# Optimization of Composite Structures by Genetic Algorithms

by

Rodolphe Le Riche

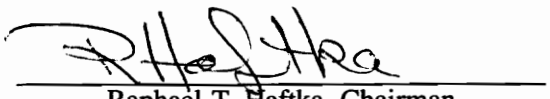
Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

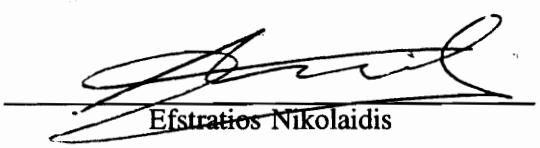
in

Aerospace Engineering

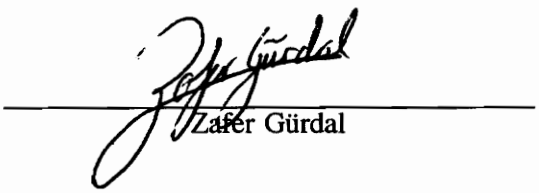
APPROVED



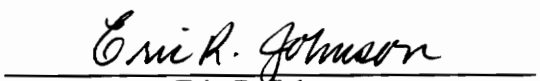
Raphael T. Haftka, Chairman



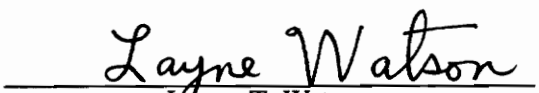
Efstratios Nikolaidis



Zafer Gürdal



Eric R. Johnson



Layne T. Watson

October, 1994

Blacksburg, Virginia

# Optimization of Composite Structures by Genetic Algorithms

by

Rodolphe Le Riche

Raphael T. Haftka, Chairman

Aerospace Engineering

(ABSTRACT)

The design of composite laminated panels is a combinatorial problem when the orientation of the fibers in each layer is restricted to a discrete pool of angles. Additionnally, composite laminates often have many optimal and near-optimal designs, and the designer may benefit by knowing many of those designs. Genetic algorithms are well suited for laminate design because they can handle the combinatorial nature of the problem and they permit the designer to obtain many near-optimal designs. However, their computational cost is high for most structural optimization problems. This work describes several attempts to reduce the cost of optimizing composite laminates using genetic algorithms.

First, the use of a genetic algorithm to maximize the buckling load of a fixed thickness composite laminate is studied. Various genetic parameters, including population size, probability of mutation, and probability of crossover are optimized by numerical experiments. A new genetic operator – stack swap – is proposed and shown to be effective in reducing the cost of the optimization.

Second, the genetic algorithm is revised and improved for minimum thickness design of composite laminated plates. The influence on the genetic search of the penalty functions enforcing failure constraints is studied. Combining fixed and proportional penalty functions is found to be the most efficient strategy. Improved selection, mutation, and stack swap operators are proposed. The use of an operator called scaling mutation that projects designs towards the feasible domain is investigated. The improvements in the genetic algorithm are shown to reduce the average price of the search by more than 50%.

Next, the improved genetic algorithm for minimum thickness laminate design is applied to a more complex wing box-beam optimization problem. Tuning the genetic algorithm on this

problem shows that, because the maximum length of a search is limited, the optimal population size does not grow with the size of the design space. If the probability of applying stack swap is reduced with the number of independent laminates in the wing box, stack swap enhances the performance of the genetic search on the wing box-beam problem.

Finally, the possibility of running many searches is investigated. It is empirically shown that several short searches can be more efficient than a long one, especially when high levels of reliability are required. An example is given where a genetic algorithm is specifically modified for better efficiency in the context of repeated short runs. A procedure is studied that enables predicting reliability at later stages of the search.

# ACKNOWLEDGEMENTS

I wish to thank Valou for being a patient Valou. Also, sincerest thanks to Doudou, Johnny and Nico.

I would like to thank Dr. R.T. Haftka for his most interesting comments and teachings.

Thanks to Eric, Wayan, Fred, Pradeep, and all the others who, among other things, helped me with technical matters.

Finally, I would like to express my gratitude to Dr. R.T. Haftka, Dr. B. Mahan, Dr. C. Vayssade and Dr. G. Touzot for making this transatlantic research program possible.

# TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

# LIST OF TABLES

# NOMENCLATURE

| | |
|---|---|
| $*$ | "Don't care" symbol. Here, it represents only one bit. |
| $a$ | Length of the plate. |
| $A_{ij}$ | Coefficients of the in-plane extensional stiffness matrix from Classical Lamination Theory. $[A]$ denotes the entire matrix. |
| $AC$ | Expected allele coverage. |
| $b$ | Width of the plate. |
| $\beta_1$ | Variable used in the critical buckling load factor due to shear calculations. |
| $C_i$ | Coefficients of the polynomial approximation to the internal loads in a box-beam (varying units). |
| $C_i^*$ | Coefficients of the polynomial approximation to the internal loads in a box-beam, normalized in lbs/in. |
| $d(H)$ | Defining length of schema H. |
| $\delta$ | Tolerance parameter in the definition of feasibility. |
| $D_{ij}$ | Coefficients of the flexural stiffness matrix from Classical Lamination Theory. $[D]$ denotes the entire matrix. |
| $e$ | Bonus parameter for having a large safety margin. |
| $E$ | One empty ply in laminate coding notation. |
| $E_1,\ E_2$ | Young's moduli in the fiber direction and perpendicular to the fiber direction, respectively. |
| $\varepsilon_1^{i,j},\ \varepsilon_2^{i,j},\ \gamma_{12}^{i,j}$ | Strains in the fiber direction, perpendicular to the fiber direction, and in shear, respectively. $i$ denotes the layer, $j$ the laminate (if necessary) |

| | |
|---|---|
| $\varepsilon_1^{ua}$, $\varepsilon_2^{ua}$, $\gamma_{12}^{ua}$ | Ultimate allowable strains in the fiber direction, perpendicular to the fiber direction, and in shear, respectively. |
| $f$ | Safety factor in the maximum strain strength failure criterion. |
| $F_i$ | Fitness associated to the $i$th individual. |
| $\bar{F}_i$ | Normalized fitness associated to the $i$th individual. |
| $F(H,t)$ | Average observed fitness of schema H at generation t. |
| $F(H)$ | Average fitness of schema H over all points in the design space. |
| $\bar{F}(t)$ | Average observed fitness of generation t. |
| $\{F\}$ | Vector of nodal forces in finite element analysis. |
| $\Gamma$ | Variable used in the critical buckling load factor due to shear calculations. |
| $G_{12}$ | Shear modulus in the laminate plane. |
| $H$ | Schema. |
| $h_t$ | Average hamming distance between the individuals of generation $t$. |
| $K$ | Number of symbols in the alphabet used for coding. |
| $[K]$ | Assembled stiffness matrix of the structure under consideration. |
| $l$ | Chromosome length. |
| $L$ | Total number of independent laminates in the chromosome. |
| $\lambda_{cc}^j$ | Critical buckling load factor of laminate $j$ due to the combined effect of normal in-plane loads and in-plane shear loads (if there is only one laminate the superscript $j$ is omitted). |
| $\lambda$ | Load amplitude parameter. |
| $\lambda^*$ | Objective function for the fixed thickness laminate problem. |

| | |
|---|---|
| $\lambda_b(m,n)$ | Buckling load factor when the laminate fails due to normal in-plane loads with $m$ and $n$ half waves in the $x$ and $y$ directions, respectively. |
| $\lambda^j_{c\epsilon}$ | Critical failure load factor of laminate $j$ due to to strains (if there is only one laminate the superscript $j$ is omitted). |
| $\lambda^j_{cn}$ | Critical buckling load factor of laminate $j$ due to normal in-plane loads (if there is only one laminate the superscript $j$ is omitted). |
| $\lambda^j_{cr}$ | Critical load factor of laminate $j$ (if there is only one laminate the superscript $j$ is omitted). |
| $\lambda^j_{cs}$ | Critical buckling load factor of laminate $j$ due to in-plane shear loads (if there is only one laminate the superscript $j$ is omitted). |
| $\lambda_F$ | Critical failure load factor in the best known feasible design. |
| $\lambda_I$ | Critical failure load factor in the best known infeasible design. |
| $m$ | Population size. |
| $m_o$ | Optimal population size. |
| $\nu_{12}$ | Poisson's ratio for transverse strain perpendicular to the fiber direction when stressed in the fiber direction. |
| $n$ | Number of analyses for one genetic search. |
| $n_c$ | Number of stack of two plies in excess of the maximum number of two in half the laminate. |
| $n_e$ | Number of analyses where the extrapolation of the reliability curve starts. |
| $n_{es}$ | Number of excess unique schemata in the population. |
| $n_s$ | Number of unique schemata in the population. |
| $n_t$ | Total number of analyses in $p$ genetic searches of $n_t/p$ analyses each. |

| | |
|---|---|
| $N$ | Total number of layers in a laminate. |
| $N_{delay}$ | Number of analyses before the upper bound of $S$ starts being adapted. |
| $N_F$ | Number of plies in the best known feasible design. |
| $N_I$ | Number of plies in the best known infeasible design. |
| $N_t$ | Total number of layers in the skin of the wing. |
| $N_x^j,\ N_y^j,\ N_{xy}^j$ | In-plane loads in the $x$, $y$ and shear load on laminate $j$, respectively (if there is only one laminate the superscript $j$ is omitted). |
| $o(H)$ | Order of schema H. |
| $p$ | Number of independent genetic runs to solve one problem. |
| $p_o$ | Optimal number of searches. |
| $p_p$ | Predicted optimal number of searches. |
| $P_{aabs}$ | Probability that an allele is absent from the population. |
| $P_{apre}$ | Probability that an allele is present in the population. |
| $P_c$ | Penalty parameter for ply contiguity constraint. |
| $P_{down}$ | The upper bound of $S$, when decreased, is decreased by $P_{down}\%$ of $P_{up}^{max}$. |
| $P_l$ | Penalty parameter for failure. |
| $P_{up}$ | The upper bound of $S$, when increased, is increased by $P_{up}\%$ of $P_{up}^{max}$. |
| $P_{up}^{max}$ | Maximum value of the upper bound of $S$. |
| $\Phi_0$ | Objective function without fixed penalty for being infeasible. |
| $\Phi$ | Objective function with fixed penalty for being infeasible. |
| $pa$ | Probability of adding one stack of two plies per half laminate through mutation (because of symmetry, two stacks are added total). |

| | |
|---|---|
| $pc$ | Probability of crossover per laminate. |
| $pd$ | Probability of deleting one stack of two plies per half laminate through mutation (because of symmetry, two stacks are deleted total). |
| $pf$ | Probability of changing the fiber orientation through mutation per allele. |
| $pm$ | Probability of changing the fiber orientation through mutation per allele. |
| $pp$ | Probability of two stacks being swapped in one half of the laminate (i.e., swapping four stacks in the whole laminate because of symmetry), per laminate. |
| $ps$ | Probability of scaling per laminate. |
| $q$ | Number of symbols in the alphabet used in the coding. |
| $Q$ | Measure of the accuracy of an approximation. |
| $r_o(n_t)$ | Reliability of the genetic algorithm after a total of $n_t$ analyses when the number number of searches is carried out. |
| $r_p(n_t)$ | Reliability of the genetic algorithm after $p$ searches for a total of $n_t$ analyses. |
| $R_q(n)$ | Predicted reliability of the genetic algorithm after $q$ searches of $n$ analyses each. |
| $s$ | Maximum distance in percent to the optimal (largest) critical load factor $\lambda_{cr}$ of the global optimum for a feasible optimal weight design to be considered as practical optimum. $s$=$\infty$ means that any feasible optimum weight design is a practical optimum. |
| $S$ | Fixed penalty for being infeasible. |
| $S1, S2, S3, S4$ | Adaptive penalty parameters. |
| $t$ | Basic ply thickness. |
| $t_s$ | Takeover time. |
| $t_x$ | Mixing time. |

| | |
|---|---|
| $\theta_i^j$ | Fiber orientation in the $i$th ply of the $j$ laminate (if there is no ambuguity on the laminate, $j$ is omitted). |
| $\{U\}$ | Vector of nodal displacements in finite element analysis. |
| $V_{0A}^k,\ V_{1A}^k,\ V_{3A}^k$ | In-plane lamination parameters in laminate $k$ (if necessary). |
| $V_{0A}^*,\ V_{1A}^*,\ V_{3A}^*$ | Nondimensional in-plane lamination parameters. |

# Chapter 1

# INTRODUCTION: OPTIMIZATION OF COMPOSITE STRUCTURES BY GENETIC ALGORITHMS

Composite structures are typically made of laminates where distinct layers are stacked. Each layer is composed of fibers of a given orientation embedded in a matrix of a different material. Fibrous composites are usually manufactured in the form of layers of fixed thickness. Designing composite structures involves finding the number of layers and the fiber orientation of each layer inside each laminate that maximize the performance of the structure under constraints such as failure, geometry, cost, etc. Compared to isotropic materials, the use of fibrous laminated composites permits higher stiffness to weight or strength to weight ratios, along with a finer tuning of the response of the structure. This is achieved at the expense of a more complex structural analysis.

Plates made of graphite epoxy material are the basic structural components used, for example, in the skin of wings of modern aircraft. Much effort has been devoted to the design of composite laminated plates. A traditional approach to the design of laminates has been to rely on the mathematically well explored continuous optimization methods. Schmit and Farshi ([Schmit77]) have formulated the design of composite laminated plates as a continuous optimization problem with ply thicknesses used as design variables. As part of the solution process, the buckling constraints were sequentially linearized with respect to the ply thicknesses. Pedersen ([Pedersen87]) used fiber orientations in each layer as continuous design variables to maximize the critical buckling load of a balanced, symmetric, specially orthotropic angle-ply laminate. Analytical solutions indicated the presence of local optima.

Because of manufacturing considerations and tradition, not only ply thicknesses are fixed, but also ply orientations are often limited to a set of angles such as $0^o$, $90^o$, and $\pm 45^o$. Designing

the laminate for flexural response then becomes a stacking-sequence optimization problem where the number of plies of a given orientation and their location in the laminate need to be chosen. In general, stacking sequences are obtained using discrete programming techniques. The laminate stacking-sequence design problem for buckling optimization has been formulated first with the number of plies as the design variables, which leads to a nonlinear integer programming problem ([Mesquita87]). The use of ply identity design variables permits a linear integer programming formulation for buckling load maximization ([Haftka92b]). However, when strength constraints are also considered the problem becomes nonlinear again and has been solved as a sequence of linearized integer programming subproblems ([Nagendra92]). The Branch-and-bound algorithm [Garfinkel72] was used to solve the integer programming problems above.

Because continuous optimizers have a low computational cost (as compared to Integer Linear Programming or pseudo-stochastic search methods) and are widely available, stacking sequence problems have alternatively been treated using continuous optimization techniques. Layer thicknesses can be chosen as the continuous design variables for an assumed stacking sequence ([Nagendra93b], Appendix C). The optimum continuous solution is then modified by rounding the thicknesses up or down. In [Gürdal91], continuous optimization was coupled to a penalty function approach in order to obtain discrete values for the ply orientations. It was noticed that a substantial reduction in buckling load resulted from restricting fiber angles to a discrete pool and that the discrete designs were often different from the continuous designs. The penalty approach was often unable to reach global optimum designs for thick laminates. Finally, a graphical procedure based on the use of lamination parameters (through the thickness integrated functions of the ply angles) has been proposed in [Miki91].

Practical composite structures, like composite aircraft wings, involve many laminates that are mechanically coupled. In [Sobieski79], the skin of a wing box was optimized with constraints on panel buckling, strain, stress, and minimum gage. The skin was made up of sandwich panels. Ply thicknesses and orientations and the total sandwich thickness were taken as continuous design variables. In an attempt to break down the complexity of the optimization problem, panels were

optimized in isolation of one another: loads applied on each panel were extracted from a global analysis of the structure, and held constant during the independent optimization of each panel. The loads were updated only after all panels were sized, and the process was iterated. However, such independent resolution of the panels does not allow for optimizing the load path in the whole structure. This can results in final designs that are not optimal. The wing box optimization problem was approached as a whole in [Starnes79] and [Haftka83]. Continuous design variables were the skin ply thicknesses, spar web and rib thicknesses and the area of the spar caps. At most 57 design variables were considered. These studies demonstrated the need for including the constraints at a global level in order to obtain a feasible final design. In [Schmit80] and [Watkins88], other approaches can be found where the wing box problem is decomposed in a series of local optimization problems at the panel level, but care is taken to account for load redistributions. In [Schmit80], component thicknesses are the design variables. Ply orientations are additionally considered in [Watkins88].

Whether concerned with independent laminates or more complex structures, all the above design procedures have pitfalls. First, problems involving the flexural and the in-plane response of laminates are intrinsically nonlinear functions of the number of plies, the ply thicknesses, and the fiber orientations. This was true in all the previously mentioned works but [Haftka92b]. Therefore, the design space will contain local optima. The existence of local optima was analytically shown in [Pedersen87] for symmetric, balanced, specially orthotropic laminates designed for buckling load maximization. The risk of producing suboptimal designs has been further observed in number of works, for example in [Nagendra93b]. Sequential linearization is a traditional approach to nonlinear problems that may also get trapped by local optima ([Nagendra92]).

Second, composite structures often exhibit many optimal and near optimal designs. The reason is that composite laminate performance is characterized by a number of parameters which is usually smaller than the number of design variables, and so different sets of design variables can produce similar results, i.e., there are many optimal or near-optimal designs. In [Shin89], a large number of designs with almost identical performance are shown. Traditional methods have

not only the drawback of sometimes converging to suboptimal designs, but also of yielding only one solution. Traditional methods do not give the designer a choice between many nearly optimal designs.

Third, stacking sequence design is inherently a discrete problem since thicknesses must be multiple of the prepreg thickness and fiber orientations are limited to a discrete set of angles. For such discrete design problems, numerical methods based on derivatives cannot be applied directly. Strategies based on rounding of continuous design are used, that need to assume a stacking sequence a priori. There is also no guarantee that the discrete optimum is in the neighborhood of the continuous optimum ([Gürdal91]). For these last two reasons, successive continuous optimization and rounding may yield suboptimal or even infeasible designs.

Last, a composite laminated plate is described by a large number of design variables (ply thickness and orientation). A strategy based on rounding of a continuous design requires $2^n$ analyses to explore all the combinations of variables rounded up and down, where $n$ is the number of design variables. The Branch and Bound algorithm also exhibits an exponential dependence on the number of design variables $n$.

Probabilistic search methods offer an appealing alternative to "traditional" search algorithms such as gradient based continuous methods or enumerative integer programming techniques. Generally speaking, probabilistic search methods sample the design space based on probabilistic rules. Probabilistic search methods enjoy two theoretical advantages over traditional methods. They are global in scope since they have a nonzero probability of eventually reaching any point of the design space, and, for the same reason, they are insensitive to the problem nonconvexities and nonlinearities. Applications of probabilistic search methods to the design of composite structures are recent.

Simulated annealing has been applied to the buckling load maximization of a simply supported composite laminate ([Lombardi90]). The algorithm, once its probability rules tuned, was able to locate near-optimal designs with a high reliability in a design space of 6561 points in

less than 1000 analyses. When the size of the design space was increased to about 43 million, near-optimal designs were reliably found in about 3000 analyses.

In the present work, the use of another probabilistic search method called genetic algorithm (GA) is explored. Early implementation and theoretical analysis of the genetic search method are credited to Holland ([Holland75]). Genetic algorithms were originally developed to simulate adaptation as in natural systems. Applications to the optimization of static functions followed rapidly ([DeJong75]). Genetic algorithms, as considered here, are probabilistic optimization methods that work on a population of designs by recombining the most desirable features of existing designs. Following the concept of Darwin's theory of evolution, a selection is performed that favors the best fitted members of the population, and assures that they transmit their attractive features to the newly created designs. In the process, new design options are created and tested. The recombination of the designs is performed by coding the designs as strings and by using genetic operators that simulate biological reproduction on these strings. Genetic algorithms do not use any gradient information, and are insensitive to the complexity of the design space. A more detailed presentation of genetic algorithms is given in Chapter 2. Section 2.1 introduces the main concepts the genetic algorithm relies on. An example of a didactic genetic algorithm is described in Section 2.2. Section 2.3 gives a short theoretical framework for the analysis of genetic algorithms and Section 2.4 discuses how constrained optimization problems can be handled by genetic algorithms. In the last decade, genetic algorithms have proven their ability to deal with a large class of problems. Applications can be found in artificial neural networks ([Fullmer91]), geophysics ([Gallagher92]), social science ([Greene87]), control ([Kristinsson92]), biology ([Lucasuis91]), diagnosis ([Potter90]), and many other fields.

In structural optimization, the genetic approach appears especially promising in cases of large, nonconvex, integer programming problems (e.g., [Hajela90], [Hajela91], [Hajela92], [Rao90], [Furuya93], [Watabe93]). Not surprisingly, genetic algorithms have recently been applied to composite optimization problems. Besides the fact that they do not need gradient information and are insensitive to the complexity of the design space, genetic algorithms also have

the advantage of generating a number of optimal and near-optimal designs during a single search. The designer has then a larger choice in deciding what is the most suitable design. In [Sargent90], the applicability of the genetic search method to the design of laminate layups is studied. [Callahan92] gives an example of stacking sequence optimization by genetic algorithm where the discrete design variables are the ply orientations.

In spite of their theoretical advantages, genetic algorithms are often too expensive to use when the analysis of one candidate solution is computationally expensive. This often occurs in structural mechanics, for example when the structure is analyzed by a detailed finite element analysis. Despite the claim that genetic algorithms are global optimization methods, some cases can cause the search to converge to suboptimal solutions, especially if genetic algorithms are used blindly (cf. Section 2.3.2.). The objective of this work is to develop expertise on how to use a genetic algorithm for composite laminate stacking sequence design. We conjecture that by accumulating knowledge on the genetic search method and on the laminate stacking sequence problem, a genetic algorithm can be developed that would be an efficient tool for the design of composite structures.

As compared to other applications of GAs to the design of composite laminate layups ([Sargent90], [Callahan92]), this work is the first to systematically use a rational measure of the price of the search to motivate implementation choices. We compared a large number of GAs by extensive testings on various laminate design problems. These GAs differed by the genetic operators, the definition of the objective function, and the tuning of the genetic parameters they used. The results of this work are a set of rules on how to efficiently and reliably optimize structures made of composite laminates by GAs.

More precisely, to alleviate the problem of the cost of the genetic search, it helps to tune the genetic algorithm to the stacking sequence design problem by selection of specialized genetic operators, and by tuning their application probabilities. This tuning of the GA has to be performed on a simple version of the problem. It is undertaken in Chapter 3 for an unstiffened panel of given thickness designed for maximum failure load. By using simply supported boundary conditions, the analysis is reduced to a simple algebraic formula, so that millions of analyses could be carried out. A genetic operator specific to the stacking sequence problem called two-stack swap is introduced and its effectiveness shown. Results are presented and some peculiarities of the stacking sequence

design problem are discussed. A tuned GA is found that can reach optimum design with high reliability in about 1000 analyses.

However, in Chapter 4, when this algorithm is applied to the dual problem of minimum weight design subject to strength and buckling constraints, the price of the optimization goes over 3000 analyses. Unlike the maximum failure load design for given thickness, this problem requires variable string length. Second, constraints have to be represented by penalty functions. Chapter 4 describes several modifications to the algorithm developed in Chapter 3 and shows how these modifications improve the performance for laminate minimum weight design. Scaling concepts that yield fully stressed designs by resizing components are incorporated into the search through a "scaling mutation" operator.

In Chapter 5, the modified genetic algorithm is applied to the global optimization of a wing box-beam. The wing box-beam problem involves the simultaneous optimization of all laminates in the skin of the wing. It is a very difficult problem, because it combines the difficulties of the single laminate problem (local optima, nonconvexity of the design space) with a very large design space and a more costly analysis. Knowledge gained on the independent laminate problem is reassessed, by considering various simplifications of the wing box: the design space is progressively enlarged by increasing the number of independent laminates making up the skin (cf. Section 5.4.1). A simpler problem is also created by freezing the internal loads in order to estimate the effect of load redistribution on the genetic search. To alleviate the problem of the cost of analyzing the wing box, a polynomial approximation to the loads within the structure is implemented (cf. Sections 5.2.1). The consequences on the genetic search of using a polynomial approximate analysis are studied (cf. Section 5.4.2). With the wing box-beam, three new factors enter into play as compared to Chapters 3 and 4: the length of the string describing the design increases, the laminates are mechanically coupled, and a single string represents many laminates. The independent effect of each of these factors on the tuning of the algorithm is discussed in Sections 5.4.3, 5.4.4 and 5.4.5, respectively.

Chapter 6 offers a different look at the problem of minimizing the cost of the genetic optimization. The possibility of running several short searches instead of a long one is studied (Section 6.3). It is shown that there is an optimal division of the search into short runs that minimizes the optimization cost while keeping the probability of finding optimal designs (reliability

of the search) at a desired level. A procedure is proposed that permits predicting the reliability at later stages of the search based on the reliability earlier in the search (Section 6.4). Prediction of the optimal number of searches to achieve a certain reliability is discussed.

# Chapter 2

# FUNCTION OPTIMIZATION BY GENETIC ALGORITHMS

## *2.1 Introduction*

### 2.1.1 Properties of Genetic Algorithms

As was stated in the previous Section, genetic algorithms are attractive search methods because they are, theoretically, global in scope, insensitive to the function nonconvexities, and can yield many solutions. We describe here the mechanisms that give GAs these unique possibilities.

Golberg [Goldberg89] brings out four properties of GAs that distinguish them from other "traditional" search methods: GAs work on a population of design points at once, and not a single point; GAs work on coded design variables, and not the design variables themselves; GAs need only payoff information (objective function value for optimization), and no other type of information (for example gradients); GAs use probabilistic transitions, as opposed to deterministic transitions.

GAs handle a population of design points at the same time. Like a tabu list ([Glover89] and [Glover90]), this population represents a constantly updated genetic memory of the problem at hand that stores some highly performing features of the designs. It enables the GA to cover many regions of the design space at once, thus protecting it some from convergence towards a local optimum. Furthermore, the GA does not yield one single solution to a problem, but a population of solutions that can vary from each other. The argument that GAs converge to the global optimum is mathematically correct only for infinitely large population sizes or an infinitely large number of analyses. It is shown by the theorem of convergence for generational methods that include GAs ([Zhigljavsky91]). In practice, populations have limited sizes so that the GA might not always find the optimum.

The second special property of GAs is that they work on coded design variables, and not the design variables themselves. If, for example, the purpose of the optimization is to minimize $f(x, y)$, the GA will usually not handle $x$ and $y$ directly, but a coded version of them that typically will be a string of characters. A simple way of coding continuous variables is to use binary coding and append the two binary numbers on the same string. Various other coding schemes can be used. The choice of the coding is crucial to the efficiency of the genetic search because the coding, in conjunction with the genetic operators that will be described later in this Chapter, composes the mechanics by the which the GA extracts information from already sampled points to create new design points.

Third, GAs need only payoff information, unlike most continuous optimization techniques which are based on derivative information. For some problems, gradients can be difficult to calculate accurately. Also, "hill-climbing" search techniques based on gradients are inherently victim of convergence to local optima since they will climb the hill on which the search is initiated, irrespective of whether this hill contains the global optimum. An example of a function having with many local optima and where gradients will probably be difficult to calculate accurately is given in Figure 2.1.

Last, GAs rely on probabilistic transitions, as opposed to deterministic transitions, which means that some of the decisions taken during the search are made by drawing pseudo random numbers and using some probabilistic rule of acceptance. Unlike gradient based methods, if a genetic search is started twice from the same population, it will take different paths during the search. Probabilistic transitions permit the GA to escape local optima by inducing random moves in the design space.

### 2.1.2 Terminology

Many ideas that led to the development of genetic algorithms were inspired from genetics and the evolution of populations (see, for example [Fraser62]). Holland's early work, which is widely recognized as the stepping stone to the development of GAs, aimed at simulating adaptation on artificial systems [Holland75]. Thus historically, genetic algorithms have been thought of as simulations of nature and were first explained using terms from the fields of genetics and evolution. Although GAs should not be restricted to simulations of biological processes, the

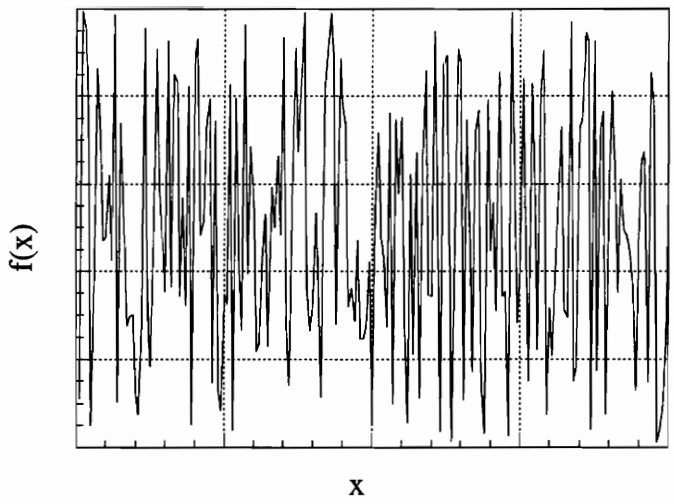



**Figure 2.1. Example of a function where gradient based methods are not suitable.**

parallelism between GAs and biology is an attractive way of describing an algorithm and has proven to be a real source of inspiration. For these reasons, we will employ here the biological terminology traditionally used by theorists of genetic algorithms. The following is a list of the most commonly used terms.

**population** Set of design points stored and used at one time of the search by the algorithm. This set will also be referred to as a **generation**.

**individual** One of the design points in the population.

**chromosome** Coded representation of one design point used by the GA to handle design variables. From Chapter 3 on, a chromosome will be a string of characters describing the stacking sequence of one or several composite laminated plate. A chromosome is composed of genes, which can take on different values or alleles.

**gene** One feature of a design, that can take on several values (see allele). For example, in the case of the stacking sequence of a laminated plate, one gene can be the orientation of the fibers in the outermost layer.

**allele** Feature value of a gene. In our laminated plate example, an allele would be one fiber orientation (say $90^o$) associated to one layer (a gene).

**locus** Position of a gene on a chromosome. For example, the locus of the gene "fiber orientation in the outermost layer" could be the leftmost digit of a string.

**genetic diversity** Measure of how diverse the alleles associated to each gene are in the population. If all the individuals have the same chromosome, there is no genetic diversity. A possible measure of genetic diversity is the average distance between individuals in a population.

**genotype** In biology, it is the noninterpreted group of chromosomes that code one individual. Humans for example have 23 pairs of chromosomes. Here a design will always be associated to a unique chromosome, so genotype and chromosome are synonymous. Thus, the genotype of a laminated plate is the string associated to it and the genotype can also be seen as the noninterpreted chromosome.

**phenotype** Decoded description of a design point (interpreted chromosome). In chapters 3 to 6, the phenotype describes the real laminated plate. It describes how many layers of what thickness are present and what is the fiber orientation in each of those layers.

**fitness** Value associated to each individual design that determines the probability of the individual of being selected for reproduction. Better designs have higher fitnesses. It is associated to the objective function value of the individual under consideration in relation to the objective function value of other individuals in the population through some mapping.

Last, the term **convergence** needs to be defined because its meaning in the context of genetic search is not well established. In this report, the term convergence describes the state of a genetic algorithm that has stopped making fast progress. Figure 2.2 illustrates, qualitatively, our definition of convergence. This definition of convergence is vague but practical, because the transition

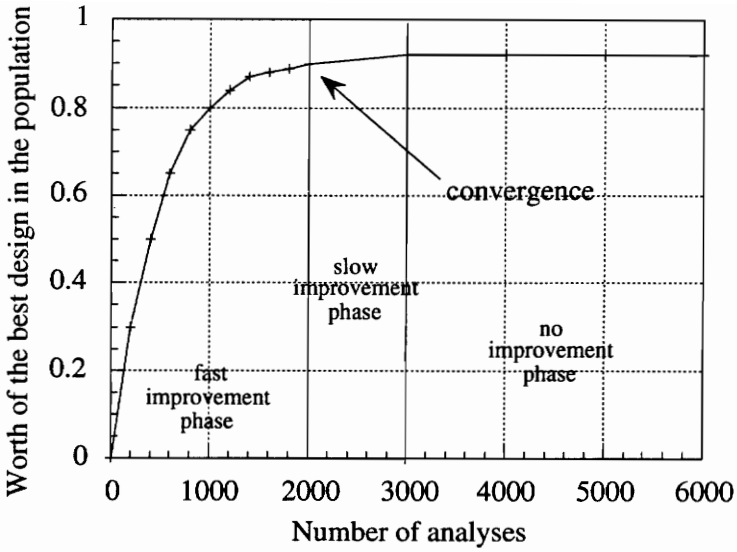



**Figure 2.2. Illustration of convergence of an elitist genetic algorithm †**
**† a GA where the best so far solution is never lost, unless a better solution is found (cf. Section 2.6.2) .**

between fast and slow improvement phases can be observed, unlike the transition between slow and no improvement phases. There are two states in which the elitist genetic algorithm (a version of the GA that does not regress, cf. 2.5.2 for more details) stops making progress: either the global optimum has been found, or the algorithm has stalled on some nonoptimal solution. Even though this last state is theoretically only a transient one (since a GA will always find the global optimum after an infinitely long time, [Zhigljavsky91]), it practically lasts long enough to be considered as a termination state. Irrespective of whether the global optimum is found or not when the GA stops improving, the final population is essentially composed of similar chromosomes. The word essentially is important here because, in practical implementations, the population is never strictly uniform. The amount of genetic diversity when the GA stops making progress is case dependent, which prevents us from using a more precise definition of convergence.

## 2.2 A Didactic Genetic Algorithm

The basic genetic algorithm and the problem used as examples in this Section are chosen on purpose to be simplistic and are meant to set the stage for the theoretical considerations presented in the next Section. Most of the theory and models currently available deal with simplified GAs, similar to the one presented here. The reader should be aware that GA implementations aimed at being efficient on real optimization problems differ substantially from this basic GA. Thus for practical implementations, the theory of GAs is most usually extrapolated, without much analytical justification, to real world problems. In this work, we will proceed very much in the same vein, by first describing a simple GA, then stating the main theoretical results governing its behavior, in order to finally generalize these results to more complex GAs used to optimize composite structures.

The problem used as example is the minimization of

$$f(x,y) \;=\; x^2 + y^2, \qquad x, y \text{ integers } \in [0,15].$$

For the purpose of the example, the design variables $x$ and $y$ are encoded as two binary strings of four bits each, that are then juxtaposed to form a chromosome. For example, the design point $x = 3$, $y = 10$ would be encoded as, 00111010 (0011 juxtaposed to 1010).

As it can be seen from Figure 2.3, the GA starts with the creation of an initial population of designs, usually at random. Each individual is evaluated and is assigned a fitness, depending on its objective function value and the objective function value of other individuals in the population. Next, pairs of designs are selected, with a probability depending on their respective fitness, to be mated. Mating is simulated on the coded chromosome associated to each design using a group of genetic operators. Crossover and mutation being the two principal genetic operators. Mating produces offspring designs that should both recombine features of the parent designs and bring in some new design concepts. The cycle selection-reproduction of parent designs is repeated until there are enough offspring designs to create a new population. The population of new designs replaces the old one, which completes one iteration of the GA. It can then be decided either to stop the search or create a new generation of designs. More details about the implementation are given next.

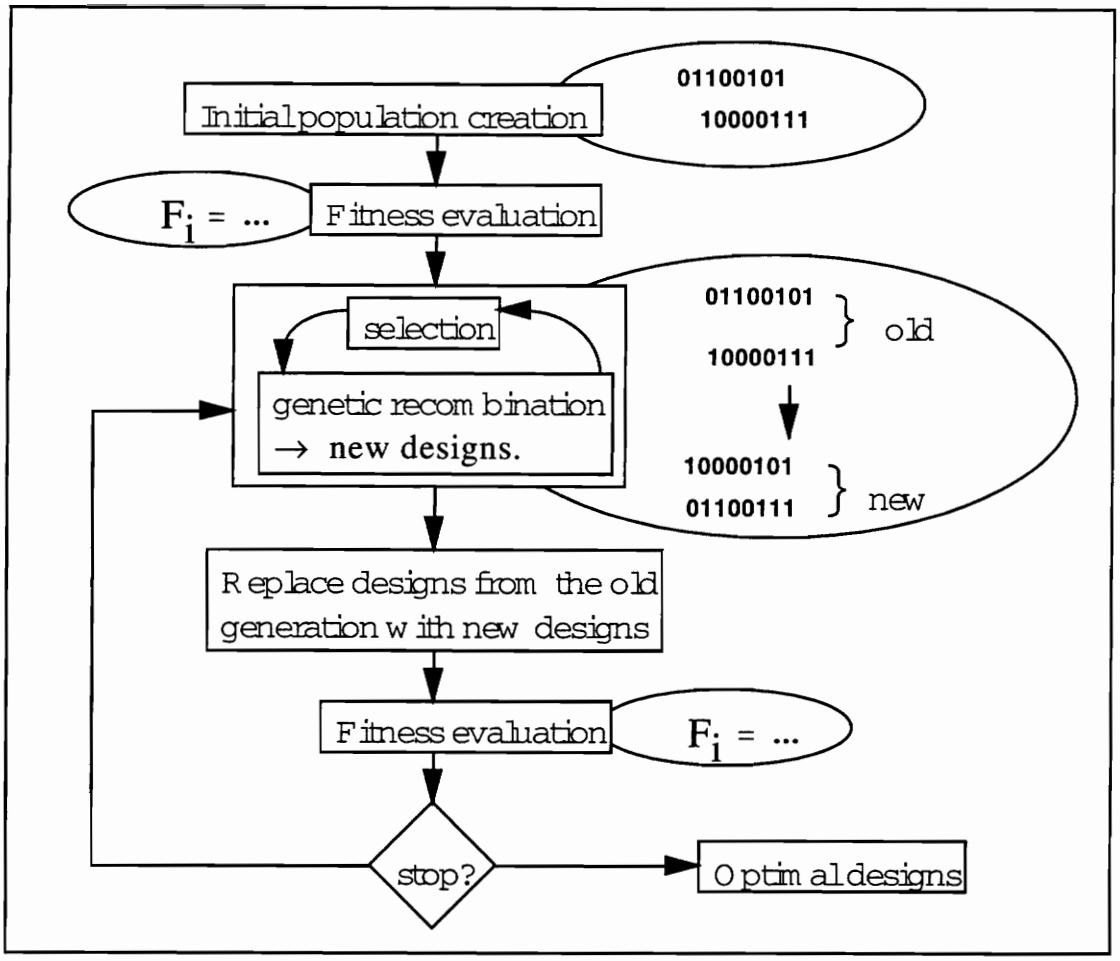



**Figure 2.3. Flow Chart of the basic genetic algorithm.**

### 2.2.1 Selection and Fitness Calculation

In this particular case, we define the fitness associated to the $i$th design as being, for example

$$F_i = 450 - f(x_i, y_i)$$

and the normalized fitness is

$$\bar{F}_i = \frac{F_i}{\sum_{j=1}^{m} F_j}. \tag{2.1}$$

where $m$ denotes the population size, so that the sum of the normalized fitnesses is equal to one. The normalized fitness is the probability the design has of being selected. A design can be chosen many times for reproduction. Even the worse design has a nonzero probability of being selected. Figure 2.4 shows how selection works. Selection can be implemented by allocating to each design $i$ in the population a portion of the segment [0-1] of length equal to $\bar{F}_i$. Then a

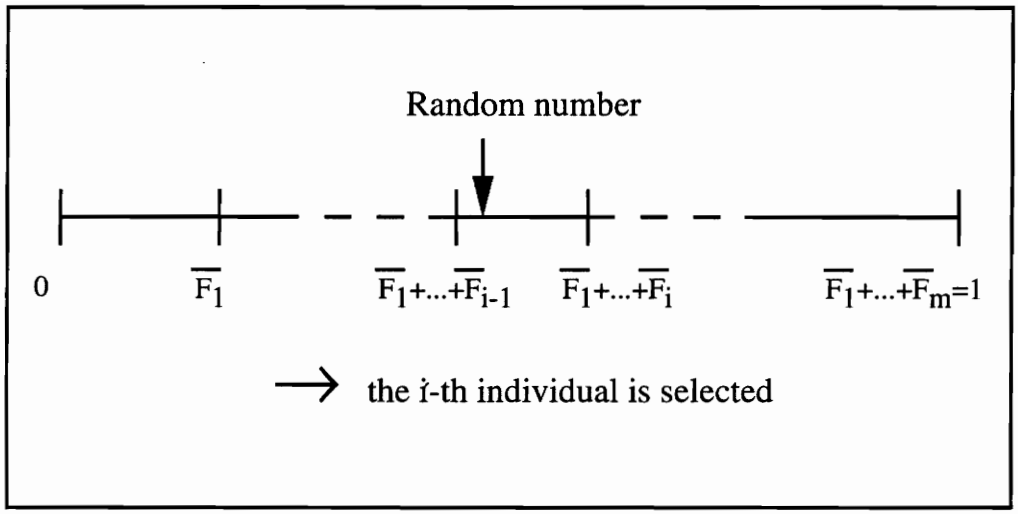



**Figure 2.4. Selection procedure.**

random number is generated between 0 and 1. The individual corresponding to the portion where the random number falls is selected.

### 2.2.2 Stopping Criterion

The genetic search can be stopped after a given number of function evaluations has been performed without improvement. This stopping criterion is well adapted to genetic algorithms, and is further discussed in Chapter 6 where the possibility of restarting the search is explored. A simpler stopping criterion is to set an upper bound on the total number of function evaluations in the search. This last stopping criterion may be preferred when testing a GA by repeating independent searches because the searches always stop after the same number of analyses. This helps when calculating statistics about the GA at the end of the search. It was the stopping criterion used in Chapters 4, 5, and 6 of this work.

### 2.2.3 Crossover

Crossover allows selected individuals to trade some of their characteristics by exchanging

parts of strings. During one-point crossover, one break point is randomly chosen in the string. Two offsprings are created by swapping the parents' substrings. Crossover is applied with a given probability, usually between 0.6 and 1.0. If crossover is not applied, parents are copied into the next generation. Figure 2.5 is an example of a one-point crossover.

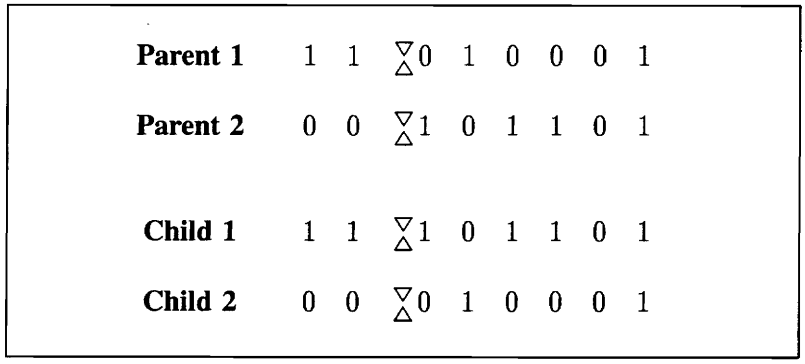



**Figure 2.5. One-point crossover.**

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

## 2.2.4 Mutation

Next, one of the two children is randomly picked and endures mutation. Mutation is a stochastic operator, usually applied with a low probability. It randomly changes some genes in the chromosome. A typical probability of mutation is of the order of one percent or less per gene. The first purpose of mutation is to protect against a complete loss of genetic material. Indeed, under the action of selection, the variety of alleles for any given gene in the population diminishes and the chromosomes all tend towards the best known chromosome. Alleles associated with lower fitness individuals are usually not transmitted to the next generation. They could disappear from the population even though they might be needed later in the search to locate the optimum. The problem is that, once the population is mainly uniform, crossover loses its ability to create new designs, and the search stalls. Thus by keeping some genetic diversity in the population, mutation helps preserving the ability of crossover to find new good designs. Mutation also directly discovers new positive features in the design through random search. Because mutation

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Before mutation** | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| **After mutation** | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

**Figure 2.6. Mutation.**

is completely random, its efficiency at directly discovering better designs decreases as the best-so-far design improves. Figure 2.6 gives an example of mutation.

The parameters associated with the GA are called **genetic parameters**. They define the working of the algorithm. Typical genetic parameters are the population size $m$, the probability of crossover $pc$, and the probability of mutation $pm$. Later in this text, as new operators are introduced, new probabilities will be added to the list of genetic parameters.

## *2.3 A Theoretical Framework for Genetic Algorithms*

The theoretical results presented hereafter contain numerous idealizations, most notably that they apply to binary strings. They are thus valid only in a qualitative way for practical implementations of GAs, and give the analytical background that explains the behavior of GAs.

### 2.3.1 Schemata and the Schema Theorem

Genetic algorithms process what is usually called schemata ([Holland75]), i.e., subsets or templates of all the design space spawned by the chromosome. Schemata are described using a "*" symbol, also called the "don't care" symbol. The "*" symbol comes from the theory of formal languages. In its most general meaning, it denotes any group of digits. Here however, in conformity with traditional use in the genetic algorithms community, "*" represents only one bit, i.e., "*" is either a "0" or a "1". For example, the schema 1*0000*1 describes the subset (10000001,11000001,10000011, 11000011). A particular string of length $l$ belongs to $2^l$ schemata, since each string position can either have its value or be a "*" symbol. Thus, a population of strings exemplifies between $2^l$ and $m2^l$ schemata (out of a total of $3^l$ schemata), depending on the

genetic diversity. John Holland has developed an argument called "implicit parallelism" according to which the GA usefully samples $m^3$ schemata per generation, even though only $m$ structures are analyzed.

Two quantities are used to characterize a schema: its order and its defining length. The order $o(H)$ of a schema $H$ is the number of fixed positions in the string. In the above example, the order of the schema was six. The defining length $d(H)$ is the distance between the first and the last fixed positions. In the above example, the defining length was equal to 7 since the first and the eighth positions were specified. The defining length of the schema $H$ = 1******* is zero and its order is one.

Let us now consider a genetic algorithm made up of selection alone. $m(H,t)$ represents the number of instances of schema $H$ at generation $t$. $F(H,t)$ represents the average fitness of the instances of $H$ at generation $t$, and $\bar{F}(t)$ is the average fitness in population $t$. The expected number of instances of $H$ at generation $(t+1)$ under the effect of selection alone is

$$m(h,t+1) \;=\; m(H,t)\frac{F(H,t)}{\bar{F}(t)}. \tag{2.2}$$

This equation tells us that a particular schema grows (or decays) in the population as the ratio of its average performance to the average performance of the population.

We now modify Equation (2.2) to include the effects of crossover and mutation. Let's consider mutation first. Mutation alters the value of an allele with a probability $pm$. The probability that an allele is not changed by mutation is $(1-pm)$. The probability that a schema of order $o(H)$ is not lost through mutation of one of its defining position is

$$(1-pm)^{o(H)} \;\approx\; 1 - o(H)pm, \tag{2.3}$$

if the probability of mutation $pm$ is small. The opposed event of having the schema lost by mutation has a probability equal to

$$1 \;-\; (1-pm)^{o(H)} \;\approx\; o(H)pm, \tag{2.4}$$

and is called probability of disruption by mutation. Crossover destroys a schema if both parents do not carry that particular schema and the break point falls between the two extreme defining

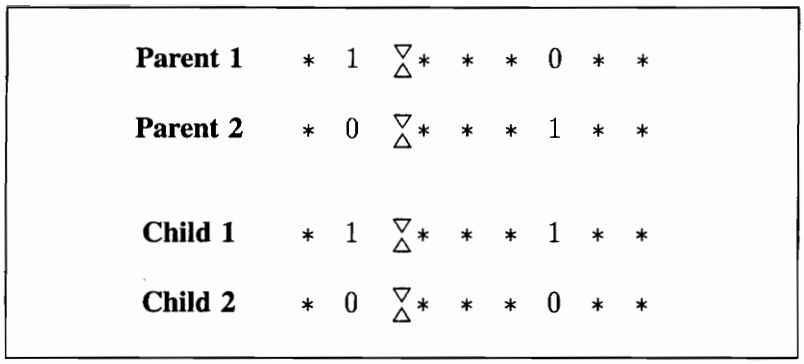



**Figure 2.7. Schema disruption by crossover.**

positions (see Figure 2.7). The probability that crossover disrupts the schema $H$ that is not being carried by both parents is

$$pc \frac{d(H)}{l-1}. \tag{2.5}$$

There are cases however where both parents carry the same schema, so, even if the break point falls between defining points, the schema is preserved in the child design. Also, mutation and crossover produce new instances of the schema $H$. Therefore, by removing from the count of instances of $H$ at generation $(t+1)$ the instances that are disrupted by mutation and crossover as estimated in Equations (2.4) and (2.5), we obtain a lower bound on the expected number of instances of $H$ at generation $(t+1)$,

$$m(h,t+1) \geq m(H,t) \frac{F(H,t)}{\bar{F}(t)} [1 - pc \frac{d(H)}{l-1} - o(H)pm]. \tag{2.6}$$

Equation (2.6) is called the **Schema Theorem**. It has important repercussions on the way GAs are implemented. The main result stemming from the Schema Theorem is that, if crossover and mutation do not excessively disrupt schemata, a schema with an above average fitness has an exponentially growing number of instances in the population. By neglecting the disruptive effects of crossover and mutation and assuming that the ratio $(\frac{F(H,t)}{\bar{F}(t)})$ remains approximately constant over the generations, one can write

$$m(h,t) \approx m(H,0)(\frac{F(H,t)}{\bar{F}(t)})^t. \tag{2.7}$$

Implicit parallelism and the exponential growth of above average schemata in the population are two arguments that explain the ability of the genetic algorithm to break down the combinatorial complexity of the design space. We now reinterpret the schema theorem qualitatively to deduce conditions for the schema theorem to apply (and thus guarantee the efficiency of the genetic search).

