a "continuous" solution may be approached. Otherwise a gradient-based method can be used to obtain a better solution with the discrete solution as a starting point. This refinement has been shown to be relatively inexpensive.

5. The real variable coding of large discrete sets seems to favor higher rates of mutation. Thus the real variable coding is expected to be more efficient (in terms of function evaluations) for long design vectors with small design spaces.

References

Arora, J.S.; Elwakeil, O.A.; Chahande, A.I.; Hsieh, C.C. 1995: Global optimization methods for engineering applications: a review. *Struct. Optim.* **9**, 137-159

Arora, J.S.; Huang, M.W.; Hsieh, C.C. 1994: Methods for optimization of nonlinear problems with discrete variables: a review. *Struct. Optim.* 8, 69-85

De Jong, K.A. 1975: *An analysis of the behaviour of a class of genetic adaptive systems.* Ph.D. Dissertation, Dept. of Computer Science, Univ. of Michigan

Galante, M. 1996: Genetic algorithms as an approach to optimize real-world trusses. *Int. J. Num. Meth. Engrg.* **39**, 361-382

Goldberg, D.E. 1989: *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley

Groenwold, A.A.; Snyman, J.A.; Stander, N. 1996: Modified trajectory method for practical global optimization problems. *AIAA J.* **34**, 2126-2131

Groenwold, A.A.; Stander, N.; Snyman, J.A. 1997: A regional genetic algorithm for the discrete optimal design of truss structures. (submitted)

Haftka, R.T.; Gürdal, Z. 1992: *Elements of structural optimization*, 3-rd ed. Dordrecht: Kluwer

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

Hajela, P.; Shih, C.J. 1989: Optimal design of laminated composites using a modified mixed integer and discrete programming algorithm. *Comp. & Struct.* **32**, 213-221

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

Huang, M.W.; Arora, J.S. 1995: Engineering optimization with discrete variables. *Proc. 36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conf. & AIAA/ASME Adaptive Structures Forum* (held in New Orleans, LA), Paper No. AIAA-95-1333-CP, pp. 1475-1485

Kam, T.Y.; Snyman, J.A. 1991: Optimal design of laminated composite plates using a global optimization technique. *Comp. & Struct.* **19**, 351-370

Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. 1983: Optimization by simulated annealing. *Science* **220**, 671-680

Kogiso, N.; Watson, L.T.; Gürdal, Z.; Haftka, R.T. 1994: Genetic algorithms with local improvement for composite laminate design. *Struct. Optim.* **7**, 207-218

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

Michalewicz, Z. 1992: *Genetic algorithms + data structures = evolution programs.* Berlin, Heidelberg, New York: Springer

Nagendra, S.; Haftka, R.T.; Gürdal, Z. 1993: Design of a blade stiffened composite panel by a genetic algorithm. *Proc. 34th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conf. and AIAA/ASME Adaptive Structures Forum* (held in La Jolla, CA), Paper No. AIAA-93-1584-CP, pp. 2418-2436

Ponslet, E.; Haftka, R.T.; Cudney, H.H. 1993: Optimal placement of tuning masses on truss structures by genetic algorithms. *Proc. 34th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conf. and AIAA/ASME Adaptive Structures Forum* (held in La Jolla, CA), Paper No. AIAA-93-1586-CP, pp. 2448-2457

Press, W.H.; Flannery, B.P.; Teukolsky, S.A.; Vetterling, W.T. 1988: *Numerical recipes in C: the art of scientific computing.* New York: Cambridge University Press

Soremekun, G.; Gürdal, Z.; Haftka, R.T.; Watson, L.T. 1996: Improving genetic algorithm efficiency and reliability in the design and optimization of composite structures. *6th AIAA/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization Part 1* (held in Bellevue, WA), Paper No. AIAA-96-4024-CP, pp. 372-383.

Tauchert, T.R.; Adibhatla, S. 1984: Design of laminated plates for maximum stiffness. *J. Comp. Mat.* **18**, 58-69

Appendix

Pseudo-algorithm

The implementation listing of a real variable coding GA is given below. The symbols are defined as follows. Maximum number of iterations MI; number of convergence iterations CI; iteration in which first best design was found BI; population size n; number of layers NL; integer counters *iter*, i, j and k; random number $ran \in [0,1]$; random integer ri; number of designs to crossover mx; length of the design vector string ℓ.

Step 1 Start program.

Step 2 Read GA parameters and problem related data.

Step 3 Initialize first population randomly:
For ($i = 1$ to n)
For ($j = 1$ to NL)
set each design variable at index $[i, j]$ randomly.

Step 4 **Begin evolution process**
For ($iter = 1$ to MI) do Steps 5 to 7.

Step 5 **Reproduction process**

Step 5.a Make population symmetric with respect to neutral plane of plate (problem specific constraint):
For ($i = 1$ to n)
For ($j = 1$ to $NL/2$)
copy variable values symmetrically with respect to neutral plane

Step 5.b Evaluate objective function for each design vector in the population:
For ($i = 1$ to n)
f_i is the fitness score for the i-th structural problem.

Step 5.c Determine sum total of function fitness values:
For ($i = 1$ to n)
$F = F + f_i$.

Step 5.d Test for convergence of the evolution process:
if $((iter - BI) > CI)$
go to Step 8

Step 5.e Determine the relative fitness probability (ranking) for each design:
For ($i = 1$ to n)
$p_i = \frac{f_i}{F}$.

Step 5.f Determine the cumulative probability (weighting sector) for each design:
For ($i = 1$ to n)
$q_i = \sum_{j=1}^{i} p_j$.

Received Sept. 8, 1997

Step 5.g Select and copy designs according to cumulative probability:
For ($j = 1$ to n)
generate a random number ran
For ($k = 1$ to n) seek corresponding sector number
if $((q_{k-1} \leq ran)$ and $(ran \leq q_k))$
where $q[0] = 0$ and $q[n] = 1$
For ($i = 1$ to NL)
copy selected design variable at index $[k, i]$ to new design variable at index $[j, i]$.

Step 6 **Crossover process**

Step 6.a Choose crossover designs randomly according to pc:
For ($i = 1$ to n)
generate a random number ran
if $(ran \leq pc)$
mark design i for further processing.

Step 6.b Make the number of selected designs even to give mx designs.

Step 6.c Execute the crossover:
For ($i = 1$ to $mx/2$)
choose vector pairs randomly from selected list and
generate a random integer
$ri \in [1, \ell - 1]$
For ($j = 1$ to ri)
copy value of vector A at index $[A, j]$
to vector B at index $[B, j]$
and *vice versa*.
See Ponslet (1993) and Arora *et al.* (1994) for variations of this implementation.

Step 7 **Mutation process**

Change design variables randomly according to pm:
For ($i = 1$ to n)
For ($j = 1$ to NL)
generate a random number ran
if $(ran \leq pm)$
select new design variable randomly from discrete set.

Step 8 Generate output.

Step 9 Stop program.