# A genetic algorithm with memory for mixed discrete-continuous design optimization

Vladimir B. Gantovnik [a,*], Christine M. Anderson-Cook [b], Zafer Gürdal [c], and Layne T. Watson [d]

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

[b] *Department of Statistics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA*

[c] *Departments of Aerospace and Ocean Engineering, and Engineering Science and Mechanics, Virginia Polytechnic Institute and State University, Blacksburg, VA, 24061*

[d] *Departments of Computer Science, and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA*

**Abstract**

This paper describes a new approach for reducing the number of the fitness function evaluations required by a genetic algorithm (GA) for optimization problems with mixed continuous and discrete design variables. The proposed additions to the GA make the search more effective and rapidly improve the fitness value from generation to generation. The additions involve *memory* as a function of both discrete and continuous design variables, multivariate approximation of the fitness function in terms of several continuous design variables, and localized search based on the multivariate approximation. The approximation is demonstrated for the minimum weight design of a composite cylindrical shell with grid stiffeners.

*Key words:* Genetic Algorithm, Composite Structure, Response Surface Approximation

* Corresponding author.
*Email address:* gantovnik@vt.edu (Vladimir B. Gantovnik).

## 1 Introduction

Mixed discrete-continuous design optimization is an active research topic. There are many diverse applications that are mathematically modelled in terms of mixed discrete-continuous variables. The optimization of such models is typically difficult because of potential existence of multiple local minima in the search space. The most general methods for solving such problems are branch and bound method, simulated annealing (SA) method, and genetic algorithms (GA) [1]. These methods do not require gradient or Hessian information. However, to reach an optimal solution with a high degree of confidence, they typically require a large number of analyses during the optimization search. Performance of these methods is even more of an issue for problems that include continuous variables. The number of analyses required is an important characteristic of any method in multidisciplinary optimization (MDO). Several studies have concentrated on improving the reliability and efficiency of GAs. Hybrid algorithms formed by the combination of a GA with local search methods provide increased performance when compared to a GA with a discrete encoding of real numbers or local search alone [2]. In order to reduce the computational cost, two of the authors earlier used local improvements and memory for discrete problems so that information from previously analyzed design points is utilized in later searches [3,4]. In the first method a memory binary tree was employed for a composite panel design problem to store pertinent information about laminate designs that have already been analyzed [3]. After the creation of a new population of designs, the tree structure is searched for either a design with identical stacking sequence or similar performance, such as a laminate with identical in-plane strains. Depending on the kind of information that can be retrieved from the tree, the analysis for a given laminate may be significantly reduced or may not be required at all. The second method is called local improvement [4]. This technique was applied to the problem of maximizing the buckling load of a rectangular laminated composite plate. The information about previously analyzed designs is used to construct an approximation to buckling load in the neighborhood of each member of the population of designs. After that, the approximations are used to search for improved designs in small discrete spaces around nominal designs. These two methods demonstrated substantial improvements in computational efficiency for purely discrete optimization problems. The implementation, however, was not suitable for handling continuous design variables.

The objective of the present work is to find a suitable algorithm for a GA with memory that can work with discrete and several continuous variables simultaneously. A local memory for the continuous part of the design space at each discrete node of a binary tree based on multivariate approximation is proposed for problems with several continuous variables. The efficiency of the proposed multivariate approximation based procedure, as well as the use

of memory for a GA that can handle continuous variables, are investigated for the weight optimization of a lattice shell with laminated composite skins subjected to axial compressive load.

## 2 Genetic algorithm package

A Fortran 90 GA framework that was designed in an earlier research effort was used for the composite laminate structure design [5]. This framework includes a module, encapsulating GA data structures, and a package of GA operators. The module and the package of operators result in what we call a standard genetic algorithm. The proposed algorithm is incorporated within the GA framework as a sample test program that illustrates performance of the binary tree memory and multivariate approximation. An integer alphabet is used to code ply genes. The continuous variables represented by floating-point numbers had already been implemented in the GA framework data structure as geometry chromosomes.