Equation (2.6) gives conditions for a schema not to be excessively disrupted: mutation should be applied with a low rate $pm$ and/or the order of the important schemata $o(H)$ should be small. Important schemata should have a short defining length $d(H)$ so as not to be too often destroyed by crossover. This leads to another important concept for genetic algorithms, namely the idea of **building blocks**. According to the schema theorem, a GA processes efficiently short, low order, highly performing schemata called building blocks. This is referred to as the Static Building Blocks Hypothesis, where the term "static" is added to outline that the effect of having a finite size population dynamically moving in the design space is not considered here (we have assumed that $(\frac{F(H,t)}{F(t)})$ does not change too much with time).

### 2.3.2 What Makes a Problem GA-hard?

All the previous theoretical properties draw too optimistic a picture of the real workings of GAs, mainly because they ignore the finite population size. If we account for the fact that the population is not infinite, we can no longer take the term $(\frac{F(H,t)}{F(t)})$ as constant over the generations, because the average relative performance of a schema in the population will strongly depend on the composition of the population. A common problem of GAs occurs when the (finite size) population loses most of its genetic diversity under the domination of one super-individual. If this individual is not optimal and is far (in the coded space of the chromosomes) from the global optimum, many of the building blocks leading to the optimum solution will have lost all representatives in the population. At that point, the only way these building blocks can be recovered is through a series of lucky mutations and selections, but such an event is very unlikely. So, once the genetic algorithm has converged to such a nonoptimal solution, it stops making progress. One says that the GA has **prematurly converged**. Apart from premature convergence, the other state in which a GA can be stopped without having located the optimal solution is when it has not converged yet, what we refer to as **unattainable convergence**. An example of unattainable convergence will be given in Chapter 5 where a very large chromosome

is considered (it has 720 genes with 4 different possible alleles at each gene). The combination of a large design space with a large population size makes the time to convergence very long, beyond the maximum number of analyses that a computer can perform.

What makes some problems hard for the GA to solve is a difficult yet crucial question that has drawn much attention in the last years. Even the existence of tight building blocks does not guarantee that a problem is easy for the GA, despite the Schema Theorem.

Historically, the first reason that has been proposed to explain the difficulties experienced by GAs on some problems was called **deception** ([Bethke81], [Goldberg87]). Deceptive problems have a response surface that will lead the GA away from the optimal region of the design space. A simple deceptive problem can be described by considering the average fitness of a schema over the entire design space $F(H)$ (as opposed to the observed average fitness of a schema in a population, $F(H,t)$). If the optimum solution to the problem is 00...0, and

$$\begin{aligned} F(0*...*) &< F(1*...*) \\ F(00*...*) &< F(01*...*) \\ F(00*...*) &< F(10*...*) \\ F(00*...*) &< F(11*...*) \end{aligned}$$

and so on, then the GA clearly will be pushed away from the optimum solution 00...0 as it is juxtaposing building blocks. In fact, most algorithm will have difficulties converging to the optimum. Such a problem is deceptive. However, deception does not fully explain the problems encountered by genetic algorithms. Goldberg has noticed that GAs performed surprisingly well on some deceptive problems ([Goldberg89]):"It is surprising that all type II problems" (a kind of deceptive problem) "converge to the best solution for most starting conditions". Grefenstette has proposed a very simple anti-deception GA based on the idea that the complement of the best known individual should always be kept in the population in order to always be able to recover from a false premature convergence ([Grefenstette93]).

Recently, two other reasons have emerged to explain false convergence of GAs: collateral convergence ([Grefenstette93] or hitchhikers in [Forrest93]) and large variance within schemata ([Grefenstette93]). These reasons, unlike the Static Building Block Hypothesis and the notion of deception, emphasize the dynamics of the convergence of the population.

Collateral convergence refers to schemata growing simultaneously and influencing each other's rate of growth. For example, suppose that $F(0*...**) = 9$, $F(1*...**) = 1$, $F(**...0*) = 6$, $F(**...1*) = 4$, that the initial population is randomly chosen and sufficiently large to reflect those average fitnesses. Suppose further that the optimal solution to the problem is 00...00. Note that the problem is not deceptive, since the average performance of the schemata $0*...**$ and $*...0*$ leads the search in the right direction. According to the schema theorem, we expect the schema $0*...**$ to occupy 90 percent of the second generation and the schema $**...0*$ to occupy 60 percent of the second generation. From the second generation on, however, the population is highly biased due to the growth of highly fitted schemata. Because of this collateral growth of schemata, it becomes impossible to apply the Schema Theorem for the schemata $0*...**$ and $*...0*$ after the second generation. For example, we expect 90% of the instances of $*...0*$ to be also instances of $0*...**$, that is $0*...0*$. If the schema $0*...0*$ does not perform well, the growth of the $**...0*$ schema might be jeopardized. Taking this collateral convergence into account, we cannot predict, on the basis of the static average performance of low order schemata, whether low order schemata will grow as the Schema Theorem postulates. The collateral growth of different schemata in the population can bias the population so that a schema containing the optimum is seen as performing poorly. If consecutively, it disappears from the population, chances are that the GA will not find the optimal design.

"Hitchhikers" are nonoptimal schemata growing in the population as a result of collateral convergence. Hitchhikers are located next to highly performing schemata so that the probability of them being separated by crossover from high performance schemata is very low. If, in the above example, the schema $011*...*$ take over the population due to the very high fitness of $0*...*$ , the alleles 11 are hitchhikers.

The last recognized reason for the GAs failing to converge to the optimum is the noise in schema evaluation. Simply stated, because the number of individuals in the population is limited, the real (static) worth of some schemata is not always well represented, especially when the fitness of these schemata varies a lot. For example, the observed average fitness of a schema at the generation $t$, $F(H,t)$, might be so underestimated with respect to the absolute average fitness $F(H)$ that the schema $H$ disappears in the subsequent generations while it contains the optimum. An extreme example of a function exhibiting large variance in fitness is the peak function: the peak function value is zero except at one point of the design space. The GA will not find the

optimum, unless, by chance, the search hits the one point with a nonzero objective function. The GA cannot extract any information on what schemata lead to the optimum because of the extreme variance in fitness between the optimum and other points close to it. Note that the schemata containing the optimum have an average fitness above the other schemata, so the peak function is not deceptive. The peak function is an extreme example, but other functions exhibit large enough variances in schemata average fitness that can mislead the GA, especially when small samples of points are used (small population size).

### 2.3.3 Exploration versus Exploitation

Premature and unattainable convergence are problems that can be seen as deficiencies in the balance between the explorative and the exploitative trends of the algorithm. Holland ([Holland75]) uses the terms exploitation and exploration to describe the tradeoff that exists between too little and too much randomness in the search. Exploitation denotes the ability of the algorithm to carefully extract positive features from the already sampled points of the design space. Exploration is the ability of the algorithm to randomly depart from already sampled regions of the design space. A GA that converges prematurly misses exploration factors. A GA that does not converge is too explorative and not exploitative enough. All the global random search methods have to deal with the trade-off between exploitation and exploration. For example in the Simulated Annealing algorithm ([Kirpatrick83]), an explorative version of the algorithm would have a slow decrease in temperature, and vice versa for a more exploitative algorithm. With the Tabu Search ([Glover89], [Glover90]), the balance between exploration and exploitation has to do with the length of the tabu list. Most of the implementation choices (coding, operators, genetic parameters) influence the balance exploitation/exploration in the GA. We list hereafter the effect of changing various probabilities on that balance.

The selection pressure varies depending on the selection scheme used and is the most straightforward way of controlling exploitation. A high selection pressure gives more opportunities to the best individuals to reproduce to the detriment of the other designs. For example, selecting according to the square of the objective function is more exploitative than selecting according to the objective function. Numerous variations on the selection procedure are possible, and some of them are discussed in Chapter 3. Increasing the mutation probability is a direct way of increasing exploration during the search. The same applies to the probability of crossover, since crossover,

by disrupting existing schemata, also contributes to the exploration of new portions of the design space. At last, increasing the population size induces a more random search, because the ratio of the total number of analyses made so far over the number of times an individual has been selected (equal to the number of generations) increases.

When implementing a GA on a particular class of problems, the purpose of the algorithm designer is to maximize the exploitative trend synonymous of efficiency without breaking the global balance of the search. Finding exploitative mechanisms that do not excessively jeopardize the globality of the search is the challenge when designing a specialized GA. Devising a GA for a class of problems is more complex than finding the right balance between exploration and exploitation by tuning the genetic parameters. Problems such as the choice of coding and genetic operators also enter into account. But a global reasoning on exploration and exploitation can be useful for diagnosing what needs to be changed to improve the algorithm. For example, an algorithm that does not converge in the allocated time suffers from too much exploration. So, neither making the population larger nor increasing the probability of mutation would help (unless other changes are simultaneously made).

## *2.4 How to Handle Constraints in Genetic Algorithms*

Constraints in GAs are handled in the following three ways: data structuring, repair operators, and penalty functions.

### 2.4.1 Constraint Removal by Data Structuring

Data structuring refers to defining the design variables such that the optimizer always produces feasible designs. For example, if there are linear equality constraints, one can eliminate one design variable per equation to keep the equality satisfied. This is not particular to GAs, and is just a way of removing constraints by reformulating the problem. But there are ways that are more specific to GAs that can be used to remove constraint, like the choice of coding. To illustrate this, suppose we are using binary coding to handle integers, and that all design variables are required to be even numbers. By removing the rightmost bit from the chromosome and assuming it always has a value of 0, one makes sure that the only integers that the GA

can ever produce are even numbers. Michalewicz and Janikow ([Michalewicz91]) have defined genetic operators that always produce points in the feasible convex domain for the special case of linear constraints. Although data structuring is always the most efficient strategy, it is not always possible to apply it.

## 2.4.2 Repair Operators

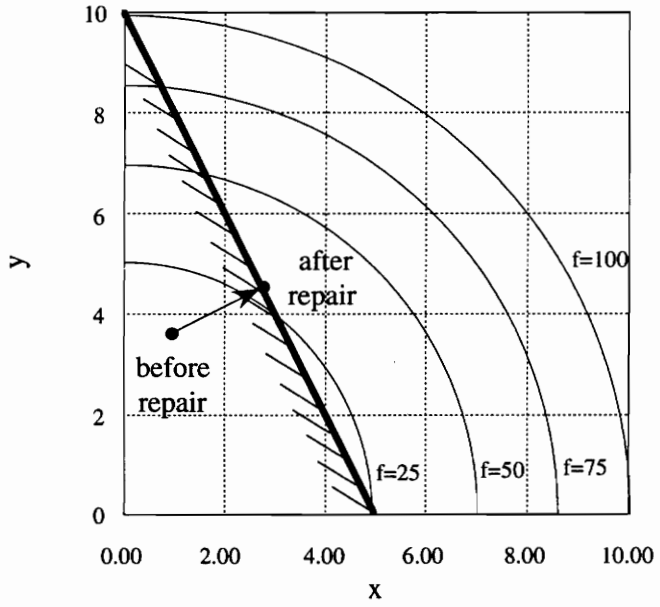



**Figure 2.8. Example of repair.**

The second way of handling constraints in GAs is through the use of repair operators. A repair operator can be thought of as a projection from an infeasible point in the design space to a feasible point. Figure 2.8 gives an example of repair. The purpose of the optimization is like in the didactic GA of Section 2.2, to minimize $f(x, y) = x^2 + y^2$, but here we add the constraint $2x + y - 10 \geq 0$. A simple repair shown on Figure 2.8 is to project infeasible points onto the constraint. Repair operators are usually applied probabilistically so as to increase the probability of finding a feasible design at the end of the search, but not reduce the GA to a heuristically based, local in scope, search method. Orvosh and Davis ([Orvosh93]) recommend using repair operators on 5% of the points sampled for combinatorial problems. Repair operators, like data

structuring, are very case specific. A repair operator called "scaling mutation" will be described later in this work.

### 2.4.3 Penalty Functions

Finally, constraints can be incorporated into the objective function through penalty functions. The use of penalty functions is the most general technique for constrained genetic optimization. For continuous optimization, the quadratic penalty function is the most popular, but this popularity is due to differentiability requirements, which are not relevant in genetic optimization. Conventional wisdom in GAs is that the penalty should be proportional to the distance to the feasible domain, so as to discriminate between designs having different levels of infeasibility ([Richardson89]). It has also been noticed that the penalty should be kept as low as possible, just above the limit below which infeasible designs become optimal ([LeRiche93], [Richardson89]). The flaw of heavily penalizing infeasible designs is that it limits the exploration of the design space to feasible regions, precluding short cuts through infeasible regions. Large penalties promote convergence to local minima in the feasible space, that is, large penalties are a cause of premature convergence. Thus it may be worthwhile to tune the penalty functions on each problem case to assure convergence in the feasible domain but still permit short cuts leading to the optimum design through the infeasible domain. There is no general solution to the problem of adjusting penalties, neither in classical numerical methods, nor in GAs. In [Schoenauer93] however, a strategy based on the Behavioural Memory Paradigm is proposed which assumes that there is some kind of memory of the topology of the design space in a population that has evolved. The idea is then to have the population satisfy the constraints one after each other. Once a constraint is satisfied by all members of the population, it is assumed that the population as a whole remembers this constraint, so there is no need to worry about it anymore. Such a strategy is equivalent to having a very large penalty put on one constraint until all the members of the population satisfy it, after which the penalty associated to this constraint could be removed. This procedure was not used later in this work, because there are complications in the implementation of the algorithm in order to maintain genetic diversity. Moreover, the Behavioural Memory Paradigm violates the minimum penalty rule of [Richardson89] and [LeRiche93], so that we do not think it would be a very efficient procedure.

Among the three aforementioned strategies for handling constraints, data structuring is always the most efficient strategy because the design space is, from the outset, reduced to the feasible domain. Having a very large penalty does not prevent the algorithm from wasting time generating infeasible designs that get no (or very little) chance to reproduce next. Another advantage of data structuring is that, contrary to repair operators and penalty functions, there is no parameter associated to it, so no tuning involved. However, data structuring is case dependent, so one cannot always resort to it. The choice between repair and penalty functions depends on the problem at hand and the implementation.

## *2.5 Other Topics of Interest*

### 2.5.1 Coding and Reordering Operators

Two arguments enter into account when choosing a coding scheme. First, because GAs process schemata, it has been argued ([Holland75], [Goldberg89a]) that the coding should maximize the number of schemata per individual. This is synonymous to using the smallest possible alphabet, i.e., a binary alphabet to maximize the length of the chromosome. For example, suppose we code integers between 1 and 8. With a decimal alphabet, one integer can be described by one character, so there are 2 schemata per individual (the decimal character and the * don't care symbol). With the binary alphabet, one individual is described by a 3 bits string, so there are $2^3 = 8$ schemata per individual.

The other argument is based on the Static Building Blocks Hypothesis. It says that the coding of the design points should be made in such a fashion so as to favor the existence of building blocks, i.e., tight low order schemata that contribute to the performance of the optimum string. In most practical cases however, it is not possible to know beforehand what kind of coding will promote building blocks.

The two previous methods for choosing the coding can lead to different decisions and have caused some controversy. While there is mathematical evidence that a 2 character alphabet is better ([Holland75], [Goldberg89a]), higher cardinality alphabets that promote tighter and easier

to recombine building blocks have often performed better on practical experiments ([Davis91], [Michalewicz91]). Chapter 3 adds to this issue.

Reordering operators like inversion ([Holland75]) have been proposed as solutions to the problem of the hitchhikers and as a way to look for the right coding. Reordering operators change the location of the genes on the chromosome, without altering the phenotype associated to the chromosome. Refering back to the previous example, the individuals

| | | | | | | |
|---|---|---|---|---|---|---|
| chromosome | 0 | 1 | 1 | $*$ | ... | $*$ |
| gene identifications | 1 | 2 | 3 | 4 | ... | $l$ |

| | | | | | | |
|---|---|---|---|---|---|---|
| chromosome | 0 | $*$ | ... | $*$ | 1 | 1 |
| gene identifications | 1 | 4 | ... | $l$ | 2 | 3 |

are identical, but the probability that crossover separates the optimal $0 * ... *$ schema from the 11 hitchhikers is much higher with the second individual than it is with the first individual because of the distance separating the hitchhickers from the defining positions of the highly fitted schema. Reciprocally, inversion might bring together genes that are coupled so as to avoid separating them too often during crossover. Messy Genetic Algorithms ([Goldberg89b]) are a more elaborate attempt at using gene reordering to gain in robustness. Although theoretically attractive, Messy GAs and reordering operators have an associated extra-cost that has limited their use in most practical implementations.

### 2.5.2 Population Replacement in Genetic Algorithms

DeJong [DeJong75] has shown that **elitism**, a strategy where the best individual is kept unchanged from one population to the next, is beneficial to the efficiency of the search in the context of function optimization. It means that the best individual never disappears at the change of generation.

More generally, it is possible renew only a fraction of the population at a time, which is what happens in natural populations. It makes sense that changing all of the individuals in the population at once involves a large, arguably excessive, loss of information about the design space. DeJong ([DeJong75]) experimented with "overlapping populations" where only a fraction of the population was selected for reproduction, and the same number of individuals was randomly chosen and deleted to make room for the offsprings. His study suggested that nonoverlapping

populations are best in the context of function optimization. Following his work, the "canonical" or "traditional" GA for function optimization was established: it is an elitist, nonoverlapping GA using binary coding, a large fixed probability of crossover, a small fixed probability of mutation, and population sizes varying between about 20 and 200 individuals.

The idea of not replacing all of the individuals reappeared with the steady state GAs ([Syswerda89], [Eshelman91]). In steady state GAs, individuals are progressively reproduced and inserted in the population. The steady population offers a more stable memory of the problem that enables the use of more explorative (more disruptive) operators (higher rates of mutation, uniform crossovers [Syswerda89]).

## *2.6 Genetic Algorithms among Evolutionary Algorithms*

GAs are part of the family of Evolutionary Algorithms that also include Evolution Strategies ([Rechenberg73]) and Evolutionary Programming algorithms ([Fogel91]). An overview and comparison of all three algorithms is presented in [Bäck93]. All Evolutionary Algorithms are based on the collective learning process of a population of individuals, each of which represent a point in the design space. The main difference between GAs and the other Evolutionary Algorithms is that Evolution Strategies and Evolutionary Programming algorithms use self-adaptation of some of their parameters. They also use mutation as their main search operator. Moreover, Evolutionary Programming algorithms do not have any recombination operators. Other differences can be drawn with respect to the canonical GA (cf. [Bäck93]), but they do not apply to most practical implementations of GAs (like the ones coming up in this report) and thus are omitted here.

# Chapter 3

# OPTIMIZATION OF LAMINATE STACKING SEQUENCE FOR BUCKLING LOAD MAXIMIZATION BY GENETIC ALGORITHMS

The objective of this Chapter is to devise a genetic algorithm for maximizing the failure load of a composite laminate. The thickness of the laminate is kept fixed. The design concerns exclusively the stacking sequence of the laminate. The genetic parameters determine how much effort is devoted to the exploration of the design space versus how much effort is used to exploit previous solutions. Random exploration of the design space is a robust search method, but it is prohibitively expensive. On the other hand, a genetic algorithm that overstresses the exploitation aspect of the search would converge prematurely to a nonoptimal solution.

In the first part of the work we describe the tuning the genetic algorithm for optimum values of various parameters. The population size, the probability of mutation, and the probability of crossover are optimized by numerical experiments. A new genetic operator – two-stack swap – is proposed and shown to be effective in reducing the cost of the genetic search on stacking-sequence design problems. Results are obtained for a graphite-epoxy plate, first when only the buckling load is considered, and then constraints on ply contiguity and strain failure are added. The influence on the genetic search of the penalty parameter enforcing the contiguity constraint is studied. Variations in performance of the genetic search as the size of the design space changes are observed. The advantage of the genetic algorithm in producing several near optimal designs is discussed.

## *3.1 Problem Description*

### 3.1.1 Analysis of the Laminated Plate

The simply supported plate, shown in Figure 3.1, is loaded in the $x$ and $y$ directions by $\lambda N_x$ and $\lambda N_y$, respectively, with $\lambda$ being an amplitude parameter. The laminate is composed of

$N$ plies. Its longitudinal and lateral dimensions are $a$ and $b$, respectively. It is further assumed to be symmetric, balanced, made of $0^o$, $90^o$ and $\pm 45^o$ plies, each of thickness $t$. Because of symmetry, the extensional-flexural coupling is suppressed (i.e., $B'_{ij}s = 0$, where $B_ij$ are the coefficients of the extensional-flexural matrix from Classical Lamination Theory). Balance of the laminate implies that there is no normal-shear extensional coupling ($A_{16} = A_{26} = 0$, where the $A_{ij}$'s are the coefficients of the extensional stiffness matrix).

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

$$\frac{\lambda_b(m,n)}{\pi^2} = \frac{[D_{11}(\frac{m}{a})^4 + 2(D_{12} + 2D_{66})(\frac{m}{a})^2(\frac{n}{b})^2 + D_{22}(\frac{n}{b})^4]}{(\frac{m}{a})^2 N_x + (\frac{n}{b})^2 N_y} . \tag{3.1}$$

It is assumed in the above formula that the laminate is specially orthotropic, i.e., the effects of $D_{16}$ and $D_{26}$ are neglected. It is a classical assumption when considering buckling of balanced and symmetric laminates since $D_{16}$ and $D_{26}$ are usually of small magnitude. The pair $(m,n)$ that yields the smallest value of $\lambda_b$, and defines the critical buckling load factor $\lambda_{cn}$, varies with the loading case, total number of layers considered, material, and the plate aspect ratio. Here, we consider all combinations of $m,n \leq 5$.

We also consider strain constraints and limit the number of contiguous plies of the same orientation to four to prevent matrix cracking. The four ply contiguity constraint is illustrated in Figure 3.2. The Figure sketches half of a laminate cross-section, the other half being the mirror image with respect to the laminate midplane. The left portion of the drawing is a case of violation of the ply contiguity constraint, and the right portion gives a similar yet feasible design.

The strain failure constraint requires all strains to remain below their allowable limits. The principal strains in the $i$th layer are related to the loads on the plate by the relations

$$\begin{aligned} \lambda N_x &= A_{11}\epsilon_x + A_{12}\epsilon_y , \\ \lambda N_y &= A_{12}\epsilon_x + A_{22}\epsilon_y , \end{aligned} \tag{3.2}$$

and

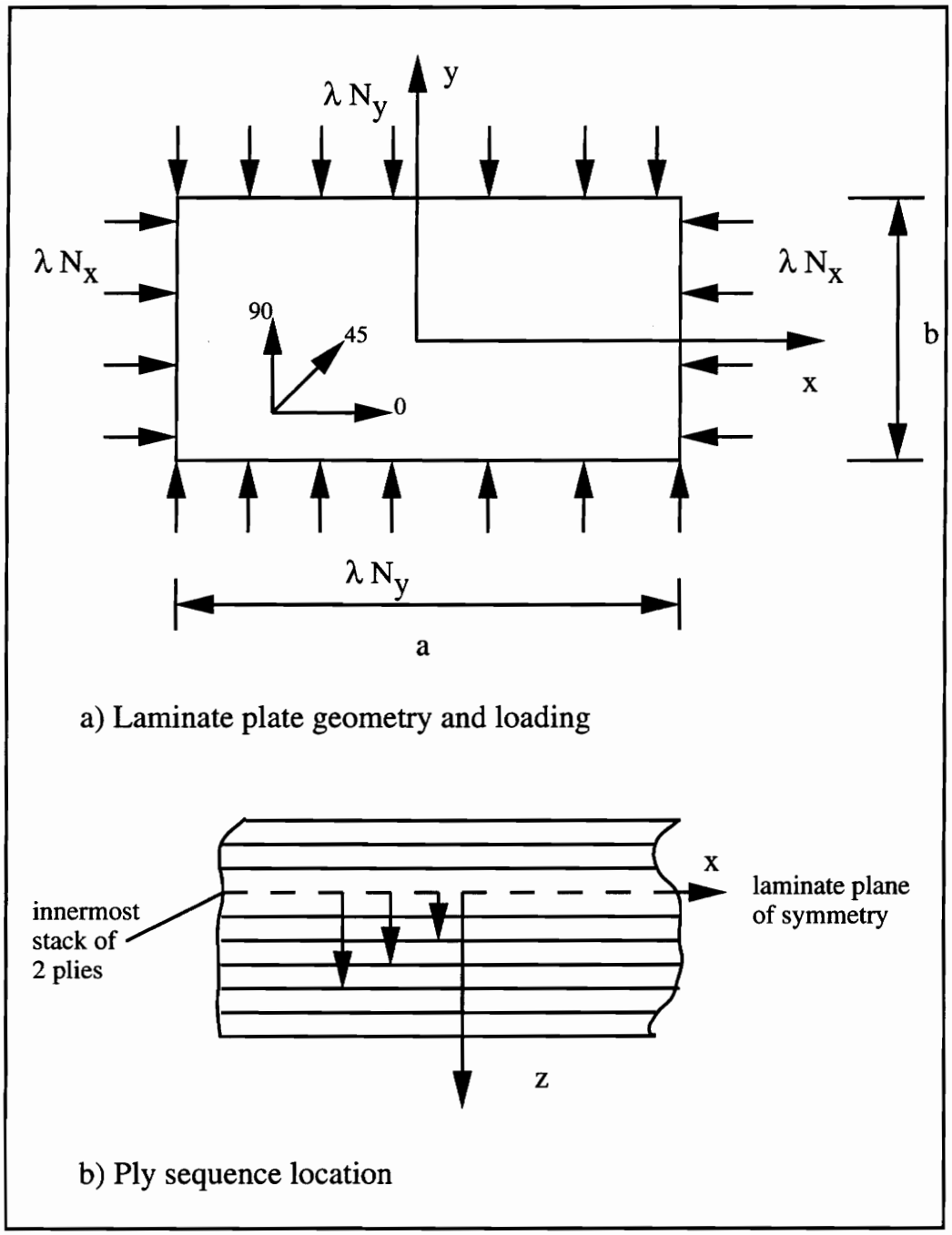



**Figure 3.1. Laminated plate geometry and loading.**

$$\begin{aligned} \epsilon_1^i &= cos^2\theta_i\ \epsilon_x + sin^2\theta_i\ \epsilon_y\ , \\ \epsilon_2^i &= sin^2\theta_i\ \epsilon_x + cos^2\theta_i\ \epsilon_y\ , \\ \gamma_{12}^i &= sin\,2\theta_i\ (\epsilon_y - \epsilon_x)\ . \end{aligned} \tag{3.3}$$

The ultimate allowable strains are $\epsilon_1^{ua} = 0.008$, $\epsilon_2^{ua} = 0.029$, $\gamma_{12}^{ua} = 0.015$. The strain failure

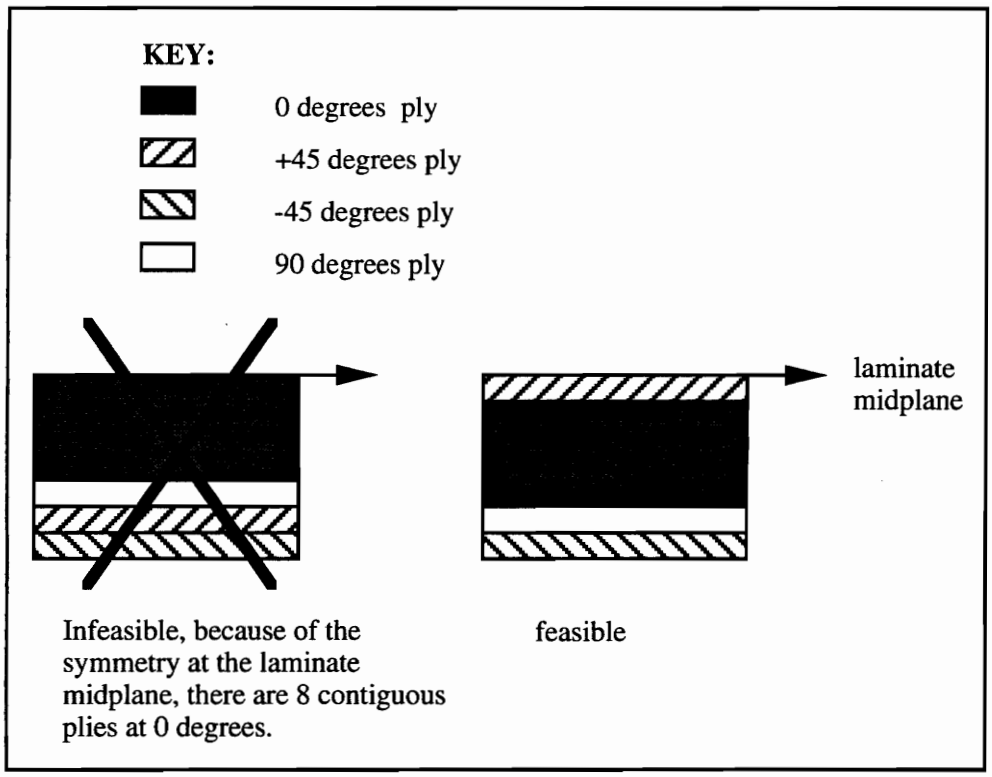



**Figure 3.2. Example of violation and satisfaction of the four ply contiguity constraint.**

load $\lambda_{c\epsilon}$ is the smallest load factor $\lambda$ such that one of the principal strains in one of the layers is above its allowable value. $\lambda_{c\epsilon}$ is calculated by

$$\lambda_{c\epsilon} = \min_i \left[ \min\left(\frac{\epsilon_1^{ua}}{f|\epsilon_1^i|}, \frac{\epsilon_2^{ua}}{f|\epsilon_2^i|}, \frac{\gamma_{12}^{ua}}{f|\gamma_{12}^i|}\right) \right], \tag{3.4}$$

where $i$ is the layer number and a safety factor $f$ of 1.5 is used.

## 3.1.2 Optimization Problem Formulation

In order to reduce the number of design variables and at the same time take into account the balanced condition, the laminate is considered to be made up of stacks of two plies of the same orientation. Within a $45^o$ stack, a $+45^o$ ply is always associated with a $-45^o$ ply, so that the balanced condition is guaranteed. Similarly, in order to implement the symmetry constraint, only one half of the laminate is coded. This means that balance and symmetry constraints are taken care of through data structuring (cf. Section 2.5.1). As a result, only $N/4$ ply orientations are required to describe the entire laminate.

Two optimization problems are considered here. The first one, referred to as buckling-only problem, has for objective the maximization of the critical buckling load of the laminate. The second problem, called buckling with strain and contiguity constraints, has for objective the maximization of the critical load factor $\lambda_{cr}$ that accounts for both buckling and strength failure, while keeping the contiguity constraint satisfied. For the buckling-only problem, the objective is to maximize $\lambda^*$, where $\lambda^* = \lambda_{cr} = \lambda_{cn}$. When strength and contiguity constraints are added, we want to maximize the objective function $\lambda^*$ defined as

$$\lambda^* = (1 - P_c)\, \lambda_{cr} \tag{3.5}$$

where $\lambda_{cr} = min\ (\lambda_{cn}, \lambda_{c\varepsilon})$.

The constraint limiting the number of contiguous plies of same orientation to four is implemented by a penalty parameter, $P_c$. Its value defines the percentage of reduction of the objective function for violation of the contiguity constraint.

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

## 3.2 Genetic Algorithm Description

A genetic algorithm is a guided random search technique that works on a population of designs. Each individual in the population represents a design, i.e., a stacking sequence, coded in the form of a string. A genetic search changes the population of strings by mimicking evolution.

A flow chart of the genetic algorithm is given on Figure 3.3. The GA starts with the generation of an initial random population of designs (three designs on Figure 3.3). In a second step, the objective function values of the designs are evaluated. Each design is assigned a fitness value depending on the objective function values of all the designs in the population. It is then processed, by so called genetic operators, to create a new population, which probabilistically combines the most desirable characteristics of the old population. The selection operator selects parent strings based on their fitness and passes them on to the operators performing the genetic recombination. The probability of a design of being chosen as parent increases with its fitness.

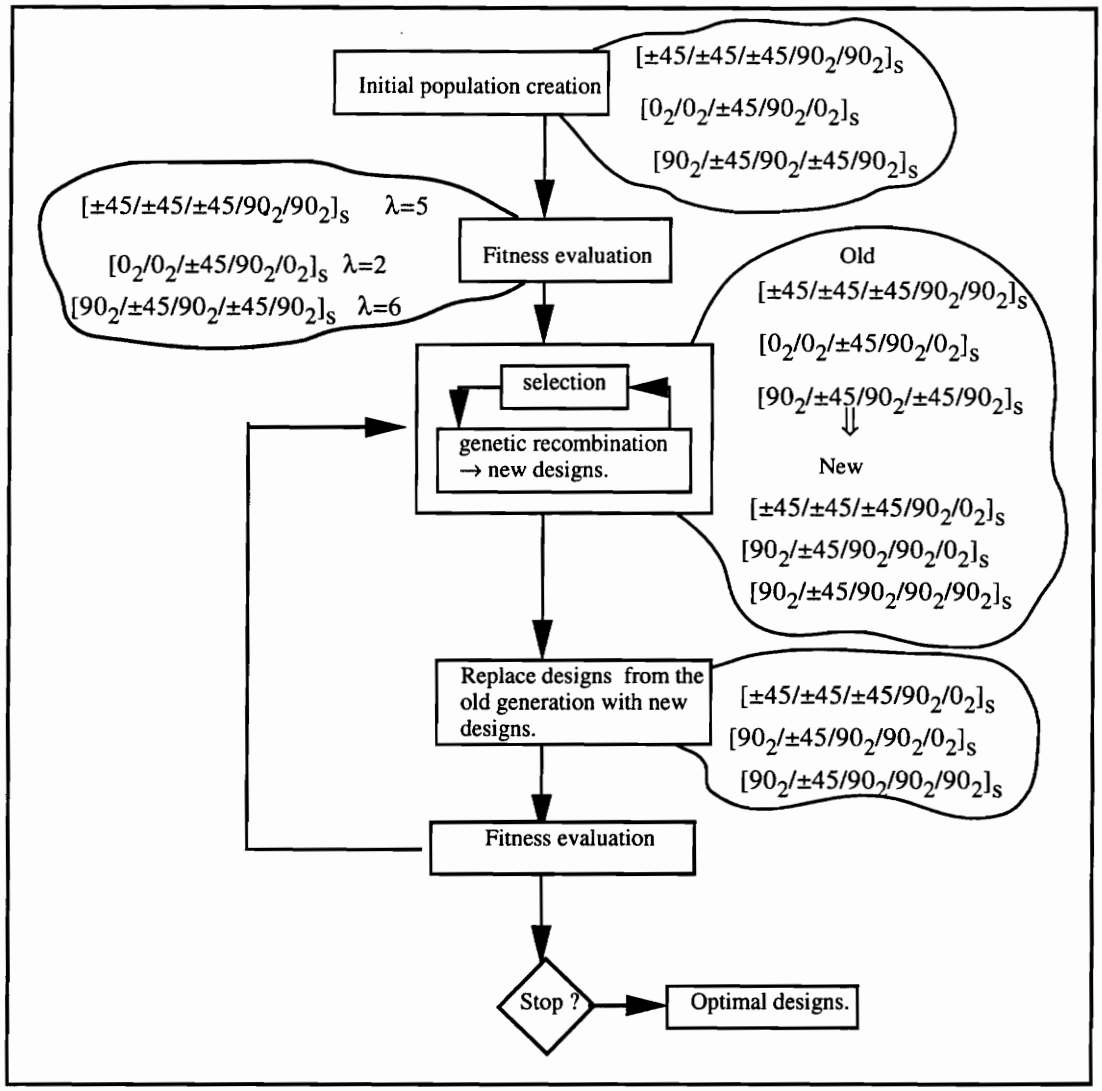



**Figure 3.3. Genetic algorithm flow chart for stacking sequence design.**

Genetic recombination is simulated by successive use of crossover, mutation, and two-stack swap operators, each applied with a given probability. The old population is then replaced by the new one, with one exception. The best of the previous generation is cloned into the next generation, which corresponds to an "elitist" version of the genetic algorithm (cf. Section 2.5.2). This process is repeated until a stopping criterion, implemented in the form of a maximum number of generations without improvement in the best design, is satisfied.

The encoding of a laminate, the calculation of its fitness, and the genetic operators are described in more detail hereafter.

### 3.2.1 Encoding of a Laminate

The coding used here to go from a design to a chromosome is the standard stacking sequence notation. Because our basic unit is a stack of two plies, laminates can be composed only of $0_2$, $\pm 45$, and $90_2$ stacks. For example, the laminate $(90_2^o/\pm 45_2^o/90_2^o/0_4^o/\pm 45^o/0_2^o)_s$ is encoded as $[90_2^o/\pm 45^o/\pm 45^o/90_2^o/0_2^o/0_2^o/\pm 45^o/0_2^o]$. By analogy to genetics, a gene is a given stack in the laminate, and the alleles can be $0_2$, $\pm 45$, or $90_2$. The rightmost position (gene) corresponds to the stack closest to the laminate midplane. The leftmost position in the chromosome describes the outermost stack of two plies.

### 3.2.2 Selection and Fitness Calculation

The first step in creating a new generation is the selection of sets of parent designs for mating. The selection process is biased so that high performance designs have a high probability of transmitting their features to the next generations. Each individual has a fitness value that determines its probability of being chosen as a future parent.

In the present Chapter, the fitness was chosen to be the relative rank in terms of the objective function $\lambda^*$ in the population. The fitness assigned to the $i$th best individual of $m$ designs is then equal to $[2(m+1-i)]/(m^2+m)$, so that the sum of all fitnesses is equal to one. The reason for using a fitness based on rank is that it keeps the selection pressure (that can be seen as the difference in fitness between designs) steady along the search. A fitness measure directly based on the objective function induces a decreasing selection pressure as the population globally converges to good designs having similar objective function values. It can be argued that having a decreasing selection pressure is a desirable feature in order to prevent premature convergence. But with a fitness directly based on the objective function value, the rate of decrease strongly depends on the problem considered. It is also dangerous to use normalized objective function values as fitness when constraints are calculated using penalty functions ([Powell93], [Whitley89]). Penalty functions tend to distort the objective function landscape and selection procedures based on the objective function are very sensitive to the tuning of the penalty: if the penalty is too harsh and most of the population is infeasible, the first individuals that satisfy the constraints will rapidly dominate the population, even though they might be mediocre solutions. If the penalty is too

small, the tendency of the population not to converge in the feasible domain is increased by the weakness of the selection pressure.

Selection is implemented, like in the didactic GA of Section 2.2, by allocating to each design a portion of the segment [0,1] equal in length to its fitness. A random number is then created between 0 and 1 that designates the selected individual.

### 3.2.3 Crossover Operator

Crossover allows selected individuals to trade characteristics of their designs by exchanging parts of strings. We used a two-point crossover, where two break points in the string are chosen randomly. Two offsprings are created by swapping the parents' substrings. Crossover is applied with a given probability, usually between 0.6 and 1. If crossover is not applied, the parents are copied into the next generation. The following is an example where a $[0_2^o/\pm 45_2^o/90_2^o/0_4^o/90_2^o/\pm 45^o]_s$ laminate is mated with a $[\pm 45_2^o/0_2^o/90_4^o/0_2^o/90_4^o]_s$ laminate to produce $[0_2^o/\pm 45^o/0_2^o/90_4^o/0_2^o/90_2^o/\pm 45^o]_s$ and $[\pm 45_3^o/90_2^o/0_4^o/90_4^o]_s$ laminates.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Parent 1** | $0_2$ | $\pm 45$ | $\overset{\nabla}{\underset{\Delta}{}}\pm 45$ | $90_2$ | $0_2$ | $0_2$ | $\overset{\nabla}{\underset{\Delta}{}}90_2$ | $\pm 45$ |
| **Parent 2** | $\pm 45$ | $\pm 45$ | $\overset{\nabla}{\underset{\Delta}{}}0_2$ | $90_2$ | $90_2$ | $0_2$ | $\overset{\nabla}{\underset{\Delta}{}}90_2$ | $90_2$ |
| **Child 1** | $0_2$ | $\pm 45$ | $\overset{\nabla}{\underset{\Delta}{}}0_2$ | $90_2$ | $90_2$ | $0_2$ | $\overset{\nabla}{\underset{\Delta}{}}90_2$ | $\pm 45$ |
| **Child 2** | $\pm 45$ | $\pm 45$ | $\overset{\nabla}{\underset{\Delta}{}}\pm 45$ | $90_2$ | $0_2$ | $0_2$ | $\overset{\nabla}{\underset{\Delta}{}}90_2$ | $90_2$ |

The children thus combine genetic information carried by the parents and previously tested by the selection.

Two-point crossover is the standard crossover operator for parameter optimization when little is known about the nonlinerarities between genes ([Goldberg89a]). The reason is that one-point crossover has a very high probability of separating bits that are located at the extremes of the (linear) chromosome. Two-point crossover, on the other hand, is not biased towards separating the bits locating around the ends of the chromosome. One way of understanding this is by thinking of a chromosome as a ring, which is possible only when two-point crossover is used (cf. Figure 3.4). Since a ring does not have ends, the end effect bias does not need to be considered. In

fact, it has been shown ([Spears91]) that two-point crossover has, on the average, the smallest disruption probability among all crossovers (the disruption probability is the probability that crossover separates $k$ digits in the string, $2 \leq k \leq L$). A discussion can be found in Chapter 4 about the effect of the number of break points in crossover on laminate design problems.

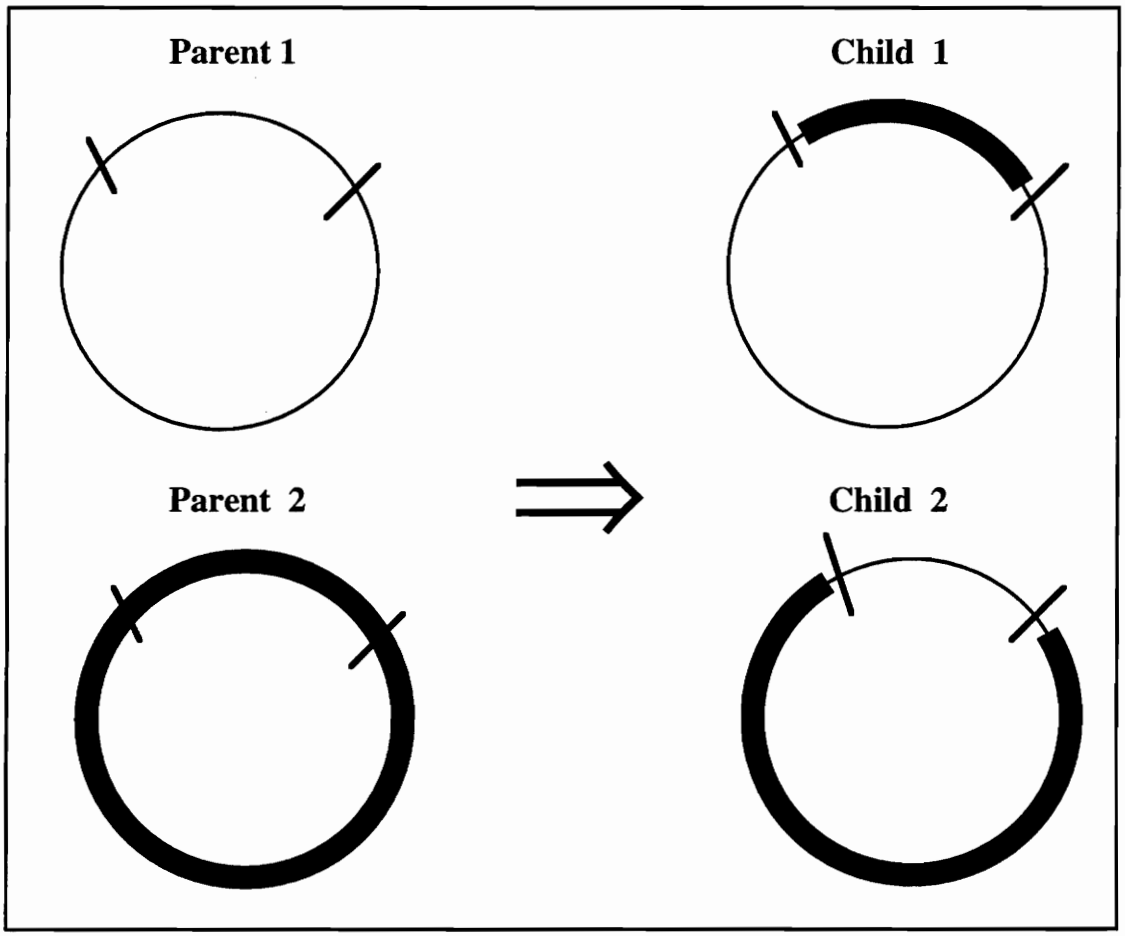



**Figure 3.4. Two-point crossover, where the chromosome is seen as a ring.**

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

### 3.2.4 Mutation

Next, one of the two children is randomly picked, and endures mutation. Mutation is a stochastic operator, applied with a low probability. It protects from a complete loss of genetic

material by changing at random an allele in a string. Its purpose is to bring in completely new bit combinations. In this chapter, each laminate string has "probability of mutation" chances of having one digit changed. A first random number is generated to decide whether to apply mutation to a given laminate. Then, a second random number decides which gene will be changed. Each gene in the string has the same probability of $1/l$ of being chosen, $l$ being the total length of the string. A third random number determines which new value the gene will take on. Since there are three possible gene values (alleles), a given gene that is subjected to mutation has 50% chances of assuming either one of the two other alleles. An example of mutation is given below.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Before Mutation** | $0_2$ | $\pm 45$ | $\underline{0_2}$ | $90_2$ | $90_2$ | $0_2$ | $90_2$ | $\pm 45$ |
| **After Mutation** | $0_2$ | $\pm 45$ | $\underline{\pm 45}$ | $90_2$ | $90_2$ | $0_2$ | $90_2$ | $\pm 45$ |

### 3.2.5 Two-Stack Swap Operator

Two-stack swap, a new operator devised for laminate design, is also a stochastic operator. Two-stack swap shuffles the location of layers without changing the composition of the laminate. Therefore, the flexural properties of the laminate are modified while its in-plane characteristics are preserved. First two groups of two stacks are selected in the string. The two groups of two stacks then exchange positions, and the two stacks within each group also exchange their respective positions. Two-stack swap has been applied with a high probability because it has some advantages over mutation that we will present with more details later. The following illustrates the working of two-stack swap.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Before two-stack swap** | $0_2$ | $\pm 45$ | $\underbrace{\pm 45}_{1}$ | $\underbrace{90_2}_{2}$ | $90_2$ | $\underbrace{0_2}_{3}$ | $\underbrace{90_2}_{4}$ | $\pm 45$ |
| **After two-stack swap** | $0_2$ | $\pm 45$ | $\underbrace{90_2}_{4}$ | $\underbrace{0_2}_{3}$ | $90_2$ | $\underbrace{90_2}_{2}$ | $\underbrace{\pm 45}_{1}$ | $\pm 45$ |

In this example, the laminate described by the string before two stack swap is $[0_2^\circ/\pm 45_2^\circ/90_4^\circ/0_2^\circ/90_2^\circ/\pm 45^\circ]_s$ and becomes $[0_2^\circ/\pm 45^\circ/90_2^\circ/0_2^\circ/90_4^\circ/\pm 45_2^\circ]_s$ after two stack swap.

An operator like two-stack swap that changes allele values based on other alleles in the chromosome should not be mistaken for re-ordering operators like inversion (cf. Section 2.6.1). Inversion does not change the design, but only the way it is coded, whereas two-stack swap changes the design.

## *3.3 Tuning the Genetic Algorithm*

### 3.3.1 Practical Optimum and Normalized Price of the Search

The design space for our problem was found to include a large number of near optimum designs. For this reason, designs that are within a 10th of a percent of the global optimum were accepted as optimal and are called here **practical optima**. The genetic algorithm was tuned by numerical experiments. The evaluation measure is the **normalized price**, which is the number of evaluations of the objective function (also called **price**) divided by the probability of reaching a practical optimum (called **practical reliability**). To compensate for random fluctuations in performance, the selection of the optimal genetic parameters was conducted by performing two hundred genetic optimizations for each of three load cases (48 plies, Ny/Nx = 0.125,0.25,0.5, respectively) to obtain average prices and reliabilities, and also standard deviations. Typically, the coefficient of variation (standard deviation over mean value) was about 0.25-0.3 for the price.

### 3.3.2 Results

The first step in the implementation of the algorithm was to choose a coding. Experiments were conducted to compare a traditional 0,1 coding (where 00 stands for $0_2$, 01 and 10 stand for $\pm 45$, and 11 corresponds to $90_2$) with the natural $0_2, \pm 45, 90_2$ alphabet. Four hundred runs were made for three different load cases. The natural $0_2, \pm 45, 90_2$ alphabet was two percent cheaper in terms of the normalized price than the 0,1 coding. The binary coding is biased because the probability of appearance of a $\pm 45$ stack of two plies is twice that of other stacks. Moreover, crossover is more disruptive with the 0,1 coding than it is with the natural coding. For example, if a break point falls within a double digit defining a stack that is 00 (i.e., $0_2^o$) for parent 1 and 11 (i.e., $90_2^o$) for parent 2, the child design will have a 01 or 10 (i.e., $\pm 45^o$) stack at that position, which

was present in none of the parent designs. Crossover would not create new stack orientations in such a fashion with the natural alphabet. That is, crossover is responsible for much more random exploration of the design with the 0,1 coding than it is with the natural coding, to the detriment of preservation of good solutions found in the past. Random exploration of the design space is not detrimental by itself, but it is a task that can be performed by mutation, whereas the juxtaposition of building blocks can only be made by crossover. The $0_2, \pm 45, 90_2$ alphabet fits well the three possible stacks of plies and was adopted for further experiments. This particular example adds to the list of cases where a natural coding of the problem is preferable to a binary coding (cf. Section 2.6.1 for a more detailed discussion of binary versus natural codings).

The population size, probabilities of crossover, mutation, and two-stack swap, the length of the search and the value of the penalty parameter were optimized. A minimum value of the practical reliability was set at 80 percent. Many experiments were carried out and only the variations near the optimal set of parameters are discussed here. Note also that rates higher than 1 can appear in the mutation, two-stack swap and inversion operators as a result of the operator being applied more than once to the string. Rates between 100 and 200% are achieved by dividing the probability by two and calling the operator twice. Similarly, rates between 200 and 300% result from dividing the probability by three and calling the operator three times. For example, a rate of 2.5 for stack swap means that stack swap is applied three times with a probability of 0.833.

Two-stack swap can be viewed as a mutation operator, where new values of the digits come from other positions in the string. Two-stack swap is a partially random perturbation of the original design: it changes the flexural properties of the laminate within a certain range without altering its stiffness. This is desirable when designing laminates for buckling and strength, because it enables moves with respect to the buckling constraint without changes in the strength constraint. Buckling constraints are typically more difficult to satisfy than strength constraints because they involve finding not only the fiber orientation in each ply but also the location of the ply in the laminate, whereas strength constraints are not sensitive to the location of the plies. So, there is more need for exploration with respect to buckling than with respect to strength, hence the relevance of stack swap operators. Even when buckling only is considered, stack swap is an important exploration tool.

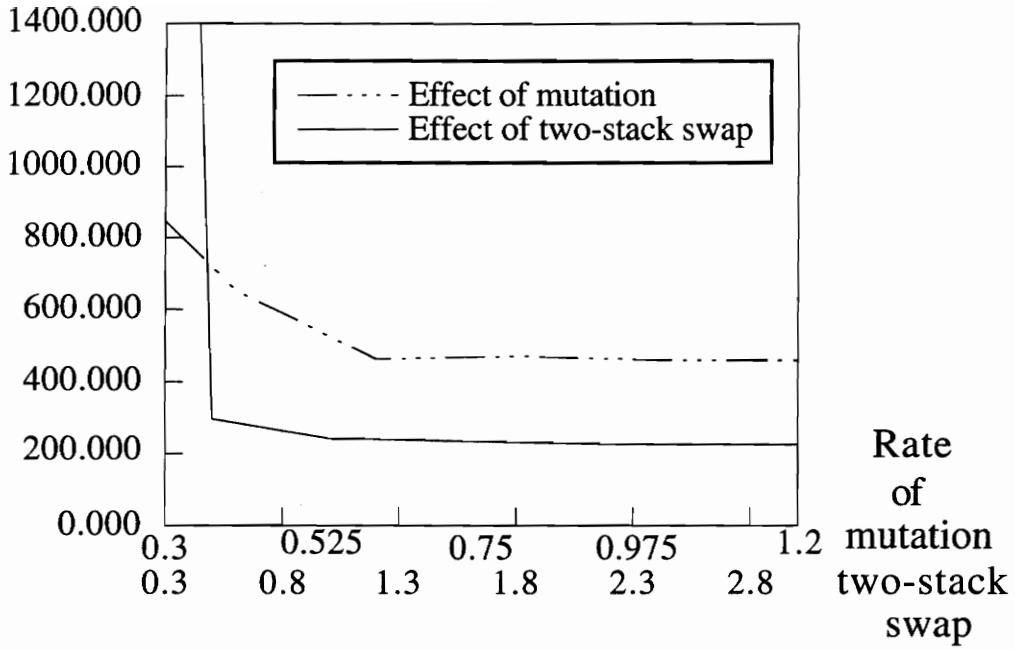



**Figure 3.5. Buckling-only problem: probability of crossover $pc$ = 1, population size $m$ = 8, search stopped after 10 generations without improvement.**

Figure 3.5 shows that the use of two-stack swap lowers the normalized price of the optimization by about a factor of two compared to mutation and that very high probabilities of two-stack swap are beneficial in the buckling-only case.

Figure 3.6 shows that a more intensive exploration of the laminate design space by stack swap can be performed on the buckling-only problem than when strain failure constraints are added. The genetic algorithm needs 300 percent two-stack swap for the buckling-only problem, whereas 100 percent is optimal in the constrained case. Compared to mutation, two-stack swap improves drastically the practical reliability without much increase in the price of the search. Two-stack swap can be used at much higher rates than mutation, because the type of random search it makes is very efficient in the laminate landscape.

A large population does not get enough mixing of the strings in a reasonable number of computations of the objective function. A good reliability would only be achieved at a very high price. On the other hand, too small a population misses so much genetic material that the search is driven by costly random mutation and two-stack swap. A rational performance measure

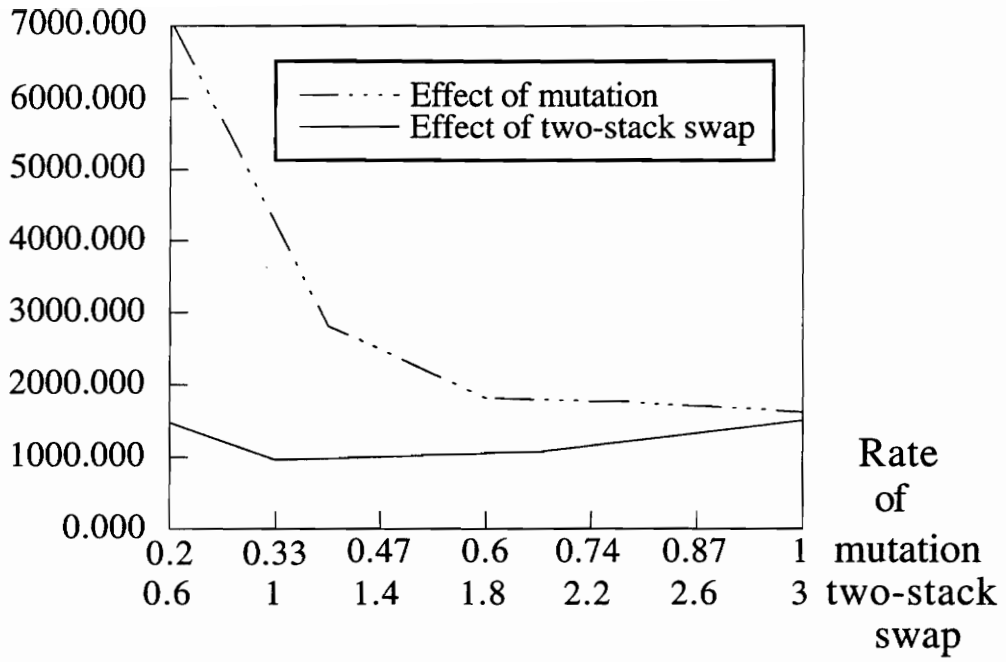



**Figure 3.6. Buckling with strain and contiguity constraint: probability of crossover $pc$ = 1, population size $m$ = 8, search stopped after 56 generations without improvement.**

has been developed in [Goldberg85], the "number of excess unique schemata per individual", whose maximization over the population size yields a theoretical optimum population size. This calculation has been adapted to our case (three character alphabet, 48 plies in the laminate that gives 12 digit long chromosomes). Details of this derivation (along with other theories for estimating population sizes) are given in Appendix A.2. The computed population size was about 22 members. This theoretical result does not account for operators like mutation and two-stack swap that create new bit combinations in the population. The theoretical optimum population size is thus overestimated. In Figure 3.7, it is shown that the minimization of the normalized price yielded an optimum population size of eight individuals for both the constrained and unconstrained buckling problems.

The effect of the probability of crossover on the normalized price is shown in Figure 3.8. The optimum probability of crossover is found to be 1 in the unconstrained as well as in the constrained buckling maximization. Part of the reason for this uncommonly high probability is the straightforward way of coding a stacking sequence into a bit string. The main flaw of

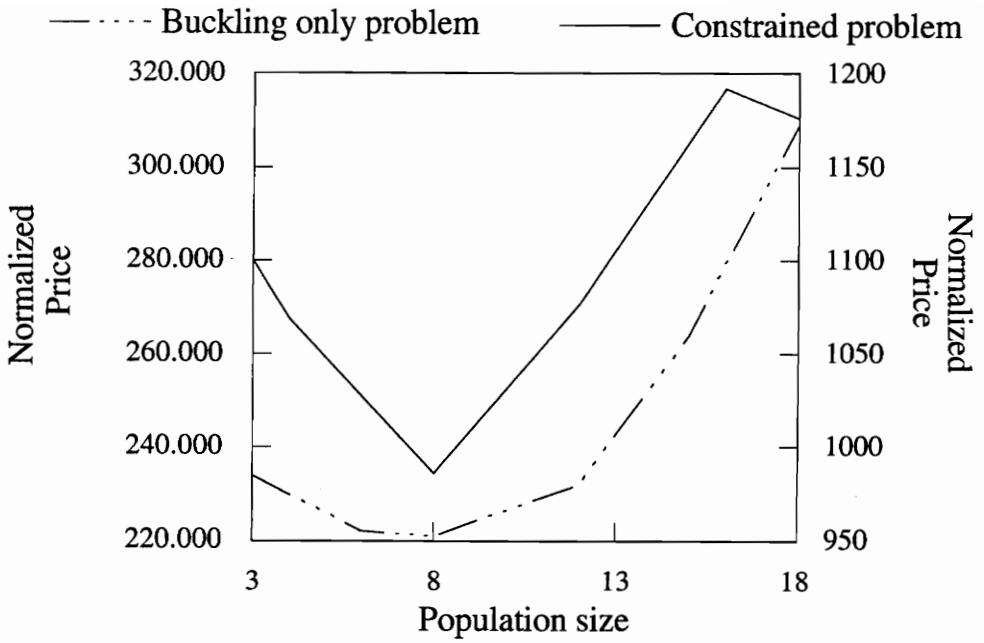



**Figure 3.7. Effect of population size, 48-ply laminate.**

crossover is its tendency to separate interacting bits that are far apart. However, for our problem, interacting plies (e.g., due to contiguous ply constraint) are next to each other in reality, and have representing digits next to each other. Therefore, the tendency of crossover to separate far-apart interacting bits is not relevant. Moreover, the loss of the best individual has often motivated a reduction in the probability of crossover. This argument does not affect the present algorithm, since it keeps the best individual unaltered.

Mutation is a pure exploration operator and its effect is theoretically clear: as the mutation rate goes up, price and reliability increase. This was confirmed by the experiments. Nevertheless, a little mutation is an indispensable guarantee against eventual loss of diversity in the population. As shown in Figure 3.9, 1% mutation per laminate in the buckling-only problem and 10% per laminate in the constrained case yielded the best normalized prices. The higher probability of mutation in the constrained optimization may be linked to a wider diversity of ply orientations for the optimum laminates and to the presence of more constraints.

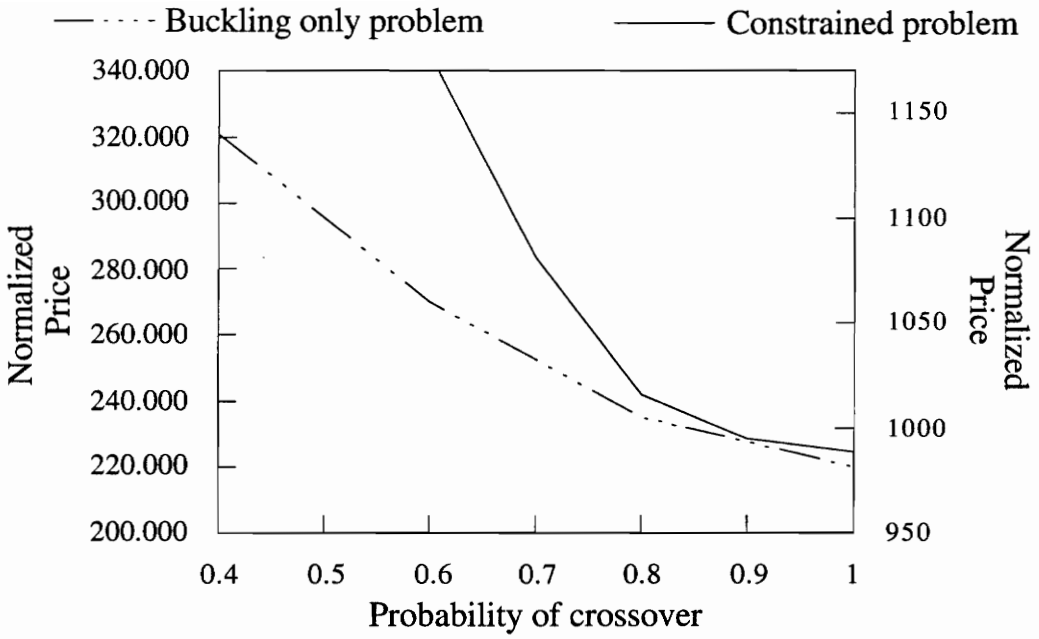



**Figure 3.8. Effect of probability of crossover, 48-ply laminate.**

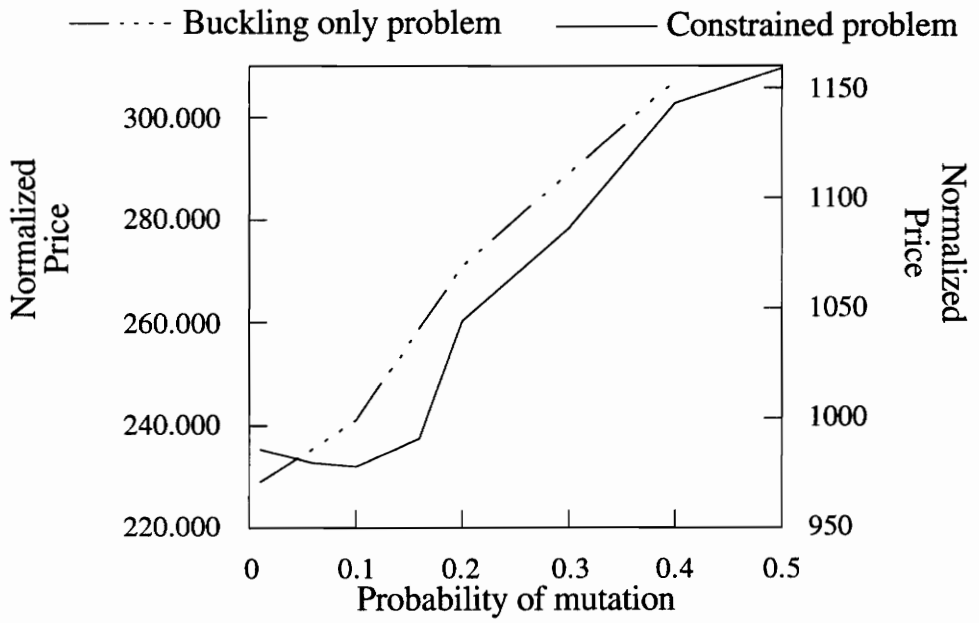



**Figure 3.9. Effect of probability of mutation, 48-ply laminate.**

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

The value of the penalty parameter, enforcing the contiguous ply constraint, was also analyzed through numerical experiments. Figure 3.10 shows normalized prices. Five loading cases were necessary to find meaningful averaged results since, among the three loading cases previously chosen, one ($N_y/N_x = 0.5$) has an atypical optimal value of the penalty parameter. 8% penalty ($P_c$ = 0.08) is the averaged optimum setting of the penalty parameter. This value is close to a minimum value for which the best infeasible designs are at the border of the practical optimum region. At about 8% penalty, designs that violate the contiguity constraint, but have high buckling and strain failure loads, are able to compete and thus breed with average-performance feasible designs. Below 8% penalty, unfeasible designs result from the genetic search. It should be noted however that in some cases, like ($N_y/N_x = 0.5$), a large penalty parameter significantly improves the price. For these loading cases, it is found that all of the practical optima naturally do not violate the contiguity constraint. The strong penalty merely limits the search to the good part of the design space.

The stopping criterion that defines the length of the search has also been optimized. Figure 3.11 illustrates that the optimum values found for the maximum number of generations without improvement in the best design is 56 for the constrained case. The stopping criterion was driven by the condition of 80 % reliability in the constrained problem.

### 3.3.3 Influence of the Performance Criterion on the optimal Tuning of the GA

Historically, the two first criteria for evaluating the performance of GAs used as optimizers were proposed by De Jong in [DeJong75]. They are called on-line and off-line performances. The on-line performance is the running average of the objective function values of all the designs evaluated during the search. The off-line performance is the running average of the objective function value of the best individuals of each generation. De Jong mentioned the possibility of weighting the on-line and off-line measures at each generation, in order to put a stronger emphasis on the last most recent generations, but he only used uniform weighting. From numerical

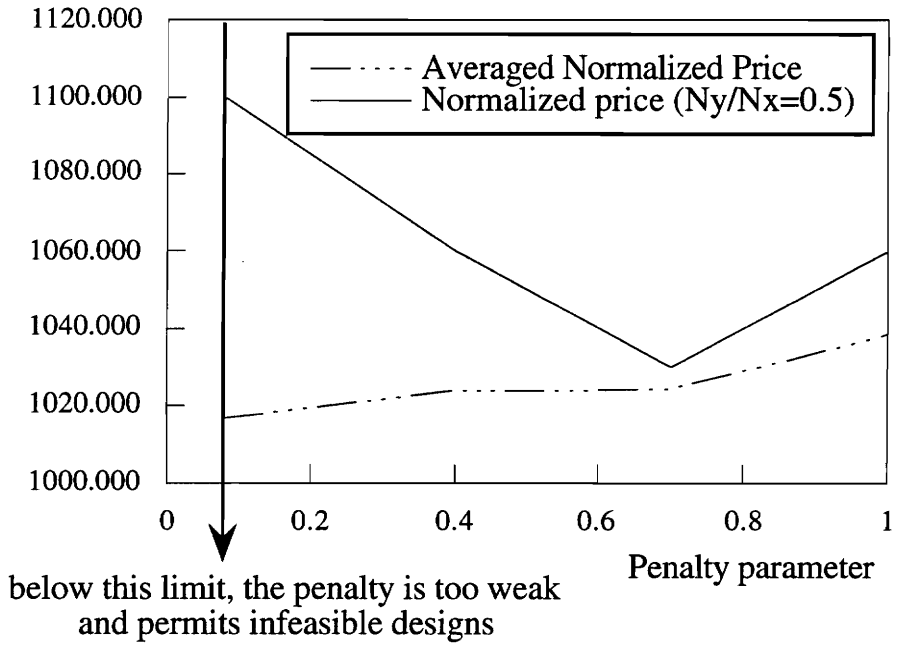



**Figure 3.10. Effect of penalty parameter for contiguity constraint $P_c$, 48-ply laminate. Buckling with strain and contiguity constraint: probability of crossover $pc$ = 1, rate of two-stack swap $pp$ = 1, probability of mutation $pm$ = 0.06, population size $m$ = 8, search stopped after 56 generations without improvement.**

experiments on a set of functions using a standard GA (cf. 2.6.2) emerged values for the genetic parameters that were optimal both for the on-line and off-line measures. They were:

| | |
|---|---|
| population size | 50-100 |
| crossover rate | 0.60 |
| mutation rate† | 0.012 |

(† Note that the above mutation rate expresses the probability that at least one allele is mutated per string. Our definition of mutation rate in this chapter is the probability that exactly one allele is mutated per string. Such a discrepancy should not however have any major effect on the results.)

Grefenstette ([Grefenstette86]) used a meta-level GA to optimize the parameters of the GA on the same test bed of functions as De Jong. He found that the optimal setting of the GA for off-line performance was the same as De Jong's. Nevertheless, he was able to improve on the

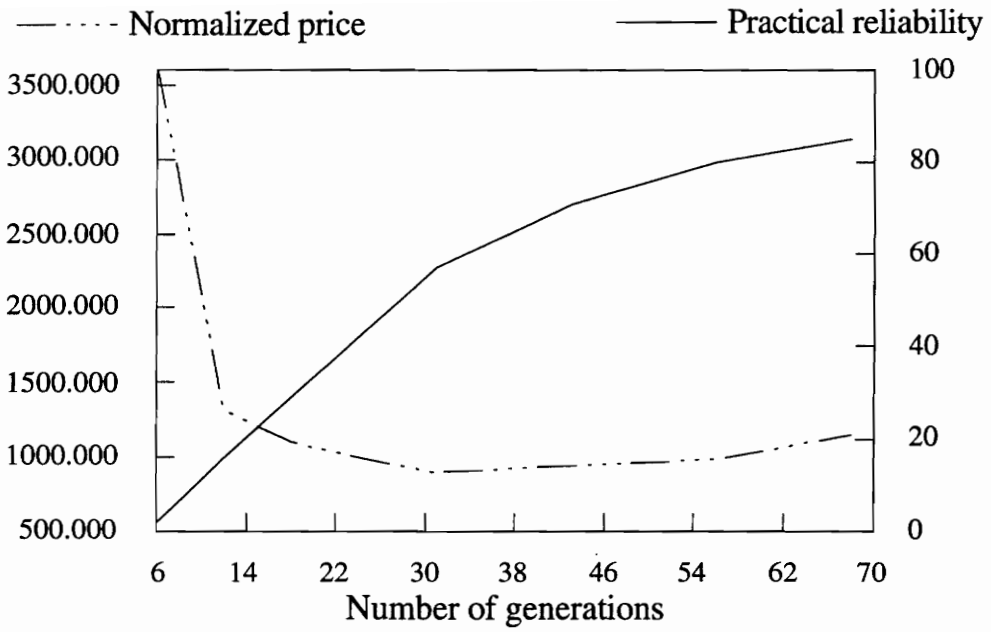



**Figure 3.11. Effect of stopping criterion (number of generations without improvement), 48-ply laminate. Buckling with strain and contiguity constraints: population size $m$ = 8, rate of two-stack swap $pp$ = 1, probability of mutation $pm$ = 0.06, probability of crossover $pc$ = 1.**

on-line performance by selecting the following genetic parameters:

| | |
|---|---|
| population size | 30 |
| crossover rate | 0.95 |
| mutation rate† | 0.114 |

Schaffer et al. ([Schaffer89]) conducted a similar study of the influence of genetic parameters on the on-line performance of the GA. They added other functions to De Jong's test bed and used a Gray coding instead of binary coding. They concluded that the following setting performed the best.

| | |
|---|---|
| population size | 20-30 |
| crossover rate | 0.75-0.95 |
| mutation rate† | 0.059-0.114 |

In this work, we found that GAs performed the best on stacking sequence design problems

with the following set of genetic parameters:

| | |
|---|---|
| population size | 8 |
| crossover rate | 1.00 |
| mutation rate | 0.01-0.10 |
| two-stack swap rate | 1.00-2.25 |

With respect to the previous conventional settings, our tuning advocates a much smaller population size and the intensive use of a different operator, stack swap. The present work differs from the three previous studies in many aspects: the performance criterion used to evaluate the GA, the algorithm employed and the class of problems considered. Those differences make our optimal setting of the GA somewhat atypical.

For example, our performance criterion, the normalized price of the search, emphasizes finding a reasonably good solution (not only the optimum) in as short a time as possible. The on-line and off-line measures used in the aforementioned studies were evaluated after long searches (of the order of 10000 analyses) as compared to ours (around 1000 analyses), and were geared towards finding the optimal solution, which partly explains the difference in population size. Our small population size is also explained by the intense use of stack swap, that acts against premature convergence. Standard GAs, that do not have specialized stochastic operators such as stack swap, need larger population sizes in order to avoid premature convergence.

## *3.4 Multiplicity of Optima*

The stacking sequence design problem has the peculiarity of many near optimal (practical) solutions, or even sometimes many global optimum designs. This feature, in conjunction with a high rate of two-stack swap as well as a fast stopping criterion, contributes to the atypical setting of the genetic parameters.

Because the genetic algorithm optimizes a population of designs and relies on probabilistic transitions, many near optimal stacking sequences can be found. Compared to other optimization methods, the genetic algorithm gives the designer more freedom in the choice of the best plate.

**Table 3.1. Optimal and near-optimal designs for buckling, strain, and contiguous ply constraints for $N_y/N_x = 0.5$.**

| Stacking sequence † | Load factor | |
|---|---|---|
| | Buckling | Strength |
| $(90_2, \pm 45_2, 90_2, \pm 45, 90_2, \pm 45_6)_s$ | 9998.19 | 10394.81 |
| $((90_2, \pm 45_2)_2, 90_2, \pm 45, 90_2, \pm 45_3)_s$ | 9997.60 | 10187.93 |
| $(\pm 45, 90_4, \pm 45, 90_2, \pm 45_5, 90_2, \pm 45)_s$ | 9976.58 | 10187.93 |

† 48 plies, $a$ = 20 in., $b$ = 5 in..

Table 3.1 shows, for example, the global optimum and the next best designs for a load ratio $N_y/N_x$ = 0.5. In the cases shown, the failure mode was buckling. The best laminate has 4% margin between strain failure and buckling failure, the second and third best designs have a margin of about 2%. The laminate with $\pm 45^o$ plies on the outside may be preferred to one with $90_2^?$ plies on the outside for its improved damage tolerance, despite a slightly lower buckling load.

**Table 3.2. Multiple optima for buckling-only problem for $N_y/N_x = 1.0$.**

| Stacking sequence † | Load factor | |
|---|---|---|
| | Buckling | Strength |
| $(90_{10}, \pm 45_2, 90_2, \pm 45_3, 90_2, \pm 45_4)_s$ | 3973.01 | 14205.18 |
| $(\pm 45, 90_{10}, \pm 45, 90_8, \pm 45, 90_8)_s$ | 3973.01 | 8935.74 |
| $(90_4, \pm 45_2, 90_{16}, \pm 45, 90_6)_s$ | 3973.01 | 8935.74 |
| $(90_2, \pm 45, 90_6, \pm 45, 90_8, \pm 45, 90_{10})_s$ | 3973.01 | 8935.74 |
| $(90_8, \pm 45, 90_2, \pm 45, 90_2, \pm 45, 90_2, \pm 45_6)_s$ | 3973.01 | 14205.18 |

† 64 plies, $a$ = 20 in., $b$ = 10 in..

For multimodal functions, the genetic search can converge to many of the optima. This property is particularly useful in stacking sequence design, where multiple global optima often occur. A single optimization by genetic algorithm yields many optima. By keeping track of

**Table 3.3. Multiple optima for buckling, strain, and contiguous ply constraint, for $N_y/N_x = 0.125$.**

| Stacking sequence † | Load factor | |
|---|---|---|
| | Buckling | Strength |
| $(90_2, \pm 45_4, 0_4, \pm 45, 0_4, \pm 45, 0_2)_s$ | 14168.12 | 13518.66 |
| $(\pm 45_3, 0_2, \pm 45_2, 0_2, 90_2, 0_4, \pm 45, 0_2)_s$ | 14134.76 | 13518.66 |
| $(90_2, \pm 45_3, 0_2, \pm 45, 0_2, \pm 45, 0_4, \pm 45, 0_2)_s$ | 14013.71 | 13518.66 |
| $(\pm 45_2, 0_2, \pm 45_2, 90_2, 0_4, \pm 45, 0_2, \pm 45, 0_2)_s$ | 13662.61 | 13518.66 |

† 48 plies, $a$ = 20 in., $b$ = 5 in..

stacking sequences whose fitnesses are within a 10th of a percent from the best-so-far design, we generate a list of practical optima. The numbers of practical optima found per search that are presented below were average values over 20 trials. For the case presented in table 3.2 (buckling-only problem, 64 plies, a = 20, b = 10, $N_y/N_x = 1.0$ ), 1.2 practical optima were found on the average during one search, with the stopping criterion of 10 generations without improvement and a normalized price of 309.7. For a slower stopping criterion (50 generations without improvement), we averaged 9.5 practical optima per search, with a normalized price of 695.0. It was observed that most of the practical optima are discovered late in the genetic search. Table 3.3 presents the constrained optimization problem, 48 plies, a = 20 in., b = 5 in., $N_y/N_x = 0.125$. Strain failure is the critical failure mode for all these laminates. On the average, 11.1 practical optima were found per search, with a normalized price of 545.

## *3.5 Effect of the Size of the Problem*

Like most pseudo-random search methods, a genetic algorithm improves its performance as the size of the problem increases. Tables 3.4 and 3.5 describe how price, normalized price and practical reliability evolve as function of the number of plies. The price and the normalized price increase with the number of layers, but in a proportion that makes the genetic search more and more effective as related to the size of the design space. Credit should be given not only to the

**Table 3.4. Effect of the size of design space for buckling-only problem.**

| Number Of Layers | Size Of Design Space | Normalized Price | Percent Of Design Space explored, % | Price | Practical Reliability |
|---|---|---|---|---|---|
| 16 | 81 | 172.3 | 213.0 | 172.3 | 100.0 |
| 32 | 6,561 | 325.4 | 4.9 | 273. | 83.9 |
| 48 | 531,441 | 327.7 | 0.06 | 323.4 | 98.7 |
| 64 | 43,046,721 | 375.0 | 0.00087 | 371.0 | 98.9 |

(Population size $m$ = 8, rate of two-stack swap $pp$ = 2.25, probability of mutation $pm$ = 0.01, probability of crossover $pc$ = 1, search stopped after 10 generations without improvement.)

intrinsic properties of the genetic algorithm but also to the fact that thick laminates have many global optima. For the buckling-only optimization, the normalized price does not have such a regular progression. The 32 plies case shows low practical reliability, i.e., a high normalized price. Most of the time, performing a short genetic search was found advantageous for the buckling-only problem, but it makes the method more sensitive to premature convergence. Some loading cases are prone to premature convergence, and two of them occurred for the 32 plies case. Note also that all the numerical experiments for the size of the problem included five load cases compared to three for the previous experiments. This explains the slight discrepancies between Tables 3.4, 3.5 and other results. For the constrained problem, 48 plies, $N_y/N_x = 0.75$ and 1 were two cases difficult to optimize. The normalized prices of these optimizations were about 4000. The averaged normalized price over the five loading cases was 2163.4.

## *3.6 Concluding Remarks*

The use of a genetic algorithm to optimize the stacking sequence of a laminated plate was presented. Buckling constraints as well as contiguity and strain constraints were considered. The genetic algorithm was adapted to take advantage of the special features of the stacking sequence design. A two-stack swap operator was implemented and improved the performance of the genetic search.

**Table 3.5. Effect of the size of design space for buckling with strain and contiguity constraints.**

| Number Of Layers | Size Of Design Space | Normalized Price | Percent Of Design Space explored, % | Price | Practical Reliability |
|---|---|---|---|---|---|
| 16 | 81 | 1546.7 | 1900.0 | 1546.7 | 100.0 |
| 32 | 6,561 | 1850.0 | 28.19 | 1796.7 | 97.1 |
| 48 | 531,441 | 2163.4 | 0.41 | 2136.7 | 98.8 |
| 64 | 43,046,721 | 3090.0 | 0.0072 | 2636.7 | 85.3 |

(Population size $m$ = 8, probability of two-stack swap $pp$ = 1, probability of mutation $pm$ = 0.06, probability of crossover $pc$ = 1, search stopped after 56 generations without improvement.)

The laminate design space was found to include multiple optima, especially for a large number of plies. The genetic algorithm was shown to be able to produce several of these optima in a single execution. It also deals well with integer variables and is not sensitive to problem nonlinearities. As a result, the genetic algorithm looks for global optimum, i.e., it can escape local optima where gradient based methods would get trapped. For example, Figure 3.12 considers two load cases for 48 plies laminates, $a$ = 20 in., $b$ = 5 in., $N_x = 1$, $N_y = 0.125$ and 0.25, respectively. In both cases, genetic search finds laminates with higher failure loads than sequential linearization with the Branch and Bound algorithm does ([Nagendra92]).

This first study showed that genetic algorithms require more objective function evaluations for stacking sequence design than traditional search methods. Therefore, the next Chapter pursues our effort to reduce the price of the search. The genetic operators are further specialized for minimum thickness design of laminated plates.

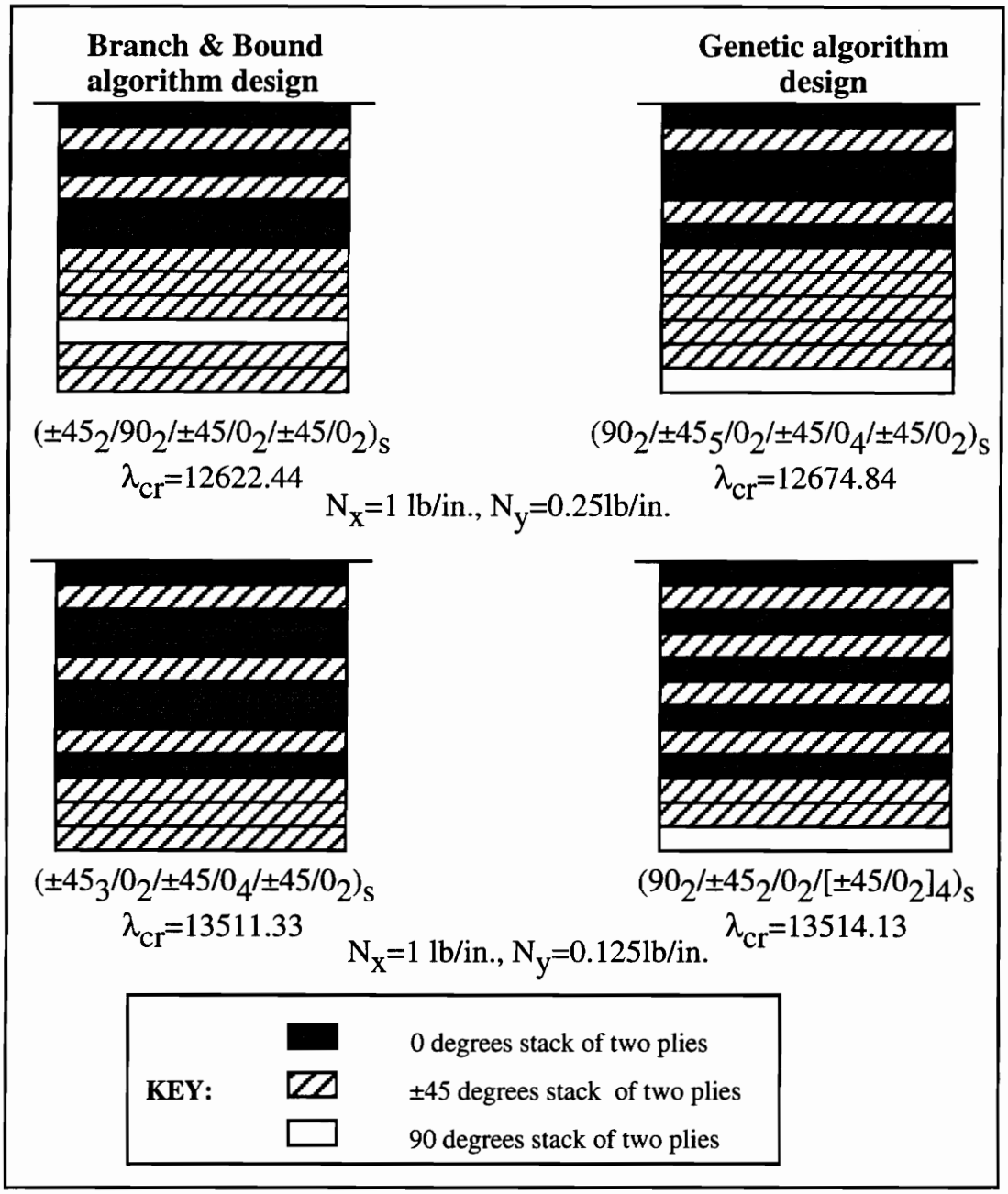



**Figure 3.12. Comparison of optimum designs, 48-ply laminate: buckling with strain and contiguity constraint.**

# Chapter 4

# IMPROVED GENETIC ALGORITHMS FOR MINIMUM THICKNESS COMPOSITE LAMINATE DESIGN

In Chapter 3, a genetic algorithm was exhaustively tested for maximizing the failure load of a fixed thickness laminate by changing its stacking sequence. Kogiso et al. ([Kogiso94b]) applied the same basic algorithm to the dual problem of minimizing the thickness with failure constraints. The variable thickness of the laminates was accommodated through variable-length chromosomes. The minimum thickness problem was found to be more costly to solve than the failure load maximization. The objective of the present study is to explore various techniques for improving the efficiency of the genetic algorithm. This is done by first studying the penalty functions that account for strength and stability constraints and then by improving the genetic operators. Note that, in [Davidor90], a GA for optimizing robot trajectories is described where the genetic operators handle variable-length strings. Finally, the use of a new operator, called scaling mutation, that projects designs toward the feasible domain is investigated. The improvements in the genetic algorithm are shown to reduce the average price of a genetic search by more than 50 %.

## *4.1 Problem Description*

The analysis process is the same as the one from Chapter 3, with the additional possibility of having a varying thickness laminate. The reader familiar with the analysis of Chapter 3 can skip this section, and keep in mind that the plate is now designed for minimum thickness.

The simply supported laminate shown in Figure 4.1 is loaded in the $x$ and $y$ direction by $N_x$ and $N_y$, respectively. The laminate is composed of $N$ plies, each of thickness $t$, and is balanced and symmetric. The orientation of the fibers in the plies are restricted to the discrete set ($0^o$, $+45^o$, $-45^o$ and $90^o$). The plate is $a$ in. long and $b$ in. wide.

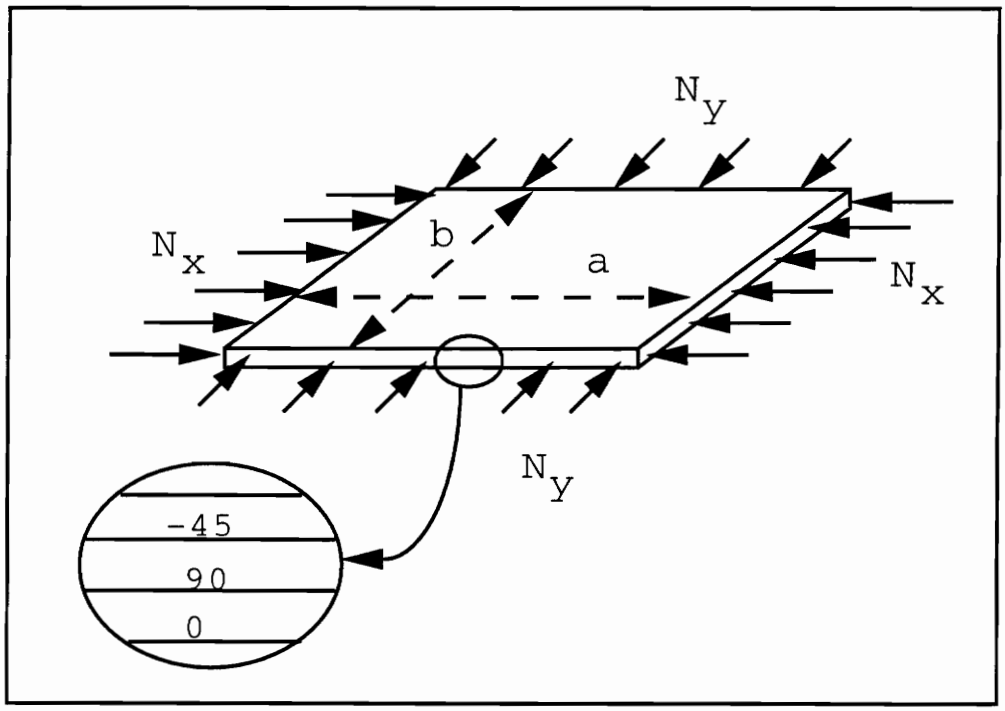



**Figure 4.1. Simply supported laminated plate subjected to normal in-plane loads.**

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

$$\lambda_b(m,n) = \pi^2 \frac{[D_{11}(m/a)^4 + 2(D_{12}+2D_{66})(m/a)^2 + D_{22}(n/b)^4]}{(m/a)^2 N_x + (n/b)^2 N_y}. \tag{4.1}$$

The smallest value of $\lambda_b$ over possible combinations of $m$ and $n$ is the critical buckling load factor $\lambda_{cn}$. If $\lambda_{cn}$ is larger than 1 the laminate can sustain the actual applied loads $N_x$ and $N_y$ without buckling.

The strength of the plate is predicted here by the maximum strain criterion. That is, the plate is assumed to fail if any of the ply strains exceeds its allowable value. In our case $\gamma_{xy}$ is zero and the principal strains in the $i$th layer are related to the loads on the plate by the relations from the classical lamination theory for symmetric and balanced laminates,

$$N_x = A_{11}\epsilon_x + A_{12}\epsilon_y$$

$$N_y = A_{12}\epsilon_x + A_{22}\epsilon_y$$

and,

$$\epsilon_1^i = cos^2\,\theta_i\epsilon_x + sin^2\,\theta_i\epsilon_y$$

$$\epsilon_2^i = sin^2\,\theta_i\epsilon_x + cos^2\,\theta_i\epsilon_y$$

$$\gamma_{12}^i = sin^2\,\theta_i(\epsilon_y - \epsilon_x)$$

where $A_{ij}$ are the coefficients of the extensional stiffness matrix and $\theta_i$ is the fiber orientation in the $i$th layer. The critical strength failure load factor $\lambda_{c\varepsilon}$ is defined as,

$$\lambda_{c\varepsilon} = \min_i \; [ \; min(\frac{\epsilon_1^{ua}}{f|\epsilon_1^i|}, \frac{\epsilon_2^{ua}}{f|\epsilon_2^i|}, \frac{\gamma_{12}^{ua}}{f|\gamma_{12}^i|}) \; ],$$

where $\epsilon_i^{ua}$ are the ultimate allowable strains, and a safety factor $f = 1.5$ is used. If $\lambda_{c\varepsilon}$ is smaller than 1, the plate is assumed to fail. The critical load factor $\lambda_{cr}$ is,

$$\lambda_{cr} = min(\lambda_{cn}, \lambda_{c\varepsilon}).$$

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

The purpose of the optimization is to find the thinnest symmetric and balanced laminate satisfying the 4-ply contiguity constraint that will not fail because of buckling or excessive strains.

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

## 4.2 A Basic Genetic Algorithm For Minimum Thickness Design

The basic genetic algorithm developed in [Kogiso94b], [Nagendra93] and Chapter 3 is used as the baseline for the present work. It is modified to accommodate plates of varying thickness. The two main changes concern coding of the designs and the formulation of the objective function. The reader is referred to Chapter 3 for a more complete description of the basic GA. In short, it is an elitist nonoverlapping GA where selection is performed according to the normalized rank of each individual in the population. A two-point crossover is used in conjonction with mutation and two-stack swap to create offspring designs.

### 4.2.1 Encoding of a Laminate

Each individual in the population represents a design, i.e., a stacking sequence, coded in the form of a string. Geometrical requirements guide the choice of coding. The symmetry constraint is readily accommodated by optimizing only one half of the laminate, the other half being obtained by symmetry. The balance constraint is enforced by associating $+45^o$ and $-45^o$ plies into one $\pm 45^o$ stack. Stacks of $0_2^o$, $\pm 45^o$, and $90_2^o$ are our basic building blocks.

The lengths of all the design strings have to be the same. Because the designs can have different number of plies, empty plies are added as "padding" to make all designs have the same nominal number of plies. The coding used here is the standard stacking sequence notation with the additional symbol $E$ for an empty ply. As a result, laminates can be composed of $0_2$, $\pm 45$, $90_2$ and $E_2$ stacks.

With the padding element $E_2$, the genetic operators of Chapter 3 can handle plates of varying thickness. An additional "compression" operator was used that pushes the empty stacks $E_2$ towards the outside of the laminate. Empty stacks imbedded in the stacking sequence can be produced by crossover, stack swap or mutation. The following crossover is an example.

| | | | | | |
|---|---|---|---|---|---|
| **Parent 1** | $90_2$ | $\overset{\nabla}{\underset{\Delta}{}}90_2$ | $\overset{\nabla}{\underset{\Delta}{}}\pm 45$ | $0_2$ | $\pm 45$ |
| **Parent 2** | $E_2$ | $\overset{\nabla}{\underset{\Delta}{}}E_2$ | $\overset{\nabla}{\underset{\Delta}{}}90_2$ | $\pm 45$ | $0_2$ |
| **Child** | $90_2$ | $\overset{\nabla}{\underset{\Delta}{}}E_2$ | $\overset{\nabla}{\underset{\Delta}{}}\pm 45$ | $0_2$ | $\pm 45$ |

The compression operator would transform the above child chromosome to $[E_2/90_2/\pm 45$ $/0_2/\pm 45]$. Without compression, disruption of the design features by crossover is expected to degrade the search efficiency. Let's consider for example crossover of two almost identical designs,

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Parent 1** | $90_2$ | $90_2$ | $\overset{\triangledown}{\underset{\triangle}{}}E_2$ | $\pm 45$ | $0_2$ | $\overset{\triangledown}{\underset{\triangle}{}}E_2$ | $\pm 45$ |
| **Parent 2** | $90_2$ | $E_2$ | $\overset{\triangledown}{\underset{\triangle}{}}90_2$ | $\pm 45$ | $E_2$ | $\overset{\triangledown}{\underset{\triangle}{}}\pm 45$ | $0_2$ |
| **Child** | $90_2$ | $90_2$ | $\overset{\triangledown}{\underset{\triangle}{}}90_2$ | $\pm 45$ | $E_2$ | $\overset{\triangledown}{\underset{\triangle}{}}E_2$ | $\pm 45$ |

The two parents have same weight and very similar compositions. Ideally, crossover should produce a child design that does not depart too dramatically from the parent designs. In the example, the child design has the same weight as the parents, but the schema $(90_2\ 90_2\ \pm 45\ *\ *)$ that was present in both parents has been lost. Things can even be worse if the break points are chosen as in the next example. Not only the $(90_2\ 90_2\ \pm 45\ *\ *)$ schema shared by both parents is lost, but also the weight of the child is different from the weight of the parents:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Parent 1** | $90_2$ | $\overset{\triangledown}{\underset{\triangle}{}}90_2$ | $\overset{\triangledown}{\underset{\triangle}{}}E_2$ | $\pm 45$ | $0_2$ | $E_2$ | $\pm 45$ |
| **Parent 2** | $90_2$ | $\overset{\triangledown}{\underset{\triangle}{}}E_2$ | $\overset{\triangledown}{\underset{\triangle}{}}90_2$ | $\pm 45$ | $E_2$ | $\pm 45$ | $0_2$ |
| **Child** | $90_2$ | $\overset{\triangledown}{\underset{\triangle}{}}E_2$ | $\overset{\triangledown}{\underset{\triangle}{}}E_2$ | $\pm 45$ | $0_2$ | $E_2$ | $\pm 45$ |

Such reasoning indicates that pushing the empty plies on the outside is necessary if we want to preserve the ability of the crossover to efficiently recombine designs.

The possibility of generating or deleting empty stack of plies $E_2$ was added to the mutation operator. Also, the probability of mutation is now a probability per allele, unlike Chapter 3 where it was a probability per laminate. This means that now many mutations per laminate can occur, although with a very small probability.

### 4.2.2 Objective Function Formulation

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

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

The treatment of the failure and stability constraints is more complicated for two reasons. First, because of the discrete nature of the problem, there may be several feasible designs of the same minimum thickness. Of these designs, we define the optimum to be the design with the largest failure load. Therefore, as in [Kogiso94b], the objective function is linearly reduced in proportion to the failure margin for designs that satisfy or almost satisfy the stability and failure constraints. For such designs the objective function $\Phi_0$ is defined as

$$if\ \lambda_{cr} \geq (1-\delta), \qquad \Phi_0 \ = \ P_c^{n_c}\ \{N \ + \ e\ [(1-\delta) \ - \ \lambda_{cr}]\} \tag{4.2}$$

$\delta$ is a small tolerance parameter added for more flexibility in the definition of feasibility, and $e$ is a small parameter for rewarding constraint margin. When the stability or failure constraints are violated, it is possible to estimate the additional thickness needed to achieve feasibility from simple scaling rule. Based on the scaling rule, the objective function $\Phi_0$ for designs that violate the stability or failure constraints is defined as

$$if\ \lambda_{cr} < (1-\delta), \qquad \Phi_0 \ = \ P_c^{n_c}\ \{\frac{N}{\lambda_{cr}^{P_l}}\} \tag{4.3}$$

where $P_l$ is an exponent used in the scaling law.