## 3 Binary tree memory

A binary tree is a linked list structure in which each node may point to up to two other nodes. In a binary search tree, each left pointer points to nodes containing elements that are smaller than the element in the current node; each right pointer points to nodes containing elements that are greater than the element in the current node. The binary tree is used to store data pertinent to the design such as the design string and its associated fitness and constraint function values. A binary tree has several properties of great practical value, one of which is that the data can be retrieved, modified, and inserted relatively quickly. If the tree is perfectly balanced, the cost of inserting of an element in a tree with $n$ nodes is proportional to $\log_2 n$ steps, and rebalancing the tree after an insertion may take as little as several steps, but at most takes $\log_2 n$ steps. Thus, the total time is of the order of $\log_2 n$ [6].

In the standard genetic algorithm, a new population may contain designs that have already been encountered in the previous generations, especially towards the end of the optimization process. The memory procedure eliminates the possibility of repeating an analysis that could be expensive. Algorithm 1 shows the pseudo code of the fitness function evaluation with the aid of the binary tree.

**Algorithm 1** Evaluation of fitness function using binary tree.

```
search for the given design in the binary tree;
if found then
  get the fitness function value from the binary tree;
else
  perform exact analysis;
end if
```

After a new generation of designs is created by the genetic operations, the binary tree is searched for each new design. If the design is found, the fitness value is retrieved from the binary tree without conducting an analysis. Otherwise, the fitness is obtained based on an exact analysis. This new design and its fitness value are then inserted in the tree as a new node.

## 4 Response surface approximation

The procedure described above works well for purely discrete optimization problems where designs are completely described by discrete strings. In case of mixed optimization problems where designs include discrete and continuous variables, the solution becomes more complicated. If the continuous variables are also discretized into a fine discrete set, the possibility of creating a child design that has the same discrete and continuous parts as one of the earlier designs diminishes substantially. In the worst case, if the continuous design variables are represented as a real numbers, which is the approach used by most recent research work, it may not be possible to create a child design that has the exact same real part as one of the parents, rendering the binary tree memory useless, and result in exact analysis even if the real part of the new child is different from one of the earlier designs by a minute amount.

The main idea of the *memory* approach proposed in this work is to construct a response surface approximation for the fitness function as a function of the continuous variables using historical data values, and estimate from the stored data whenever appropriate. The *memory* in this case consists of two parts: a binary tree, which consists of the nodes that have different discrete parts of the design, and a storage part at each node that keeps the continuous values and their associated fitness values. That is, each node contains a real array that stores the continuous variable points' value and their corresponding fitness function values. In order for the *memory* to be functional, it is necessary to have accumulated a sufficient number of designs with different continuous values for a particular discrete design point so that the approximation can be constructed. Naturally, some of the discrete nodes will not have more than a few designs with different continuous values. However, it is possible that as the evolution progresses good discrete parts will start appearing repeatedly

with different continuous values. In this case, one will be able to construct a good quality response surface approximation to the data.

The response surface approximation approach is an extension of the previous work by the authors where a spline-based approach was used for only one continuous variable [7]. An evolving database of continuous variable points is used in the current work to construct a multivariate response surface approximation at those discrete nodes that are processed frequently. The modified quadratic Shepard's method is a local smoothing method used for the approximation of scattered data for the case of two independent continuous design variables [8]. This method may be the best known among all scattered data interpolants for a general number of variables. Shepard's method for fitting a surface to data values has the advantage of small storage requirements and easy generalization to more than two independent variables.

In addition to building a multivariate approximation, it is important to assess accuracy of the multivariate approximation at new continuous points, so that a decision may be made either to accept the approximation or perform exact function evaluation. Based on the bivariate approximation, the proposed algorithm described by the following pseudo code is then used to decide when to retrieve the fitness function value from the approximation, and when to do an exact analysis and add the new data point to the approximation database. For the description of the pseudo code, let $v \in Z^k$ be a $k$-dimensional integer design vector for the discrete space, $x^{(i)} \in E^m$ a real $m$-dimensional design vector for the continuous variables, and $f\left(v, x^{(i)}\right)$ the corresponding fitness value of the individual defined by $\left(v, x^{(i)}\right)$. Furthermore, define $d_i \in E$ to be a real distance corresponding to point $x^{(i)}$ to measure its proximity any given point, $c \in Z$ an integer counter, initially zero, and $r \in E$ a real range value. Let $T$ be the set of observed exact analyses and their corresponding information within a given discrete node, precisely

$$T = \left\{ \left( x^{(i)}, f\left(v, x^{(i)}\right), d_i \right) \right\}_{i=1}^{n}.$$

Each node in the binary tree memory structure records a tuple of the form $(v, T, c, r)$. The pseudo code for processing a candidate individual $(v, x)$ is defined by Algorithm 2.