Three assumptions were made in [Kogiso94b] to estimate the values of $e$ and $P_l$:

1–. Scaling rule: the strain failure loads are approximately proportional to the thickness of the laminate. The buckling loads are approximately proportional to the cube of the laminate thickness.

2–.The optimum design is located at the boundary between feasible and infeasible domains, i.e. $\lambda_{cr} \simeq 1$.

3–. When estimating parameters for the failure constraints, the contiguity constraints can be neglected ($n_c = 0$).

Too large a value for $e$ would favor thick designs having a large feasibility margin over the optimum design. The maximum value of $e$ was estimated in [Kogiso94b] by comparing the objective function of a hypothetical optimum design having $N$ plies and $\lambda_{cr} = (1-\delta)$ with a heavier feasible design of $cN$ plies ($c > 1$). Neglecting the discreteness of the number of plies, we assume that $c$ can vary continuously. Then, under the above assumptions, the requirement that a design thicker than the optimal should have a higher objective becomes,

$$\begin{aligned} N &< cN + e[(1-\delta) - c^3], \quad \textit{if buckling is critical,} \\ N &< cN + e[(1-\delta) - c], \quad \textit{if strains are critical.} \end{aligned} \tag{4.4}$$

Buckling failure yields the tighter bound on $e$,

$$e < \frac{N(c-1)}{c^3 - (1-\delta)}. \tag{4.5}$$

For $\delta < 0.01$, the upper bound on $e$ increases with $c$ until a point $1 < c_{max} < 1.1$ and then decreases. To be on the safe side, we calculate the upper bound on $e$ for $c = 2$ and $\delta = 0$, what yields $e < \frac{N}{7}$. For the cases considered hereafter, $N > 44$ so Equation (4.4) will be satisfied for $e \leq 6$.

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

$$N < \frac{c\,N}{\lambda_I^{P_l}}. \tag{4.6}$$

According to the scaling rule, $\lambda_I = c$ if the strains are critical and $\lambda_I = c^3$ if buckling is critical. The bounds found in [Kogiso94b] were $P_l > 1$ if strains are critical and $P_l > 1/3$ if buckling is critical.

The main problem of the previous objective function formulation is that for some load cases, the critical load of infeasible designs that are one stack lighter than the optimum feasible designs is very close to 1. Then $P_l$ needs to be very large for the objective function $\Phi_0$ of feasible optimum designs to be smaller than the objective function of good infeasible designs. But a large $P_l$ was seen as being detrimental to the efficiency of the genetic search (Chapter 3, [Richardson89]).

The objective function and operators described so far are those used in [Kogiso94a], [Kogiso94b], [Nagendra93], and Chapter 3. We will now consider modifications for better efficiency in minimum weight laminate design.

## *4.3 Improved Genetic Algorithm for Minimum Thickness Design*

### 4.3.1 Modified Objective Function Formulation

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

$$\begin{aligned} &if\ \lambda_{cr} \geq (1-\delta), \quad \Phi = P_c^{n_c}\,\{N + \varepsilon\,[(1-\delta) - \lambda_{cr}]\}, \\ &if\ \lambda_{cr} < (1-\delta), \quad \Phi = P_c^{n_c}\,\{\frac{N}{\lambda_{cr}^{P_l}}\} + S. \end{aligned} \tag{4.7}$$

$S$ and $(1/\lambda_{cr}^{P_l})$ are both penalties, but their action differs in that $(1/\lambda_{cr}^{P_l})$ is function of the distance to feasibility while $S$ is not. If $S > 0$, $P_l$ can be chosen smaller than the minimum value of 1 derived in [Kogiso94b].

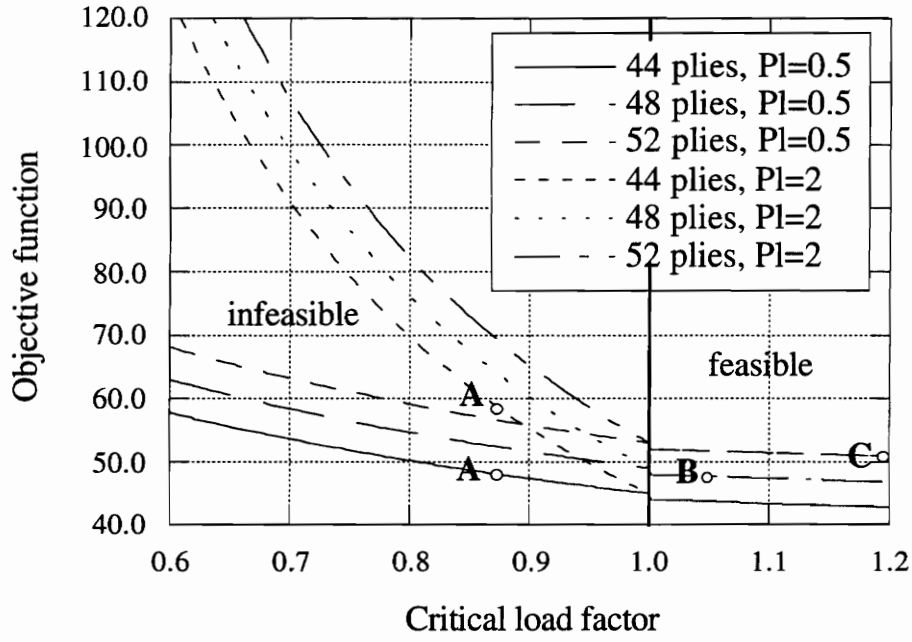



**Figure 4.2. Objective function $\Phi$, $S$ = 1.0, $n_c = 0$. Points A, B, and C correspond to best designs obtained for load case 1.**

Figure 4.2 shows how the objective function $\Phi$ varies as a function of the most critical constraint for $N = 44$, 48, and 52, $P_l = 0.5$ and $P_l = 2$, $S = 1.$, $\delta = 0$, and $\varepsilon = 6$. As an example, we show on the figure the best 44, 48 and 52 ply designs obtained for load case 1 (see Results Section for complete description). The critical failure load factors $\lambda_{cr}$ for these designs are $\lambda_{cr} = 0.879$, 1.040 and 1.201, respectively, and are marked on the curves by points A, B, and C respectively. For $S = 1$ and $P_l = 2$ the optimal design (point B) has a substantial lower objective function than either the infeasible design (A) or the heavier feasible design (C). When $P_l = 0.5$, the advantage of B over A becomes more marginal. Figure 4.3 gives a magnified representation of $\Phi$ such as shown on Figure 4.2 for $P_l = 0.5$ and $S = 1$, with the additional case $P_l = 0.5$ and $S = 0$. The setting $P_l = 0.5$, $S = 0$ makes the infeasible

design (point A) have a lower objective function than the optimum (point B). Figures 4.2 and 4.3 together illustrate the use of $S$ to allow us to reduce the required value of $P_l$.

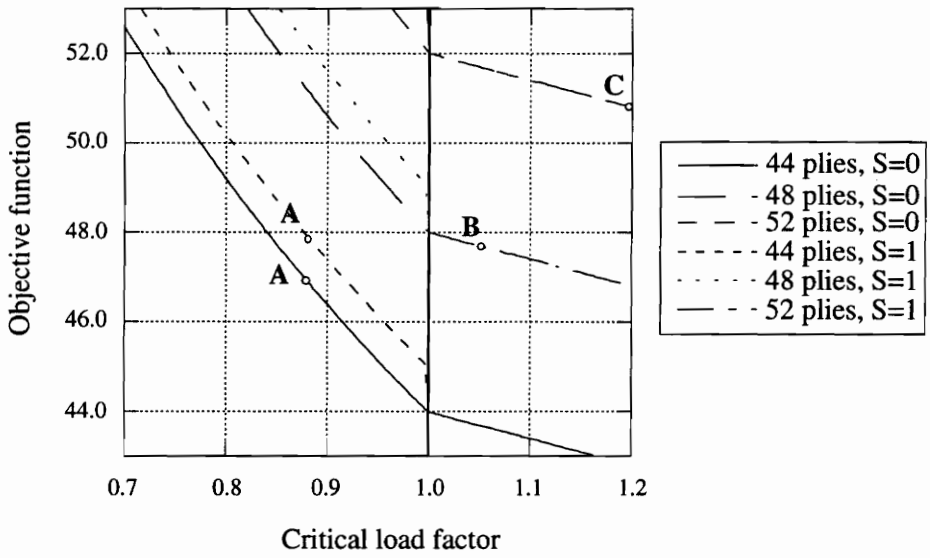



**Figure 4.3. Objective function $\Phi$, $P_l = 0.5$, $n_c = 0$. Points A, B, and C correspond to best designs obtained for load case 1.**

Considering bounds on $S$, we enforce $S > 0$ so as not to penalize feasible designs. Because of balance and symmetry constraints, the best feasible design is two stacks (4 plies) heavier than the best infeasible design. Neglecting the ply contiguity constraint, for the most pessimistic scenario where $\lambda_{cr}$ of the best feasible and infeasible tend towards 1, $S$ can be chosen so as to guarantee the optimality of the feasible design,

$$N + S \geq N + 4. \tag{4.8}$$

So a value of $S = 4$ will guarantee the optimality of the feasible design. More refined bounds on $S$ can be calculated on-line once $P_l$ is set using Equation (4.6). Various strategies have been tested for adapting the value of the step size $S$ in the course of the run, and are discussed in Section 4.4.2.

### 4.3.2 Modified Selection Procedure

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

### 4.3.3 Crossover for Varying Thickness Laminates

We consider three questions for tailoring the crossover operator to the laminate design problem. First, how many break points are optimal; second, should we accord special treatment to empty plies because they control the weight of the laminate; and third, should we always transmit the outermost plies of the thinner parent laminate. We compared the performance of one-point, two-point and uniform crossovers for laminate design. Uniform crossover ([Syswerda89]) is a recombination process where each bit of the offspring has 50% probability of coming from one parent or the other. For $L$-digit strings uniform crossover has a maximum of $L-1$ break points , a minimum of 0 break points, and an average of $(L-1)/2$ break points. The two-point crossover is a standard among practitioners of genetic algorithms because, on the average, it minimizes the probability that crossover separates $k$ $(2 \leq k \leq L)$ beneficially interacting digits in the string (Spears and De Jong 1991). This effect of crossover to separate bits carried by one of the parents is called "disruption". However, empirical evidence has shown that two-point crossover does not always perform the best (cf. [Eschelman89], [Syswerda89]). In [Spears91], it is conjectured that disruption is not always a drawback, but can make the offspring more different from their parents, which might be crucial in avoiding premature convergence, especially for small population sizes. Moreover, two-point crossover minimizes disruption **on the average**, i.e. when sets of digits far apart on the string as well as digits close to each other are considered. In our case, since the string is the direct representation of the laminate cross-section, layers that are close interact more (in terms of bending and contiguity) than layers that are far from each other. One point crossover is less disruptive of adjacent layers.

When two laminates of different thicknesses are mated, the locations of the break points determine the thickness of the child design. Also, with a single-point crossover when the break point falls in the empty part of both parent laminates (in the middle of the $E$'s on the left of the string), the first parent design is cloned. We considered four crossover strategies that have different effects on the thickness of the child design. The first strategy corresponds to the crossover used

in [Kogiso94b] and [Nagendra93], where the break points can fall anywhere in the parent strings. The second strategy called "thin crossover" restricts the location of the break points to the full part of the thinner parent laminate (not in the $E$'s). For such a crossover, the thickness of the offspring is equal to the thickness of one of its parents, but no averaging of the thicknesses is possible. The third strategy called "thick crossover"restricts the break points to the full part of the thicker parent laminate. In that case, some averaging of the parents' thicknesses might occur. The fourth strategy, called "averaging crossover", cuts and splices the parents' strings so that the offspring will always have a thickness which is near the average thickness of the parents.

Finally, we wish to evaluate a heuristic according to which the outermost plies of the offspring should always come from the thinner laminate. The heuristic is based on the fact that the outermost plies influence the laminate buckling behavior more than the innermost plies. In cases where buckling is critical, the highly fitted outermost plies need to be rapidly found by the algorithm and rapidly propagated within the population of designs because they are the "building blocks" necessary to reach optimal region of the design space. If after a substantial number of generations, a thin laminate is competing with a thicker laminate, it must not be far from being feasible and probably carries the crucial outermost plies. We will compare crossover operators that do and do not accommodate this rule.

The aforementioned features were combined in the eleven crossover operators described in Table 4.1. They will be evaluated later. Figures 4.4 and 4.5 show examples of crossovers (the ones omitted can be easily deduced from the others). To facilitate tracking of where offspring's bits come from, a new notation was used instead of the usual laminate notation. The $E$'s symbols are kept for empty plies, but $I_{PI}$'s replace the fiber orientations, where $I$ is the digit position number and $_{PI}$ designates the parent.

### 4.3.4 New Mutation

The traditional mutation operator previously described, appears to have a biased and uneven effect depending on the string length (i.e., maximal laminate thickness) and number of digits denoting full stacks (i.e., actual laminate thickness). To illustrate this, let us calculate the probabilities of adding at least two stacks, deleting at least two stacks and changing the fiber

**Table 4.1. Description of crossover operators.**

| | |
|---|---|
| X1 | One-point crossover, the break point can be chosen anywhere in the string. |
| X1-thin | One-point crossover, the break point is chosen in the full part of the thinner laminate. |
| X1-thick | One-point crossover, the break point is chosen in the full part of the thicker laminate. |
| X1-thick-out | similar to X1-thick + the outermost plies of the offspring come from the thinner parent laminate. |
| X2 | Two-point crossover, both break points are chosen anywhere in the parent string. |
| X2-thin | Two-point crossover, both break points are chosen in the full part of the thinner parent string. |
| X2-thick | Two-point crossover, both break points are chosen in the full part of the thicker parent string. |
| X2-av | Two-point averaging crossover, the offspring has the parents' averaged number of layers ($\pm 4$). A random number of plies is taken from the outside of the first parent laminate, and the inside is completed from the second parent laminate, which is equivalent to breaking each parent string at two points. If the parents have the same weight, it becomes a one-point crossover. |
| X2-thin-out | similar to X2-thin + the outermost plies of the offspring come from the thinner parent design. |
| X2-av-out | similar to X2-av + the outermost plies of the offspring come from the thinner parent design. |
| UX | Uniform crossover |

orientation of at least two stacks per laminate (remember that because of symmetry one digit in the string actually represents two stacks). For the example problems presented in this report, optimal designs have 12 full stacks and a string length of 15. A mutation hitting a full stack has 67% chance of changing the ply orientation and 33% chance of having this stack deleted. If the probability of mutating a stack is set to 0.01, a design with 12 full stacks has $0.67\ (1 - .99^{12})$ $\simeq 0.0750$ probability of having at least two stacks changed, 0.0375 probability of having two stacks deleted and $(1 - .99^3) \simeq 0.0300$ probability of having two stacks added. If the number of full stacks decreases, the probability of stack deletion per string decreases, the probability of stack addition per string increases, and the probability of change of fiber orientation per string decreases. The opposite holds when the number of full stacks increases.

Thus the effect of the mutation operator will vary depending on the thickness of the design

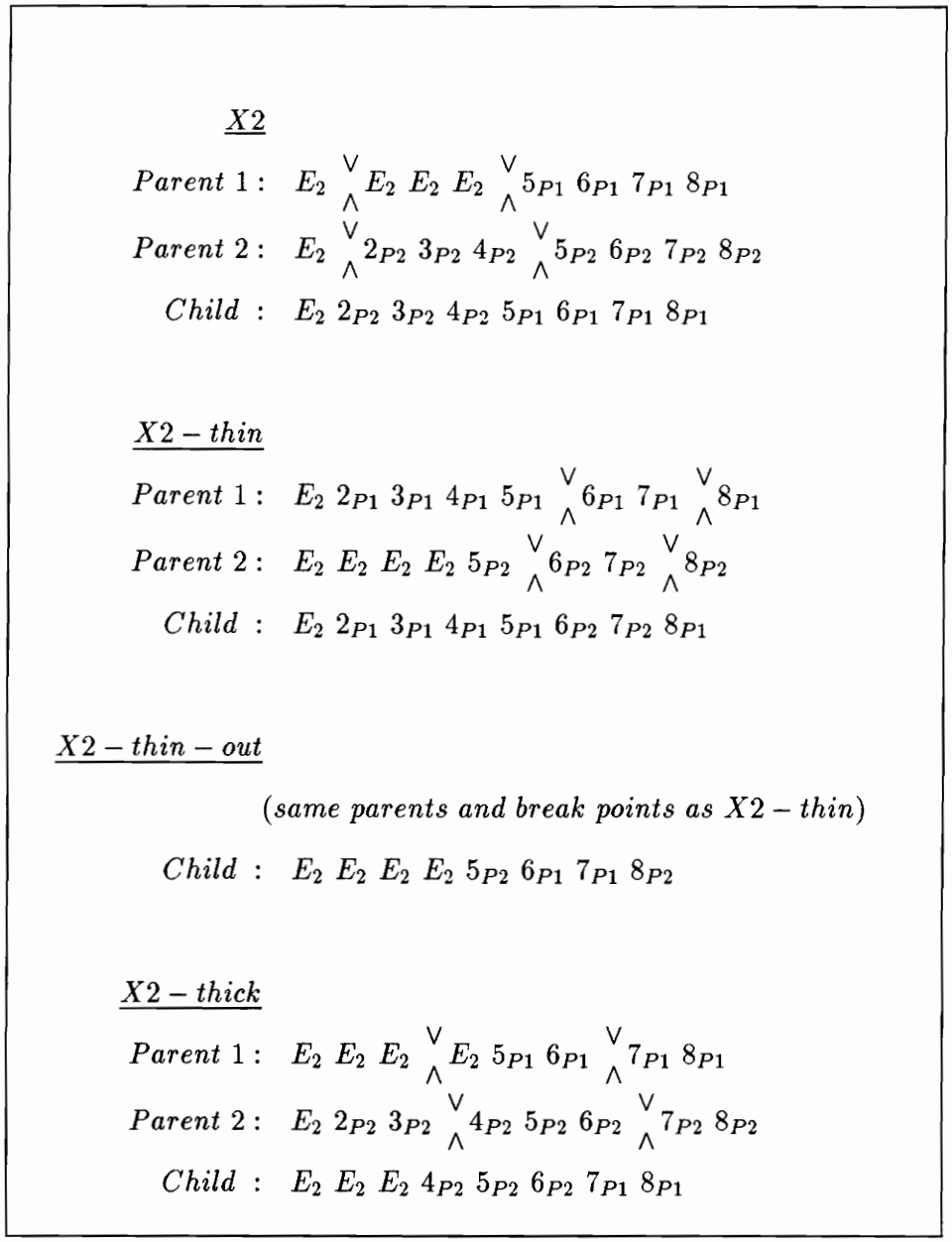



**Figure 4.4. Examples of crossovers.**

and the maximum thickness that can be coded. Moreover, it is not possible to tune independently the probabilities of adding plies, deleting plies and changing fiber orientations.

To correct this, three separate mutation probabilities are used: a probability of adding plies per string ($pa$), a probability of deleting plies per string ($pd$), and a probability of changing the fiber orientation per digit ($pf$). The new mutation adds or deletes two stacks somewhere in the

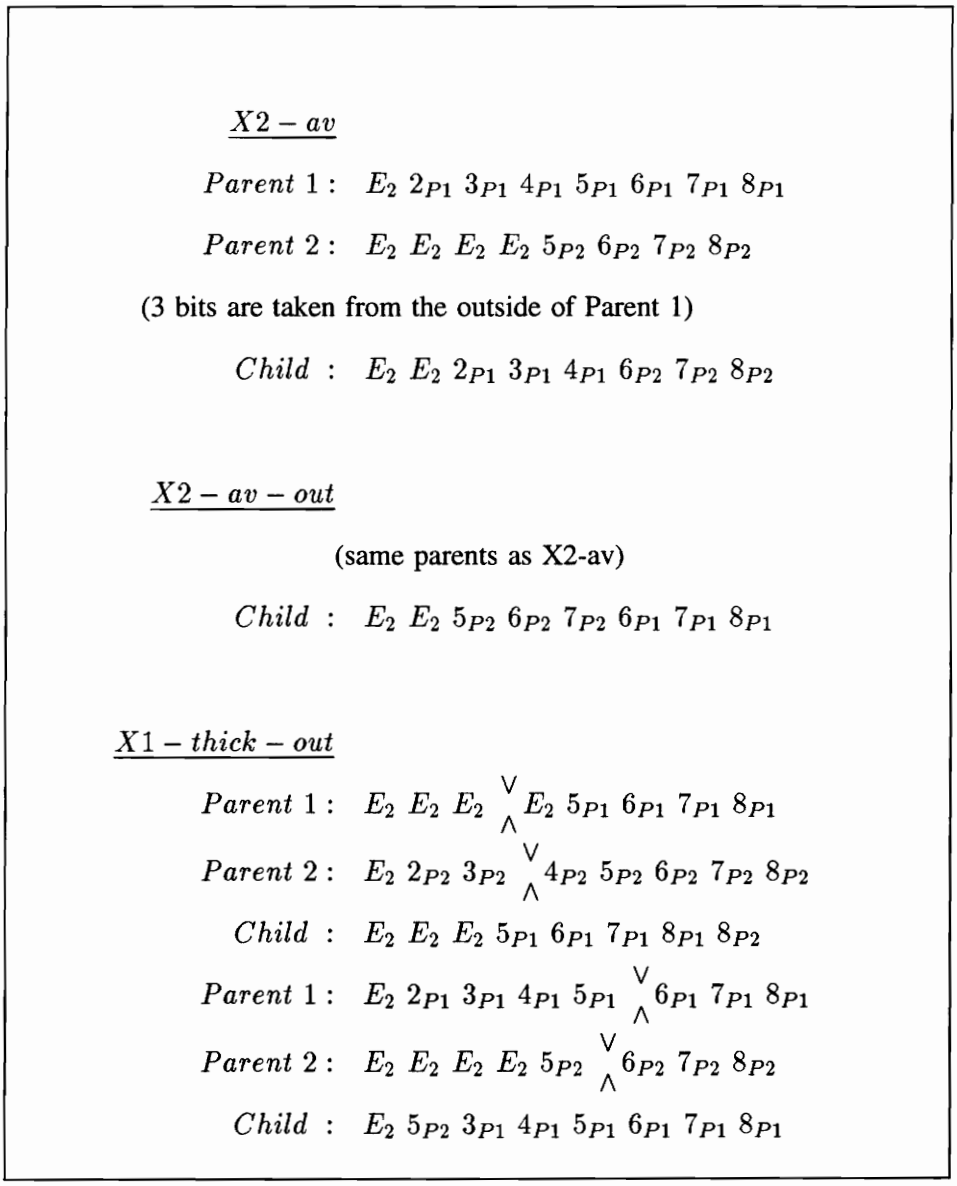



**Figure 4.5. Examples of crossovers (continued).**

laminate if the corresponding probabilistic test is passed. For change of fiber orientation, a test is implemented for each digit coding two full stacks, and if it is passed, the fiber orientation is changed.

### 4.3.5 One-Stack Swap

The original algorithm (cf. Chapter 3) has been developed for a class of relatively easy

problems for which only one load case was considered in the optimization problem. However, for more difficult problems where many loadings are considered when designing a laminate, we believe that two-stack swap shuffles the digits too much. Accordingly, we propose and test a new one-stack swap operator inducing less changes in the string. This operator selects randomly two stacks in the full part of the laminate and swaps them (cf. Figure 4.6).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Before one-stack swap** | $0_2$ | $\pm 45$ | $\underbrace{\pm 45}_{1}$ | $90_2$ | $90_2$ | $0_2$ | $\underbrace{90_2}_{2}$ | $\pm 45$ |
| **After one-stack swap** | $0_2$ | $\pm 45$ | $\underbrace{90_2}_{2}$ | $90_2$ | $90_2$ | $0_2$ | $\underbrace{\pm 45}_{1}$ | $\pm 45$ |

**Figure 4.6. One-stack swap operator.**

## 4.3.6 Scaling Mutation

When a design is infeasible, it makes sense to bias mutations toward adding rather than deleting stacks. Furthermore, using the scaling rule, we can estimate the thickness deficiency and use it to determine the magnitude of the bias. Similarly, for a feasible design, we can bias mutations to favor deleting stacks. The strain load factor ($\lambda_{c\epsilon}$) varies proportionally to the laminate thickness and the critical buckling load factor ($\lambda_{cn}$) varies proportionally to the cube of the laminate thickness. These results are exact if each ply thickness is scaled up or down by the same ratio. When the thickness is changed by adding or deleting a stack, the scaling rules provide only rough approximations. However, they help to direct the population of designs towards optimal weight regions of the design space at a very low calculation cost. Hence scaling can be seen as a "repair operator" ([Michalewicz91], [Orvosh93], [Underbrink94]), where the buckling and strain constraints are being repaired. When performing scaling, the two following equations are solved for $h_b^{new}$ and $h_{st}^{new}$,

$$1 = \lambda_{cn}^{new} = \lambda_{cn} \left(\frac{h_b^{new}}{h}\right)^3 , \tag{4.9}$$

$$1 = \lambda_{c\varepsilon}^{new} = \lambda_{c\varepsilon} \left(\frac{h_{st}^{new}}{h}\right). \tag{4.10}$$

where $h$ is the present laminate thickness, $h_b^{new}$ and $h_{st}^{new}$ are the desirable laminate thicknesses after scaling due to buckling and strains, respectively. The laminate thickness after scaling is $h_{new} = max(h_b^{new}, h_{st}^{new})$. If $h_{new} < h$, the number of plies that would yield laminate thickness the closest to $h_{new}$ is deleted. If $h_{new} > h$, $(h_{new} - h)$ is truncated to the inferior multiple of the stack thickness, and the corresponding number of stacks is added. The consequence of this last truncation is that when a design is almost feasible, scaling is not applied. In that case indeed, it is likely that feasibility can be achieved without increase in thickness only by finding a better stacking sequence. To minimize the influence of the fiber orientations, the plies are added or deleted at the laminate midplane. The orientation of the added plies is random except that no contiguity constraint violation is introduced in the process. Figure 4.7 illustrates how scaling works.

$$\textit{Before Scaling}: \quad [E_4/(\pm 45)_2/90_2/\pm 45/90_4/\pm 45/0_2/90_2/(\pm 45/0_2)_2]_s$$

$$\lambda_{c\varepsilon} = 1.2, \quad \lambda_{cn} = 1.3, \quad h = 0.26\ in., \quad t = 0.005\ in.$$

$$\textit{After Scaling}: \quad [E_4/(\pm 45)_2/90_2/\pm 45/90_4/\pm 45/0_2/90_2/\pm 45/0_2/\pm 45]_s$$

$$h_b^{new} = 0.238\ in., \quad h_{st}^{new} = 0.216\ in., \quad h = 0.24\ in.$$

**Figure 4.7. Scaling mutation.**

Scaling is performed after fitness evaluation and before the selection-recombination process (cf. Figure 3.3 for a flow chart of the algorithm) in order to have all the needed data available ($\lambda$'s and weights). For the designs that have been changed during scaling, an approximate analysis is performed to update their objective function value $\Phi$. In the approximate analysis, the weight and the number of contiguous plies in excess of four are known exactly, and it is assumed that the most critical constraint after scaling has a value equal to 1 (which would be true if we were

scaling all the layers by the same amount, and if $h$ was not limited to an integer multiple of the ply thickness).

## 4.4 Results

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

We consider a graphite-epoxy plate with longitudinal and lateral dimensions $a = 20\ in.$ and $b = 5\ in.$ respectively. The material properties are: $E_1$ = 18.50 $\times$ $10^6$ psi; $E_2$ = 1.89 $\times$ $10^6$ psi; $G_{12}$ = 0.93 $\times$ $10^6$ psi; $\nu_{12}$ = 0.3; $t$ = 0.005 in. ($t$ is the basic ply thickness); $\epsilon_1^{ua}$ = 0.008; $\epsilon_2^{ua}$ = 0.029; $\gamma_{12}^{ua}$ = 0.015. The maximum thickness for a laminate is assumed to be 64 plies (i.e. string length = 64/4 = 16).

Four different load cases are considered: load case 1, $N_x = 13,000.$ lb/in and $N_y = 1,625.$ lb/in; load case 2, $N_x = 12,500.$ lb/in and $N_y = 3,125.$ lb/in; load case 3, $N_x = 9,800.$ lb/in and $N_y = 4,900.$ lb/in; and a multiple load case where the loadings [$N_x = 12,000.$ lb/in, $N_y = 1,500.$ lb/in], [$N_x$ = 10,800. lb/in, $N_y$ = 2,700. lb/in] and [$N_x$ = 9,000. lb/in, $N_y$ = 4,500. lb/in] are considered simultaneously, i.e. the optimal design should be able to withstand each of these loadings. For each load case and each tuning of the genetic algorithm 200 independent searches are performed, each of them stopped after 6000 analyses. The multiple load case was not studied in Chapter 3, and this prevents direct comparisons on the price of the searches between Chapter 3 and this chapter. All the load cases considered here were first treated in [Kogiso94b]. Table 4.2 summarizes the main characteristics of the optimal designs for each of the load cases considered here.

Among the single load cases, load case 1 was found to be easy for the genetic algorithm because strain constraints are by far the most critical. There are many practical optima that are different combinations of the same pool of fiber orientations. Load case 1 has many more

**Table 4.2. Summary of optimal designs.**

| Load case | Optimal design | Failure mode | Number of practical optima |
|---|---|---|---|
| 1 | $[\pm 45_5/0_4/\pm 45/0_4/90_2/0_2]_s$ | strain | >13‡ |
| 2 | $[\pm 45_2/90_2/\pm 45_3/0_2/\pm 45/0_4/\pm 45/0_2]_s$ | strain | 3 |
| 3 | $[90_2/\pm 45_2/(90_2/\pm 45)_2/\pm 45_5]_s$ | buckling | 13 |
| mult. | $[90_4/\pm 45_3/0_4/\pm 45/0_4/90_2/0_2]_s$ | buckling and strain † | 4 |

†: many constraints active.

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

practical optima than any other load case considered here (cf. Table 4.2). Load case 2 is more difficult since it has only 3 practical optima. Although load case 3 has 13 practical optima, it was found to be misleading for the genetic algorithm (Kogiso et al. 1994b): practical optimal designs are composed of $\pm$ 45 and 90 degrees plies exclusively. However, in the course of the search, the algorithm has difficulties removing the 0 degree plies because the strain constraint is critical in these plies. As the number of 0 degree plies in the laminate is reduced, the strains in the remaining 0 degree plies increase. However, once the last 0 degree ply has been removed, the strain constraint is no longer critical, 48 ply feasible designs that fail in buckling can be found. This kind of problem is known as a singular optimum and as being "GA-deceptive", leading to convergence to non-optimal designs. The multiple load problem is also a difficult test case because four failure modes (strain failure due to the first load set and buckling failure due to the three load sets) occur in the optimal region of the design space. Many active constraints around the optimum design mean that many of the design points surrounding the optimum will be infeasible, therefore penalized, which lowers their probability of being selected for reproduction. However, some of these infeasible designs might have given a quick access to the optimum. Typically, the presence of many active constraints around the optimum reduces the number of ways in which designs can be recombined to produce the optimum, thus slowing down the search. The GA exhibits difficulties of a different nature for load case 3 and for the multiple load case. As a result, changes made to the GA often had opposite effects on the performance of the multiple load case and of load case 3. The GA needs to be able to make radical changes to the designs to allow removal of several $0^o$ plies and solve load case 3. On the other hand, the large number

of constraints in the multiple load case makes too much random exploration of the design space a rather inefficient strategy compared to a more careful juxtaposition of building blocks.

In Appendix D, the four load cases are applied to a plate simplified so that it can be encoded as a four-gene long chromosome. Because of the restricted length of the chromosome, it is possible to explore the whole design space. The average objective function value and the associated variance of all schemata are generated . It is observed that the variances of first order schemata are larger for the difficult load cases. The difficult load cases also exhibit deceptive first order schemata (misleading average performance of the schemata). Our experiments confirm the relation between deceptiveness, variance in schemata, and GA hardness.

The rest of the chapter describes the effect of each of the proposed modifications on the genetic search. A new GA, called "GA variable" (GAvar) composed of all the modifications will be used as reference and compared to versions of the GA where the modifications are replaced one at a time by their old counterparts. GAvar is made of the new selection, X1-thick crossover (probability $pc$ = 1), new mutation (probabilities of adding and deleting a stack $pa$ and $pd$ = 0.05, probability of changing the fiber orientation $pf$ = 0.01), one-stack swap ( probability $pp$ = 1). Other parameters are: population size $m$ = 8, $P_l = 0.5$, $S = 1$, $P_c = \sqrt{\frac{10}{9}}$, $\delta = 0.005$. The old GA, called "GA fixed" (GAfix), such as used in [Kogiso94a], [Kogiso94b], and Chapter 3, is composed of the old selection, X2 crossover (probability $pc$ = 1), the old mutation (probability $pm$ = 0.01), two-stack swap ( probability $pp$ = 1), has $P_l = 2$, $S = 0$, and has otherwise the same settings as GAvar. GAfix and GAvar are compared on Figure 4.8 in terms of the reliability $r_1(n_t)$ as a function of the number of analyses $n_t$. The plot represents averages from 200 independent runs made on each one of the 4 test case described above. Because the multiple load case is a particularly important application, the reliabilities specific to that case are also given on Figure 4.9.

The intersections of the curves with the 80% reliability line represent the price of the associated searches. GAvar costs 1450 analyses, compared to 3300 for GAfix, which represents a 56% decrease in the price of the search. Note, however, that with GAfix, the reliability is 0.6 at 1500 analyses, so making two runs of 1500 analyses has a reliability of $(1 - 0.4^2) = 0.84$. This is better than one run of 3000 analyses which has a reliability of 0.78. GAfix needs around 500 analyses to start finding practical optima on the multiple load case (Figure 4.9). Then, GAfix is

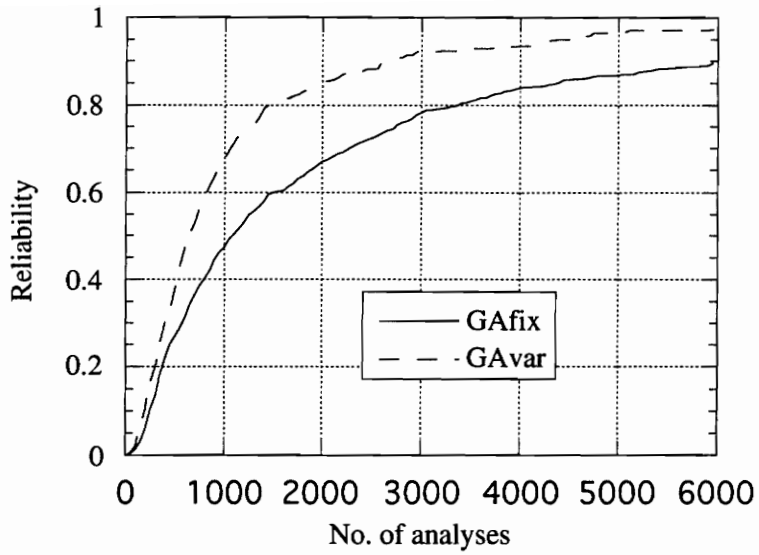



**Figure 4.8. Comparison of GAfix and GAvar, average of all load cases, 200 runs.**

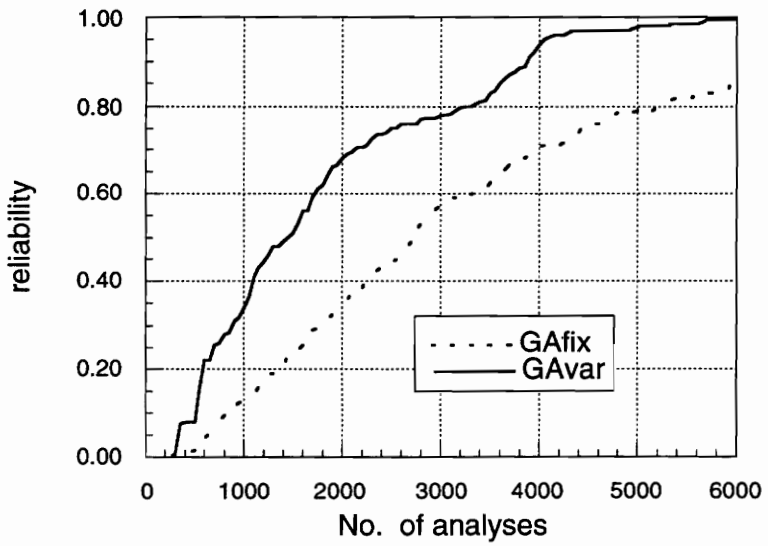



**Figure 4.9. Comparison of GAfix and GAvar, multiple load case, 200 runs.**

slower than GAvar at discovering practical optima for the multiple load case. It is also slower on the multiple load case than it is on all other load cases. The 80% reliability threshold is reached by GAfix after 5200 analyses, versus 3250 for GAvar.

Even though 200 tests are performed for each experiment, non-negligible deviations in the price of the searches have been noticed. The 200 runs have been repeated five times. Assuming a normal distribution of the price of the search, we calculated that at a confidence level of 90% the price of the search is 440±14 for load case 1, 1180±71 for load case 2, 1490±106 for load case 3, 3250±252 for the multiple load case, and 1450±135 on average. Note that the price of the search is more subject to variations on difficult problem cases (third and multiple load case) than on easy cases (first and second load cases). We will now detail the different improvements made to GAfix.

### 4.4.1 Effect of the Penalty Parameters

We discuss in this section the effect of the penalty parameters $P_l$ and $S$ on the price of the search of GAvar. Table 4.3 shows the price of the search and the reliability after 6000 analyses for the different load cases and on the average when $P_l = 0.5$ for $S = 0, 0.5, 1, 2,$ and 4. Tables 4.4 and 4.5 show the effect of changing $P_l$ for $S = 0$ and $S = 1$ respectively.

**Table 4.3. Effect of $S$ on the price of the search / reliability after 6000 analyses, $p_l$ = 0.5.**

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

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

On the average, the trend shown in the tables follows the rule that the minimum penalty is the most efficient (as long as it remains high enough to force convergence to the feasible

**Table 4.4. Effect of $P_l$ on the price of the search / reliability after 6000 analyses, $S$ = 0.**

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

* the GA did not achieve achieve the required 80% reliability.

— the GA did not converge in the feasible domain.

**Table 4.5. Effect of $P_l$ on the price of the search / reliability after 6000 analyses, $S$ = 1.**

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

* the GA did not achieve achieve the required 80% reliability.

— the GA did not converge in the feasible domain.

domain) (Chapter 3, [Richardson89]). There seems to be an advantage at using a combination of fixed penalty and penalty proportional to the distance to the constraints boundary on the third and multiple load cases. This is an experimental argument coming in favor of the use of some fixed penalty. Two other arguments were previously described: some fixed penalty makes it possible to lower the pressure for feasibility in the infeasible domain (i.e. lower $P_l$), allowing a more careful sampling of the infeasible domain while the step guarantees convergence to feasible designs. The use of the fixed step penalty $S$ removes the need for infinitely large $P_l$ for load cases where light infeasible designs can be found very close to the constraint boundary.

## 4.4.2 Adaptive Penalty Parameters

Various strategies for setting the value of the step penalty parameter $S$ adaptively are proposed and tested. The value of $P_l$ is fixed beforehand at 0.5. As shown in Equation (4.8),

assuming that the best infeasible design is one digit (= 2 stacks = 4 plies) lighter than the best feasible results in an absolute upper bound on $S$ equal to 4.

The first strategy for calculating $S$ is to make sure that the objective function (Equation (4.7)) of the best known feasible design is slightly smaller than the objective function of the best infeasible design, i.e.,

$$N_F + \varepsilon\,[(1-\delta) - \lambda_F] < \frac{N_I}{\lambda_I^{P_i}} + S1,$$

or,

$$S1 > N_F + \varepsilon\,[(1-\delta) - \lambda_F] - \frac{N_I}{\lambda_I^{P_i}}, \tag{4.11}$$

where $N_F$ and $N_I$ are the number of plies of the best known feasible and infeasible designs, respectively, and $\lambda_F$ and $\lambda_I$ are the most critical constraints of the best known feasible and infeasible designs, respectively. The best feasible design is defined as the lightest design found so far that satisfies all of the constraints, and the best known infeasible design is the design with no more than 4 contiguous plies, with a most critical load smaller than $(1-\delta)$ and which maximizes the step $S$ necessary for inequality (4.11) to be satisfied. Equation (4.11) gives a lower bound for $S$ called $S1$ based on the designs already sampled by the search. The problem with such a bound arises at the beginning of the search, when very little is known about the design space. Typically, good infeasible designs are found before good feasible designs so that the bound on $S$ calculated by equation (4.11) is a gross overestimate.

In order to get a better estimate of $S$, especially at the beginning of the search, we can add scaling information. In turn, we assume that either the best feasible design or the best infeasible design is of better quality.

If the best so far feasible design is more representative of good designs than the best known infeasible design, it is necessary to estimate what a good infeasible design could be using the scaling rule. The most critical case is when strains are active, i.e., $N_I = N_F - 4$ and $\lambda_I \simeq (N_F - 4)/N_F$, which when substituted in Equation (4.11) yields,

$$S2 > N_F - (N_F - 4)^{(1-P_l)} N_F^{P_l}. \tag{4.12}$$

Typically, when $P_l = 0.5$, $S2$ has values ranging from 2 to 2.2 for $N_F$ varying between 10 and 140.

Conversely, if the best so far infeasible design is chosen as being more reliable in Equation (4.11), and if we assume that $N_F = N_I + 4$ and $\lambda_F \simeq (1 - \delta)$, then Equation (4.11) yields,

$$S3 > N_I + 4 - \frac{N_I}{\lambda_I^{P_l}}. \tag{4.13}$$

The last lower bound on $S$ that can be used is one known from experience. Here, we use the fixed lower bound $S4 = 1.0$, as derived in the previous section.

To maximize the performance of the genetic search, we took $S$ as being equal to its calculated lower bound, for various combinations of $S1$, $S2$, $S3$ and $S4$. The constraint $0 < S < 4$ is always imposed. Results are reported on Figure 4.10. In terms of the price of the search, $S = min(S1, S4)$ is the best strategy, closely followed first by $S = S4$, then by $S$ = min $(S1, S2, S3)$ and $S$ = min $(S1, S2, S3, S4)$. The less efficient strategy (by far) is $S = S1$. As it has already been said, $S1$ is too large early in the search which slows down progress. Overall, the performance of the previous adaptive schemes for tuning the penalty step $S$ was found disappointing since only the scheme $S = min(S1, S4)$ did slightly better than the fixed step $S = S4$.

Therefore, another class of adaptive strategies for tuning $S$ was tested, where the basic idea is to start the search with a small $S$, which is known to permit short cuts through the infeasible domain (cf. Section 3.3.2, [Richardson89]). Later on, $S$ is adjusted depending on where the population has started converging. A value of $S$ is calculated from $S1$ and bounds are superimposed. Here, the value of the upper bound is tuned during the run. To start with, the upperbound and the lowerbound of $S$ are taken to be 0, so that the search is initiated with the smallest possible $S$. $S = 0$ is kept unchanged for $N_{delay}$ analyses. Then, if the $m/2$ best designs are infeasible, where $m$ is the population size, the upperbound on $S$ is increased by $P_{up}$ % of its maximum value, $P_{up}^{max}$ = 4. Reciprocally, if the $m/2$ best designs are feasible, the

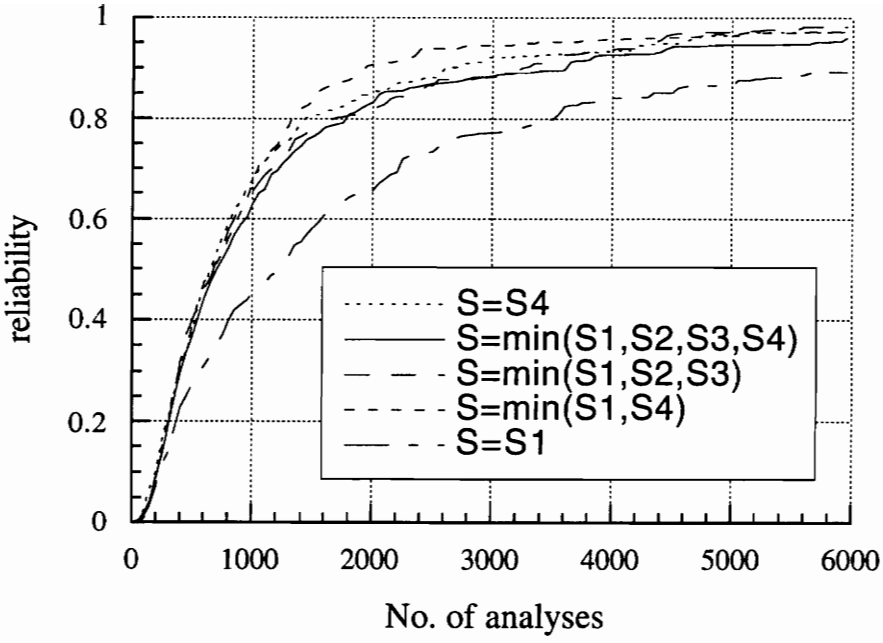


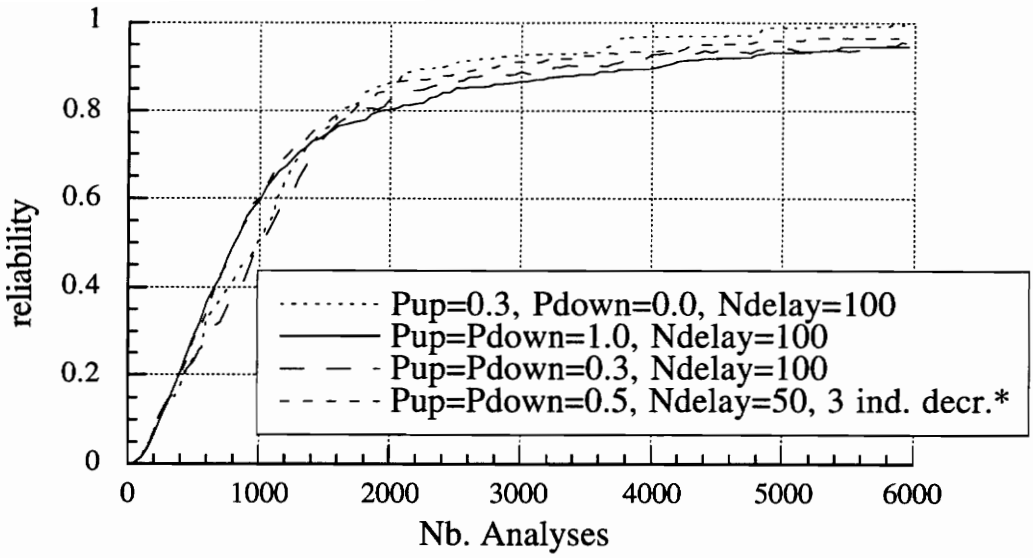

**Figure 4.10. Comparison of different procedures for adapting $S$, $P_l = 0.5$, 200 runs.**

upperbound on $S$ is decreased by $P_{down}$ % of its maximum value, $P_{up}^{max} = 4$ (but it never goes below 0). Because it is difficult to have one half of the population satisfying all of the constraints, the conditions for a decrease in the upperbound of $S$ are rarely satisfied. We think that all the previous strategies might keep too high an upperbound on $S$. So an alternative strategy was tried where the decrease in the upperbound of $S$ occurred when only the 3 best individuals were feasible. Results showing the reliability of the search for various settings are given on Figure 4.11.

Once again, the performance of these adaptive schemes was not noticeably better than the performance yielded by keeping $S$ fixed at a correct value. Moreover, these strategies are seen as missing their primary goal which was to avoid tuning one parameter, $S$. The adaptive scheme itself requires some tuning in order to be efficient, so there is no gain in terms of the number of parameters to be tuned. Adaptive penalty parameters were not used in the final version of GAvar.

### 4.4.3 Selection

Table 4.6 compares the price of genetic searches with the new (no duplicate parents) and




* 3 ind. decr.: if only the 3 best individuals in the population are feasible, then the upperbound on S is decreased.

**Figure 4.11. Comparison of different procedures for adapting the upperbound of $S$, $P_l = 0.5$, 200 runs.**

old selection schemes for each load case and on the average, using the reference GA previously described. For all cases except for load case 2, the new selection procedure was better. The gain in performance is explained by enhanced genetic diversity due to the prohibition on two identical parents. On average, the new selection procedure is responsible for a drop in price of the search from 1600 to 1450 analyses, a saving of 9%.

**Table 4.6. Effect of selection scheme on the price (top entry) and reliability after 6000 analyses (bottom entry).**

| | load case 1 | load case 2 | load case 3 | multiple load case | average |
|---|---|---|---|---|---|
| Old selection | 480<br>1.00 | 980<br>1.00 | 3050<br>0.91 | 3300<br>0.97 | 1600<br>0.97 |
| New selection (no duplicate parents) | 440<br>1.00 | 1180<br>1.00 | 1490<br>0.94 | 3250<br>1.00 | 1450<br>0.98 |

### 4.4.4 Crossover

The prices of the search and reliabilities after 6000 analyses associated with the use of different crossover operators are shown on Table 4.7.

**Table 4.7. Price of the search / reliability after 6000 analyses for various crossovers (crossovers defined in Table 4.1).**

| crossover | load case 1 | load case 2 | load case 3 | multiple load case | average |
|---|---|---|---|---|---|
| X1 | 530 / 1.00 | 1220 / 1.00 | 1920 / 0.97 | 4250 / 0.89 | 1710 / 0.96 |
| X2 | 500 / 1.00 | 1080 / 1.00 | 1780 / 0.99 | 3550 / 0.90 | 1680 / 0.97 |
| UX | 490 / 1.00 | 1440 / 1.00 | 880 / 0.99 | 4430 / 0.87 | 1610 / 0.96 |
| X1-thin | 520 / 1.00 | 1230 / 1.00 | 1420 / 0.98 | 3890 / 0.98 | 1470 / 0.98 |
| X2-thin | 550 / 1.00 | 1260 / 1.00 | 2330 / 0.95 | 3760 / 0.91 | 2070 / 0.96 |
| X1-thick | 440 / 1.00 | 1180 / 0.98 | 1490 / 0.94 | 3250 / 1.00 | 1450 / 0.98 |
| X2-thick | 490 / 1.00 | 1190 / 1.00 | 1380 / 0.92 | 3700 / 0.89 | 1580 / 0.95 |
| X2-av | 540 / 1.00 | 1200 / 1.00 | 1400 / 0.98 | 3800 / 0.88 | 1670 / 0.96 |
| X1-thick-out | 460 / 1.00 | 1220 / 1.00 | 1550 / 0.97 | 3400 / 0.95 | 1480 / 0.97 |
| X2-thin-out | */ 0.10 | */ 0.025 | * / 0.335 | * / 0.005 | * / 0. |
| X2-av-out | 540 / 1.00 | 1370 / 0.99 | 1430 / 0.97 | 3730 / 0.91 | 1740 / 0.96 |

* : undefined because the required 80% reliability was not achieved

Based on the extensive testing reported in Table 4.7, we can answer our three questions on how many break points are best, the treatment of empty plies in crossover, and the propagation of the outside of the thinner laminates.

For the -thick and -thin versions of crossover, the best performance is yielded by the one-point crossovers. When break points are taken anywhere in the parent strings, uniform crossover performs the best and one-point crossover the worst. A possible explanation for this result is that one-point crossover is more likely to degenerate to cloning which occurs when the break point falls in the empty part of the laminates. The better performance of X1-thick is related to its performance on the difficult multiple load case. We conjecture that the multiple load case is efficiently solved when a careful juxtaposition of short substrings occurs. In contrast, the simultaneous suppression of the last $0^o$ plies necessary to reach practical optimum designs in load case 3 benefits from crossover operators highly disruptive at a local level. The most effective crossover operator for solving load case 3 is the uniform crossover, followed by two-point crossover and one-point crossover. Figure 4.12 is a plot of the average number of stacks of

two plies in excess of 2 in the best designs as a function of the number of analyses for the multiple load case, which is the case that causes the most difficulty for satisfaction of the 4-ply contiguity constraint. Figure 4.12 shows that the satisfaction of the contiguity constraints does not seem to play a role in the respective effect of the crossover operators, because the GA always locates good laminates without more than 4 contiguous plies in less than 10 generations for X1-thick and X2-thick. (It can also be argued from Figure 4.12 that the value of $P_c$ is too high, but no other testing has been performed on this subject.) X1-thick has an average price of 1450 while X2-thick costs 1580 analyses. About 8% analyses can be saved by selecting the right number of break points for laminate design.

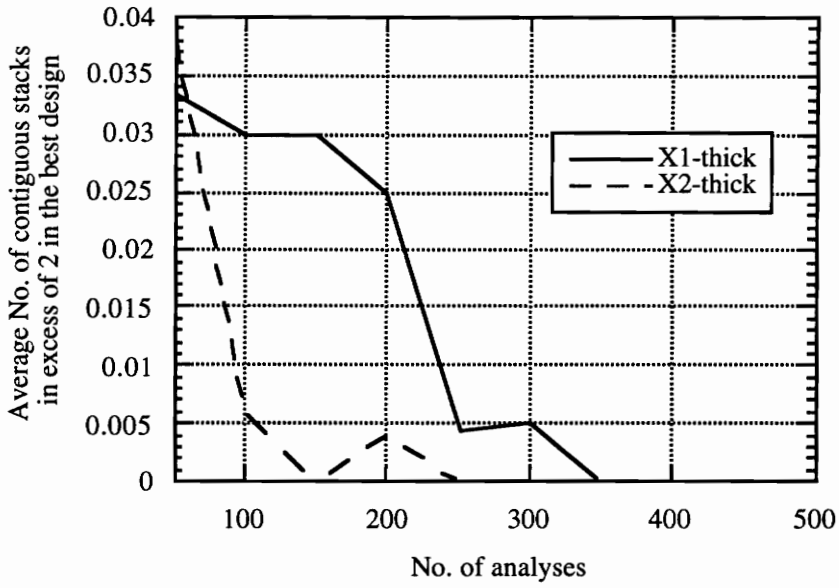



**Figure 4.12. Average number of contiguous stacks in excess of two in the best designs, multiple load case, 200 runs.**

An evaluation of the different strategies for recombining laminate thicknesses can be made either by comparing the effects of X2, X2-thin, and X2-thick, or by comparing X1, X1-thin and X1-thick. The -thick versions of crossover, where the break points are taken in the full part of the

thicker laminate, create thickness averaging when one break point falls inside the thicker laminate but outside the thinner. With the -thin crossovers, the offspring's thickness will be equal to the weight of one of its parents. It can be seen on Table 4.7 that the -thick versions yield the best performance. Except for X1-thin on the load case 3 problem, -thin crossovers are not efficient. X2-av yields a price of the search intermediate between X2-thin and X2-thick. We conclude that for minimum laminate thickness design, a good crossover has a dual role of recombining both the fiber orientations in the layers and the total number of layers. Even though X2-av performs a rigorous weight averaging, it has the flaw of sometimes moving the plies closer or further from the laminate midplane (which increases the risk of loosing the good buckling properties coded in the parents). The -thick crossovers are the only ones that both recombine the parents laminate thicknesses and preserve the ply original location.

Finally, we observe the effect of forcing the offspring's outermost plies to come from the thinner parent by comparing X1-thick with X1-thick-out, X2-thin with X2-thin-out and X2-av with X2-av-out. On the average, such a scheme is always detrimental to the search. Indeed, the -thick-out and -av-out crossovers suffer, like previously mentioned for the -av crossovers, from periodic ply change of location. The -thin-out crossovers create offsprings whose weights are always the weight of the lightest parent, thus biasing the search towards light infeasible designs. X2-thin-out rarely finds feasible designs.

The crossover operator that exhibits best all the above desirable features is X1-thick.

In [Spears93], the respective roles of crossover and mutation are discussed. It is claimed that the primary effect of crossover is to maximize the average payoff in the population, but that, despite conventional wisdom, the advantage of crossover over mutation is not well established on static optimization problems: "If optimality is sought, crossover may be deleterious. If the maximization of accumulated payoff is thought, mutation may be insufficient". In response to this study, a new set of genetic parameters was tried where the probability of crossover was set to $pc$ = 0. To compensate, stochastic exploration was increased (probability of adding and deleting two stacks per laminate $pa = pd$ = 0.3, probability of changing the fiber orientation per allele $pf$ = 0.1, probability of one-stack swap $pp$ = 1.), and the selection pressure was increased by reducing the population size to $m$ = 6 individuals. The performance of this algorithm was terrible: no practical optimum design was found for load case 1, the price of the search for load cases 2 and

3 were 4350 and 5020, respectively, and the reliability for the multiple load case only reached 54 % after 6000 analyses. Those results prove clearly that crossover is needed for laminate design.

### 4.4.5 Mutation

The new mutation operator provides individual control over the probabilities of adding and deleting plies, and of changing fiber orientations. Table 4.8 reports tests performed with the new and old mutations operators. Various combinations of probabilities of adding/deleting plies and changing the fiber orientations are tried. The best set had the probability of adding plies and probability of deleting plies $pa$ and $pd$ equal to 0.05, and probability of changing the fiber orientation $pf = 0.01$. The corresponding probabilities are quite close in the old mutation operator (probability of adding plies $pa = 0.03$, probability of deleting plies $pd = 0.0375$, probability of changing the fiber orientation $pf = 0.0067$). The main difference between new and old mutations is that the new mutation can add plies anywhere in the laminate, whereas the old mutation could only add new plies on the outside (where the $E$'s are). As a result, the new mutation is a more uniformly random operator than the old mutation was. A minor improvement in the price of the search ensues from the new mutation operator (1450 against 1490 analyses).

### 4.4.6 Stack Swap

Because we believe that the digit shuffling caused by two-stack swap was excessive, we introduced one-stack swap that interchanges fewer digits. We also tested whether instead of one-stack swap we can improve performance by reducing the probability of using two-stack swap. Results presented in Table 4.9 confirm that the optimal probability of applying two-stack swap is at or close to 100%. Stack swap is thus a crucial operator. Indeed, while tracking the origin of practical optima for the multiple load case, we found that in 8 cases out of 10 stack swap directly created the optimal design.

One-stack swap improves performance for all cases, but it particularly helps the multiple load case. Table 4.9 shows that lowering the probability of stack swap does not make two-stack swap perform like one. It is logical that making few changes of a large magnitude to the designs (i.e. using two-stack swap at a low rate) is not equivalent to making many small changes (i.e.

**Table 4.8. Effect of the mutation on the price of the search (top entry) and reliability after 6000 analyses (bottom entry).**

| | load case 1 | load case 2 | load case 3 | multiple load case | average |
|---|---|---|---|---|---|
| Old mutation | 530 | 980 | 1750 | 3280 | 1490 |
| pm = 0.01 | 1.00 | 1.00 | 0.96 | 0.92 | 0.97 |
| New mutation | 1025 | 1340 | 1700 | 5460 | 1850 |
| pf=0.01,pa/pd = 0.01 | 1.00 | 0.94 | 0.95 | 0.82 | 0.93 |
| New mutation | 440 | 1180 | 1490 | 3250 | 1450 |
| pf = 0.01,pa/pd = 0.05 | 1.00 | 1.00 | 0.94 | 1.00 | 0.98 |
| New mutation | 390 | 1740 | 1180 | 4030 | 1860 |
| pf = 0.01,pa/pd = 0.1 | 1.00 | 1.00 | 1.00 | 0.88 | 0.97 |
| New mutation | 530 | 1060 | 2250 | 3600 | 1880 |
| pf = 0.001,pa/pd = 0.05 | 1.00 | 0.97 | 0.93 | 0.93 | 0.96 |
| New mutation | 490 | 1690 | 1200 | 4230 | 1820 |
| pf = 0.05,pa/pd = 0.05 | 1.00 | 1.00 | 1.00 | 0.86 | 0.96 |
| New mutation | 640 | 3630 | 1360 | 5920 | 3010 |
| pf = 0.1,pa/pd = 0.05 | 1.00 | 1.00 | 1.00 | 0.81 | 0.95 |

pm: probability of applying the old mutation / bit.

pf: probability of changing the fiber orientation / bit.

pa/pd: probability of adding and deleting a stack / string.

**Table 4.9. Effect of stack swap on the price of the search (top entry) and reliability after 6000 analyses (bottom entry).**

| | load case 1 | load case 2 | load case 3 | multiple load case | average |
|---|---|---|---|---|---|
| Two-stack swap | 460 | 1340 | 3870 | 4840 | 2400 |
| pp = 0.5 | 1.00 | 1.00 | 0.86 | 0.88 | 0.93 |
| Two-stack swap | 440 | 1480 | 1670 | 4750 | 2080 |
| pp = 0.8 | 1.00 | 0.99 | 0.95 | 0.86 | 0.95 |
| Two-stack swap | 460 | 1480 | 1570 | 4020 | 1990 |
| pp = 1.0 | 1.00 | 1.00 | 0.97 | 0.85 | 0.96 |
| One-stack swap | 490 | 1210 | 5170 | 3110 | 1820 |
| pp = 0.5 | 1.00 | 1.00 | 0.82 | 0.92 | 0.93 |
| One-stack swap | 440 | 1080 | 2190 | 3100 | 1500 |
| pp = 0.8 | 1.00 | 1.00 | 0.94 | 0.92 | 0.96 |
| One-stack swap | 440 | 1180 | 1490 | 3250 | 1450 |
| pp = 1.0 | 1.00 | 1.00 | 0.94 | 1.00 | 0.98 |

*pp*: probability of stack swap.

intensively using one-stack swap). Table 4.9 also shows that the optimal probability of applying one-stack swap is about 1.0.

### 4.4.7 Scaling Mutation

We tried a strategy where most of the additions and deletions of stacks were taken care of by scaling mutation, while reducing the rate of random mutation. We lowered the probabilities of adding and deleting stacks from 5% to 1%, and we applied scaling mutation on 10% of the designs. Table 4.10 reports what consequences scaling mutation has on the price of the search and reliability after 6000 analyses.

**Table 4.10. Effect of scaling mutation on the price of the search (top entry) and reliability after 6000 analyses (bottom entry).**

| | load case 1 | load case 2 | load case 3 | multiple load case | average |
|---|---|---|---|---|---|
| Scaling | 320<br>1.00 | 990<br>1.00 | 1570<br>0.96 | 3110<br>0.96 | 1310<br>0.97 |
| No scaling | 440<br>1.00 | 1180<br>1.00 | 1490<br>0.94 | 3250<br>1.00 | 1450<br>0.98 |

Scaling is beneficial to load cases 1, 2 and the multiple load case. In every load case, scaling helps finding designs in the optimal weight region early on in the search (before 500 analyses). Load case 1 is the ideal case where there are sufficiently many practical optima for scaling to have some chance of getting close to one. On the difficult third and multiple load cases, locating the optimal weight region does not mean getting very much closer to the optimum. On these cases, scaling yields mainly largely infeasible designs that have little chance to survive. It just wastes some of the genetic memory carried in the population. That explains why no decrease in price of the search is seen on load case 3, and why the improvement noted on the multiple load case is marginal (remember that this load case has a large standard deviation in price). Figure 4.13 shows that scaling helps early on in the search. After 4600 analyses, the GA without scaling has a higher reliability than the GA with scaling. It can be seen from Figure 4.14 that scaling

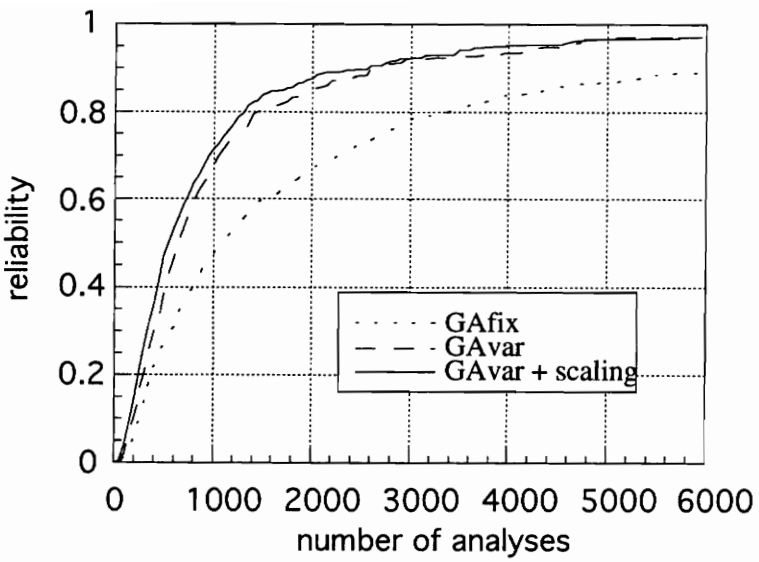



**Figure 4.13. Effect of scaling mutation on the reliability, 200 runs.**

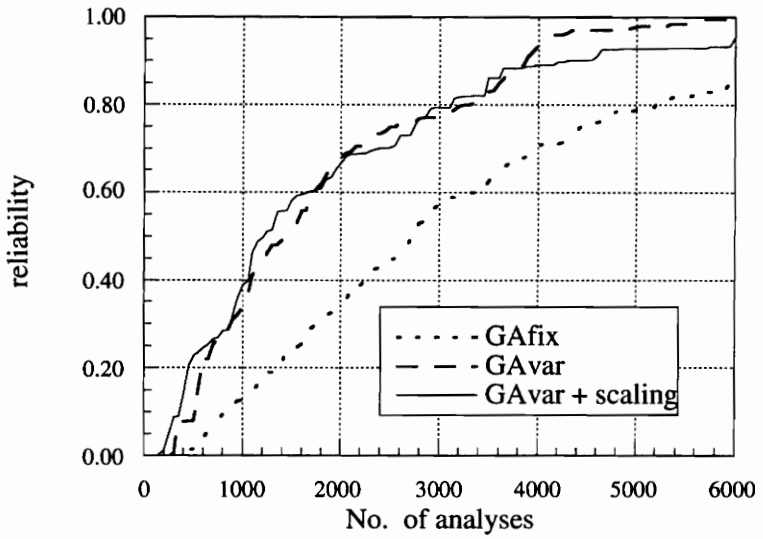



**Figure 4.14. Effect of scaling mutation on the reliability, multiple load case, 200 runs.**

marginally helps on the multiple load case, and even seems detrimental to the search after about 4000 analyses.

More generally, Figures 4.13 and 4.14 indicate that scaling mutation should be turned off after the early stages of the search. Scaling is really intended at helping the GA locating optimal weight regions. Note that locating optimal weight regions is not a critical issue for the test problems treated here. The GA without scaling will always converge to 48 plies designs in about 500 analyses. On the other hand, the use of scaling mutation is very promising on cases where finding the optimal weight is a challenge. Such problems might arise when the string is much longer.

## *4.5 Minimum Thickness Design by Genetic Algorithm: Observations*

The genetic algorithm devised in Chapter 3 has been improved and tested for minimum thickness design of composite plates.

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

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

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

In the next chapter, we study how the knowledge gained on single laminate problems generalize to the case of more complex structures, where many laminates are interacting. The structure under investigation is a wing box beam.

# Chapter 5

# COMPOSITE WING BOX OPTIMIZATION BY GENETIC ALGORITHMS

Thin-walled box beams are extremely efficient structures. The thin-walled box beam in some form has become a fundamental structural element in the construction of aircraft, ships, offshore platforms, bridges and the cores of tall buildings. The advantage of the hollow section is that the material is efficiently used both in flexure and in torsion. The wing box structure we will be considering hereafter is a generic and simplified composite wing box used previously in [Haftka83] for the study of global damage tolerance. Another example of wing box optimization by genetic algorithm for minimum weight and frequency response with stress and displacement constraints can be found in [Hajela92]. In [Quilici92], parts of helicopter structures are modelled as composite box beams and optimized for minimum weight with displacement, stress failure, buckling and frequency constraints. Composite box-beam structures are optimized in [Ragon94] and a detailed analysis of the effects of subcomponents interaction is given.

The wing box is a complex, highly redundant structure, where all the components interact through mechanical coupling. The use of composite materials in the construction still increases the complexity.

The purpose of this Chapter is to evaluate the potential of genetic algorithms for optimizing composite wing boxes. We start by describing the wing box model and formulate the optimization problem. We then propose and study an approximate analysis, that provides information about couplings between elements of the wing box and permits faster analysis. Necessary changes to the GA developed for single laminate so that it can handle wing boxes are described next. Finally, optimization results are given. Two aspects of the design problem are expected to affect the genetic search and are investigated: first, the design space is larger because multiple laminates are optimized; second, couplings between laminates are of a different nature than couplings between layers within the same laminate. Repercussions of these two factors on the optimal population size and probability of stack swap are discussed.

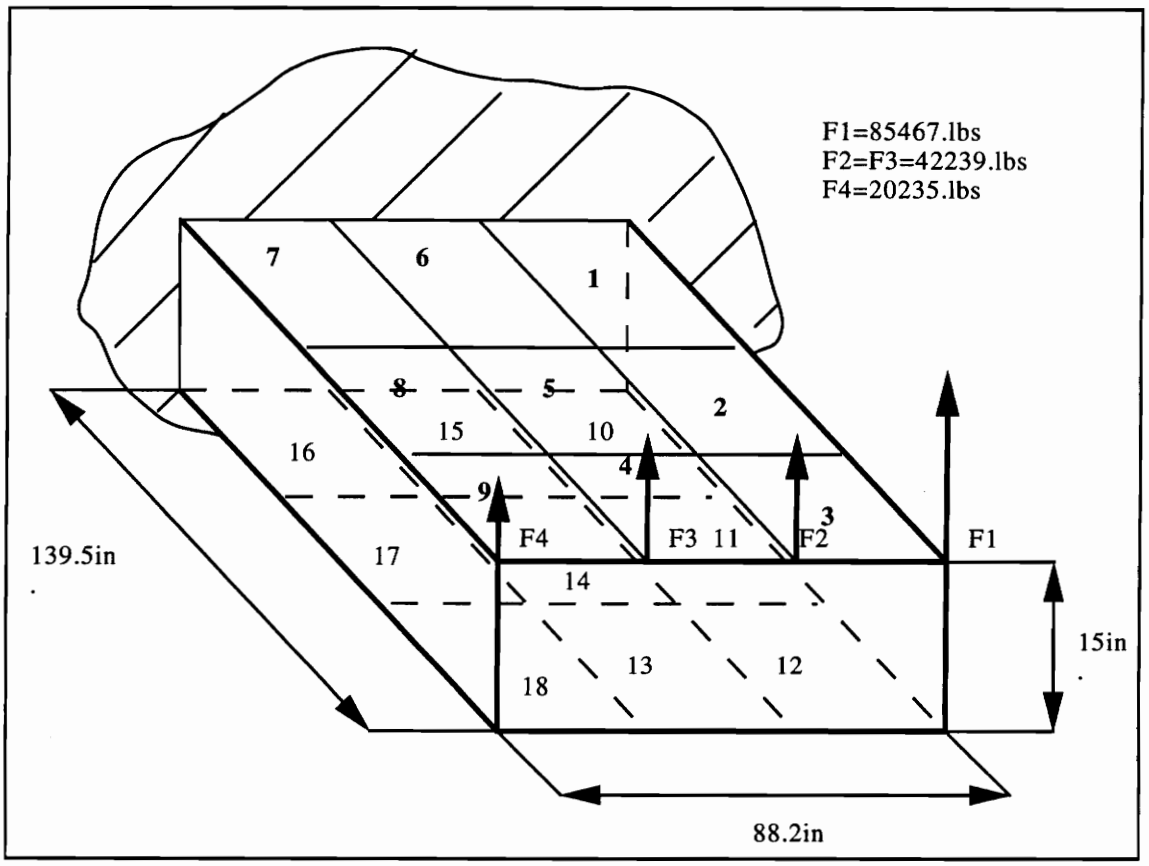



**Figure 5.1. Wing box structure.**

# *5.1 Problem Description and Analysis*

## 5.1.1 Description of the Wing Box

The structure we consider here is an unswept, untapered, wing box with four spars and three ribs. The wing box is clamped at the root and subjected at the tip to the applied load distribution shown in Figure 5.1. The loads on the wing were calculated for a plane with a weight of 50715 lbs., a load factor of 2.5 and a safety factor of 1.5 in agreement with the Federal Aviation Administration regulations. The width, length and depth dimensions of the rectangular wing box are shown in Figure 5.1. There are nine upper and nine lower wing skin panels, whose numbering is shown in Figure 5.1. Webs of ribs and spars are modelled by 21 shear web elements. All the panels are made of graphite-epoxy T300/5208, whose material properties are, $E_1$ = 18.50 $10^6$ psi (127.59 GPa), $E_2$ = 1.89 $10^6$ psi (13.03 GPa), $G_{12}$ = 0.93 $10^6$ psi (6.41 GPa), $\nu_{12}$ = 0.3, the

basic ply thickness is $t$ = 0.005 in. (0.0127 cm.). The ultimate allowable strains are $\epsilon_1^{ua}$ = 0.008, $\epsilon_2^{ua}$ = 0.029, and $\gamma_{12}^{ua}$ = 0.015. The panel laminates are required to be symmetric and balanced. Normal- shear extensional, and extensional-flexural couplings are thus removed. The webs of ribs and spars have a fixed stacking sequence of $[(\pm 45)_5]_s$. The stacking sequences of the panels making the upper and lower skin of the wing are the object of the design process.

## 5.1.2 Finite Element Analysis and Buckling Models

The global wing box analysis is organized as shown in Figure 5.2.

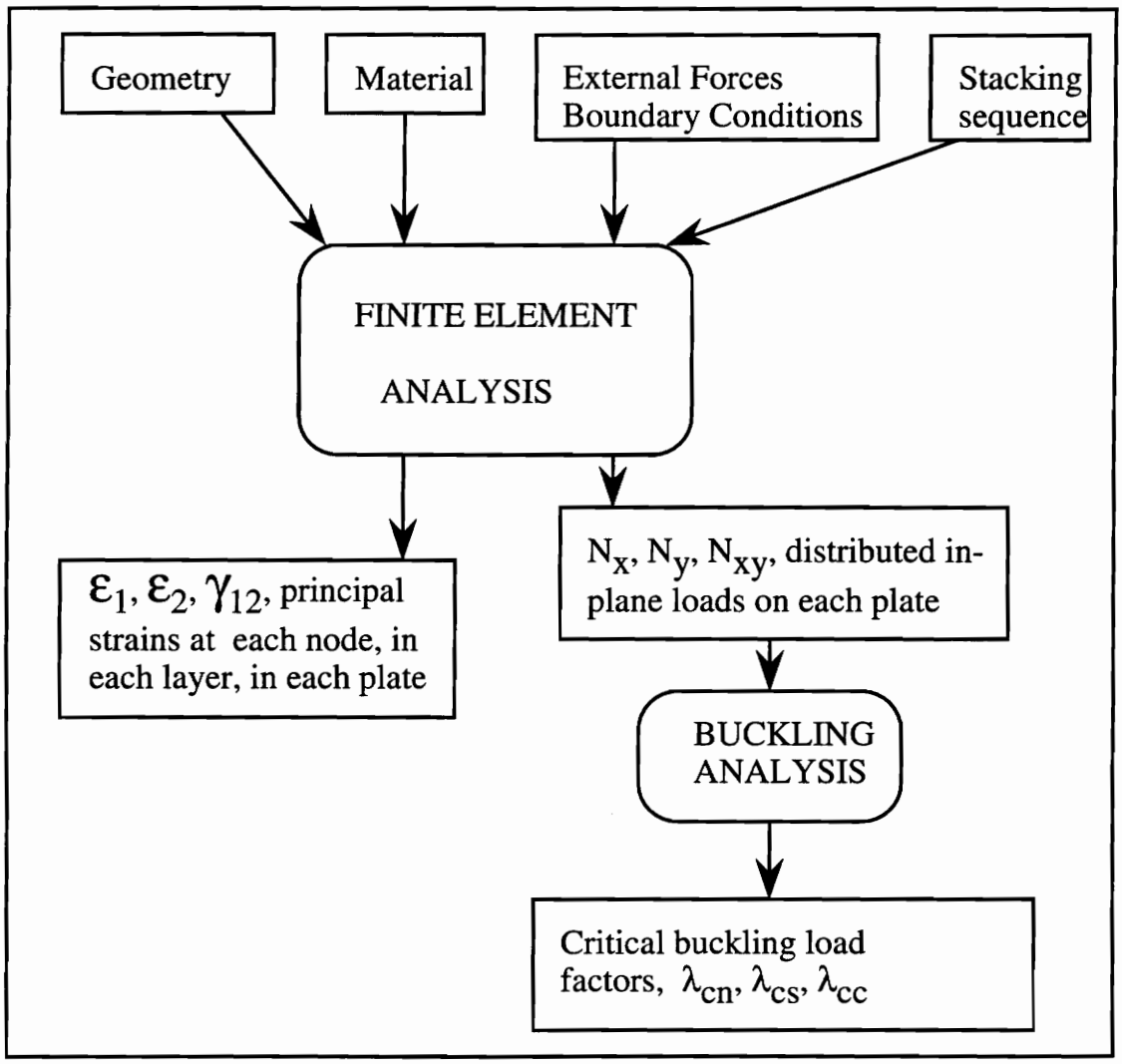



**Figure 5.2. Wing Box Analysis Flow Chart.**

The finite-element model has 32 nodes, and 72 degrees of freedom. The 18 upper and lower surface panels and the 21 webs of ribs and spars are rectangular membrane (plane stress elements) with 4 nodes per element and 2 degrees of freedom per node. Displacements of the nodes are obtained by solving the equations of equilibrium

$$[K]\,\{U\} \;=\; \{F\}, \tag{5.1}$$

where $[K]$ is the global stiffness matrix of the structure, $\{U\}$ is the vector of nodal displacements, and $\{F\}$ is the global force vector (data). The derivation of the element stiffness matrix and details of the finite element analysis are given in Appendix B. The complete wing box analysis requires 0.09 CPU seconds on an IBM AIX RISC/model 6000 Version 3 computer. It was possible to shorten the time of one analysis by taking advantage of the fact that only the skin stacking sequence changes during the optimization. So the part of the assembled stiffness matrix $[K]$ corresponding to the webs of ribs and spars can be calculated once at the beginning of the genetic search and stored for future use. With this modification, one reanalysis of a wing box takes 0.065 CPU seconds, about 28% less than a complete analysis.

After solving Equation (5.1) for the nodal displacements, element strains and in-plane loads are calculated ( cf. Appendix B for more details). Strains are obtained in the principal material directions, in each layer, at each node of each plate. The strain with the largest magnitude among the four nodes is considered to be the strain associated with the element and layer. We use $\epsilon_1^{i,j}$, $\epsilon_2^{i,j}$ and $\gamma_{12}^{i,j}$ to denote the principal strains in the $i$th layer of the $j$th laminate.

The critical strength failure load factor in the $j$th laminate, $\lambda_{c\epsilon}^{j}$, is defined as,

$$\lambda_{c\epsilon}^{j} \;=\; \min_{i} \;\; [ \;\; min(\frac{\epsilon_1^{ua}}{f|\epsilon_1^{i,j}|}, \frac{\epsilon_2^{ua}}{f|\epsilon_2^{i,j}|}, \frac{\gamma_{12}^{ua}}{f|\gamma_{12}^{i,j}|}) \;\; ],$$

where $i$ is the layer number and a safety factor $f \;=\; 1.5$ is used. If $\lambda_{c\epsilon}^{j}$ is smaller than 1, the $j$th plate is assumed to fail.

Values of element in-plane loads $N_x$, $N_y$, $N_{xy}$ (cf. Figure 5.4) are used to calculate buckling constraints.

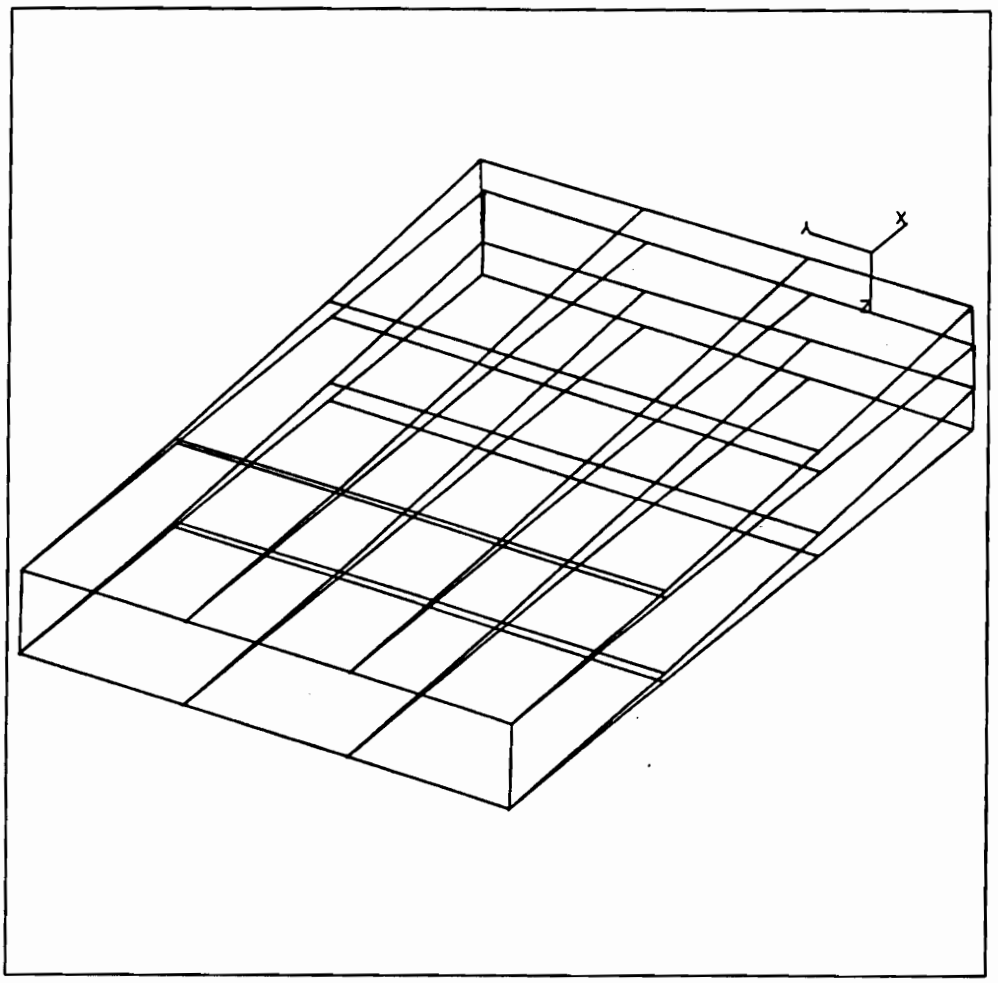



**Figure 5.3. Initial and deformed wing box.**

### 5.1.3 Buckling Analysis

The three different buckling modes we consider are buckling due to normal in-plane loads $N_x$ and $N_y$, buckling due to in-plane shear loads $N_{xy}$, and buckling due to combined in-plane normal and shear loads. The buckling models used assume, like in Chapter 3, specially orthotropic laminates, i.e., the bending-twisting coupling is neglected ($D_{16}$ and $D_{26}$ are assumed small). The buckling model for normal in-plane loads $N_x$ and $N_y$ is identical to the one described in Section 3.1.1. It yields a critical normal buckling load factor in the $j$th plate $\lambda^j_{cn}$.

In the shear buckling model, we additionally assume that the plate has an infinite length. Modelling of the buckling of a finite plate under uniform shear loading involves solving an eigenvalue problem iteratively ([Whitney85]), which is too expensive computationally. Instead,

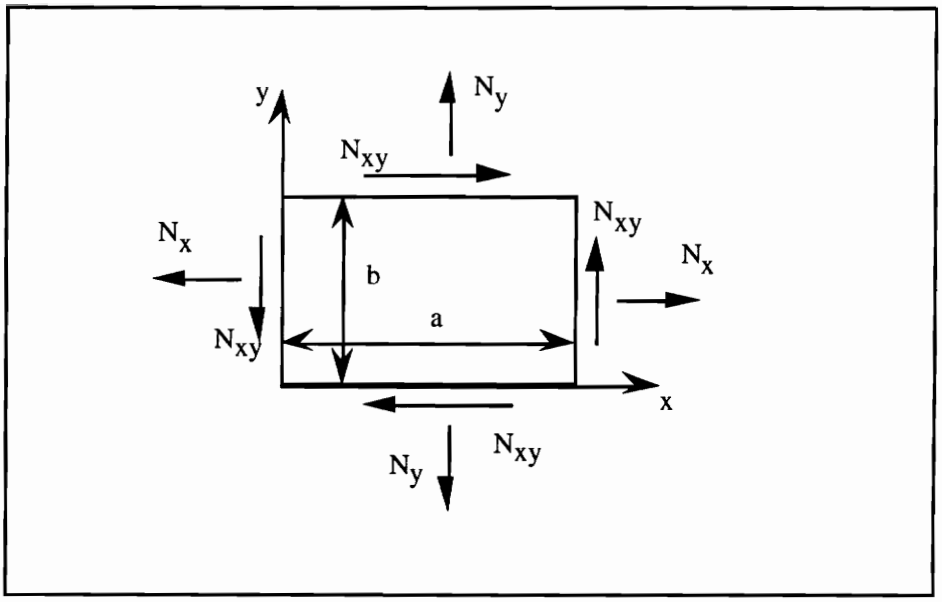



**Figure 5.4. Laminated panel under in-plane loads.**

analytical solutions available for plates infinite in length in the $x$ direction ([Whitney85]) are used here as approximations to the buckling load of a finite length plate in order to limit the computational cost. Figure 5.5 shows the infinite strip under consideration. $b$, the width of the strip, is chosen as the smallest side of the rectangular plate.

Solutions for the critical buckling load factor $\lambda_{cs}$ due to in-plane shear are presented in ([Whitney85]) for a complete range of the variable $\Gamma$, where

$$\Gamma = \frac{\sqrt{D_{11}D_{22}}}{D_{12} + 2D_{66}} \tag{5.2}$$

For $1 \leq \Gamma \leq +\infty$, the critical buckling load factor due to in-plane shear $\lambda_{cs}$ is,

$$\lambda_{cs} = \frac{4\beta_1 (D_{11}D_{22}^3)^{1/4}}{b^2 N_{xy}}, \tag{5.3}$$

and for $0 \leq \Gamma \leq 1$,

$$\lambda_{cs} = \frac{4\beta_1 \sqrt{D_{22}(D_{12} + 2D_{66})}}{b^2 N_{xy}}. \tag{5.4}$$

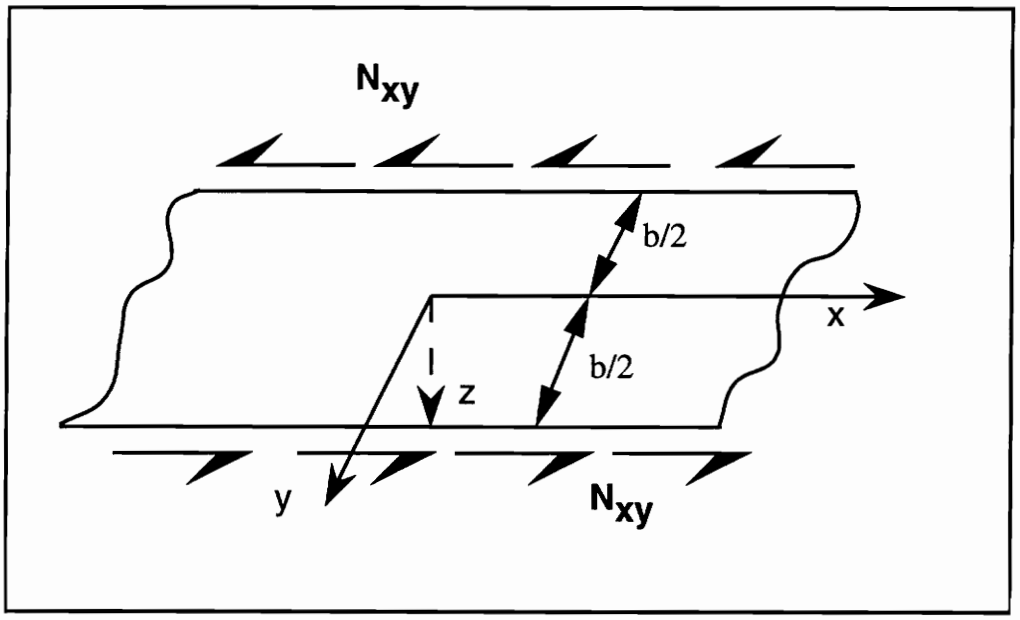



**Figure 5.5. Infinite strip under shear load.**

The critical shear buckling load factor in the $j$th laminate is denoted $\lambda_{cs}^{j}$ . Values for $\beta_1$ can be found in Table 5.1. A more complete derivation of the critical buckling load due to in-plane shear is given in Appendix B.

**Table 5.1. Coefficients $\beta_1$ for critical buckling load factor due to shear (from [Whitney85]).**

| $\Gamma$ | $\beta_1$ |
|---|---|
| 0.0 | 11.71 |
| 0.2 | 11.80 |
| 0.5 | 12.20 |
| 1.0 | 13.17 |
| 2.0 | 10.80 |
| 3.0 | 9.95 |
| 5.0 | 9.25 |
| 10.0 | 8.70 |
| 20.0 | 8.40 |
| 40.0 | 8.25 |
| $\infty$ | 8.13 |

In a first approximation, the combined effect of in-plane normal and shear loads on buckling of simply supported plates can be accounted for through the expression ([Lekhnitskii68]),

$$\frac{1}{\lambda_{cn}} + \frac{1}{\lambda_{cs}^2} \leq 1, \tag{5.5}$$

for buckling not to occur. By analogy with $\lambda_{cn}$ and $\lambda_{cn}$, a critical buckling load factor $\lambda_{cc}$ is defined that accounts for the combined effect of normal and shear loads,

$$\lambda_{cc} = \frac{1}{\frac{1}{\lambda_{cn}} + \frac{1}{\lambda_{cs}^2}}. \tag{5.6}$$

Equation (5.6) indicates that $\lambda_{cc}$ will always be smaller (hence more critical) than $\lambda_{cn}$ if $N_x$ and $N_y$ are compressive (in that case the failure load factors vary between 0 and $\infty$). Panels of the lower skin have $N_x$ in tension and the previous argument might not be true. However, buckling due to normal in-plane loads is not critical there and is disregarded. The critical combined buckling load factor on plate $j$ will be denoted $\lambda_{cc}^j$.

### 5.1.4 Optimization Problem Formulation

The purpose of the optimization is to minimize the total number of plies making up the upper and lower skin, by changing the number of plies and the orientation of the fibers in each ply. Once again, because of manufacturing practices, the orientation of the fibers within each layer is restricted to the discrete pool ($0^o$, $90^o, \pm 45^o$). The laminates of the upper and lower surfaces should not fail in any of the three buckling modes considered (normal, shear and combined buckling), or because of strength deficiency. Additionally, we apply the previously described four-ply contiguity constraint in order to reduce risks of matrix cracking. The loads imposed on the structure make only panels of the upper surface prone to buckling due to normal compressive loads, so there is no need to calculate $\lambda_{cn}$ and $\lambda_{cc}$ in the lower skin (laminates numbered $j$ = 10, 18, cf. Figure 5.1). The optimization of the skin of the wing box is formulated as follows:

| | |
|---|---|
| **Minimize** | $N$, the total number of layers in the upper and lower skin panels, |
| **by changing** | $N$ and the fiber orientation $\theta_i^j$ in each ply $i$ of each panel $j$ = 1, 18 , |
| **subjected to** | $\theta_i^j \in (0^o, 90^o, 45^o)$, $j$ = 1, 18, |
| | laminates are balanced and symmetric, |
| | there are no more than four contiguous plies with the same fiber orientation, |
| (strength) | $\lambda_{c\epsilon}^j \geq 1$, $j$ = 1, 18, |
| (buckling upper skin) | $\lambda_{cs}^j \geq 1$, $\lambda_{cc}^j \geq 1$, $j$ = 1, 9, |
| (buckling lower skin) | $\lambda_{cs}^j \geq 1$, $j$ = 10, 18. |

Four optimization problems based on the above wing box problem will be considered. They correspond to four levels of complexity, because they have increasing sizes of design space, and because they give evidence of an increasing number of false optima. These different variations on the wing box problem are called One-Laminate, Two-Laminate, Six-Laminate and 18-Laminate Wing Box problems respectively. In the One-Laminate Wing Box Problem, all the panels making up the upper and lower skin have the same stacking sequence. In the Two-Laminate Wing Box problem, all the panels of the upper skin are the same, and all the panels of the lower skin are the same, but the lower skin can be different from the upper skin. In the Six-Laminate Wing Box, the panels are the same chordwise, e.g., panels 1,6 and 7 are identical, panels 2,5, and 8 are identical, etc. (cf. Figure 5.1).

## *5.2 Simplified Wing Box Problems*

Compared to the single laminate problem that we have studied so far, the wing box problem has four complications: the analysis is much more expensive; the chromosome is substantially longer; many laminates are handled simultaneously; the loads on each laminate vary as the laminates change in the course of the optimization.

Because of the increased cost of the problem, it is difficult to explore good strategies for the GA by the type of thorough testing conducted for the single laminate case. Instead, we consider

in depth several simplified problems. Reducing the number of laminates that vary independently reduces the size of the chromosome.

For the One-Laminate Problem, we employ a polynomial approximation of the internal loads as a mean of reducing the cost of an analysis. Then, a simplified wing box model, where the loads on each laminate are fixed and kept constant during the optimization, is implemented in order to isolate the effect of the load redistribution. Finally, we modify the single laminate problem from Chapter 4 by reducing the basic layer thickness to study the effects of chromosome length on the search.

### 5.2.1 Approximate Analysis of Internal Loads

In order to test a genetic algorithm on the wing box problem, we usually perform 50 runs of 20000 analyses each. With 0.065 CPU seconds per analysis, testing one implementation of the GA costs 65000 CPU seconds or 18 CPU hours in time spent exclusively analyzing designs. Practically, evaluating the performance of a particular GA on the wing box problem takes over 2 days. Therefore, minimizing the time needed to analyze a design is crucial to conducting extensive testing. The present Section describes a polynomial approximation of the internal loads in the structure that eliminates the computing time of the finite element analysis.

A cubic polynomial is used to approximate the relationships between three lamination parameters characterizing the laminate and each load $N_i^j$ ($i = x, y$ or $xy$), acting on each panel $j$ of the skin. The three in-plane lamination parameters of each skin laminate $k$ ([Miki91]) are defined as

$$V_{0A}^k = h, \quad V_{1A}^k = \int_{-h/2}^{h/2} cos\, 2\theta dz, \quad V_{3A}^k = \int_{-h/2}^{h/2} cos\, 4\theta dz, \tag{5.7}$$

where $h$ is the laminate thickness, $\theta$ is the fiber orientation in each ply, and $z$ is the through-the-thickness coordinate.

For the One-Laminate Wing Box problem, a design is fully specified by one laminate, i.e., only three in-plane lamination parameters. To simplify the coming expressions, we drop the $_A$ and $^k$ indices. Each of the three loads $N_x^j$, $N_y^j$ and $N_{xy}^j$ applied on each skin laminate $j = 1, 18$ is approximated by a complete cubic polynomial of the form

$$N_i^j(V_0, V_1, V_3) = C_0 + C_1V_0 + C_2V_1 + C_3V_3 + C_4V_0V_1 + C_5V_1V_3 + C_6V_0V_3 + C_7V_0^2 + C_8V_1^2 + C_9V_3^2 + C_{10}V_0V_1V_3 + C_{11}V_0^2V_1 + C_{12}V_0^2V_3 + C_{13}V_1^2V_0 + C_{14}V_1^2V_3 + C_{15}V_3^2V_0 + C_{16}V_3^2V_1 + C_{17}V_0^3 + C_{18}V_1^3 + C_{19}V_3^3. \quad (5.8)$$

The 20 coefficients were obtained by a least square fit based on internal loads from 1000 randomly generated structures. The 1000 structures were analyzed by finite element analysis, which provides 18×3 internal loads for each structure. 20 coefficients were calculated for each of the 54 internal loads. Equation (5.8) can be rewritten using nondimensional in-plane lamination parameters $V_i^* = V_i/h$ and coefficients $C_i^*$ having the dimension of lb/in . For example, the 20 coefficients of the polynomial approximating the spanwise load on the first laminate $N_x^1$ are given in Table 5.2.

**Table 5.2. Coefficients $C_i^*$ for approximating $N_x$ in the first panel (lbs/in).**

| | | | | | | |
|---|---|---|---|---|---|---|
| $C_0^*$ 4880 | $C_1^*$ 43120 | $C_2^*$ 58400 | $C_3^*$ 22080 | $C_4^*$ -165120 | $C_5^*$ 3968 | $C_6^*$ -60544 |
| $C_7^*$ -57792 | $C_8^*$ -51200 | $C_9^*$ 8576 | $C_{10}^*$ -7168 | $C_{11}^*$ 115712 | $C_{12}^*$ 43264 | $C_{13}^*$ 67584 |
| $C_{14}^*$ -660 | $C_{15}^*$ -11724 | $C_{16}^*$ 593 | $C_{17}^*$ 25466 | $C_{18}^*$ 24217 | $C_{19}^*$ 1679 | |

The coefficients show that all three of the in-plane lamination parameters, the thickness of the laminate included, are influential on the load distribution. The smallest three coefficients are $C_{14}^*$, $C_{16}^*$ and $C_{19}^*$, which weight respectively the terms $V_1^{*2}V_3^*$, $V_3^{*2}V_1^*$, and $V_3^{*3}$. Table 5.2 also confirms that, as the skin laminate changes, the loads in the panels of the skin are redistributed. The accuracy of the approximation is measured using the statistic $Q$. $Q$ is a relative norm of the error in one of the internal loads. It is defined as

$$Q = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(N_i^{ap} - N_i^{ex})^2}}{\bar{N}}, \quad (5.9)$$

where $N_i^{ap}$ is the approximated load on the $i$th randomly generated sample structure, $N_i^{ex}$ is the corresponding exact load, and $\bar{N}$ is

$$\bar{N} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(N_i^{ex})^2}. \tag{5.10}$$

In each of the randomly generated structures, the laminate thickness was uniformly distributed between four and 160 plies (by increments of four), and the orientation of the fibers in each ply was randomly chosen among ($0^o$, $90^o$, and $\pm 45^o$). Table 5.3 gives $Q$ for all the loads from the $n = 1000$ sample structures used in the least square fit. Table 5.4 also gives $Q$ for $n = 1000$ different randomly selected sample structures.