The algorithm uses three real user-specified parameters, $d_0$, $\delta$, and $\epsilon$. The parameter $d_0 > 0$ is an upper bound on the trust region radius about each sample point $x^{(i)}$. The parameter $\delta$ is chosen to satisfy $0 < \delta < 1$, and in higher dimensions protects against large variations in $f$ in unsampled directions. Finally, the parameter $\epsilon > 0$ is the selected acceptable approximation accuracy, and is solely based on engineering considerations.

**Algorithm 2** Evaluation of fitness function using binary tree and $m$-dimensional approximation.

**if** $v$ is not found in the tree **then**
  evaluate $f(v, x)$;
  $T := \Big\{(x, f(v, x), 0)\Big\}$;
  add a node corresponding to $(v, T, 1, 0)$;
  **return** $f(v, x)$;
**else**
  **if** $c < (m + 2)(m + 1)$ **then**
    evaluate $f(v, x)$;
    $T := T \cup \Big\{(x, f(v, x), 0)\Big\}$;
    $c := c + 1$;
    $r := \max\limits_{t \in T} t_2 - \min\limits_{t \in T} t_2$;
    **return** $f(v, x)$;
  **else**
    construct an approximation $S(x)$ using
    the data in $T = \left\{\Big(x^{(i)}, f\Big(v, x^{(i)}\Big), d_i\Big)\right\}_{i=1}^{n}$;
    define $k$ and $d^*$ by $d^* = d_k - \|x - x^{(k)}\| = \max\limits_{1 \le i \le n} d_i - \|x - x^{(i)}\|$;
    **if** $d^* \ge 0$ and $\Big|f\Big(v, x^{(k)}\Big) - S(x)\Big| < \delta\, r$ **then**
      **return** $S(x)$;
    **else**
      evaluate $f(v, x)$;
      **if** $|f(v, x) - S(x)| > \epsilon$ **then**
        $T := T \cup \Big\{(x, f(v, x), 0)\Big\}$;
      **else**
        $d := \min\Big\{d_0, \|x - x^{(k)}\|\Big\}$; $d_k := d$;
        $T := T \cup \Big\{(x, f(v, x), d)\Big\}$;
      **end if**
      $r := \max\limits_{t \in T} t_2 - \min\limits_{t \in T} t_2$;
      **return** $f(v, x)$;
    **end if**
  **end if**
**end if**

## 5 Local improvement

Once a multivariate approximation is in place at a given discrete node, a local improvement procedure may be implemented to improve the performance of the GA. The values of the continuous variables at a given discrete node are either randomly assigned (if mutation operator is used) or obtained through a crossover operation. If an explicit multivariate approximation and its first

partial derivatives are available at a given node, it is possible to generate good candidates for the continuous design variables for the next child at that node rather than depend on random action from the crossover operator. That is, after construction of an initial approximation $S(x)$ of $f\Big(x)$ based on the objective function values obtained at the design sites, one can easily find the point $x^*$ that optimizes the approximate function $S(x)$ in some compact subset $\Omega \subset E^m$. This optimal $x^*$ value is stored at the discrete node in addition to the rest of the $T$ database. If, in future generations, a discrete node that has a stored $x^*$ value is reached through the crossover operation on the discrete part $v$ of the design, then, rather than performing crossover on the real part, $(v, x^*)$ is used as the child design for the next generation. This child design will then be treated like the other new designs in the child population and will be checked if an approximation to it can be used without exact analysis.

## 6 Design optimization problem

The design of a lattice shell with specified radius, length, and axial load level is considered. Such shells supported by a lattice have been considered as a replacement to solid shells, stiffened shells and honeycomb structures [9–11]. Consider a lattice cylindrical shell loaded with compressive axial force $P$. Proper design would involve, in general, determination of rib parameters (dimension of cross section, material, spacing, and orientation angle, $\varphi$), skin parameters (the number of layers, their materials, thicknesses, and orientation angles, $\theta_k$) that satisfy strength and stability constraints while minimizing the weight of the shell. Constraints considered include rib strength constraint $(C_1)$ , skin strength constraint $(C_2)$, rib local buckling constraint $(C_3)$, and general buckling constraint $(C_4)$. All constraint equations are based on the lattice cylindrical shell model developed by Bunakov [12–14].