For $N_x$, $Q$ is close to 0.04 or 0.05, indicating a fairly accurate approximation of the $N_x$ internal loads. Larger relative errors occur with $N_y$, and sometimes with $N_{xy}$. It was determined by looking at the individual errors ($N_i^{ex} - N_i^{ap}$) that $N_y$ is badly predicted when its value is close to zero. The reason is that cases where $N_y$ is large pull the fitting response surface towards them. For optimization purpose however, occasional bad load predictions have little influence on the final design as long as they occur for small loads that are not related to the active constraints.

**Table 5.3. $Q$ measure of the quality of the approximated loads based on the structures used in the least square fit.**

| laminate | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $N_x$ | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 |
| $N_y$ | 0.07 | 0.33 | 0.14 | 0.12 | 0.20 | 0.07 |
| $N_{xy}$ | 0.04 | 0.02 | 0.03 | 0.03 | 0.03 | 0.02 |
| laminate | 7 | 8 | 9 | 10 | 11 | 12 |
| $N_x$ | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 |
| $N_y$ | 0.07 | 0.33 | 0.14 | 0.07 | 0.32 | 0.10 |
| $N_{xy}$ | 0.05 | 0.11 | 0.06 | 0.04 | 0.02 | 0.03 |
| laminate | 13 | 14 | 15 | 16 | 17 | 18 |
| $N_x$ | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 |
| $N_y$ | 0.10 | 0.19 | 0.07 | 0.07 | 0.32 | 0.10 |
| $N_{xy}$ | 0.03 | 0.03 | 0.02 | 0.05 | 0.11 | 0.06 |

By comparing Tables 5.3 and 5.4, it can be seen that the approximation is not really sensitive to whether the structures were used in the least square fit or not. Errors are very similar in both Tables. As expected, $N_x$ and $N_{xy}$ are somewhat better predicted in Table 5.3 than in Table 5.4,

**Table 5.4.** $Q$ **measure of the quality of the approximated loads for structures different from the ones used in the least square fit.**

| laminate | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $N_x$ | 0.05 | 0.05 | 0.05 | 0.04 | 0.04 | 0.04 |
| $N_y$ | 0.07 | 0.31 | 0.13 | 0.12 | 0.19 | 0.07 |
| $N_{xy}$ | 0.04 | 0.02 | 0.03 | 0.04 | 0.03 | 0.02 |
| laminate | 7 | 8 | 9 | 10 | 11 | 12 |
| $N_x$ | 0.04 | 0.04 | 0.04 | 0.05 | 0.05 | 0.05 |
| $N_y$ | 0.07 | 0.32 | 0.14 | 0.07 | 0.31 | 0.10 |
| $N_{xy}$ | 0.05 | 0.11 | 0.06 | 0.04 | 0.02 | 0.03 |
| laminate | 13 | 14 | 15 | 16 | 17 | 18 |
| $N_x$ | 0.05 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 |
| $N_y$ | 0.10 | 0.19 | 0.07 | 0.07 | 0.30 | 0.10 |
| $N_{xy}$ | 0.04 | 0.03 | 0.02 | 0.04 | 0.11 | 0.06 |

since the reference wing box beams of Table 5.3 were used when constructing the approximation. Surprisingly, $N_y$ is, on average, slightly better predicted in Table 5.4 than in Table 5.3.

The accuracy in the prediction of the largest loads, $N_x$, propagates to the buckling load factors that are calculated from closed form formula using the loads as input. The error is of the order of a few percent in the worst case. However the error on strains is larger, because strains are calculated in a different fashion in the finite element and in the approximate analyses. In the finite element analysis, strains are estimated at each node from the geometrical distortion of the elements. The largest strain in magnitude over the four elemental nodes is taken as the elemental strain. In the approximate analysis, the strains $\{\epsilon^j\} = [\epsilon_x^j \ \epsilon_y^j \ \gamma_{xy}^j]^T$ in each plate $j$ are obtained by solving $\{N^j\} = [A^j]\{\epsilon^j\}$, where $[A^j]$ is the in-plane extensional stiffness matrix of laminate $j$ from Classical Lamination Theory. That is, the strains in the approximate analysis are known in an average sense in each element and not at each node. In the worst cases, strains found from the approximate analysis underestimated the strains coming from the finite element analysis by as much as 50 percent. Note that the possibility of also approximating nodal loads and strains by polynomials in order to reduce approximation errors was ruled out because it would have involved eight times more polynomials (four for the nodal loads and four for the nodal strains).

## 5.2.2 Wing Box Without Load Redistribution

The effects of loads redistribution inside the structure can be suppressed by fixing the loads $\{N^j\} = [N_x^j \ N_y^j \ N_{xy}^j]^T$ on each laminate $j$ and keeping them constant during the optimization.

We selected the loads on each laminate as the ones occurring in the best known design obtained with loads redistribution. For example, the loads in the Six-Laminate Wing Box best known design are given in Table 5.5, and the corresponding design is given in the Results section of this Chapter. Like in the previous approximate analysis, the strains $\{\epsilon^j\}$ in each plate $j$ are obtained by solving $\{N^j\} = [A^j]\{\epsilon^j\}$. The critical buckling load factors are obtained from the analytical formula given in Section 5.1.3. The "$L$-Laminate Wing Box" problem without load redistribution is equivalent to optimizing the stacking sequences of $L$ laminates simultaneously but independently.

**Table 5.5. Panel Loadings (lb/in) † for the best known Six-Laminate Wing Box design.**

| laminate | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $N_x$ | 15207 | 7187 | 2430 | 2321 | 104.40 | 7628 |
| $N_y$ | 2505 | 62 | 419 | 243 | 632 | 2401 |
| $N_{xy}$ | 976 | 1624 | 1614 | 1250 | 1148 | 707 |
| laminate | 7 | 8 | 9 | 10 | 11 | 12 |
| $N_x$ | 13308 | 7260 | 2346 | -14566 | -6957 | -2197 |
| $N_y$ | 2034 | 265 | 245 | -191 | -748 | 110 |
| $N_{xy}$ | 1138 | 394 | 478 | 443 | 1384 | 1573 |
| laminate | 13 | 14 | 15 | 16 | 17 | 18 |
| $N_x$ | -2323 | -7060 | -13121 | -12626 | -6970 | -2341 |
| $N_y$ | 26 | -969 | -193 | -96 | -755 | 114 |
| $N_{xy}$ | 1277 | 1113 | 444 | 437 | 669 | 492 |

† compressive loads are positive, tensile loads are negative.

### 5.2.3 Single Laminate Problem

To study the effect of increased chromosome length, we modified the single laminate problem treated in Chapter 4 by allowing the basic layer thickness $t$ to be smaller than 0.005 in.. The laminate we consider is similar to the one used in Chapter 4. It is subjected to the loads $N_x$ = 9,800. lb/in and $N_y$ = 4,900. lb/in . For $t$ = 0.005, 0.0025, 0.001, 0.0006 in., the lengths of the optimal chromosomes (that is accounting for the full stacks only) are 12, 24, 60, and 100, respectively.

An alternative way of changing the chromosome length would have been to increase the loads on the laminate, which would lead to increased total thickness. However, with the strains

changing linearly with one over the thickness and the buckling load changing as one over the cube of the thickness, increased loads would lead to laminates where buckling was not critical. Such laminates are easy to optimize because only the total number of plies of each fiber orientation, not the stacking sequence, influence the strength. There are two drawbacks associated with changing the ply thickness $t$: the problem is no longer realistic because the layer thickness is too small, and, as the layer thickness is reduced, the number of designs with critical load factors within a certain distance from the optimal critical load factor increases, making it easier to find a practical optimum design. This phenomenon however, in contrast to the effect of changing the loads, can be corrected by changing the definition of a practical optimum: to compensate for the increasing number of practical optima when ply thickness decreases, $s$ needs to be lowered.

## *5.3 Genetic Algorithm Implementation*

The structure of the genetic algorithm used on the wing box problem is similar to the ones of the GAs used in the previous chapters. It is an elitist generational GA, i.e., all individuals but the best are replaced by new designs during the change of generation. Once again, the fitness which is the probability of an individual to be selected for reproduction is defined as the normalized rank of the individual in the population.

### 5.3.1 Encoding of a Wing Box

Crossover, mutation, stack swap operators are identical to the ones presented in Chapter 4 for the varying thickness genetic algorithm. The only changes regard the coding of the wing box and the objective function calculation.

The One-, Two-, Six- and 18-Laminate Wing Box problems have respectively one, two, six and 18 independent laminates to be optimized. The chromosome describing the whole wing box is the assembly of the subchromosomes corresponding to each independent laminate. The way a single laminate is coded as a chromosome was explained in Chapter 4 and is kept here. If we consider, for example, a Two-Laminate Wing Box design with $[(\pm 45^o)_4/90_2^o/\pm 45^o/0_2]_s$ as upper skin laminate and $[\pm 45^o/90_2^o/0_2^o/90_4^o/0_2]_s$ as lower skin laminate, the associated chromosome is

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $E_2$ | $E_2$ | $\pm 45$ | $\pm 45$ | $\pm 45$ | $\pm 45$ | $90_2$ | $\pm 45$ | $0_2$ |
| $E_2$ | $E_2$ | $E_2$ | $\pm 45$ | $90_2$ | $0_2$ | $90_2$ | $90_2$ | $0_2$ |

In the example above, the maximum plate thickness is 36 plies. The genetic operators are successively applied to the subchromosome of each independent laminate. For example, a crossover between Two-Laminate Wing Box designs can be

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Parent 1** | $E_2$ | $E_2$ | $\overset{\nabla}{\underset{\Delta}{}} \pm 45$ | $\pm 45$ | $\pm 45$ | $\pm 45$ | $90_2$ | $\pm 45$ | $0_2$ |
| | $E_2$ | $E_2$ | $E_2$ | $\pm 45$ | $90_2$ | $0_2$ | $90_2$ | $\overset{\nabla}{\underset{\Delta}{}} 90_2$ | $0_2$ |
| **Parent 2** | $E_2$ | $90_2$ | $\overset{\nabla}{\underset{\Delta}{}} \pm 45$ | $90$ | $\pm 45$ | $0_2$ | $0_2$ | $\pm 45$ | $\pm 45$ |
| | $E_2$ | $E_2$ | $E_2$ | $E_2$ | $\pm 45$ | $90_2$ | $\pm 45$ | $\overset{\nabla}{\underset{\Delta}{}} 0_2$ | $90_2$ |
| **Child** | $E_2$ | $E_2$ | $\overset{\nabla}{\underset{\Delta}{}} \pm 45$ | $90$ | $\pm 45$ | $0_2$ | $0_2$ | $\pm 45$ | $\pm 45$ |
| | $E_2$ | $E_2$ | $E_2$ | $\pm 45$ | $90_2$ | $0_2$ | $90_2$ | $\overset{\nabla}{\underset{\Delta}{}} 0_2$ | $90_2$ |

With $L$ laminates, this crossover can be seen as a $(2L-1)$-point crossover, where $L-1$ break points always fall on the boundaries between laminates, and the rest is distributed such that one break point occurs within each subchromosome. Such a crossover is biased towards separating the left from the right part of each subchromosome. In Chapter 4, this one-point crossover bias was tested against the biases induced by two-point and uniform crossovers and proved to help the GA on single laminate problems. The crossover used on the wing box problem has the same kind of bias, but it affects many laminates at the same time. In the Two-Laminate problem for example, the outside of the upper skin laminate will usually be separated from the inside of the lower and upper skin laminates during reproduction. Nevertheless, this crossover was chosen because it is the most direct extension of the thoroughly tested single laminate crossover. Other crossovers would have other biases, and the repercussions on the search efficiency are difficult to predict. The one-point crossover performed satisfactorily well on the single laminate problem so, by selecting the aforementioned crossover for the wing box, it is hoped that the results derived on the single laminate problems (type and optimal application rates of the genetic operators) will apply.

Like crossover, we used the decoupled mutation and the one-stack swap that have been implemented in Chapter 4 for the variable thickness GA. To simplify the expressions, one-stack swap will be called stack swap in this chapter. All genetic operators are applied independently to each subchromosome.

### 5.3.2 Objective Function Formulation

The objective function and the associated penalty functions are basically identical to the ones defined for the single laminate problem of Chapter 4, Equation (4.7). We want to minimize objective function $\Phi$ defined as

$$\begin{aligned} &if\ \lambda_{cr} \geq (1-\delta), \quad \Phi = P_c^{n_c}\ \{N_t + e\ [(1-\delta) - \lambda_{cr}]\}, \\ &if\ \lambda_{cr} < (1-\delta), \quad \Phi = P_c^{n_c}\ \{\frac{N_t}{\lambda_{cr}^{P_l}}\} + S. \end{aligned} \tag{5.11}$$

One difference with Equation (4.7) is that $N$ is replaced by $N_t$, the sum of the plies over the independent laminates. Also $\lambda_{cr}$, the critical failure load factor, accommodates not only normal buckling and strength like it was done on the single laminate problems, but also shear and combined buckling,

$$\lambda_{cr} = \min_j(\lambda_{c\epsilon}^j, \lambda_{cs}^j, \lambda_{cc}^j), \qquad j = 1, 18. \tag{5.12}$$

$\delta$ is a tolerance parameter in the definition of feasibility, $\varepsilon$ is a bonus factor for safety margin, $P_c$ is the penalty parameter for contiguity constraint, $n_c$ is the total number of violations of the ply contiguity rule, and $P_l$ is the penalty parameter for failure.

## 5.4 Results

The results presented here concentrate on both the effects of the population size $m$ and the stack swap probability $pp$. The population size is a central parameter influencing both the exploration and the exploitation trends of a GA. Stack swap is an operator of crucial importance to the single laminate problem whose relevance needs to be checked on other composite structures. The performance of all six combinations of $m$ and $pp$ for $m \in (5, 10, 50)$ and $pp \in (0, 1)$ are compared. Unless otherwise specified, the other genetic parameters are kept at their optimal values presented in Chapter 4, i.e., the probability of crossover $pc$ is 1, the probabilities of adding and deleting two stacks per laminate $pa$ and $pd$ are 0.05, the probability of changing the fiber orientation $pf$ is 0.01. Parameters related to the objective function definition are $e = 6$, $\delta = 0.005$, $P_l = 1$, $P_c = \sqrt{\frac{10}{9}}$ and $S = 4$.

Like in Chapters 4 and 5, the performance of a GA will be estimated in terms of the reliability $r_1(n)$, which is the probability a GA has of finding a practical optimum after one search of $n$ analyses. However, because we compare here the performance of GAs on problems of varying difficulties, we will keep flexible the definition of what a practical optimum is: a practical optimum is a feasible optimum weight design whose critical load factor is within $s$ percent of the optimal critical load factor. The choice $s = \infty$ means that a practical optimum is an optimum weight feasible design ($\lambda_{cr}$ has to be larger than $1 - \delta$), without any other restrictions on the critical load factor. Each value of $r_1(n)$ is an average of the performance of the GA over 50 independent runs, except for the 18-Laminate Wing Box problem where only 25 tests are performed, and the single laminate problem where 100 tests could be afforded.

### 5.4.1 Wing Box Problems

**One-Laminate Wing Box Problem**

The best One-Laminate Wing Box design found by a genetic algorithm using the exact analysis is

$$[\pm 45^o_{20}/0^o_2/(90^o_2/0^o_4)_4/90^o_2/0^o_2]_s,$$

for a total of 18 $\times$ 140 = 2520 plies in the wing skin. The most critical constraint is combined buckling of the first laminate, $\lambda_{cr} = \lambda_{cc}^1 = 1.047604$. A common feature of good One-Laminate Wing Box designs is to have combined buckling of the first laminate as most critical constraint, which is an intuitive result since the first panel is the one at the root of the wing, on the side where the largest loads are applied. It is clamped on one side, and endures the largest compressive $N_x$ load resulting from the tip loading. For the sake of comparison, an equivalent problem that yet does not account for the ply contiguity constraint, was solved using a continuous sequential quadratic programming package. In the continuous formulation of the stacking sequence problem, the fiber orientations are fixed beforehand to $[(\pm 45)_{x_1^j}/(90_2)_{x_2^j}/(0_2)_{x_3^j}/(\pm 45)_{x_4^j}/(90_2)_{x_5^j}/(0_2)_{x_6^j}]_s$ and the design variables are the ply thicknesses, $x_i$, $i = 1, 6$, in each laminate $j = 1, L$. In order to obtain a laminate that can be manufactured, the continuous design is rounded so that ply thicknesses are multiples of the basic ply thickness $t$. The interested reader should refer to Appendix C for more details about the continuous optimization procedure and subsequent rounding of the designs. After rounding, the optimal continuous One-Laminate Wing Box design is

$$[\pm 45_{21}^{\circ}/90_4^{\circ}/0_{24}^{\circ}]_s,$$

and its most critical constraint is $\lambda_{cr} = \lambda_{cc}^1 = 1.046692$. The continuous optimum design has the same weight as the genetic best design, but it does not satisfy the ply contiguity constraint since it was not considered during design.

Figure 5.6 shows the reliability of the search when $s = 0.1$ for various values of the population size $m$ and the probability of stack swap $pp$. It is seen that, for the best setting of the GA ($m = 5$, $pp = 1$), 80% reliability is achieved in less than 2000 analyses. The settings $m = 5$, $pp = 1$ and $m = 10$, $pp = 0$ have very similar reliabilities for the first 3000 analyses. $m = 50$, $pp = 1$ yields to an unreliable genetic search in that case. Table 5.5 presents the results of the tests performed on the One-Laminate Wing Box problem in a condensed form: it describes the optimum setting of the GA (population size $m$ / probability of stack swap $pp$) as a function of the number of analyses, $n$, and the definition of what a practical optimum is, $s$. Note that in all cases, for $n < 2000$, a population size $m$ equal to 5 is the best setting. However, the first setting to achieve 100 percent reliability for $s = 0.1$ (a stringent definition of what a practical optimum is) is $m = 10$, $pp = 0$, at 4000 analyses. This corresponds to the well known tradeoff

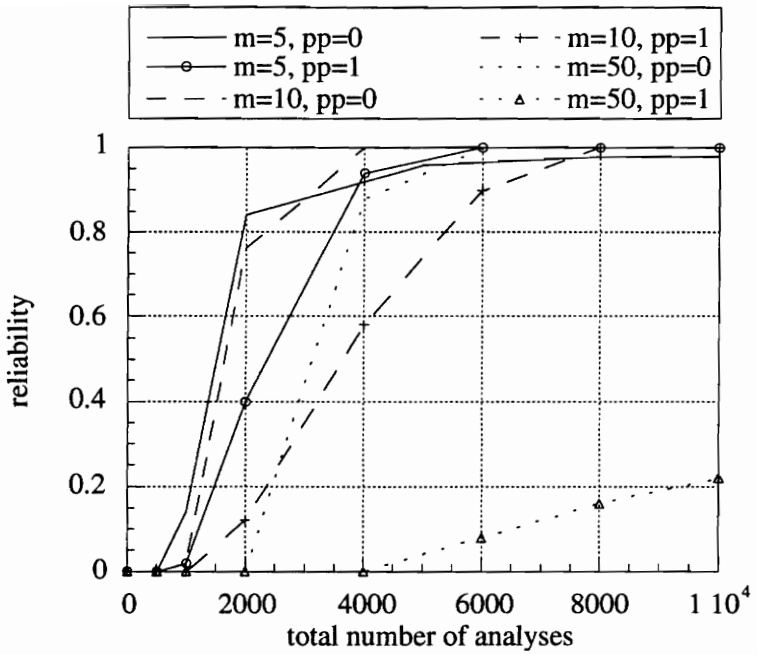



**Figure 5.6. Reliability ($s$ = 0.1%) versus Number of analyses, One-Laminate Wing Box problems, 50 tests.**

between exploration and exploitation in GAs ([Holland75], [DeJong75], [Goldberg89], Chapter 2), according to which larger population sizes converge more slowly but probabilistically to a better quality of optimum. For $s$ = 1 and $\infty$ (slack definitions of practical optima), 100 percent stack swap improves convergence in the first generations. Table 5.5 is also discussed in Section 5.4.3 that deals with the effects of chromosome length.

### Two-Laminate Wing Box Problem

The best genetic Two-Laminate Wing Box design is

$$[\pm 45^{\circ}_{16}/0^{\circ}_{4}/(90^{\circ}_{2}/0^{\circ}_{4})_{5}/90^{\circ}_{2}/0^{\circ}_{2}]_{s}, \quad \text{for the upper skin panels,}$$
$$[90^{\circ}_{2}/\pm 45^{\circ}/90^{\circ}_{4}/0^{\circ}_{2}/90^{\circ}_{4}/(0^{\circ}_{4}/90^{\circ}_{2})_{2}]_{s}, \quad \text{for the lower skin panels,}$$

for a total number of plies in the skin equal to 9 $\times$ (140 + 52) = 1728. The most critical constraint is shear buckling of panel 12, $\lambda_{cr} = \lambda^{12}_{cs} = 1.035174$. The corresponding optimum continuous design is

$$[\pm 45^{\circ}_{19}/90^{\circ}_{2}/0^{\circ}_{30}]_{s}, \quad \text{for the upper skin panels,}$$
$$[90^{\circ}_{12}/0^{\circ}_{10}/90^{\circ}_{6}]_{s}, \quad \text{for the lower skin panels.}$$

**Table 5.6. (optimum population size / optimum probability of stack swap), One-Laminate Wing Box Problem.**

| $n$ / $s$ | 500 | 1000 | 2000 | 4000 | 6000 | 8000 |
|---|---|---|---|---|---|---|
| 0.1 | – | (5/0) | (5/0) | (10/0)* | * | * (5/1)* (50/0)* |
| 0.2 | – | (5/0) | (5/0) | (5/0)*(5/1)* (10/0)* | ***(10/1)* (50/0)* | ***** |
| 1.0 | (5/0) (5/1) | (5/0)* | *(5/1)* (10/0)* | ***(10/1)* (50/0)* | ***** | ***** |
| $\infty$ | (5/1) | (5/0)*(5/1)* (10/1)* | ***(10/0)* (50/1)* | **** (50/0)* | ***** | ***** |

– means that no practical optimum was found
$(A/B)$* means that the GA with population size $m$ = A and probability of stack swap $pp$ = B just reached 100% reliability, the * alone is used in subsequent generations as a reminder, but $(A/B)$ is omitted.

The continous design has 9 × (140+56) = 1764 plies, and thus is 2% heavier than the genetic design. In addition to that, it does not satisfy the four ply contiguity constraint, and has as most critical load factor $\lambda_{cr} = \lambda_{cs}^{12} = 1.034232$.

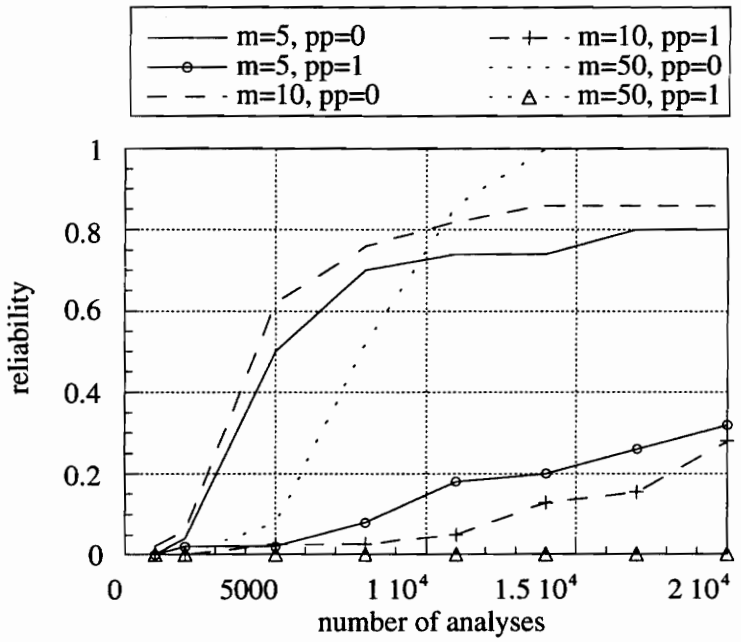



**Figure 5.7. Reliability ($s = \infty$) versus Number of analyses, Two-Laminate Wing Box problems, 50 tests.**

**Table 5.7. (optimum population size / optimum probability of stack swap), Two-Laminate Wing Box Problem.**

| $n$ / $s$ | 1000 | 2000 | 5000 | 8000 | 14000 | 20000 |
|---|---|---|---|---|---|---|
| 0.1 | – | – | – | – | (5/0)(50/0) | (5/0)(50/0) |
| 1.0 | – | – | – | (50/0) | (50/0) | (50/0) |
| 2.0 | – | (10/0) | (50/0) | (50/0) | (50/0) | (50/0) |
| $\infty$ | (10/0) | (10/0) | (10/0) | (10/0) | (50/0)* | * |

– means that no practical optimum was found
$(A/B)^*$ means that the GA with population size $m$ = A and probability of stack swap $pp$ = B just reached 100% reliability, the * alone is used in subsequent generations as a reminder, but $(A/B)$ is omitted.

Figure 5.7 gives the reliability ($s = \infty$) versus the number of analyses for different population sizes $m$ and probabilities of stack swap $pp$. The 80% reliability limit is first reached after 10000 analyses when $m = 10$ and $pp = 0$. The setting $m = 50$ and $pp = 0$ achieves 100% reliability first after 14500 analyses. Table 5.6 shows what is the optimal $(m/pp)$ setting for the GA at different times of the search $n$, for varying definitions of practical optima $s$. There are no conditions under which 100 percent stack swap is beneficial to the GA performance. For $s = 2$ and $s = \infty$, a population size $m = 10$ is better early in the search, but $m = 50$ outperforms it later on. $m = 10$ seems to be the best choice early in the search for large $s$ ( = 2.0, $\infty$), versus $m = 50$ for smaller $s$ (0.1, 1.0), which, once again, confirms the higher quality of designs obtained using a large population size when long searches are affordable. Section 5.4.3 addresses this issue under a broader spectrum.

### Six-Laminate Wing Box Problem

The best genetic Six-Laminate Wing Box design is

| stacking sequence | panels |
|---|---|
| $[\pm 45^\circ_{11}/90^\circ_2/0^\circ_2/90^\circ_2/\pm 45^\circ/0^\circ_2/\pm 45^\circ/90^\circ_2/90^\circ_2/0^\circ_4/90^\circ_4/0^\circ_2$ $90^\circ_2/0^\circ_2/90^\circ_4/0^\circ_4/90^\circ_2/0^\circ_2/90^\circ_4/0^\circ_2]_s$, | 1,6 and 7 |
| $[\pm 45^\circ_8/90^\circ_2/\pm 45/90^\circ_4/\pm 45^\circ_3/90^\circ_2/0^\circ_2/\pm 45/(90^\circ_4/0^\circ_2)_3]_s$, | 2,5 and 8 |
| $[\pm 45^\circ_{20}]_s$, | 3,4 and 9 |
| $[90^\circ_2/\pm 45/(0^\circ_4/90^\circ_2)_2/0^\circ_2/90^\circ_2]_s$, | 10,15 and 16 |
| $[90^\circ_4/0^\circ_2/90^\circ_2/\pm 45^\circ_2/90^\circ_2/(90^\circ_2/\pm 45/90^\circ_2)_2]_s$, | 11,14 and 17 |
| $[\pm 45^\circ_2/90^\circ_4/\pm 45^\circ_{10}]_s$, | 12,13 and 18 |

The total number of plies in the skin is 3 × (140+108+80+40+52+56) = 1428, and the most critical constraint is $\lambda_{cr} = \lambda^{12}_{cs} = 1.057810$. No continuous optimization was performed on this

case. Figure 5.8 shows the probability of finding an optimal weight feasible design (reliability with $s = \infty$) as a function of the cost of the search, for different settings of the GA. Finding optimum weight designs becomes difficult in that case since, at best, 28% probability of finding optimum weight designs can be achieved after 20000 analyses if $m = 5$, $pp = 0$. All the other settings of the GA, except $m = 10$, $pp = 0$, never find any optimum weight design. Table 5.7 shows the optimal combination of $m$ and $pp$ as a function of $n$ and $s$. Notice that as the difficulty increases (lower $s$), the finding of practical optima takes more analyses. Independently of the number of analyses and the criterion used to define practical optima, the pair ($m = 5$, $pp = 0$) performs the best. It is the parameter setting with the less randomness (since no stack swap is performed) and the highest pressure for selection that was tested.

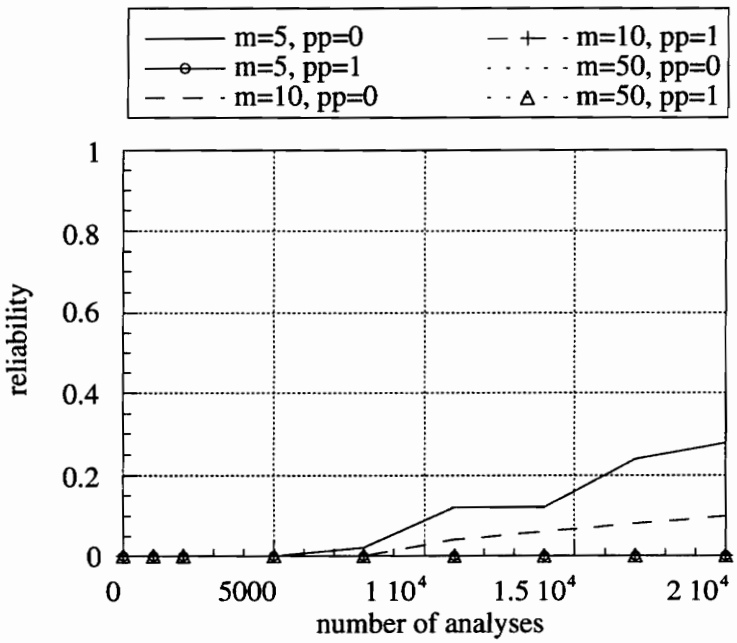



**Figure 5.8. Reliability ($s = \infty$) versus Number of analyses, Six-Laminate Wing Box problems, 50 tests.**

**Table 5.8. (optimum population size / optimum probability of stack swap), Six-Laminate Wing Box Problem.**

| $n$ / $s$ | 1000 | 5000 | 8000 | 11000 | 17000 | 20000 |
|---|---|---|---|---|---|---|
| 0.1 | – | – | – | – | – | (5/0) |
| 1.5 | – | – | – | – | (5/0) | (5/0) |
| 2.0 | – | – | – | (5/0) | (5/0) | (5/0) |
| $\infty$ | – | – | (5/0) | (5/0) | (5/0) | (5/0) |

– means that no practical optimum was found.

### 18-Laminate Wing Box Problem

The continuous optimization and rounding yielded the 18-Laminate Wing Box design

| stacking sequence | panel | No. of plies |
|---|---|---|
| $[\pm 45^{\circ}_{14}/90^{\circ}_{2}/0^{\circ}_{54}]_s$, | 1 | 168 |
| $[\pm 45^{\circ}_{23}/90^{\circ}_{10}/0^{\circ}_{10}]_s$, | 2 | 132 |
| $[\pm 45^{\circ}_{13}/90^{\circ}_{6}/0^{\circ}_{14}]_s$, | 3 | 92 |
| $[\pm 45^{\circ}_{7}/90^{\circ}_{14}/0^{\circ}_{4}]_s$, | 4 | 64 |
| $[\pm 45^{\circ}_{17}/90^{\circ}_{18}]_s$, | 5 | 104 |
| $[\pm 45^{\circ}_{14}/90^{\circ}_{28}/0^{\circ}_{2}]_s$, | 6 | 116 |
| $[\pm 45^{\circ}_{5}/90^{\circ}_{44}]_s$, | 7 | 108 |
| $[\pm 45^{\circ}_{10}/90^{\circ}_{24}]_s$, | 8 | 88 |
| $[\pm 45^{\circ}_{6}/90^{\circ}_{14}]_s$, | 9 | 52 |
| $[0^{\circ}_{20}]_s$, | 10 | 40 |
| $[\pm 45^{\circ}/90^{\circ}_{4}/0^{\circ}_{22}]_s$, | 11 | 56 |
| $[\pm 45^{\circ}/90^{\circ}_{30}/0^{\circ}_{8}]_s$, | 12 | 80 |
| $[90^{\circ}_{8}/0^{\circ}_{4}]_s$, | 13 | 24 |
| $[\pm 45^{\circ}_{7}/90^{\circ}_{4}/0^{\circ}_{4}]_s$, | 14 | 44 |
| $[90^{\circ}_{2}/0^{\circ}_{8}]_s$, | 15 | 20 |
| $[90^{\circ}_{10}/0^{\circ}_{4}]_s$, | 16 | 28 |
| $[\pm 45^{\circ}_{3}/0^{\circ}_{6}]_s$, | 17 | 24 |
| $[\pm 45^{\circ}/90^{\circ}_{4}/0^{\circ}_{2}]_s$, | 18 | 16 |

For this design, the total number of plies in the skin is 1256, and the most critical constraint is $\lambda^1_{cc} = 1.029953$.

Optimizing the 18-Laminate Wing Box problem by GAs is costly since the chromosome is very large: there are $18 \times 40 = 720$ genes with four possible allele values. This represents a total design space size of $4^{720}$. Larger chromosomes not only make the cost in terms of number of analyses higher, but also the computer time necessary to process chromosomes increases. For this reason, testing on the 18-Laminate Wing Box problem was limited to 25 runs of 30000 analyses for the settings: ($m$ = 5, $pp$ = 0), ($m$ = 5, $pp$ = 0.055), and ($m$ = 10, $pp$ = 0). The

previous notion of reliability cannot be used on the 18-Laminate Wing Box problem, because the GA never converges to the same weight design. This indicates the difficulties met by the search. Instead, Table 5.8 gives the number of feasible designs found, the minimum, the maximum, and the average of the weights of the feasible designs found as a function of the number of analyses.

**Table 5.9. Performance of the GA on the 18-Laminate Wing Box Problem, 25 runs.**

| | $n$ | 5000 | 8000 | 13000 | 18000 | 23000 | 30000 |
|---|---|---|---|---|---|---|---|
| $m = 5$ | *nfeas.* | 2 | 7 | 10 | 14 | 18 | 19 |
| $pp = 0$ | *wmin.* | 1700 | 1576 | 1508 | 1448 | 1400 | 1396 |
| $pf = 0.01$ | *wmax.* | 1708 | 1764 | 1708 | 1708 | 1708 | 1708 |
| | *wav.* | 1704 | 1660 | 1604 | 1568 | 1532 | 1508 |
| $m = 10$ | *nfeas.* | 0 | 1 | 5 | 13 | 21 | 23 |
| $pp = 0$ | *wmin.* | – | 1764 | 1520 | 1472 | 1456 | 1444 |
| $pf = 0.01$ | *wmax.* | – | 1764 | 1852 | 1788 | 1756 | 1696 |
| | *wav.* | – | 1764 | 1692 | 1600 | 1588 | 1552 |
| $m = 5$ | *nfeas.* | 1 | 8 | 13 | 15 | 19 | 19 |
| $pp = 0.055$ | *wmin.* | 1676 | 1520 | 1400 | 1352 | 1340 | 1316 |
| $pf = 0.005$ | *wmax.* | 1676 | 1768 | 1628 | 1580 | 1544 | 1492 |
| | *wav.* | 1676 | 1612 | 1532 | 1488 | 1456 | 1420 |

*nfeas.* is the number of feasible designs, *wmin.* is the number of plies of the lightest feasible design found, *wmax.* is the number of plies of the heaviest feasible design found, *wav.* is the average number of plies of feasible designs.
– means that no feasible design was found.

The comparison of the performance between $m = 5$ and $m = 10$ tells us, like for the Six-Laminate Wing Box problem, that there is advantage at using a small population size, since it finds feasible designs more rapidly and keeps improving them at a faster pace than for a larger population. The best 18-Laminate Wing Box design, which is also the best wing box design ever found by the GA, has 1316 plies total (against 1428 plies for the best Six-Laminate Wing Box design) and was found by a GA tuned with $m = 5$, $pp = 0.055$, $pf = 0.005$ (cf. Section 5.4.5 for the reasons of this tuning). For a more readable description of the design, we will denote $0^\circ_2$, $\pm 45^\circ$ and $90^\circ_2$ by 1, 2, and 3 respectively. The optimum 18 laminate wing box design is

| stacking sequence | panel | No. of Plies |
|---|---|---|
| $[22222131121222122112312211231331131331]_s$, | 1 | 156 |
| $[22223221211322133212122112313332]_s$, | 2 | 124 |
| $[11331213213321233232223]_s$, | 3 | 92 |
| $[22322321211331211231]_s$, | 4 | 80 |
| $[22221222212232131211213112112]_s$, | 5 | 116 |
| $[22212121221222222121321122131211232]_s$, | 6 | 140 |
| $[33223322322213311232231212 32]_s$, | 7 | 112 |
| $[223311332223223122232]_s$, | 8 | 84 |
| $[23222222312222]_s$, | 9 | 56 |
| $[13113112113131]_s$, | 10 | 56 |
| $[331321232232121222223]_s$, | 11 | 84 |
| $[323122222121]_s$, | 12 | 48 |
| $[1312331]_s$, | 13 | 28 |
| $[31313]_s$, | 14 | 20 |
| $[2313113]_s$, | 15 | 28 |
| $[3323323222]_s$, | 16 | 40 |
| $[3131]_s$, | 17 | 16 |
| $[222331323]_s$, | 18 | 36 |

It can be argued that the ply contiguity constraint takes on overwhelming importance as the number of independent laminates $L$ increases because the penalty is based on the total number of violations, which often will be large when $L$ is large. Excessive penalty is known to impair the efficiency of the genetic search (cf. Chapter 2). For this reason, we performed a few more runs where the penalty parameter associated to ply contiguity constraint was reduced from $P_c = \sqrt{\frac{10}{9}}$ to $P_c = 1.02$. Apart for $P_c$, the tuning of the GA was $pp = 0.055$, $pf = 0.005$, $m = 5$, and the other genetic parameters kept their standard values (cf. beginning of Section 5.4). Four independent runs of 150000 analyses did not yield any better design than the one already known. Since it was noticed that the GA stopped improving at about 60000 analyses, we increased the population size to 50 individuals and repeated the experiment. The results were not better than before either. This seems to indicate that the difficulties experienced by the GA on the 18-Laminate Wing Box problem are not only related to fine tuning. We conjecture that the algorithm probably needs more suited exploration mechanisms, like scaling (cf. Chapter 4) or intralaminate stack swap ([Nagendra94]), in order to deal more efficiently with very large design spaces. Multilevel optimization, where a complex problem is reduced to a series of simpler problems, might also be an appropriate way of handling the 18-Laminate Wing Box problem. The simplified wing box

problem where the internal loads are kept fixed during the optimization (cf. Section 5.2.2) is one step in that direction.

### 5.4.2 Approximate One-Laminate Wing Box Problem

The best known design for the One-Laminate Wing Box with approximate loads problem is

$$[\pm 45^{\circ}_{16}/90^{\circ}_{2}/(0^{\circ}_{2}/90^{\circ}_{2}/0^{\circ}_{2})_{6}]_{s}.$$

This design has the same weight as the optimum One-Laminate Wing Box design, i.e., the wing skin panels is made of 2520 plies. The approximate optimum design resembles the exact optimum design: the 32 outermost and 30 innermost stacks of two plies of the optimum designs of the exact and approximate problems are identical, only 8 stacks out of 70 are different. Like the exact optimum, the optimum design of the approximate wing box model fails because of combined buckling at panel 1, $\lambda_{cr} = \lambda^{1}_{cc} = 1.032240$ (versus 1.047604 with the exact analysis).

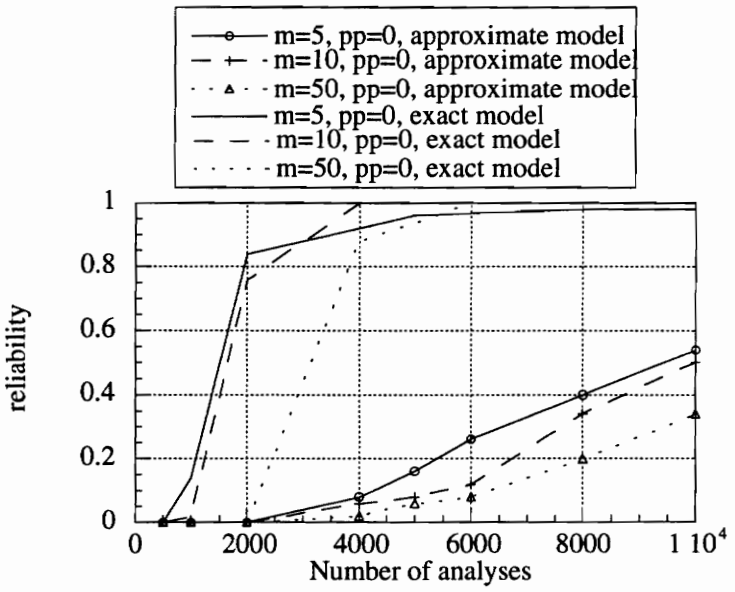



**Figure 5.9. Comparison of reliability curves for the exact and approximate One-Laminate Wing Box problems, $s$ = 0.1%, 50 tests.**

Figure 5.9 shows the reliability of the GA as a function of the number of analyses for population sizes equal to 5, 10, 50, using the exact and the approximate model. Practical optima have a critical load factor within $s = 0.1$ percent of the critical load factor of the optimum. The reliability achieved by the GA on the approximate problem is smaller than the reliability achieved on the exact problem, all parameters of the search being equal. Even though the approximate model reproduces well the exact model from a structural point of view, it represents a higher level of difficulty to the GA. Because it is derived through least square fit from a random sample of structures, the approximate model tends to overestimate the loads for optimal weight designs: most of the optimal weight designs fail because the loads they endure in critical regions of the wing box (typically for the One-Laminate Wing Box problem, $N_x$ in panel 1) are too large. Around the optimal weight region of the design space, the load response surface is more influenced by the numerous infeasible optimal weight designs with high loads than by the few feasible optimal weight designs that have smaller loads. As a result, the load response surface will be pulled towards high loads in the optimal region of the design space, so that the approximate model is expected not to have as many feasible optimum weight designs as the exact model. The approximate problem ends up being more difficult.

It is possible to compensate for the smaller density of feasible optimum designs in the approximate problem by widening the definition of practical optima for the approximate problem only. For example, the reliability of the GA on the approximate wing box when $s = 1.5$ percent and the reliability on the exact problem when $s = 0.1$ are shown on Figure 5.10. It can be seen that the reliability curves of the exact and approximate problems compare better when the additional difficulty of the approximate model has been compensated for by increasing $s$.

If a slack definition of a practical optimum is used, testing of the GA can be performed on the polynomial approximation of the structure. Figure 5.10 shows that the influence of the population size on the reliability of the search is well captured by the approximate model.

A pertinent improvement to the approximation scheme is studied in [Harrison94]: exact and approximate analyses are both used during the genetic search. The response surface fit is updated in the course of the search from the last exact analyses. The idea is to have a density of sampled structure increasing in the optimal region of the design space as the search progresses, therefore enhancing the fit in that region. Such a strategy, if applied to the wing box, would probably

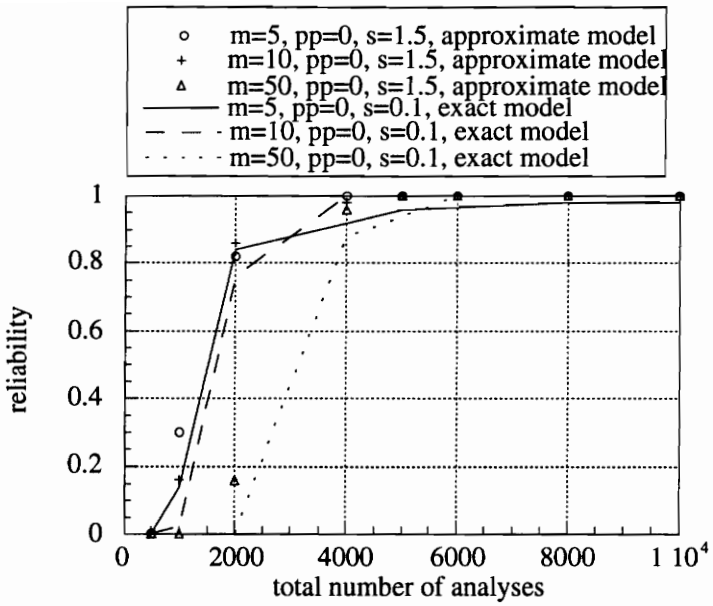



**Figure 5.10. Comparison of reliability curves for the exact ($s$ = 0.1%) and approximate ($s$ = 1.5%) One-Laminate Wing Box problems, 50 tests.**

reduce the tendency of the approximate analysis to overestimate the loads acting on critical parts of the wing box.

### 5.4.3 A Dimensional Study of the Effect of Chromosome Length

One of the major changes with respect to the single laminate problems induced by the wing box is an increase in the length of the chromosomes. This is equivalent to having a larger design space. In Chapter 3, a restricted study on the effect of increasing the chromosome length on the performance of the GA was made for single fixed thickness laminates. It was seen that, as the design space gets larger, the GA seems to be exploring smaller relative fractions of the design space before locating practical optima. Here we enter more into the details of the dynamics of GAs. A survey of the dimensional models in GA theory that relate population size, length of chromosome, and time to convergence is given in Appendix A.

We describe the results of experiments made on a the modified single laminate problem where the length of the chromosome is changed by changing the basic ply thickness $t$. The purpose of this example case, already described in Section 5.2.3, is to isolate the effect of the chromosome

length. Results presented in Table 5.9 show the relation between chromosome length, time to convergence, optimum population size and probability of stack swap. Table 5.9 shows the set of parameters (optimum population size / probability of stack swap) having the highest reliability at a given number of analyses, among the six parameter combinations (5/0), (5/1), (10/0), (10/1), (50/0), (50/1). Two definitions of practical optimum are taken, $s = \infty$ and $s = 0.5$, and four basic ply thicknesses are used. Recall from Section 5.2.3 that different optimum lengths of chromosome $l$ = 12, 24, 60, 100 correspond to the different basic ply thicknesses $t$ = 0.005, 0.0025, 0.001, 0.0006 in., respectively.

**Table 5.10. (optimum population size / optimum probability of stack swap), single laminate problem.**

| $n$ | 1000 | 2000 | 5000 | 10000 | 20000 | 30000 |
|---|---|---|---|---|---|---|
| $t$ = .005in<br>$s$ = 0.5 | (5/1) | (5/1) | (10/1) | (10/1) | (10/1) | (10/1)*<br>(50/1)* |
| $t$ = .005in<br>$s = \infty$ | (5/1) | (5/1) | (50/0) | (50/0)* | (5/0)*(5/1)*<br>(10/0)*(10/1)*<br>* (50/1)* | ***<br>*** |
| $t$ = .0025in<br>$s$ = 0.5 | (5/1) | (5/1) | (5/1) | (5/1)<br>(50/1) | (50/1) | (50/1) |
| $t$ = .0025in<br>$s = \infty$ | (5/1) | (5/1) | (5/1)*<br>(10/1)* | ** (5/0)*<br>(10/0)*<br>(50/0)* | *****<br>(50/1)* | ***<br>*** |
| $t$ = .001in<br>$s$ = 0.5 | – | – | (5/1) | (10/0) | (10/0) | (10/0) |
| $t$ = .001in<br>$s = \infty$ | (5/1) | (10/0) | (5/1)* | * (5/0)*<br>(10/0)*(10/1)*<br>(50/0)* | ***<br>* * | ***<br>*** |
| $t$ = .0006in<br>$s$ = 0.5 | – | – | (10/0) | (5/0) | (10/0) | (10/0) |
| $t$ = .0006in<br>$s = \infty$ | – | (10/0) | (10/0) | (5/0)*<br>(5/1)*<br>(10/0)* | ***<br>(10/1)*<br>(50/0)* | ***<br>** |

– means that no practical optimum was found
$(A/B)$* means that the GA with population size $m$ = A and probability of stack swap $pp$ = B just reached 100% reliability, the * alone is used in subsequent generations as a reminder, but $(A/B)$ is omitted.

Two main conclusions ensue from Table 5.9: as the length of the chromosome increases ($t$ decreases), a large population ($m$ = 50) is no longer optimal for large numbers of analyses $n$. Furthermore, as $l$ increases, the advantage at using stack swap disappears. We next elaborate on these two conclusions.