The mixed optimization problem considered here operates on three design variables $v$, $x_1$, and $x_2$. The discrete variable is the stacking sequence of the skins, $v = \{\theta_1, \ldots, \theta_n\}$, where $n$ is an implicit design variable dictated by the number of layers in the skin stacking sequence. We shell restrict our consideration to two continuous design variables, namely, the angle of helical ribs, $x_1 = \varphi$, and the rib height, $x_2 = H$. The optimization problem can be formulated as finding the stacking sequences of the skins, the angle of helical ribs, and the rib height in order to minimize the mass of the shell, $M$. The set of design variables is expressed as a vector $\tau = (v, x_1, x_2)$. The design problem is typically formulated to provide a minimum mass structure:

$$M = 4\pi\rho L \left[ h \left( R + h + \frac{H}{2} \right) + H \frac{\delta}{a} (R + h) \right], \tag{1}$$

where $\rho$ is the material density, $L$ is the length of the shell, $R$ is the shell radius, $h = \sum_{k=1}^{n} h_0^{(k)}$ is the skin thickness, $h_0$ is the single ply thickness, $\delta$ is the rib width, $a$ is the rib spacing. The optimization problem can be written as

$$\min_{\tau} M(\tau) \tag{2}$$

such that

$$\begin{aligned}
&C_1(\tau) \geq 0 \quad \text{(rib strength)},\\
&C_2(\tau) \geq 0 \quad \text{(skin strength)},\\
&C_3(\tau) \geq 0 \quad \text{(rib local buckling)},\\
&C_4(\tau) \geq 0 \quad \text{(general buckling)},\\
&H \in [H_{min}, H_{max}],\\
&\varphi \in [\varphi_{min}, \varphi_{max}],\\
&\theta_k \in \{0^\circ, \pm 45^\circ, 90^\circ\}, \quad (k = 1, n),\\
&n \in [n_{min}, n_{max}],
\end{aligned}$$

where $H_{min}$ and $H_{max}$ are lower and upper bounds of the rib height; $\varphi_{min}$ and $\varphi_{max}$ are are lower and upper bounds of the angle of helical ribs, $\theta_k$ is the ply orientation angle in the $k$-th skin ply, $n$ is the total number of skin plies, $n_{min}$ and $n_{max}$ are minimum and maximum possible values of $n$. The critical constraint is defined as

$$C_{cr}(\tau) = \min_{i=1,4}\{C_i(\tau)\}, \tag{3}$$

and the constrained optimization problem is transformed into an unconstrained maximization problem for the genetic algorithm. This is done by using penalty parameters. The fitness function $\Phi$ to be maximized is defined as

$$\Phi(\tau) = \begin{cases} -M(\tau) + C_{cr}(\tau)q, & C_{cr}(\tau) \geq 0,\\ -M(\tau)(1 - C_{cr}(\tau))^p, & C_{cr}(\tau) < 0, \end{cases} \tag{4}$$

where $q$ and $p$ are bonus and penalty parameters, respectively.

## 7 Results

A cylindrical lattice shell is made of fiberglass-epoxy composite material with density $\rho = 2100$ kg/m$^3$. The specified axial compressive load is $P = 10^6$ N/m. The shell radius and length are $R = 1.0$ m and $L = 1.5$ m, respectively. The lattice shell has $\pm\varphi$ unidirectional helical ribs with elastic modulus $E = 45.0$ GPa, and shear modulus $G = 1.0$ GPa. The compressive strength of ribs is $\bar{\sigma}^r = 240.0$ MPa. The shell has external and internal skins made of T300/5208 graphite-epoxy unidirectional plies. The material properties of the skin plies

are given in Table 1. The possible ranges for the design variables are given in Table 2. The range of the shell mass throughout the entire design space is approximately $31.12 \leq M \leq 303.74$ kg.

### *7.1 GA parameters*

The values of the GA parameters used in the experiments are shown in Table 3. The GA stopping condition is a limit on the total number of function evaluations conducted by the standard GA, $(n_i)_{max} = 500000$. The best known global optimal design obtained by the standard GA is presented in Table 4. The table gives the average number of exact analyses from ten runs of individuals ($\bar{n}_e$), the continuous design variables ($x_1$, $x_2$), the discrete design variable ($v$), the critical constraint value ($C_{cr}$), the mass ($M$), and fitness function value ($\Phi$). This design was obtained in an average of about 274545 function evaluations by the standard GA. Table 5 shows reliability (fraction of runs in which the optimum was found, out of 50 runs) for various geometry chromosome mutation probabilities $p_m$ and population sizes. The conclusion from the table indicates that that the GA works better with the large population size along with very small mutation probability. A small population size causes the GA to quickly converge on a local minimum, because it insufficiently samples the parameter space. Note that the results given in Table 4 and in the rest of study provided below were obtained with the population size of 20 and with geometry chromosome mutation probability of 0.01.

### *7.2 Effect of continuous memory*

The results presented in this section focus on the ability of the proposed algorithm to save computational time during GA optimization with the multivariate approximation used as a memory device. The performance of the GA with the multivariate approximation is presented in Table 6, which shows averages from ten runs. The table shows the best design $(x_1, x_2, v, \Phi)$ after $n_i$ attempts to evaluate the fitness function, of which $n_e$ are the number of exact fitness function evaluations with $\epsilon = 0.01$, $\delta = 0.1$, $d_0 = 0.5$. In addition, Table 6 contains the average percent savings ($\bar{\xi}$) in terms of fitness function evaluations, mass ($M$) and critical constraint ($C_{cr}$) corresponding to the best design which are identical to the results presented earlier for the baseline algorithm. The percent savings ($\xi$) in terms of number of fitness function evaluations is defined by

$$\xi = \left(1 - \frac{n_e}{n_i}\right) \times 100\%. \tag{5}$$

The influence of the number of generations on the function evaluation savings are shown in Table 7. This table contains average values for fitness function

($\bar{\Phi}$), savings ($\bar{\xi}$), and reliability (fraction of runs the optimum was found, out of 20 runs). The number of successful runs and the percent savings ($\bar{\xi}$) increase when the algorithm is run for a longer time, as expected.

These results show that the cost of the GA with continuous variables could be reduced up to 60% relative to the standard GA by using the approximation procedure. For the problem considered, the computation of the fitness function is not very expensive in terms of CPU time. However, this procedure has great potential in problems with expensive objective functions.

The mean absolute error $\bar{\varepsilon}$ due to the approximation and the savings ($\bar{\xi}$) in terms of the number of fitness evaluations for different values of the parameters $\epsilon$, and $\delta$ with $d_0 = 0.5$ are shown in Table 8. The mean absolute error $\bar{\varepsilon}$ is defined as

$$\bar{\varepsilon} = \frac{1}{n_s} \sum_{i=1}^{n_s} |\Phi(x_i) - S(x_i)|, \tag{6}$$

where $n_s = n_i - n_e$ is the total number of acceptable approximate evaluations. This error is computed every time that the algorithm decides to extract an approximation of the fitness value without an exact analysis. It is possible to further enhance the performance of the algorithm by a more precise tuning of its parameters. Table 8 shows the expected trends; both average savings ($\bar{\xi}$) and average absolute error $\bar{\varepsilon}$ increase as either $\epsilon$ or $\delta$ increases. Table 8 also shows that the saving are significant, but of course these percentages are likely to be smaller for longer chromosomes that results in a larger design space.

For the above results, Algorithm 2 used the test $c < 20$ rather than $c < (m+2)(m+1) = 12$, because of constraints in the Shepard algorithm code from [8].

### *7.3 Effect of local improvement*

Finally, the performance comparison of the considered GA modifications, namely, (1) standard GA, (2) GA with approximation, and (3) GA with approximation and local improvement are presented in Table 9. The average number of attempts to evaluate the fitness function $\bar{n}_i$, the number of exact fitness function evaluations $\bar{n}_e$, the percent savings $\bar{\xi}$ in terms of number of exact fitness evaluations, and the percent savings $\bar{\zeta}$ defined as

$$\bar{\zeta} = \left(1 - \frac{\bar{n}_e}{\bar{n}_i(\text{Standard GA})}\right) \times 100\% \tag{7}$$

in terms of number of exact fitness evaluations as compared with the standard GA are shown in the table.

As one would expect, the GA with local improvement converges faster in terms of number of fitness function evaluations than the GA with approximation. The two algorithms with approximation demonstrate good convergence in comparison with the standard GA, and very substantial decrease in the number of exact analyses required to find the optimal solution.