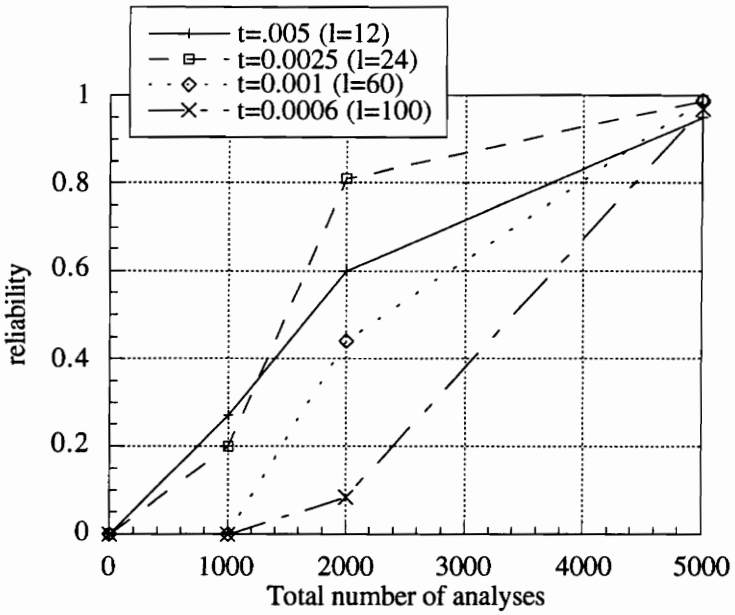



**Figure 5.11. Comparison of reliability curves for varying chromosome length $l$, single laminate problem, $s = \infty$, $m$ = 10, $pp$ = 0, 100 tests.**

Figure 5.11 shows the reliabilities at finding feasible optimal weight designs ($s = \infty$) for the single laminate problem and varying string lengths. Figure 5.12 shows the same data for the Wing Box problems. Clearly, the GA becomes slower at locating optimal weight designs as the string length increases. The same phenomenon can also be seen, although less clearly, from Table 5.9 where the GA needs over 1000 analyses to be able to locate only one optimum weight design when the chromosome is 100 digits long ($t$ = 0.0006 in.). Appendix A.1 gives analytical and other experimental arguments from the literature on GAs explaining and predicting how much longer convergence takes as the chromosome length increases.

The slowing down of the GA when the chromosome length increases in conjunction with a limited total number of analyses per search is responsible for the optimal population size remaining small as the chromosome grows. To our knowledge, the part of the literature on GAs that deals with optimum population size has always claimed that the population size should grow with the

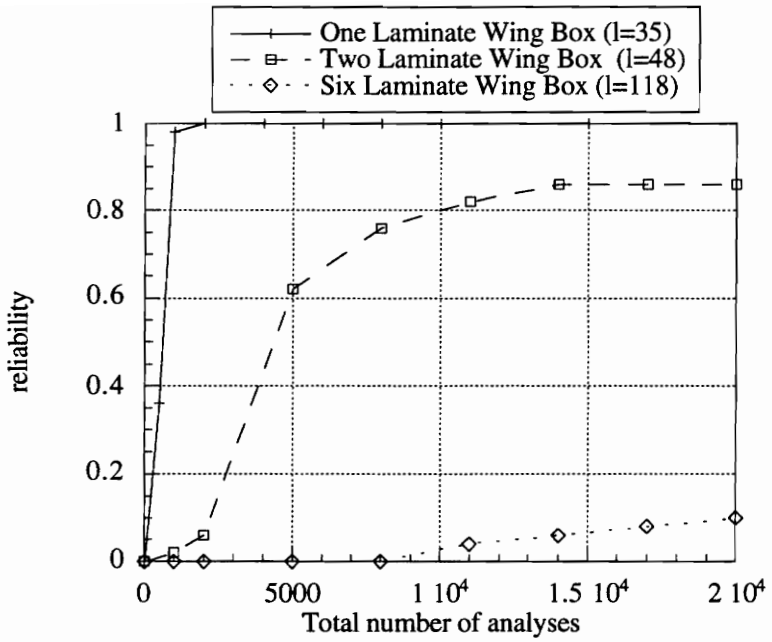



**Figure 5.12. Comparison of reliability curves for varying chromosome length $l$, wing box problem, $s = \infty$, $m$ = 10, $pp$ = 0, 50 tests.**

chromosome length. All of these studies are based on different forms of the same argument: the complexity of the problem (the design space size) increases with the length of the chromosome, which calls for a more extensive sampling of the design space, that is to say a larger population size. Rates of growth between the population size and the chromosome length were proposed that range from exponential [Goldberg85] to linear ([Goldberg92], [Alander92]). However, all of these studies disregard the fact that in practical optimization problems, the number of analyses is limited (especially when finite element codes are used to model the structure). Large population sizes are more efficient if hundreds of thousands of analyses can be afforded. Small population sizes converge faster but chances are that they converge to an optimum of lower quality. In the wing box problem, we could not usually analyze more than 30000 structures. Therefore, as can be seen in Tables 5.5 to 5.9, the genetic algorithm performed better using small population sizes than large population sizes during the 30000 first analyses of the search, especially as the length of the chromosome increases!

Referring now to Figure 5.13, if we rank the population in order of decreasing reliability, for $t$ = 0.005 in. we obtain

5 / 10 equivalent to 50, at $n$ = 1000 analyses

50 / 10 / 5, at $n$ = 10000 analyses

50 / 10 equivalent to 5, at $n$ = 30000 analyses

For a longer chromosome ($t$ = 0.0006 in.), the same observation yields

10 / 5 equivalent to 50, at $n$ = 1000 analyses

5 / 10 / 50, at $n$ = 10000 analyses

10 / 5 / 50, at $n$ = 30000 analyses

If we consider reliability at $n$ = 10000, the population sizes rank as 50 / 10 / 5 in the order of decreasing performance for $t$ = 0.005 in., but they rank as 5 / 10 / 50 for $t$ = 0.0006 in. The optimal population size decreases with increasing chromosome length as a result of observing performance at a given number of analyses. The chromosome lengths $l$ = 12 and 24, 30000 analyses represent long enough a search to permit convergence of the GA even for a population size of 50 individuals. In those cases, 50 individuals is the optimal population size at $n$ = 30000 analyses.

However, for the longer chromosomes ($l$ = 60 and 100), 30000 analyses are not sufficient for a GA with 50 individuals per generation to reach a high reliability. Population sizes of 5 and 10 individuals have, at 30000 analyses, already converged and their rate of progress is slowing down. On the contrary, a GA with a population size of 50 individuals still makes fast progress at 30000 analyses.

### Chromosome Length and Optimal Probability of Stack Swap

Table 5.9 shows that 100 percent stack swap is the most efficient strategy as long as the chromosome remains relatively short ($t$ = 0.005 in. and $t$ = 0.0025 in.). However, as the chromosome gets longer, the advantage of using intensively stack swap disappears. Figure 5.13 gives another surprising illustration of the relation between stack swap and chromosome length: even though the problem corresponding to the short chromosome ($t$ = 0.005 in.) is much easier than the problem corresponding to the long chromosome ($t$ = 0.0006 in.), only 62 percent reliability could be achieved without stack swap when $t$ = 0.005 in. ($m$ = 50), versus 100 percent when $t$ = 0.0006 in. ($m$ = 5). In other terms, the performance of the GA degrades dramatically on the short chromosome problems if no intensive stack swap is used, whereas it improves on

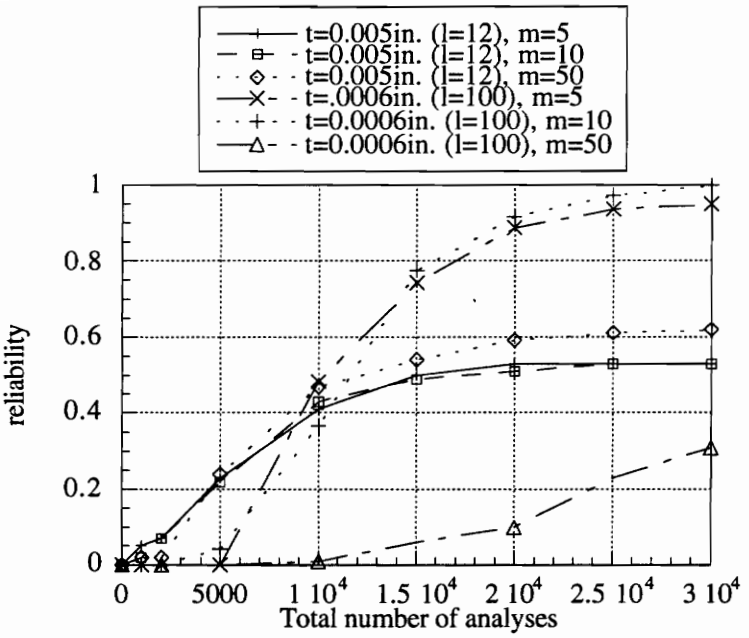



**Figure 5.13. Reliability curves for varying chromosome length $l$ and population sizes $m$, single laminate problem, $s = 0.5$, $pp$ = 0, 100 tests.**

the long chromosome problems. Insufficient and excessive pressure for convergence explain this phenomenon. For short chromosome problems, intensive stack swap is needed as an agent against premature convergence. When the chromosome is long, the risk of converging prematurly is not as critical. The additional pseudo stochastic exploration of the design space performed by stack swap is not as much needed when chromosomes are long as when they are short. With the long chromosomes, the price of stack swap which is the destruction of some design ideas, is not balanced by any need for the type of exploration it performs. Our claim is thus that the optimal probability of stack swap decreases as the chromosome gets longer.

It can be objected to the above demonstration that 100 percent stack swap is not needed on the single laminate problem with $t$ = 0.0006 in. because there are very many optimal weight designs. As it has been noticed in previous chapters, problems with many practical optima typically do not need intense use of explorative operators such as stack swap. However, if we consider a stringent definition of what a practical optimum is, as a way of reducing the number of acceptable designs, 100 percent stack swap still degrade the performance of the GA (cf. Table 5.10).

**Table 5.11. Reliability at 30000 analyses, Single Laminate problem, $t$ = 0.0006 in., $s$ = 0.2, 100 tests.**

| $m$ / $pp$ | 5 | 10 | 50 |
|---|---|---|---|
| 0 | 0.27 | 0.2 | 0.01 |
| 1 | 0.13 | 0.03 | 0.00 |

The relation between stack swap and chromosome length has been observed from the experiments made on the single laminate problem. None of the tests made on the wing box (whose chromosomes are at least 35 alleles long) call for 100 percent stack swap either. The last result however is to be considered with caution because load redistribution and the presence of many laminates in one chromosome are other factors that influence the optimal probability of stack swap. A discussion of stack swap and multiple laminates chromosomes is presented in Section 5.4.5.

### 5.4.4 Effect of the Load Redistribution on the Genetic Search

The Two-and Six-Laminate Wing Box problems have been implemented without load redistribution. The best known design for the Two-Laminate Wing Box without load redistribution is

$$[\pm 45^o_{34}]_s, \qquad \text{for the upper skin panels,}$$

$$[90^o_4/\pm 45^o/(90^o_2/0^o_2)_2/0^o_2/\pm 45^o/(\pm 45^o/0^o_2)_2]_s, \qquad \text{for the lower skin panels.}$$

This design is lighter by one gene (i.e., four plies) than the one obtained when loads are redistributed. The corresponding total number of plies in the skin is 1692 (versus 1728 if load redistribution is accounted for). The most critical constraint is combined buckling in the first laminate, $\lambda_{cr} = \lambda^1_{cc} = 1.011355$.

Similarly, the best design for the Six-Laminate Wing Box problem without load redistribution is

| stacking sequence | panels | No. of plies |
|---|---|---|
| $[\pm 45^o_{20}/(0^o_2/\pm 45^o)_2/\pm 45^o$ | 1,6 and 7, | 136 |
| $/(\pm 45^o/90^o_2/\pm 45^o)_2/90^o_2/\pm 45^o_2]_s$, | | |
| $[\pm 45^o_5/0^o_2/\pm 45^o_2/90^o_2/\pm 45^o_6/0^o_2/\pm 45^o_3/$ | 2,5 and 8 | 108 |
| $90^o_2/\pm 45^o/(0^o_2/90^o_2)_2/\pm 45^o/90^o_2]_s$, | | |
| $[90^o_2/\pm 45^o_9/0^o_2/90^o_4/\pm 45^o/0^o_2/(0^o_2/\pm 45^o)_2/90^o_2]_s$, | 3,4 and 9 | 80 |
| $[90^o_4/\pm 45^o/(0^o_2/\pm 45^o)_2]_s$, | 10,15 and 16 | 36 |
| $[90^o_4/\pm 45^o/90^o_4/(0^o_2/\pm 45^o)_2/90^o_4/\pm 45^o/90^o_2]_s$, | 11,14 and 17 | 52 |
| $[90^o_4/\pm 45^o/0^o_2/90^o_4/0^o_2/\pm 45^o_5/90^o_2/\pm 45^o]_s$, | 12,13 and 18 | 56 |

This design has two stacks less than the corresponding optimum with load redistribution. The wing skin has a total of 1404 plies. Its most critical constraint concerns strength failure due to excessive strains in the fiber direction of a $0^o$ ply in laminate 10, $\lambda_{cr} = \lambda^{10}_{c\varepsilon} = 1.026229$.

Figure 5.14 compares the reliability of the GA when optimizing the Two-Laminate Wing Box problems with and without load redistribution. The reliabilities are very close when considering feasible optimal weight designs ($s = \infty$), but when $s = 1.5$, the reliability drops dramatically for the problem with load redistribution, while it does not change for the problem with fixed loads. The efficiency of the genetic search is impaired by the load redistribution on the Two-Laminate Wing Box problem. The same is observed for the Six-Laminate Wing Box problem (see Figure 5.15). Load redistribution is responsible for additional epistasis between alleles and consequently additional noise in the schemata evaluation, two factors known for slowing down the GA [Grefenstette93, Radcliffe93]. To see this, let us consider a subset of all the structures that can be encoded, a subset defined by fixing some ply orientations in a given laminate. Suppose we calculate the critical buckling loads of the considered laminate in all the structures obtained by changing the non-fixed ply orientations and by emptying them. The standard deviation of these critical buckling loads will be larger for the problem where loads are redistributed than when the loads are fixed. Indeed, not only the $D_{ij}$'s coefficients but also the applied loads $N_{ij}$ vary in the buckling formula because of load redistribution, which is an additional source of noise in the buckling load factor statistics.

Notice also on Figure 5.15 that $m = 10$ performs much better when compared to $m = 5$ on the problem without load redistribution. As explained above, load redistribution generates a noise in the fitness evaluation of the schemata. The noise acts both as an anti-premature convergence

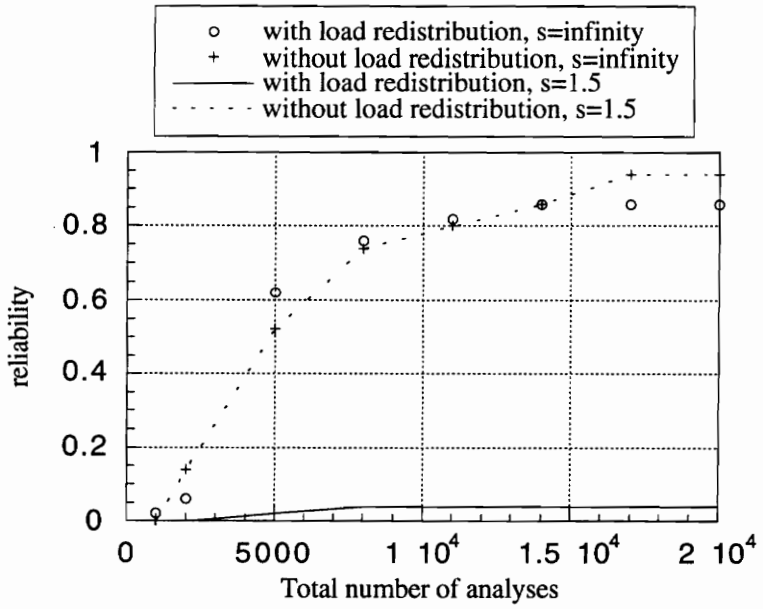



**Figure 5.14. Comparison of reliability curves with and without load redistribution, Two-Laminate Wing Box problem, $m$ = 10, $pp$ = 0, 50 tests.**

factor and calls for a higher selection pressure in order to get rid of schemata that were too optimistically reproduced because of favorable noise in their schemata fitness evaluation. These two reasons explain why the wing box problem with load redistribution benefits from smaller population sizes than the problem without load redistribution.

### 5.4.5 Effect of having Many Laminates in One Chromosome

We discuss in this Section the consequences of having multiple laminates carried by a single chromosome on the genetic search. The way the genetic operators are implemented to accommodate the multiplicity of laminates was presented earlier in Section 5.3.

The probabilities of applying various genetic operators need to be adapted depending on the number of laminates carried by the chromosome. For example, 100 percent stack swap is beneficial to the single laminate case up to a chromosome length of 60, while it is detrimental to the wing box as early as when the chromosome is 48 digits long (Two-Laminate Wing Box optimal design). The detrimental effect of stack swap on the wing box problem should not be linked to load redistribution. It was found on the Six-Laminate Wing Box problem without load

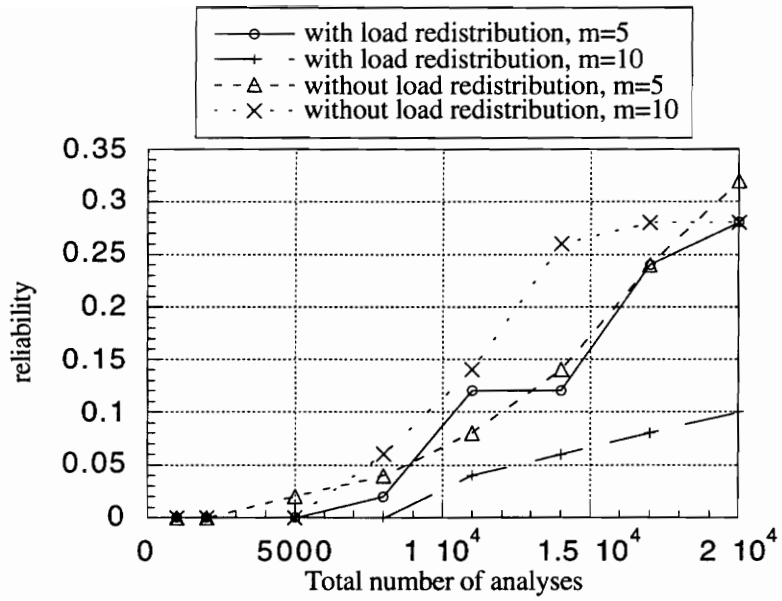



**Figure 5.15. Comparison of reliability curves with and without load redistribution, Six-Laminate Wing Box problem, $s = \infty$, $pp$ = 0, 50 tests.**

redistribution that when 100 percent stack swap is used, the GA does not locate any feasible optimum weight design in 50 runs of 20000 analyses with a population size $m$ = 10. Without stack swap, the same GA locates feasible optimum weight design 28 percent of the time after 20000 analyses. Because the genetic operators are independently applied to each laminate in the chromosome, their effect on the wing box design is cumulative. For example, a Two-Laminate Wing Box design submitted to 100 percent stack swap will in fact endure two series of stack swap (one on each independent laminate, i.e., eight stacks of two plies are relocated). A Six-Laminate Wing Box design submitted to 100 percent stack swap has 24 stacks relocated. It is very likely that the wing box design has endured major changes as a result of such a perturbation. The same reasoning applies to the change of fiber orientation mutation operator. For the stack swap and change of fiber orientation mutation operators thus, the probabilities of application need to be reduced. Dividing the optimal probability of stack swap for the single laminate case by the number of independent laminates in the chromosome seems to be a reasonable rule of thumb.

On the other hand, the probabilities of adding and deleting stacks through mutation should not be as drastically reduced. Indeed, as the number of independent laminates increases, ex-

ploration of weight distribution between the laminates becomes increasingly critical. Unless inter-laminate stack swap is used [Nagendra94], addition and deletion of stacks through mutation is the primary factor that creates new weight distributions between the laminates all along the genetic search.

A GA was tested on the Six-Laminate Wing Box problem with $pp = \frac{1}{6} \simeq 0.16$ and the probability of changing fiber orientation $pf$ reduced from $pf = 0.01$ to $pf = 0.005$. These new probabilities applied with a population size $m$ equal to 5 yielded a new feasible optimum design that is one gene (equivalent to $4 \times 3$ =12 plies in all of the skin) lighter than the old optimum design. This new optimum design has 1416 plies total in the skin, versus 1428 without stack swap.

| stacking sequence | panels | No. of plies |
|---|---|---|
| $[\pm 45^o_{17}/0^o_4/\pm 45^o/0^o_2/90^o_2/\pm 45^o/90^o_2/0^o_2/$ $90^o_2/\pm 45^o/(0^o_4/90^o_2)_2/90^o_2/0^o_2]_s$, | 1,6 and 7 | 140 |
| $[\pm 45^o_5/0^o_2/\pm 45^o_3/(\pm 45^o/90^o_2)_4/90^o_2$ $/\pm 45^o/0^o_4/90^o_4/0^o_2/\pm 45^o/90^o_2]_s$, | 2,5 and 8 | 104 |
| $[\pm 45^o_{20}]_s$, | 3,4 and 9 | 80 |
| $[\pm 45^o/(90^o_2/0^o_4/90^o_2)_2/0^o_2]_s$, | 10,15 and 16 | 40 |
| $[\pm 45^o_3/90^o_2/\pm 45^o_4/90^o_4/\pm 45^o_3/0^o_2]_s$, | 11,14 and 17 | 52 |
| $[\pm 45^o_2/90^o_4/\pm 45^o_{10}]_s$, | 12,13 and 18 | 56 |

With this last tuning, the GA finds feasible designs having the new optimum weight 11 percent of the time after 20000 analyses (based on 71 tests). If the population size is increased to $m = 10$, designs with the new optimum weight are not found anymore. However, the probability of finding feasible designs with the old optimum weight is 22 percent after 20000 analyses ($m = 10$, $pp = 0.055$, $pf = 0.005$) versus 10 percent ($m = 10$, $pp = 0.$, $pf = 0.01$) and 0 percent ($m = 10$, $pp = 1.$, $pf = 0.01$).

Similarly, a GA with $pp = \frac{1}{18} \simeq 0.055$ and $pf = 0.005$ performed better on the 18-Laminate Wing Box Problem than the same GA without stack swap and with $pf = 0.01$. On the average, the designs found with the first setting of the GA were 6% lighter than the designs found with the second setting (cf. Table 5.8, Section 5.4.1).

So, stack swap is also a very efficient operator on the wing box problem when its probability is reduced with increasing number of independent laminates being optimized.

## 5.5 Optimization of Coupled Composite Panels: Observations

A genetic algorithm has been developed for the optimization of a composite wing box-beam, where many mechanically coupled laminates are simultaneously designed for minimum weight. Because chromosome representations can be found that fit well the discreteness and topology of the wing box problem, the GA can easily handle many laminates. The One-, Two- and Six-Laminate Wing Box problems were successfully solved by genetic optimization. In contrast, optimizing the wing box with traditional continuous methods involves a priori choices about the stacking sequence, heuristics to transform the continuous design into a discrete design, and simple constraints such as the ply contiguity constraint are very difficult to implement.

Nevertheless, increasing design space size and design variable interactions were found to slow down convergence of the GA towards optimal designs. A first consequence is that there is no advantage at increasing the population size as the design space size increases if the total number of analyses performed in the search remains of the same order.

Even though the proportion of the design space sampled by the GA decreases as the design space size increases (cf. Section 3.5), the number of analyses to achieve convergence grows. For the 18-Laminate Wing Box problem that has a chromosome of 720 genes with four possible allele values at each gene, it was very difficult to reliably converge to low weight designs in a realistic time (we tried up to 150000 analyses). Possible ways to speed the genetic search on this kind of problem were proposed and partially investigated: an approximation of the analysis can be used to shorten calculation times; simplified statements of the problem, like the wing box problem with frozen internal loads, can be solved recursively; other specialized operators, like inter-laminate swap and scaling mutation, that are efficient at finding the optimal weight designs can be used.

The next Chapter takes a different look at the problem of minimizing the cost of the genetic search. The possibility of controlling the number of searches and the length of each search is considered. After the choice of the operators (Chapter 4), the tuning of their respective probabilities of being applied (Chapter 3), the decomposition of the search into many small searches is seen to have an influence on the cost of achieving a given reliability with the GA.

# Chapter 6

# OPTIMAL NUMBER OF SEARCHES AND SEARCH LENGTH FOR RELIABLE GENETIC OPTIMIZATION

The reliability of a genetic optimizer as a function of the number and length of the searches is studied. It is empirically shown that there is an advantage in performing many short genetic searches rather than a long one, especially when high levels of reliability are required. An example is given where a genetic algorithm is specifically modified for better efficiency in the context of repeated short runs. Knowing the reliability of a genetic optimizer after a certain number of analyses, a procedure is studied that enables predicting the reliability at later stages of the search, and the optimal number and length of searches to achieve it.

## *6.1 Introduction: Convergence of Genetic Algorithms*

A genetic algorithm (GA), when used as optimizer on a static function, typically goes through two phases, one of fast progress towards better points in the domain searched, followed by a period of lower efficiency. The end of the fast progress phase is usually called (although this term is not mathematically rigorous) convergence (cf. Chapter 2).

The Schema Theorem ([Holland75], Chapter 2) proves the fast convergence of the population towards good regions of the design space, provided that selection pressure is maintained and that the genetic operators do not excessively disrupt the positive features carried by good individuals. Louis and Rawlins [Louis93] have developed predictions of the time to convergence for generational GAs without mutation and with restrictions on the crossover operator used. Goldberg and Deb ([Goldberg90]) have estimated the "takeover time" of various selection procedures, which is the expected number of generations after which the population will be composed of all but one identical individuals. The calculations applied to generational GAs, where the effects of mutation and crossover were neglected. Thierens and Goldberg ([Thierens93]) introduced the

notion of "mixing time", which is an expected number of generations for the GA to create the optimum individual.

Much efforts for improving GAs has focused on lengthening the fast improvement phase based on the aforementioned studies. For example, by specifying that the takeover time is larger than the mixing time, a relation is derived in [Thierens93] that a GA must satisfy in order to find the optimum before it has converged. However, such results are not readily applicable to GAs which do not use binary coding, and have mutation and problem specific operators.

In this chapter, the only three parameters that are considered are the number of searches, their respective length, and a practical measure of the performance of the GA called reliability (cf. Chapters 3 and 4). The purpose of the GA here is to find practical optima, i.e., optimum and near optimum points, with the highest possible reliability after a given number of analyses. Accordingly, our measure of the performance of a GA is the reliability $r(n)$, which is the probability a GA has of finding a practical optimum after $n$ analyses. The reliability curves $r(n)$ used as data here come from Chapter 4, where the purpose of the optimization is to minimize the thickness of a composite laminated plate subject to strength and buckling constraints. One advantage of this example is that it is computationally inexpensive, so that a large number of runs can be made to accurately estimate $r(n)$. Moreover, it is a problem of practical importance because it represents well more complex problems, in the field of composites engineering in particular, and in design oriented optimization problems in general.

This chapter is an investigation of the relation between maximum reliability and the number of searches. Indeed, the reliability of a GA can be increased by executing multiple short searches instead of a single long search. It is shown that the GA can be specifically tailored to take advantage of repeated runs of short length. In the last part of this paper, a procedure is proposed to predict the reliability of a GA at later stages of a run, and to tell how many searches of which length will yield such a reliability.

## 6.2 *Example Problem: Minimum Thickness of a Composite Laminated Plate*

### 6.2.1 Problem Description

The purpose of this work is not to obtain a better understanding of how GAs work, but instead to develop a strategy for using best existing GAs. As such, our demonstration needs to be based on an existing practical optimization problem and the corresponding GA. Moreover, the problem needs to be computationally inexpensive to permit exhaustive testing. The example problem of minimizing the thickness of a composite laminated plate fulfills all of these requirements.

A GA was thus implemented in Chapter 4 in order to find the minimum number of layers and the orientations of the fibers within each layer such that the laminate does not fail. More precisely, the first objective was to find the thinnest plate that would not fail, and if several of those plates existed, the one with the largest safety margin. Simple mathematical models were used to predict the various failure modes of a laminated composite plate subject to a given set of loads $(X_i, Y_i)$ (see Figure 6.1). The design variables were the fiber orientations restricted to the discrete pool of 0°, 90°, or ±45°. A more complete description of the analysis and optimization problem formulation is given in Chapter 4.

The present study uses two GAs as examples, one called "GA fixed" (GAfix) and the other "GA variable" (GAvar). GAfix was originally implemented in Chapter 3 for safety margin maximization for a plate of fixed thickness. In Chapter 4, the genetic operators were modified for better efficiency when the laminate thickness can vary, which produced GAvar. The coding used in both GAfix and GAvar is the direct representation of one half of the layer stacking sequence (the other half being found by symmetry). Because $+45°$ layers are conveniently paired with $-45°$, the basic building block was selected to be two layers with possible alleles as $0^\circ_2$, $90^\circ_2$, $\pm 45°$ or $E_2$, with $E_2$ denoting two empty layers. We thus deal with a 4-letter alphabet. Each chromosome had 16 genes, so the size of the design space was $4^{16}$. Figure 6.1 shows an example of a chromosome (shortened for graphical purpose) and the associated plate. Both GAfix and GAvar are nonoverlapping elitist generational GAs, where selection is based on a linear normalization

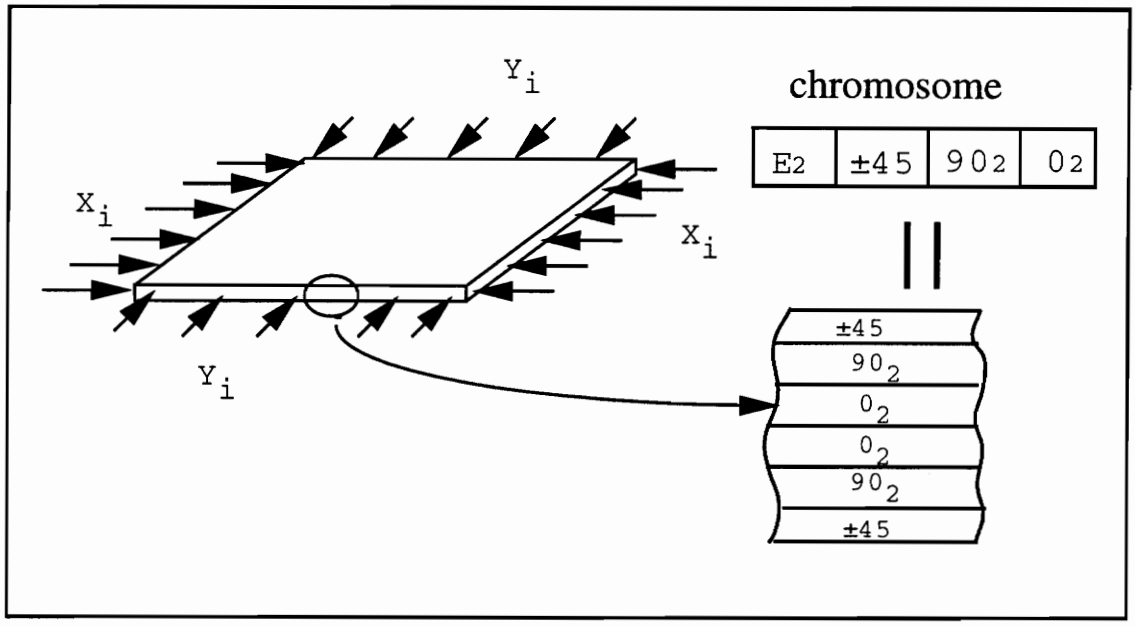



**Figure 6.1. Composite laminated plate, loading, and associated chromosome.**

of the rank of each individual. That is, the $i$-th best individual in a population of m individuals has a probability of being selected as parent equal to $2(m-i+1)/(m^2+m)$. GAfix and GAvar both use a problem specific operator (although different versions of it) called stack swap. GAfix uses a two point crossover while GAvar uses a one point crossover. In contrast to GAfix, the selection procedure in GAvar does not allow two identical designs to be paired. Both GAfix and GAvar have a probability of crossover equal to 1, a probability of stack swap equal to 1, and a population size equal to 8. GAfix has a probability of mutation per allele equal to 0.01. GAvar has a probability of changing the fiber orientation per allele equal to 0.01, and probabilities of adding and deleting one stack of two layers per individual equal to 0.05. These probabilities were obtained by extensive experimentation (Chapters 3 and 4).

### 6.2.2 Practical Optima and Reliability of the GA

The composite plate problem often has many optimal and/or near optimal solutions. In Chapters 3 and 4, a practical optimum was defined as an optimal weight design that has a safety margin within 0.1% of the optimal safety margin. This is a fairly stringent definition of practical optimum that would fulfill any real design requirement. It was made possible by the low computational cost of an analysis in relation with the design space size and structure, so that

the GA could locate practical optima in a reasonable time ($\leq$ 6000 analyses). To avoid having results that depend too much on a particular load combination, we consider four different sets of loads $(X_i, Y_i)$, i.e., four different optimization problems. The problem cases had 3, 4, 13 and over 13 practical optima, respectively.

The reliability associated with each optimization problem defined by a set $(X_i, Y_i)$ was estimated by averaging the frequencies at which practical optima are found over many independent GA runs. In this study, we ran 200 optimizations for each of the four load cases. The reliability $r_1(n_t)$ that we will be discussing in the rest of this study is the average of the four load related reliabilities at a given number of analyses $n_t$. The subscript 1 here denoted that the reliability is associated with a single search (even though to obtain $r_1(n_t)$ here we run 200 optimizations and average).

## 6.3 Optimal Number and Length of Searches for Reliable Genetic Optimization

Because the later stage of a genetic search shows slow progress $r_1(n_t)$ tends to become flat as $n_t$ increases. It is a case of diminishing returns. This occurs because some searches converge to the wrong region of the design space and take very long to extricate themselves out of that region. To reduce the chances of getting stuck in the wrong region, it may be more economical to make several short searches rather than have $n$ large enough so that every single search would have time to reach a practical optimum.

We denote the reliability obtained by repeating the genetic optimization $p$ times, each time with $n$ searches as $r_p(n_t)$, where $n_t = (pn)$ is the total number of analyses, also called price of the search. Then, because the $p$ searches are independent

$$r_p(n_t) = 1 - (1 - r_1(n))^p. \tag{6.1}$$

The first problem we address here is the following: given a single-search reliability curve $r_1(n_t)$, what is the number of searches $p$ and their length $n$ that minimizes the price of the search $n_t$ and achieves a reliability of at least $R$. That is,

$$\begin{aligned} \underset{n,p}{\text{Minimize}} \quad & np, \\ \text{such that} \quad & 1 - [1 - r_1(n)]^p \geq R. \end{aligned}$$

We can alternatively solve the dual problem of maximizing the reliability $r_p(n_t)$ for a given search price $n_t$

$$\begin{aligned} \underset{n,p}{\text{Maximize}} \quad & r_p(pn) = 1 - (1 - r_1(n))^p, \\ \text{such that} \quad & np \leq n_t. \end{aligned}$$

For an elitist GA, $r_p(n_t)$ is an increasing function of $n_t$, and the inequality constraint in the dual problem becomes an equality because it will always be active. Consequently, $n = n_t/p$, and there is only one variable left, $p$, so that the dual problem is easier to solve than the primal problem. For the examples here we found the optimum number of searches $p_o$, which maximizes $r_p(n_t)$ for a given $n_t = pn$ by enumeration, trying all values of $p$ between 1 and 20. To simplify our notation we use $r_o(n_t)$ to mean the reliability that can be achieved with $n_t$ analyses when the optimum number of searches is carried out.

Figures 6.2 and 6.3 show the reliability $r_p(n_t)$ versus the total number of analyses $n_t = (pn)$ for one, two, three and the optimum number $p_o$ of consecutive GA searches, for GAfix and GAvar, respectively. It is seen from Figure 6.2 that, for a required reliability $R = 80\%$, GAfix requires about 3300 analyses with one search, and about 2600 analyses with two or three shorter searches. Figure 6.3 shows that, for $R = 80\%$, one or two searches with GAvar are optimal and require about 1450 analyses. However, from Figure 6.3 we see that a reliability $R = 90\%$ may be achieved by two searches of 1050 analyses each, versus one search of 2650 analyses. That is a saving of 20% in the number of analyses can be realized by making two runs instead of one. The advantage of splitting the search into several shorter searches increases when higher reliability is sought. $R = 95\%$ is yielded by three searches of 900 analyses each ($n_t = 2700$), versus one search of 4600 analyses, which represents 41% saving.

Another important conclusion appears to emerge from Figures 6.2 and 6.3: in order to obtain the complete $r_o(n_t)$ curve, it is sufficient to calculate $r_1(n_t)$ until the optimum number of searches $p_o$ changes from 1 to 2. If we consider GAvar for example, the optimal number of searches $p_o$ changes from 1 to 2 at $n_t = 1450$ analyses, and from 2 to 3 at $n_t = 2550$ analyses.

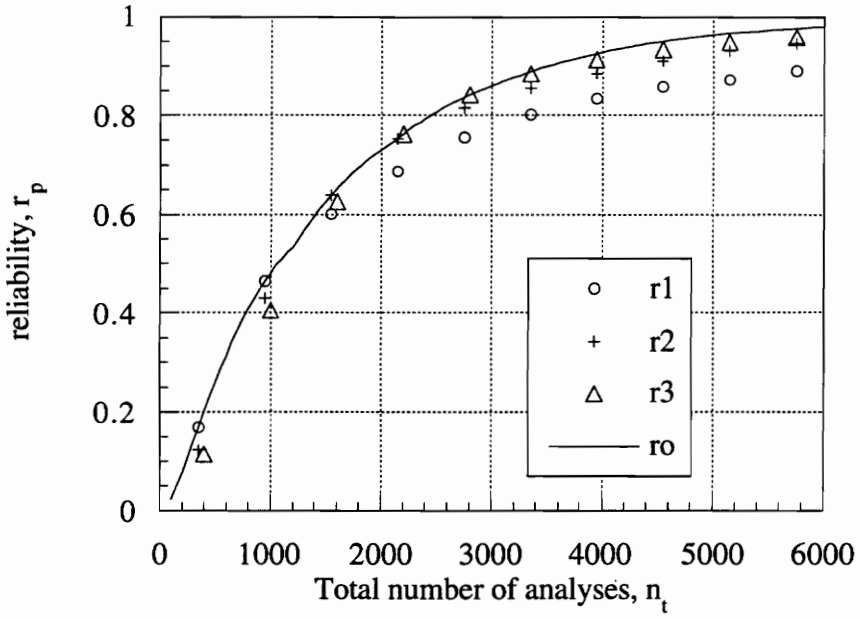



**Figure 6.2. reliability versus total number of analyses ($n_t$), GAfix.**

The portion of the curve $r_o(n_t) = r_2(n_t)$ ($p_o = 2$) needs only information from $r_1(n_t)$ up to the point $n_t = 2550/2 = 1275 < 1450$. The same applies to the rest of the $r_o(n_t)$ reliability curve and it also applies to GAfix. Furthermore, if the target reliability is high enough, it is even possible to stop generating $r_1(n_t)$ earlier. For example, in order to achieve 80 percent reliability, it is necessary to run 1450 analyses with GAvar. But if the target reliability is raised to 90 percent, 1050 analyses will be sufficient, and for 95 percent reliability only 900 analyses are needed. Practically, this means that GAs need not to be tested over a very large number of analyses when establishing $r_1(n_t)$. This conclusion, however, depends on the shape of the $r_1(n_t)$ curve, which for our two examples, has a negative second derivative $d^2r_1/d^2n$. If the derivative changes sign as in a sigmoid curve, or even if it becomes close to zero, $p_o$ may not increase monotonically with $n_t$.

We also explored the possibility of specially tuning the GA in order to take advantage of repeated runs. The idea is to maximize the speed at which the GA makes progress early on. On our particular problem, one way of forcing the GA to make fast progress at the beginning of the

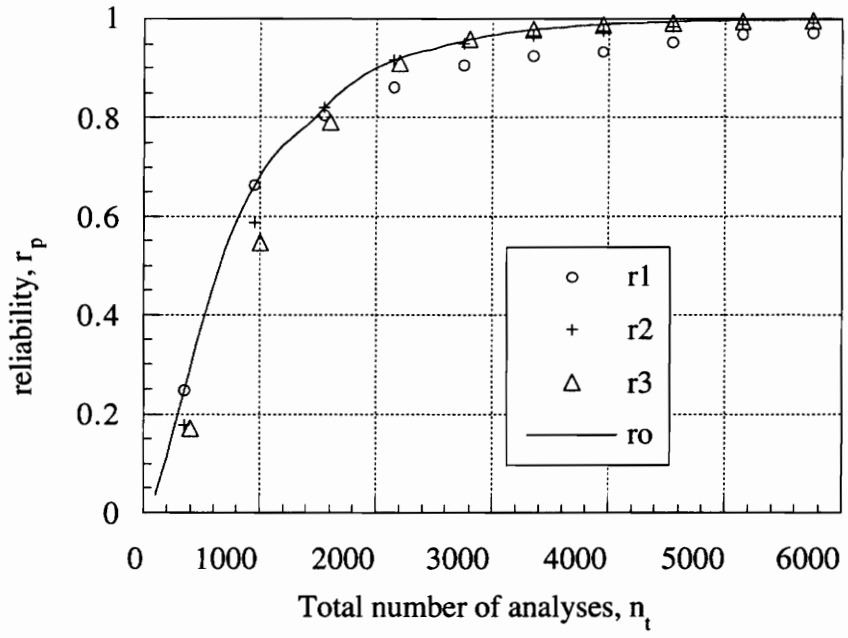



**Figure 6.3. Reliability versus total number of analyses ($n_t$), GAvar.**

run is to increase the selection pressure. Although this strategy will not work on every problem, it is expected to apply to many cases. By increasing the selection pressure, we allow more runs to converge rapidly, but we also increase the chance that they will converge to the wrong answer. Which effect will dominate is problem dependent. However, when the number of analyses $n_t$ is small, we expect the first effect to be more important than when $n_t$ is high.

The GA used on Figure 6.4 is in every aspect similar to GAvar, except for the selection of designs which is based on the rank of the design squared, i.e., the probability of the i-th best individual in the population to be selected as parent is $6(m+1-i)^2/[m(m+1)(2m+1)]$. With this modification, GAvar when run twice for 650 analyses has a reliability of 0.8015%. For $R = 80\%$, the modified GAvar run twice is about 10% cheaper than GAvar (1300 against 1450 analyses). When run once, the modified GAvar is about 10% more expensive than GAvar (1590 against 1450 analyses to achieve 80% reliability). That is, the stronger selection pressure increases the reliability for $n_t$ = 650 from 0.50 to 0.55. This happens because without strong selection pressure, even runs that are headed in the right direction do not usually have a chance to

reach a practical optimum for this low $n_t$. On the other hand, the increased selection pressure also reduces the chances of runs that started in the wrong part of design space to extricate themselves, so for $n_t$ = 1450, the strong selection pressure reduces the reliability from 0.80 to 0.79.

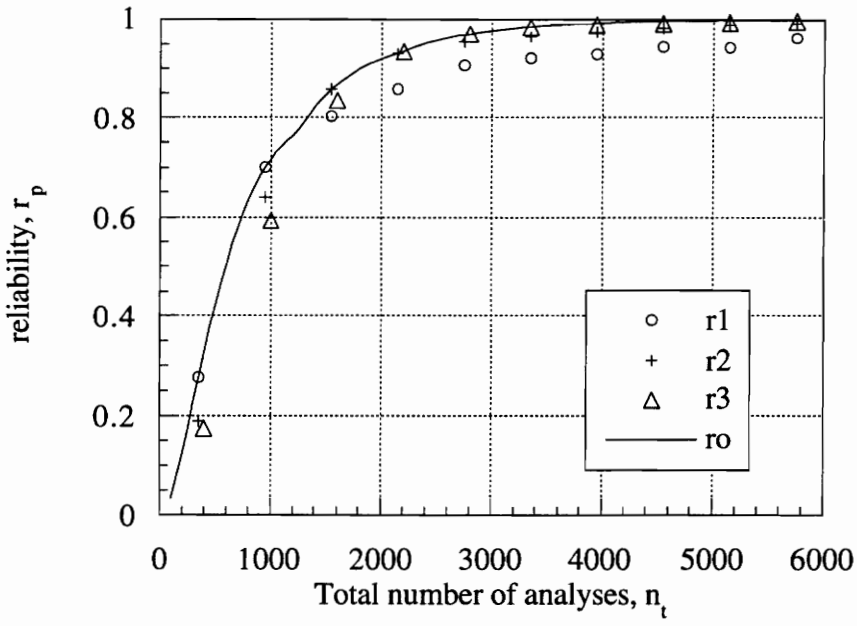



**Figure 6.4. Reliability versus total number of analyses ($n_t$), (modified) GAvar: fitness = square of the rank.**

# *6.4 Prediction of the Reliability*

## 6.4.1 Extrapolation of the Reliability Curve $r_1(n_t)$

The second issue we consider here is the extrapolation of the reliability curve $r_1(n_t)$. We need to extrapolate the reliability $r_1(n_t)$ when a problem is too costly to test to a high enough reliability. For example, running 100 searches of 2000 analyses might be affordable while running 100 searches of 3000 analyses might not. In that case, the reliability of the GA after 3000 analyses

needs to be derived by extrapolation. There are also cases where the GA that is being tested has a stopping criterion such as: "Stop the search after $k$ analyses without improvement". With such a stopping criterion, different runs of the same GA stop after different number of analyses. It is necessary to extrapolate the performance of the short GA runs to study the reliability of the GA at later stages of the search. Finally, as is shown later in this study, extrapolating $r_1(n_t)$ permits us to obtain an estimate $p_p$ to the optimal number of runs for a given total cost of the search $n_t$, and also to predict up to what point the single search reliability curve $r_1(n_t)$ is needed.

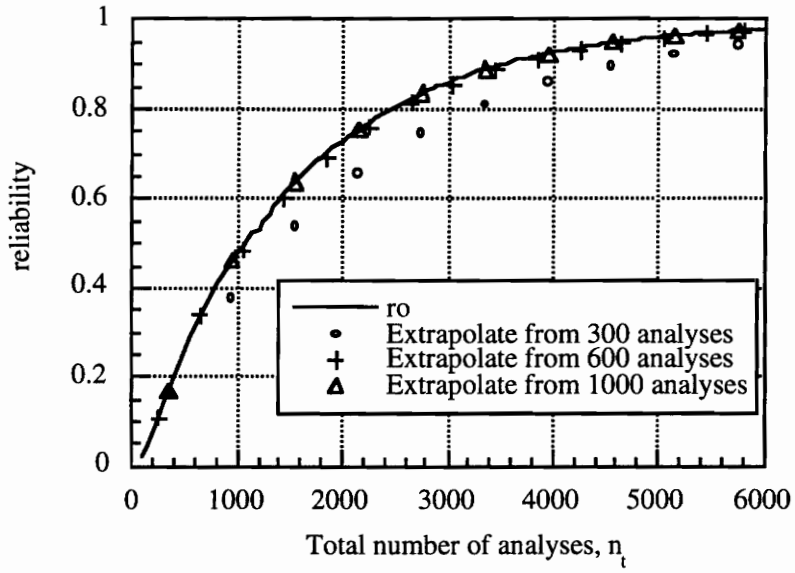



**Figure 6.5. Reliability extrapolation, GAfix.**

The reliability $r_1(n_t)$ can be extrapolated using Equation (6.1). For the sake of extrapolation, we assume that the number of GA runs $p$ is a continuous variable. Then, the number of searches of length $n_e$ necessary to achieve a reliability $R$ is

$$p = \frac{ln\ (1 - R)}{ln\ (1 - r_1(n_e))}. \tag{6.2}$$

The point ( $pn_e$ , $R$ ) is being extrapolated from the point ( $n_e$ , $r_1(n_e)$ ). Figures 6.5 and 6.6 show, for GAfix and GAvar respectively, the reliability $r_o(n_t)$ obtained for the optimum

number of searches $p_o$, and three extrapolated reliability curves starting from $n_e$ = 300, 600, and 1000 respectively. The reason for showing $r_o(n_t)$ on the Figures and not $r_1(n_t)$, is that we are interested in knowing $r_1(n_t)$ only as long as one search is optimal, i.e., as long as $r_1(n_t) = r_o(n_t)$.

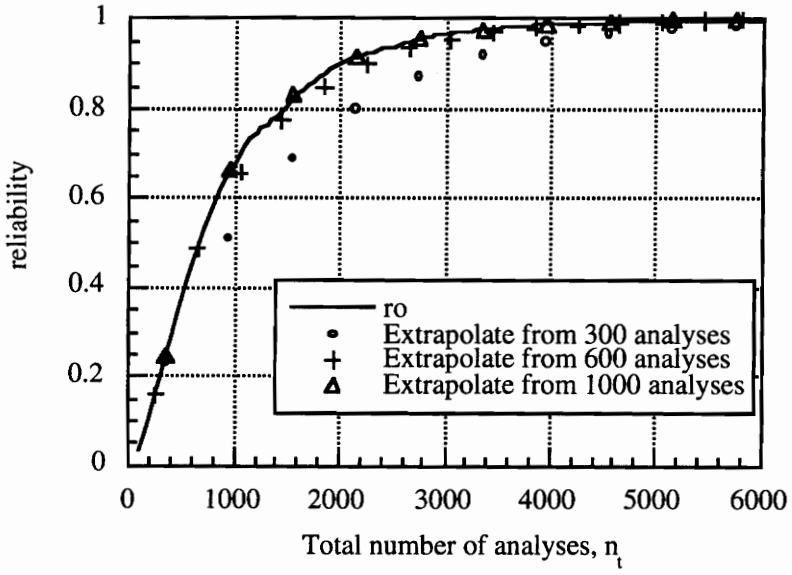



**Figure 6.6. Reliability extrapolation, GAvar.**

For both GAfix and GAvar, the extrapolations starting from $n_e$ = 300, 600, and 1000 usually underestimate the reliability that can be obtained using the optimum number of searches $p_o$. Figures 6.5 and 6.6 show that the extrapolation scheme proposed permits a reasonable long range prediction of $r_o(n_t)$. The largest error seen on the extrapolated data from Figures 6.5 and 6.6 is about 20%. Note that extrapolating $r_1(n_t)$ after 1500 analyses, both for GAvar and GAfix, would not make sense because beyond 1500 analyses the optimal number of searches $p_o$ changes from 1 to 2: as was pointed out when discussing Figures 6.2 and 6.3, knowing $r_1(n_t)$ at and after 1500 analyses is of no interest neither for constructing $r_o(n_t)$ nor for deciding how to use the GA.