## 8 Conclusions

A GA with memory along with multivariate approximation was applied to the problem of weight minimization of a lattice shell with mixed discrete design variable and two continuous design variables. The use of memory based on binary tree for discrete part of the design variables avoids repeating analyses of previously encountered designs. The multivariate approximation for continuous variables saves unnecessary exact analyses for points close to previous values. Moreover, it is also demonstrated that the multivariate approximation can be used to provide local improvement during the search and further reduce the number of exact function evaluations required to reach an improved solution.

## 9 Acknowledgements

The authors are indebted to Dr. Samy Missoum for suggesting improvements in the multivariate approximation algorithm. This research was supported in part by Air Force Office of Scientific Research grant F49620-99-1-0128 and National Science Foundation grant DMS-9625968.

## References

[1] Stelmack MA, Nakashima N, Batill SM. Genetic algortihms for mixed discrete-continuous optimization in multidisciplinary design. 38th AIAA/ ASME/ ASCE/ AHS/ ASC Structures, Structural Dynamics and Materials Conference, AIAA Paper No. 1998-2033. Long Beach, California; 1998.

[2] Seront G, Bersini H. A new GA-local search hybrid for optimization based on multi level single linkage clustering. Genetic and Evolutionary Computation Conference (GECCO-2000). Las Vegas, Nevada; 2000.

[3] Kogiso N, Watson LT, Gürdal Z, Haftka RT. Genetic algorithms with local improvement for composite laminate design. Structural Optimization 1994;7(4):207-218.

[4] Kogiso N, Watson LT, Gürdal Z, Haftka RT, Nagendra S. Design of composite laminates by a genetic algorithm with memory. Mechanics of Composite Materials and Structures 1994;1(1):95-117.

[5] McMahon MT, Watson LT, Soremekun GA, Gürdal Z, Haftka RT. A Fortran 90 genetic algorithm module for composite laminate structure design. Eng. Computers 1998;14:260-273.

[6] Vowels RA. Algorithms and data structures in F and Fortran. Tucson, Arizona: Unicomp, Inc; 1998.

[7] Gantovnik VB, Gürdal Z, Watson LT. A genetic algorithm with memory for optimal design of laminated sandwich composite panels. 43rd AIAA/ ASME/ ASCE/ AHS/ ASC Structures, Structural Dynamics, and Materials Conference, AIAA Paper No. 2002-1221. Denver, Colorado; 2002.

[8] Renka RJ. Multivariate interpolation of large sets of scattered data. ACM Transactionson Mathematical Software 1988;14(2):139-148.

[9] Vasiliev VV, Lopatin AV. Theory of lattice and stiffened composite shells. Mechanics of composite materials, edited by Tarnopolskii YM. (in Russian) Riga: Zinatne; 1992:82-88.

[10] Vasiliev VV, Barynin VA, Rasin AF. Anisogrid lattice structures - survey of development and application. Composite Structures 2001;54:361-370.

[11] Slinchenko D, Verijenko VE. Structural analysis of composite lattice shells of revolution on the basis of smearing stiffness. Composite Structures 2001;54:341-348.

[12] Bunakov VA, Protasov VD. Cylindrical lattice composite shells. Mechanics of Composite Materials (in Russian) 1989;6:1046-1053.

[13] Belousov PS, Bunakov VA. Bending of cylindrical lattice composite shells. Mechanics of Composite Materials (in Russian) 1992;2:225-231.

[14] Bunakov VA. Design of axially compressed composite cylindrical shells with lattice stiffeners. Optimal Design, edited by Vasiliev VV, Gürdal Z. Lancaster,PA: Technomic Publishing Co.; 1999:207-246.

**List of Tables**

Table 1
The material properties of the skin (T300/5208).

| Stiffness parameters, GPa | | | | Strength parameters, MPa | | | | |
|---|---|---|---|---|---|---|---|---|
| $E_1$ | $E_2$ | $G_{12}$ | $\nu_{12}$ | $X_t$ | $Y_t$ | $X_c$ | $Y_c$ | $S$ |
| 181.0 | 10.3 | 7.17 | 0.28 | 1500.0 | 57.0 | 1340.0 | 212.0 | 68.0 |

Table 2
Ranges for the design variables.

| Design variable | Range |
|---|---|
| $H \in [H_{min}, H_{max}]$ | $[0.001, 0.1]$ m |
| $\varphi \in [\varphi_{min}, \psi_{max}]$ | $[5°, 85°]$ |
| $\theta_k,\ k = 1, n$ | $\{0°, \pm 45°, 90°\}$ |
| $n \in [n_{min}, n_{max}]$ | $[2, 20]$ |

Table 3
GA parameters used in the experiments.

| Parameter | Value |
|---|---|
| Maximal number of generations | 25000 |
| Population size | 20 |
| Laminate chromosome length | 7 |
| Probability of crossover ($p_c$): | |
| • for laminate chromosomes | 1.0 |
| • for geometry chromosomes | 1.0 |
| Probability of mutation ($p_m$): | |
| • for laminate chromosomes | 0.05 |
| • for geometry chromosomes | 0.01 |
| Crossover type: | |
| • for laminate chromosomes | two-point |
| • for geometry chromosomes | one-point |

Table 4
The best known optimal design using standard GA.

| $\bar{n}_e$ | $x_1$ | $x_2$ | $v$ | $C_{cr}$ | $M$ | $\Phi$ |
|---|---|---|---|---|---|---|
| 274545 | 1.0000 | 0.0052 | 1111100 | 0.0 | 43.8228 | -0.1443 |

Table 5
The percent reliability for various geometry chromosome mutation probabilities $p_m$ and population sizes.

| $p_m$ | Population size | | |
|---|---|---|---|
| | 20 | 50 | 100 |
| $> 0.5$ | 0.00 | 0.00 | 0.00 |
| 0.5 | 0.04 | 0.02 | 0.00 |
| 0.4 | 0.10 | 0.00 | 0.02 |
| 0.3 | 0.14 | 0.20 | 0.00 |
| 0.2 | 0.46 | 0.38 | 0.14 |
| 0.1 | 0.60 | 0.78 | 0.52 |
| 0.01 | 0.52 | 0.72 | 0.96 |
| 0.001 | 0.36 | 0.78 | 0.94 |
| 0.0 | 0.28 | 0.44 | 0.96 |

Table 6
The efficiency of the multivariate approximation.

| $\bar{n}_i$ | $\bar{n}_e$ | $\bar{\xi}$, (%) | $x_1$ | $x_2$ | $v$ | $C_{cr}$ | $M$ | $\Phi$ |
|---|---|---|---|---|---|---|---|---|
| 183995 | 74626 | 59.4 | 1.0000 | 0.0052 | 1111100 | 0.0 | 43.8228 | -0.1443 |

Table 7
The performance of the GA with multivariate approximation for different number of generations.

| Number of generations | Reliability | $\bar{\Phi}$ | $\bar{\xi}$, (%) |
|---|---|---|---|
| 5000 | 0.1 | -0.1508 | 42.8 |
| 10000 | 0.4 | -0.1464 | 49.9 |
| 20000 | 0.9 | -0.1444 | 53.3 |

Table 8
The error of the multivariate approximation ($\bar{\varepsilon}$) and the percent savings ($\bar{\xi}$) as a function of the parameters $\epsilon$ and $\delta$ with $d_0 = 0.5$.

| $\epsilon$ | $\delta$ | $\bar{\varepsilon}$ | $\bar{\xi}$, (%) |
|---|---|---|---|
| 0.001 | 0.1 | 0.00097 | 47.9 |
| | 0.5 | 0.00128 | 50.5 |
| | 1.0 | 0.00445 | 52.2 |
| 0.005 | 0.1 | 0.00185 | 55.6 |
| | 0.5 | 0.01440 | 56.5 |
| | 1.0 | 0.03040 | 58.3 |
| 0.01 | 0.1 | 0.02490 | 59.4 |
| | 0.5 | 0.03310 | 62.0 |
| | 1.0 | 0.04170 | 66.8 |

Table 9
The performance comparison of the GA modifications.

| Modification | $\bar{n}_i$ | $\bar{n}_e$ | $\bar{\xi}$, (%) | $\bar{\zeta}$, (%) |
|---|---|---|---|---|
| • standard | 274545 | 274545 | 0.0 | 0.0 |
| • with approximation | 183995 | 74626 | 59.4 | 72.8 |
| • with approximation and local improvement | 53690 | 38280 | 28.7 | 86.1 |