### 6.4.2 Prediction of the Optimum Number of Searches

The extrapolated reliability curves can be taken as $r_1(n_t)$ in order to obtain an estimate $p_p$ to the optimal number of searches for a given total number of analyses $n_t$. The process of obtaining this estimate $p_p$ is exactly the same as the one we used to calculate $p_o$ in the first part of this study, except that we use the extrapolated reliability curve for one search instead of the exact complete data $r_1(n_t)$.

Note however that, when $i \times n_e \leq n_t \leq (i+1) \times n_e \quad (i \geq 2)$, the predicted reliability of any number of searches $j$, $1 \leq j \leq i$, is the same. For example, if $r(n)$ is extrapolated after $n_e$ = 300 analyses, the predicted reliabilities yielded by one search of 700 analyses and two searches of 350 analyses will be exactly the same. Similarly, the predicted reliability at 1000 analyses for one, two, and three searches will be the same. This is due to the fact that Equation (6.1) is used twice, first for extrapolating $r_1(n_t)$ from $r_1(n_e)$ to $r_1(n_e + \Delta)$, and second for calculating the reliability of j repeated searches using $r_1(n_e + \Delta)$ to get $r_1(j \times (n_e + \Delta))$. Whatever the number of times Equation (6.1) is used, the prediction of the reliability is the same because it comes from the same knowledge ($r_1(n_e)$) through the same procedure (Equation (6.1)).

We prove hereafter that, when considering extrapolated reliability curves, one search of $n_t$ = $pn_e$ analyses has the same predicted reliability as $q$ searches of $(p/q)n_e$ analyses each (for a same total number of analyses $n_t$ and $p > q$). $n_e$ is the number of analyses after which the extrapolation is started, and $r_1(n_e)$ is the corresponding reliability for one search. According to the extrapolation procedure (Equations (6.1) and (6.2)), the reliability for one search is

$$R_1(n_t) \;=\; 1 \;-\; (1 - r_1(n_e))^p \;=\; 1 \;-\; (1 - r_1(n_e))^{n_t/n_e}, \tag{6.3}$$

where $R_q(n)$ denotes the reliability of $q$ searches of $n$ analyses each, such as predicted from the extrapolation. The reliability at $n_t$ analyses yielded by $q$ searches of $(p/qn_e)$ analyses is

$$R_q(pn_e) \;=\; 1 \;-\; \{1 - R_1((\frac{p}{q})n_e)\}^q \;=\; 1 \;-\; \{1 - [1 - (1 - r_1(n_e))^{\frac{pn_e}{qn_e}}]\}^q,$$

or,

$$R_q(n_t) = 1 - \{(1 - r_1(n_e))^{p/q}\}^q = 1 - (1 - r_1(n_e))^p = R_1(n_t).$$

The condition for this result to apply is that $\frac{p}{q} n_e > n_e$, i.e., $p > q$. Figure 6.7 illustrates the relation for GAfix and $n_e$ = 1000. It shows that, using the extrapolated reliability curve for one search, one and two searches are equivalent after 2000 analyses, one, two and three searches are equivalent after 3000 analyses, and so on.

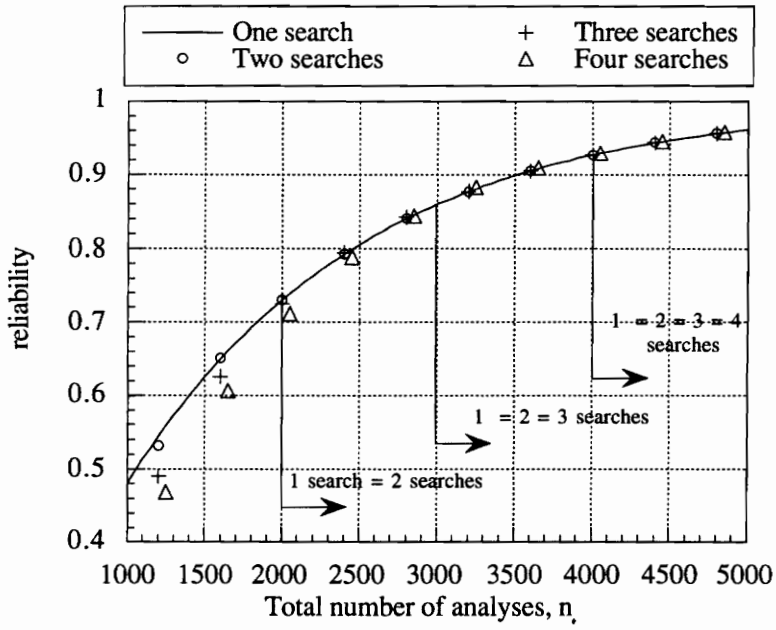



**Figure 6.7. Extrapolated reliability for 1,2,3, and 4 searches, GAfix, the reliability for 1 search is extrapolated from 1000 analyses on.**

As a result, when predicting the optimal number of searches $p_p$ for $n_t$ analyses ($i \times n_e \leq n_t \leq (i+1) \times n_e$), two situations can occur: first, if $i < 2$ (i.e., $n_e \leq n_t \leq 2n_e$) or $p_p > i$ (i.e., $n_t/p_p < n_e$), the predicted number of searches $p_p$ is unique because Equation (6.1) is used once. In the other case ( $n_t > 2n_e$ and $p_p \leq i$), the prediction only tells us that $p_p \in [1, i]$.

Figures 6.8 and 6.9 show the predicted optimum number of searches $p_p$ for extrapolations starting at $n_e$ = 300 and 1000 respectively, using GAfix. When the inaccuracy in $p_p$ discussed

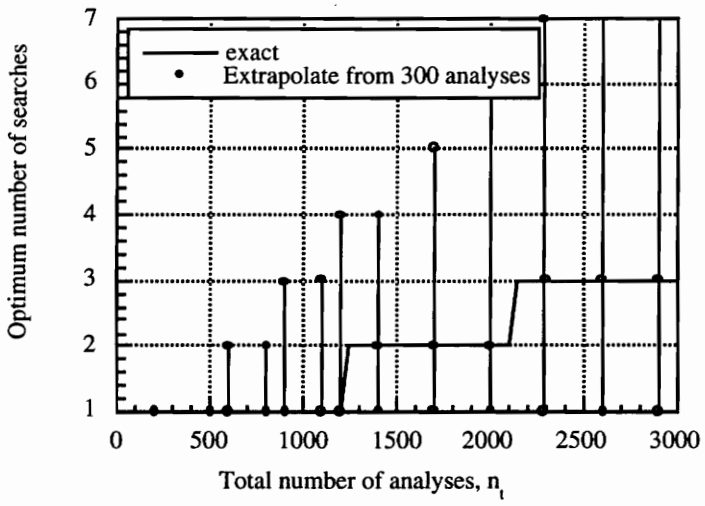



**Figure 6.8. Optimal number of searches, GAfix. The vertical bars cover the whole range of predicted optimal number of searches such as found from the extrapolated data.**

above occurs, vertical bars are put that denote the whole range of $p_p$. As can be seen on Figure 6.8, the prediction of the optimum number of searches is poor when the extrapolation is started early (the error grows approximately linearly with the total number of analyses $n_t$). However, when the extrapolation is started at 1000 analyses, the prediction of the optimal number of searches is good (cf. Figure 6.9). Similar conclusions can be drawn for GAvar.

The most important point to predict is when the optimal number of searches changes from 1 to 2. When starting the extrapolation from 300 analyses, the prediction of the switch is poor because one and two searches are equivalent from 600 analyses on. That is, all we can predict is that the switch occurs after 600 analyses. In fact, the switch from $p_o = 1$ to $p_o = 2$ occurs at $n = 1250$ analyses for GAfix (cf. Figure 6.8). On the other hand, when the extrapolation starts at 1000 analyses, the switch is predicted at 1550 analyses (cf. Figure 6.9), which corresponds to about 25% error.

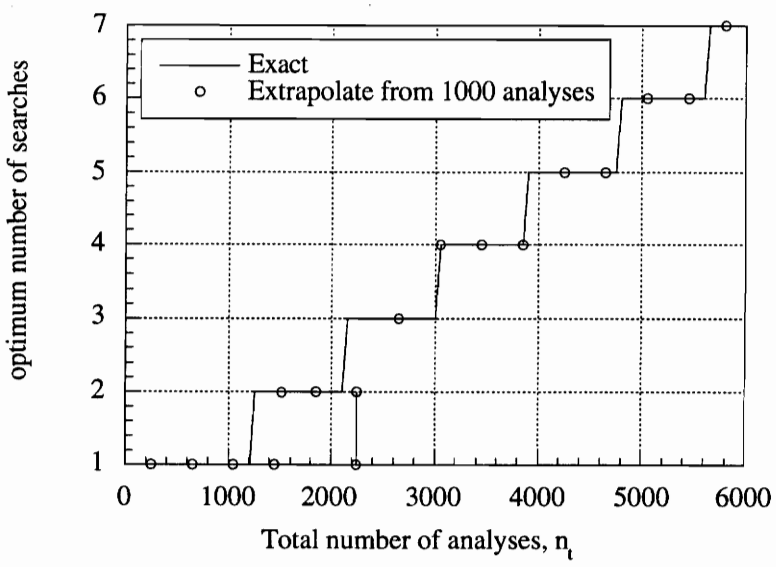



**Figure 6.9. Optimal number of searches, GAfix. The vertical bars cover the whole range of predicted optimal number of searches such as found from the extrapolated data.**

## *6.5 Multiple Number of Searches and Stopping Criterion*

This Chapter investigates the effect of the number of searches and their respective length on the reliability of a GA. The reliability curve departs from zero only when a non-trivial percentage of searches have converged to a practical optimum. Increasing the number of analyses for a fixed number of searches works when there are still many searches heading in the right direction but which have not converged yet. Eventually most of these converge, and increasing the number of analyses for a fixed number of searches has two bad consequences: analyses are wasted either on searches that are already optimal, or they are wasted trying to rescue searches that have converged to the wrong optimum.

As a result, it was found beneficial to increase the number of searches while keeping the total number of analyses constant for laminate design problems. The advantage of repeated short searches increased as a higher reliability was required from the GA. It was also shown that using a strategy of multiple short searches may favor GAs with stronger selection pressure.

For the examples presented here, there is no need to know the reliability of one search $r_1(n_t)$ beyond the point where one search is no longer optimal. Furthermore an extrapolation procedure was developed to predict the reliability in the long run based on knowledge of the reliability early in the search. This may allow large savings in number of analyses when testing the reliability of a GA.

With respect to the stopping criterion, this study shows that a stopping criterion based on a maximum number of analyses without improvement is best. It will avoid spending analyses on a stalled genetic search. The computer resources left would be better spent by starting a new search. Because the optimal number of searches increases with the required reliability, it is better to set the stopping criterion to a small number of generations without improvement if high reliabilities are sought.

# Chapter 7
# CONCLUDING REMARKS

The use of a genetic algorithm to optimize the stacking sequence of one or many laminated plates was presented. Buckling constraints as well as ply contiguity and strain constraints were considered. The laminate design space was found to include multiple optima, especially for a large number of plies. The genetic algorithm was shown to be able to produce several of these optima in a single execution. It also deals well with integer variables and is not sensitive to problem nonlinearities. As a result, the genetic algorithm looks for global optimum, i.e., it can escape local optima where gradient based methods would get trapped. Genetic algorithms are powerful tools for the design of composite laminates, but they require a large number of analyses. Several approaches to reducing the cost and increasing the reliability of the GAs were described.

The genetic algorithm was adapted to take advantage of the special features of stacking sequence design. That is, knowledge about composite laminates can be used to create more efficient sampling mechanisms that yet do not jeopardize the globality of the search as gradient information does. The first way this is done is by choosing a coding of the designs in association with a crossover operator. This is the genetic formulation of the problem, that defines how data are structured. In this work, a natural coding of the laminate stacking sequence was employed. A stack swap operator was implemented that permits perturbing the flexural properties of plates while keeping the in-plane response unchanged for a more efficient pseudo-random exploration of the design space.

The specialized genetic algorithm was first implemented for a simply supported plate designed for maximal buckling load. The cost of the search was sufficiently low to investigate the effect of the genetic parameters on the reliability of the search by large scale testing. After tuning, it was possible to obtain near-optimal designs with a high probability by analyzing fewer than 1000 out of 531,441. design points. This compared favorably with simulated annealing applied

elsewhere for the same problem. Tuning of the GA yielded the following optimal set of genetic parameters:

| | |
|---|---|
| population size | 8 |
| crossover rate | 1.00 |
| mutation rate (per string) | 0.01-0.10 |
| two-stack swap rate | 1.00-2.25 |

As compared to conventional genetic algorithms, the tuned genetic algorithm for stacking sequence design has an extra operator, stack swap. The combination of intensive stack swap with an uncommonly small population yielded the lowest price of the search. The effect of the magnitude of the penalty for violating the ply-contiguity constraint on the genetic search was also investigated. It was found that the smallest penalty that yields feasible designs is optimal.

Next, the dual problem of minimizing the weight while maximizing the failure load was explored. The performance of the first genetic algorithm, devised for buckling load maximization for a fixed thickness plate, degraded when the laminate was allowed to change thickness. New genetic operators that account for varying thickness were implemented and tested. A mix of fixed penalty and penalty proportional to the distance to feasibility was proposed to account for the constraints. The combined penalties were found to perform better than proportional penalty alone. Scaling mutation, an operator that changes the thickness of the laminates according to the most critical constraint value, enabled further decrease in price of the search by guiding mutations towards the optimal weight region of the design space. The dual problem of minimizing the weight while maximizing the failure load could be solved with a high reliability by sampling 1350 out of more than 4 billion design points.

Application to a wing box-beam structure showed that GAs can be applied successfully to more complex problems. Even though chromosomes longer than in the independent plate problems

were considered, the optimal population size did not increase, as could have been expected from other studies ([Goldberg85], [Goldberg93], [Thierens93], [Tate93], [Louis93]). The reason is that the number of analyses per search has not been increased in proportion to the increase in chromosome length. It is a practical result since, in most real cases of structural optimization, the number of analyses is limited. Stack swap was still an efficient operator on the wing box problem, but at a probability reduced by division by the number of laminates in the structure. That is, the probability that the whole structure to endure stack swap should remain constant. The cost of the genetic search was found to grow as a result of load redistribution in the wing box.

We also studied the influence of the number and length of searches on the cost and reliability of the genetic algorithm. For the laminate design problem, we found that it is preferable to split the search into several short searches if the required reliability is high. The genetic algorithm can be also specially tuned, for example by increasing the selection pressure, to take full advantage of multiple short runs. A procedure is proposed to extrapolate the reliability curve. One practical consequence of this study is that there is no need to test the GA beyond the point where many searches start having a higher reliability than one search. A procedure to predict the optimal number of searches was proposed and discussed.

A few implementation alternatives still need to be studied: selection procedures exist that have a lower stochastic sampling noise than the selection procedure implemented in this work, like for example the Stochastic Universal Sampling ([Baker87]); the steady state GA ([Syswerda89], [Eschelman91]), where new individual designs are introduced one after each other in the population enjoys a growing popularity for optimization purpose and needs to be tested for stacking sequence design; self-adaptation ([Bäck92]) is a strategy where the GA optimizes its own genetic parameters as the search proceeds, by treating the probabilities of applying operators as individual

features, coding them on the chromosome and subjecting them to selection and reproduction. Ideally, self-adaptation is a way to avoid tuning the GA. However, there is a cost for self-adaptation that still needs to be assessed.

More efforts need to be devoted to the wing box problem in order to efficiently locate the optimal weight region. A first idea is to implement scaling mutation (cf. Sections 4.3.6 and 4.4.6) on the wing box problem. Inter-laminate stack swap ([Nagendra94]) is an operator where stacks are swapped between different laminates. It explores new weight distributions in the structure while keeping the total weight constant. The current version of the GA lacks this kind of weight redistribution mechanisms, and has to rely on the combination of two lucky mutations to produce the same effect, which is a very unlikely event.

Most structural optimization problems of practical interest have costly analysis in terms of computer resources. Particular attention should be put on the use of approximate analysis within the genetic search. A first step in that direction has been taken in Section 6.2.1 and 6.4.2 for the wing box-beam problem. In [Harrison94], the approximate analysis is constructed during the GA run and is progressively mixed with the exact analysis. GAs are known for not being very sensitive to noise in the objective function and so, should be able to easily cope with the switch between exact and approximate analysis. Further research in that direction is needed. Another approach to reduce the cost of the genetic search is to hybridize it with more efficient local search techniques. The genetic algorithm would be in charge of keeping the search global, while heuristic methods would perform local optimization for better efficiency. The GA plus scaling mutation can be seen as an example of hybrid GA. An effort to both use approximate analysis in conjunction with local improvement techniques has already been made in [Kogiso94b] on the minimum thickness laminate design problem.

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

# REFERENCES


**Adali90** Adali, S., and Duffy, K.J.,"Design of Antisymmetric Hybrid Laminates for Maximum Buckling Load: I.Optimal Fiber Orientation. II. Optimal Layer Thickness", *Composite Structures*, 14, 1990, No.1, pp.49-60, and No.2, pp. 113-124.

**Alander93** Alander, J.T.,"On Optimal Population Size of Genetic Algorithms", *Proceedings of Compeuro 92*, IEEE Computer Society Press, 1992, pp. 65-70.

**Bäck92** Bäck, T., "Self-Adaptation in Genetic Algorithms", *Proceedings of the First European Conference on Artificial Life*, F.J.Varela and P.Bourgines eds., the MIT Press, Cambridge MA, 1992, pp. 263-271.

**Bäck93** Bäck, T. and Schwefel, H.P., "An Overview of Evolutionary Algorithms for Parameter Optimization", *Evolutionary Computation*, 1(1), 1993, pp.1-23.

**Baker87** Baker, J.E., "Reducing Bias and Inefficiency in the Genetic Algorithm", *Proceedings of the Second International Conference on Genetic Algorithms*, MIT, Cambridge, MA, July 28-31, 1987, pp. 14-21.

**Ball93** Ball, N.R., Sargent, P.M., and Ige, D.O., "Genetic Algorithm Representations for Laminate Layups", *Artificial Intelligence in Engineering*, 8(2), 1993, pp. 99-108.

**Batoz90** Batoz, J.-L., and Dhatt, G., *Modélisation des Structures par Eléments Finis*, Hermès Ed., Paris, France, 1990.

**Bethke81** Bethke, A.D., *Genetic Algorithms as Function Optimizers*, Doctoral Dissertation, Univ. of Michigan, *Dissertation Abstracts International 41*(9), 3503B, (University Microfilms No.8106101), 1981.

**Booker85** Booker, L.B., "Improving the Performance of Genetic Algorithms in Classifier Systems", *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Carnegie-Mellon Univ., Pittsburg, Pa., July 24-26, 1985, pp. 80-92.

**Bushnell87** Bushnell, D., "PANDA2 – Program for Minimum Weight Design of Stiffened, Composite, Locally Buckled Panels", *Computers and Structures*, 25(4), 1987, pp. 469-605.

**Callahan92** Callahan, K.J., Weeks, G.E., "Optimum Design of Composite Laminates Using Genetic Algorithms", *Composite Engineering*, 2(3), 1992, pp. 149-160.

**Davidor90** Davidor, Y., *Genetic Algorithms and Robotics: a Heuristic Strategy for Optimization*, World Scientific Series in Robotics and Automated Systems, 1, World Scientific Publ., 1990.

**Davis91** Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold Publ., 1991.

**DeJong75** De Jong, K.A., "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", Doctoral Dissertation, University of Michigan, Dissertation Abstract International, 36 (10), 5140B, (University Microfilms No. 76-9381), 1975.

**Eschelman89** Eschelman, L.J., Caruana, R. and Schaffer, D., "Biases in the Crossover Landscape," *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Fairfax, Va., June 1989.

**Eschelman91** Eschelman, L.J., "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination," in *Foundations of Genetic Algorithms*, Rawlins G.J.E. (Eds.), Morgan Kaufmann (Publ.), San Mateo, CA, 1991, pp. 265-283.

**Fogel91** Fogel, D.B., "System Identification Through Simulated Evolution: a Machine Learning Approach to Modeling", Needham Heights, MA:Ginn, 1991.

**Forrest93** Forrest, S. and Mitchell, M.,"Relative Building Block Fitness and the Building Block Hypothesis", in *Foundations of Genetic Algorithms – 2*, Whitley L.D. (Eds.), Morgan Kaufmann (Publ.), San Mateo, CA, 1993, pp. 109-126.

**Fraser62** Fraser, A.S., "Simulation of Genetic Systems" , *Journal of Theoritical Biology*, 2, 1962, pp. 329-346.

**Fullmer91** Fullmer, B., and Mkkulainen, R., "Using Marker-Based Genetic Encoding of Neural Networks To evolve Finite-state Behaviour", *Proceedings of the First European Conference on Artificial Life* (ECAL-91), Paris, 1991.

**Furuya93** Furuya, H., and Haftka, R.T., "Genetic Algorithms for Placing Actuators on Space Structures", *Proceedings of the fifth International Conference on Genetic Algorithms*, Univ. of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers, July 17-21, 1993, pp. 536-542.

**Gallagher92** Gallagher, K., Sambridge, M., and Drijkoningen, G., "Genetic Algorithms: an Evolution from Monte Carlo Methods for Strongly Nonlinear Geophysical Optimization", *Geophysical Research Letters*, 18(3), May-June 1992, pp. 735-740.

**Garfinkel72** Garfinkel, R.S., and Nemhauser, G.L.,*Integer Programming*, John Wiley and Sons, Inc., New York, 1972.

**Glover89** Glover, F., "Tabu Search – *Part I*, ORSA Journal on Computing", 1(3), 1989, pp. 190-206.

**Glover90** Glover, F., "Tabu Search – *Part II*, ORSA Journal on Computing", 2(1), 1990, pp. 4-32.

**Goldberg85** Goldberg, D.E., "Optimal Initial Population Size for Binary-Coded Genetic Algorithms", *TCGA Report* No. 85001, Department of Engineering Mechanics, The University of Alabama, Tuscaloosa, AL 35487-2908, November 1985.

**Goldberg87a** Goldberg, D.E. and Santani, T.A., "Engineering Optimization by a Genetic Algorithm", *Proceedings of the 9th Conference on Electronic Computation*, ASCE, 1987, pp. 471-482.

**Goldberg87b** Goldberg, D.E., "Simple Genetic Algorithms and the Minimal Deceptive Problem", *Genetic Algorithms and Simulated Annealing*, Davis L. Ed., London: Pitman, 1987, pp. 74-88.

**Goldberg89a** Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison - Wesley Publishing Company, Inc., Reading, MA. 1989.

**Goldberg89b** Goldberg, D.E., Korb, B., and Deb, K., "Messy Genetic Algorithms: Motivation, Analysis, and First Results", *TCGA Report* No. 89003, Departement of Engineering Mechanics, The University of Alabama, Tuscaloosa, AL 35487, May 1989.

**Goldberg90** Goldberg, D.E., and Deb, K., "A Comparative Analysis of Selection Schemes Used In Genetic Algorithms", *TCGA Report* No. 90007, The University of Alabama, Tuscaloosa, AL 35487, July 1990.

**Goldberg91** Goldberg, D.E., and Deb, K., "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms", *Foundations of Genetic Algorithms*, Morgan Kaufmann Publ., San Mateo, CA, 1991, pp. 69-93.

**Goldberg92** Goldberg, D.E., Deb, K., and Clark, J.H., "Genetic Algorithms, Noise, and the Sizing of Populations", *Complex Systems*, 6, 1992, pp. 333-362.

**Goldberg93** Goldberg, D.E., Deb, K., and Clark, J.H., "Accounting for Noise in the Sizing of Populations", in *Foundations of Genetic Algorithms – 2*, Whitley L.D. (Eds.), Morgan Kaufmann (Publ.), San Mateo, CA, 1993, pp. 127-140.

**Greene87** Greene, D.P., and Smith, S.F., "A Genetic System For Learning Models of Consumer Choice", *Proceedings of the Second International Conference on Genetic Algorithms*, MIT, Cambridge, MA, July 28-31, 1987, pp. 217-223.

**Grefenstette86** Grefenstette, J.J., "Optimization of Control Parameters for Genetic Algorithms", in *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-16(1), 1986, pp. 122-128.

**Grefenstette93** Grefenstette, J.J., "Deception Considered Harmful", in *Foundations of Genetic Algorithms – 2*, Whitley L.D. (Eds.), Morgan Kaufmann (Publ.), San Mateo, CA, 1993, pp. 75-92.

**Gürdal91** Gürdal, Z., and Haftka, R.T., "Optimization of Composite Laminates", presented at the *NATO Advanced Study Institute on Optimization of Large Structural Systems*, Berchtesgaden, Germany, Sept.23 - Oct. 4, 1991.

**Haftka83** Haftka, R.T., Starnes, J.H., and Nair, S., "Design for Global Damage Tolerance and Associated Mass Penalties", *Journal of Aircraft*, 20(1), January 1983, pp. 83-88.

**Haftka92a** Haftka, R.T., and Gürdal, Z., *Elements of Structural Optimization*, Kluwer Academic Publishers, Dordrecht, 1992.

**Haftka92b** Haftka, R.T., and Walsh, J.L., "Stacking-Sequence Optimization for Buckling of Laminated Plates by Integer Programming," *AIAA Journal*, 30(3), 1992, pp. 814-819.

**Hajela90** Hajela, P., "Genetic Search – An Approach to the Nonconvex Optimization problem", *AIAA Journal*, 26(7), 1990, pp. 1205-1210.

**Hajela91** Hajela, P., "Optimal Design of Viscoelastically Damped Beam Structures", *Applied Mechanics Review*, 44(11), part 2, supplement, Nov. 1991, pp. S96-S106.

**Hajela92** Hajela, P., and Lin, C.Y., "Genetic Search Strategies in Multicriterion Optimal Design", *Structural Optimization*, 4, 1992, pp. 99-107.

**Harrison94** Harrison, P.N., Le Riche, R. and Haftka, R.T., "Design of Stiffened Composite Panels by Genetic Algorithm and Response Surface Approximations", to be published in *Proceedings of the 36th AIAA/ ASME/ ASCE/ AHS/ ASC Structural Dynamics and Materials Conference"*, New Orleans, April 10-13, 1995.

**Holland75** Holland, J.H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

**IMSL89** IMSL MATH/LIBRARY, *User's Manual*, Version 1.1, December 89, pp. 895-902.

**Kirpatrick83** Kirpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by Simulated Annealing", *Science*, 220(4598), pp. 671-680, 1983.

**Kogiso94a** Kogiso, N., Watson, L.T., Gürdal, Z., and Haftka, R.T., "Genetic Algorithms with Local Improvement for Composite Laminate Design", *Structural Optimization*, 7(4), pp. 207-218, 1994.

**Kogiso94b** Kogiso, N., Watson, L.T., Gürdal, Z., Haftka, R.T. and Nagendra, S., "Minimum Thickness Design of Composite Laminates subject to Buckling Strength Constraints by Genetic Algorithm", *Mechanics of Composite Materials and Structures*, 1(1), 1994, pp. 95-117.

**Kreizig83** Kreiszig, E., *Advanced Engineering Mathematics*, Wiley Eastern Limited Ed., New Delhi, 1983.

**Kristinsson92** Kristinsson, K.K., and Dumont, G.A., "System Identification and Control Using Genetic Algorithms", IEEE Transactions on Systems, Man and Cybernetics, 22(5), Sept./Oct. 1992, pp. 1033-1046.

**Lekhnitskii68** Lekhnitskii, S.G., *Anisotropic Plates*, translated by Tsai S.W., and Cheron T., Gordon and Breach Sci. Publ., Inc., 1968.

**LeRiche93** Le Riche, R., and Haftka, R. T., "Optimization of Laminate Stacking Sequence for Buckling Load Maximization by Genetic Algorithm", *AIAA Journal*, 31(5), May 1993, pp. 951-970.

**LeRiche94a** Le Riche, R., and Haftka, R. T., "Improved Genetic Algorithm for Minimum Thickness Composite Laminate Design", (submitted for publication in *Composites Engineering*), 1994.

**LeRiche94b** Le Riche, R., and Haftka, R. T., "Optimal Number of Searches and Search Length for Reliable Genetic Optimization", (submitted for publication in *Evolutionary Computation*), 1994.

**Lombardi92** Lombardi, M., Haftka, R.T. and Cinquini, C., "Optimization of Laminate stacking Sequence for Buckling Load Maximization by Simulated Annealing", *Proceedings of the 33rd AIAA/ASME/ASCE SDM Conference*, Dallas, Texas, 1992.

**Louis93** Louis, S.J., and Rawlins, G.J.E., "Syntactic Analysis of Convergence in Genetic Algorithms", *Foundations of Genetic Algorithms – 2*, Morgan Kaufmann Publ., San Mateo, CA, 1993, pp. 141-151.

**Lucasius91** Lucasius, C.B., Blommers, M.J.J., Buydens, L.M.C., and Kateman, G., "A Genetic Algorithm for Conformational Analysis of DNA", in *Handbook of Genetic Algorithms*, L.Davis (Ed.), Van Nostrand Reinhold Publ., New York, 1991, pp. 251-281.

**Mesquita87** Mesquita, L., and Kamat, M.P., "Optimization of Stiffened Laminated Composite Plate with Frequency Constraints," *Engineering Optimization*, 11, 1987, pp. 77-88.

**Michalewicz91** Michalewicz, Z., and Janikow, C.Z., "Handling Constraints in Genetic Algorithms," *Proceedings of the fourth International Conference on Genetic Algorithms*, San Mateo, Ca, Morgan Kaufmann, 1991, pp. 151-157.

**Miki79** Miki, M., "Optimum Design of Fibrous Laminated Composite Plates Subject to Axial Compression", *Proceedings of the 3rd Japan-US Composite Materials Conference*, Tokyo, Technomic Publishing Co., Lancaster, 1979, pp. 1017-1019.

**Miki91** Miki, M., and Sugiyama, Y., "Optimum Design of Laminated Composite Plates Using Lamination Parameters", *Proceedings of the 32nd AIAA/ ASME/ ASCE/ AHS/ ASC Structures, Structural Dynamics and Materials Conference*, (Baltimore MD), AIAA, Washington, DC, AIAA Paper No.91-0971-CP, April 1991, pp. 275-283.

**Nagendra92** Nagendra, S., Haftka, R. T., and Gürdal, Z., "Stacking Sequence Optimization of Simply Supported Laminates with Stability and Strain Constraints", *AIAA Journal*, 30(8), 1992, pp. 2132-2137.

**Nagendra93a** Nagendra, S., Haftka, R.T., and Gürdal, Z., "Design of Blade Stiffened Composite Panel by a Genetic Algorithm, " *Proceedings of the 34th AIAA/ ASME/ ASCE/ AHS/ ASC Structures, Structural Dynamics and Material Conference*, La Jolla, CA, April 19-21, 1993, pp. 2418-2436.

**Nagendra93b** Nagendra, S., "Optimal Stacking Sequence Design of Stiffened Composite Panels with Cutouts," Ph.D. Dissertation, Virginia Polytechnic and State University, June 1993.

**Nagendra94** Nagendra, S., Jestin, D., Gürdal, Z., Haftka, R.T., and Watson, L.T., "Improved Genetic Algorithm for the Design of Stiffened Composite Panels," submitted for publication to *Composites and Structures*, 1994.

**Orvosh93** Orvosh, D. and Davis, L., "Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints," *Proceedings of the fifth International Conference on Genetic Algorithms*, Univ. of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers, July 17-21, 1993, p. 650.

**Pedersen87** Pedersen, P., "On Sensitivity Analysis and Optimal Design of Specially Orthotropic Laminates", *Engineering Optimization*, 11, 1987, pp. 305-316.

**Powell93** Powell, D., and Skolnick, M.M., "Using Genetic Algorithms in Engineering Design Optimization with Non-linear Constraints", *Proceedings of the fifth International Conference on Genetic Algorithms*, Univ. of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers, July 17-21, 1993, pp. 424-431.

**Potter90** Potter, W.D., Tonn, B.E., Hilliar, M.R., Liepens, G.E., Goeltz, R.T., and Purucker, S.L., "Diagnosis, Parsimony, and Genetic Algorithms", *Proceedings of the Third Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 1990, pp. 1-8.

**Quilici92** Quilici, S., *Prédimensionnement de Structures Composites Stratifiées - La Prise en Compte de Critères mécaniques*, Doctoral dissertation, Dépt. de Mécanique des Solides, Université d'Aix-Marseilles II, France, 1992.

**Radcliffe90** Radcliffe, N.J., *Genetic Neural Networks on MIMD Computers*, Ph.D. dissertation, University of Edinburgh, Edinburgh, UK, 1990.

**Radcliffe93** Radcliffe, N.J., *A Study in Set Recombination, Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Univ. of Illinois at Urbana-Champaign, July 17-21, 1993, pp. 23-30.

**Ragon94** Ragon, S.A., *Optimization of Composite Box-Beam Structures Including Effects of Subcomponents Interaction*, Master Thesis, Dept. of Engineering Mechanics, Virginia Polytechnic Institute and State University, 1994.

**Ramsey93** Ramsey, C.L., and Grefenstette, J.J., "Case-based Initialization of Genetic Algorithms", *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Univ. of Illinois at Urbana-Champaign, July 17-21, 1993, pp. 84-91.

**Rao90** Rao, S.S., Pan, T.S., and Venkayya, V.B., "Optimal Placement of Actuators in Actively Controlled Structures using Genetic Algorithms", *AIAA Journal* , 28(6), 1990, pp. 942-943.

**Rechenberg65** Rechenberg, I., *Cybernetic Solution Path of an Experimental Problem*, Royal Aircraft Establishment, Library Translation 1122, Farnborough, UK, 1965.

**Rechenberg73** Rechenberg, I., *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Stuttgart, Frommann-Holzboog, 1973.

**Reddy84** Reddy, J.N., *An Introduction To The Finite Element Method*, McGraw-Hill, 1984.

**Richardson89** Richardson, J.T., Palmer, M.R., Liepins, G. and Hilliard, M., "Some Guidelines For Genetic Algorithms with Penalty Functions," *Proceedings of the Third International Conference on Genetic Algorithms*, George Mason Univ., Morgan Kaufmann Publishers, June 4-7, 1989, pp. 191-197.

**Schaffer89** Schaffer, J.D., "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization", *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Fairfax, Va., June 1989, pp. 51-59.

**Schittkowski86** Schittkowski, K., "NLPQL: a FORTRAN subroutine solving nonlinear programming problems", (edited by Clyde L. Monma), *Annals of Operations Research*, 5, pp. 485-500.

**Schmit79** Schmit, L.A., and Farshi, B., "Optimum Design of Laminates Composite Plates", *International Journal for Numerical Methods in Engineering*, 11, No.4, 1979, pp. 623-640.

**Schmit80** Schmit, L.A., and Mehrinfar, M., "Multilevel Optimum Design of Structures with Fiber-Composite Stiffened-Panel Components", *AIAA Journal*, 20(1), 1980.

**Schoenauer93** Schoenauer, M., and Xanthakis, S.,"Constrained GA Optimization", *Proceedings of the fifth International Conference on Genetic Algorithms*, Univ. of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers, July 17-21, 1993, pp. 573-580.

**Shin89** Shin, Y.S., Haftka, R.T., Watson, L.T., and Plaut, R.T., "Design of Laminated Plates for Maximum Buckling Load", *Journal of Composite Materials*, 23,1989, pp. 348-369.

**Sobieski79** Sobieszczanski-Sobieski, J., "An Integrated Computer Procedure for Sizing Composite Airframe Structures," NASA TP-1300, 1979.

**Spears91** Spears, W.M. and De Jong, K.A., "An Analysis of Multi-point Crossover," in *Foundations of Genetic Algorithms*, Rawlins G.J.E. (Eds), Morgan Kaufmann Publishers, San Mateo, CA., 1991, pp. 301-315.

**Spears93** Spears, W.M., "Crossover or Mutation?," in *Foundations of Genetic Algorithms – 2*, Whitley L.D. (Eds.), Morgan Kaufmann (Publ.), San Mateo, CA, 1993, pp. 221-237.

**Starnes79** Starnes, J.H., Jr., and Haftka, R.T., "Preliminary Design of Composite Wings for Buckling, Strength, and Displacement Constraints," in *Journal of Aircraft*, 16, 1979, pp. 564-570.

**Syswerda89** Syswerda, G., "Uniform Crossover in Genetic Algorithms," *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Fairfax, Va., June 1989, pp. 2-9.

**Tate93** Tate, D.M., and Smith, A.E., "Expected Allele Coverage and the Role of Mutation in Genetic Algorithms", *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Univ. of Illinois at Urbana-Champaign, July 17-21, 1993, pp. 31-37.

**Thierens93** Thierens, D., and Goldberg, D.E., "Mixing in Genetic Algorithms", *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publ., Univ. of Illinois at Urbana-Champaign, July 17-21, 1993, pp.38-47.

**Underbrink94** Underbrink, A.J.Jr., and Williams, G.P.W.Jr., "Genetic Algorithms Applied to Assembly Operations," to appear in *Proceedings of the 1994 SPIE Conference KB AI Systems in Aerospace and Industry*, SPIE code number 2244-04.

**Vose91** Vose, M.D., "Formalizing Genetic Algorithms" , Technical Report CS-91-127, The Univ. of Tenessee, Computer Science Dept., Feb. 1991.

**Watabe93** Watabe, H., and Okino, N., "A study on Genetic Shape Design", *Proceedings of the fifth International Conference on Genetic Algorithms*, Univ. of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers, July 17-21, 1993, pp. 445-450.

**Watkins88** Watkins, R.I., "Optimal Design of Large Laminated Structures", *ICAS Paper*, No. 88-1.10.2, 1988, pp. 1480-1486.

**Whitley89** Whitley, D., "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best", *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Fairfax, Va., June 1989, pp. 116-123.

**Whitley93** *Foundations of Genetic Algorithms – 2*, Whitley, L.D., Editor, Morgan Kaufmann Publishers, San Mateo, California, 1993.

**Whitney85** Whitney, J.M., *Structural Analysis Of Laminated Anisotropic Plates*, Technomic Publishing Company, Lancaster, 1985.

**Zhigljavsky91** Zhigljavsky, A.A., *Theory of Global Random Search*, Springer Verlag, 1991.

# Appendix A

# DIMENSIONAL MODELS OF GENETIC ALGORITHMS

This Appendix discusses analytical models to predict optimal population size and time to convergence in genetic algorithms. Numerical applications are given for the composite optimization problems of Chapters 3, 4 and 5. The reader is referred to Chapter 2 for a definition of the basic concepts about GAs, like schemata, genes, alleles, etc..

## *A.1 Predictions of the Population Size*

We present hereafter analytically derived population sizes based on four theories: the expected allele coverage ([Tate93]), the number of excess unique schemata ([Goldberg85]), the population-sizing equation ([Goldberg92]) and an empirical relation ([Alander92]).

### A.1.1 Expected Allele Coverage

The expected allele coverage $AC$ is the expected proportion of all possible alleles which occur in a population of size $m$. $AC$ should not be mistaken for the probability that all alleles are present in the population. For example, if $K = 4$ and $m = 3$, the probability that all alleles are present in the population is zero, while the proportion of all possible alleles that occur in the population, AC, is larger than zero. By setting requirements on the value of $AC$, we can calculate a lower bound on the size of the population.

Let us suppose that chromosomes have length $l$ and that there are $K$ distinct symbols used for coding.

From [Tate93], the expected allele coverage $AC$ is,

$$AC = 1 - \frac{1}{K}\sum_{j=1}^{K-1} j p_j(m,K), \tag{A.1}$$

where

$$p_j(m,K) = \frac{1}{K^m}\binom{K}{j}\sum_{i=0}^{K-j-i}(-1)^i\binom{K-j}{i}(K-j-i)^m. \tag{A.2}$$

Note that $AC$ does not depend on the length $l$ of the string. Using this formula, we found that the smallest population soze to achieve $AC = 0.999$ is 18 if $K = 3$, and 25 if $K = 4$.

Although the number of symbols, $K$, is 4, $K = 3$ was observed to account for the special treatment $E$ alleles receive during recombination. Recall from Chapter 4 that the $E$'s are pushed on the outside of the string, that the crossover rarely breaks the strings in the middle of the $E$'s, and that other genetic operators also treat the $E$'s separately. Rapidly after the beginning of the search, the GA mixes mainly $0_2$, $\pm 45$, $90_2$ alleles that are located on the right hand side of the string, to the detriment of the $E$'s. Such an implementation is equivalent to shortening the string and reducing the number of symbols handled.

The expected allele coverage reasoning only considers alleles, i.e., schemata of order 1, and neglects associations of alleles. In contrast, the next counting argument for deriving population sizes accounts for all schemata.

## A.1.2 Number of Excess Unique Schemata per Individual

The following calculations generalize to any number of symbols per alphabet $K$ the derivation of optimal population size for binary alphabets given in [Goldberg85].

The probability of each gene to assume a given value is $1/K$. The probability that a string matches an order $j$ schema is,

$$P_{\text{single match on order j schema}} = (1/K)^j. \tag{A.3}$$

The probability that none of the $m$ strings in the population is an example of a given schema of order $j$ is,

$$P_{\text{no order j success in m ind.}} = \left[1 - (1/K)^j\right]^m. \tag{A.4}$$

The probability that at least one member of the population is an instance of a given order $j$ schema is,

$$P_{\text{at least one order j success in m ind.}} = 1 - \left[1 - (1/K)^j\right]^m. \tag{A.5}$$

For $j$ fixed positions, there are $K^j$ different schemata. There are $\binom{l}{j}$ ways to select the $j$ positions in a $l$-digits string. The expected number of unique schemata is,

$$n_s = \sum_{j=1}^{l} \binom{l}{j} K^j \{1 - \left[1 - (1/K)^j\right]^m\}. \tag{A.6}$$

A single string is an instance of $2^l$ schemata since each position can be a "don't care" symbol $*$ or the value it has. Furthermore, a single string is the zero point in terms of schemata processing through recombination, since at least two strings are necessary for recombination . A measure of the useful schemata in the population is the excess schemata count $n_es$,

$$n_{es} = n_s - 2^l. \tag{A.7}$$

Assuming that schemata sampled by the GA lead the way to the optimum solution, it is desirable to maximize the number of excess unique schemata per individual,

$$max \frac{n_s - 2^l}{m}. \tag{A.8}$$

Solutions to that problem yield optimal population sizes. Results for different values of $l$ and $K$ are given in Table A.1.

**Table A.1. Optimal population sizes using the number of excess unique schemata per individual.**

| l<br>K | 30 | 40 | 60 |
|---|---|---|---|
| 3 | 700 | 5400 | 300000 |
| 4 | 2000 | 22500 | |

Optimal population sizes are clearly overestimated by using the number of excess unique schemata. For chromosome lengths of 360 and above, the optimal population sizes obtained in such a fashion are ridiculously large, because the optimal population size grows exponentially with the string length. Accounting for schemata of orders up to the string length is excessive. What really counts is that the GA processes efficiently building blocks, which typically have short defining lengths than the strings. The next study for estimating population sizes in GAs is based on building blocks sampling.

### A.1.3 The Population-Sizing Equation

The population-sizing equation is derived in [Goldberg92] by considering the probability of choosing the right schema between different competing schemata. The schemata are carried by different members of the population. A schema can be characterized by the mean of its performance and the associated mean variance. If all deceptive effects are neglected, the schema leading to the optimal solution will have the largest mean among competing schemata. In that case, we want to maximize the probability of selecting the individual that carries the schema with the largest mean performance. An error occurs when the observed mean performances of competing schemata do not reflect their actual respective performances. In GAs, the sample is the population. As the population increases, the variance of the mean of the schemata decreases, meaning that we can become more confident in our ability to choose better schemata as the population size increases. Applying statistical decision theories, Goldberg et al. ([Goldberg92]) derived an expression for the probability of making an error. This expression is referred to as the population-sizing equation. It depends, among other things, on the population size and the string length. One interpretation of the population-sizing equation is that, in order to keep the probability of making an error constant, the size of population should be a linear function of the string length, i.e., $m = O(l)$. Note that this result is valid for any number $K$ of symbols used in the coding, and is independent from the GA implementation.

It has been found empirically that the optimal population size for the One Laminate Wing Box problem is around 8 individuals. The length of the string for the One Laminate Wing Box problem is somewhere between 30 and 40, and the length of the string for the 18 Laminate Wing Box problem is comprised between 360 and 720. From $m = O(l)$, we deduce that a population size for the 18 Laminate Wing Box problem comprised between 96 and 144 individuals would

keep the schema sampling error rate at the same level as it was for the One Laminate Wing Box problem.

Another theory presented in [Thierens93] argues that, in order for the GA to have enough time to correctly mix building blocks before the population is uniform, the population size should vary exponentially with the string length. This is explained in greater detail in the next section on time to convergence.

### A.1.4 An Empirical Study on Optimal Population Size

In [Alander92], experiments are reported where a GA is used to find a given substring. Based on the minimization of the time to find the optimal string, averaged over 100 trials, the author observes that the optimal population size $m_o$ seems to satisfy the relation,

$$l \leq m_o \leq 2l. \tag{A.9}$$

Note that the above result has been established using binary coding and population sizes below 24 individuals. Nevertheless, if we apply Equation (A.11) to the single laminate problem ($30 \leq l \leq 40$), we obtain optimal population sizes between 30 and 80 individuals. For the 18 Laminate Wing Box problem ($360 \leq l \leq 720$), we obtain optimal population sizes between 360 and 1440.

All of the previous analytical results that help choose the population size recommend using much larger population sizes than the ones found experimentally on our composite design problems. However, these studies concern generic versions of the GA that are fundamentally different from the specialized GA devised in this work. As stated in [Goldberg92], "at small population sizes the GA makes many errors of decision, and the quality of convergence is largely left to chance or to the serial fix-up of flawed results through mutation or other serial injection of diversity". Precisely, the GA for stacking sequence design has efficient mechanisms to inject diversity in the population (stack-swap, decoupled mutation), which make the use of a small population the most efficient strategy. Among the previous studies, the one that yields the smallest population sizes is the prediction using expected allele coverage. It is interesting to note that the authors

argue in that study ([Tate93]) that high mutation rates and non-binary encodings can be beneficial. The GA devised in this work for stacking sequence design satisfies these conditions.

The population size is one of the factors that determine the rate of convergence of the GA. Models for the time to convergence of GAs are discussed next.

## *A.2 Time to Convergence*

In an attempt to predict the cost of the genetic search, we review some of the studies that relate the population size, the length of the chromosome and the time to convergence.

In [Goldberg91], the rate of growth of the best individual in the population is estimated as a result of selection only, that is, disruptions due to other genetic operators are neglected. A measure of time to convergence called takeover time $t_s$ is used, which is the number of generations after which the population is composed of all the same individuals but one. It is found that, when the selection procedure is linear ranking (which is the selection procedure of the GA for composite optimization implemented here), takeover time $t_s$ is proportional to $log\, m$,i.e., convergence occurs in $O(m\ log\, m)$ function evaluations. Note that this result is independent of the chromosome length, because recombination is neglected. However, if we use the population-sizing equation ([Goldberg92], see also previous Section) according to which $m = O(l)$, we obtain a dimensional relation according to which GAs can converge in $O(l\ log\, l)$ function evaluations. This suggests that if the population is properly sized to permit accurate decision making among schemata in the presence of noise, global results are predicted in times that are $O(l\ log\, l)$.

The population-sizing equation and the takeover time both neglect the effect of crossover, which is to mix building blocks. In [Thierens93], the notion of mixing of building blocks is introduced. A mixing time $t_x$ is calculated, which is the expected number of generations before the right building blocks are juxtaposed to create the global optimum. Calculations are performed on a simple problem that yields a conservative bound on the mixing time. Takeover time, which corresponds to convergence under the influence of selection only, and mixing time are then interrelated. If the takeover time $t_s$ is bigger than the mixing time $t_x$, the mixing process is

expected to have enough time to juxtapose all the building blocks in one single individual, the global optimum. The relation

$$t_s > ct_x, \tag{A.10}$$

where $c$ is a constant is interpreted in [Thierens93] as: the population size $m$ should grow exponentially with the number of building blocks in the string to permit sufficient building block mixing before convergence. The number of building blocks in the string is in general proportional to the length of the string, so, Equation (A.12) indicates that the population size should grow exponentially with the string length.

Another analysis of time to convergence is proposed in [Louis93]. The hamming distance is the number of digits that separates two chromosomes. For example, the hamming distance between 0001 and 0000 is 1, while the hamming distance between 0011 and 0000 is 2. In [Louis93], the average hamming distance in generation $t$, $h_t$, is considered. The average hamming distance is the sum of the hamming distances between every pair of individuals in the population, divided by the number of pairs of individuals. For a GA with proportionate selection (selection probability proportional to the objective function value), without mutation, and with a crossover that does not change the average hamming distance in the population, it is argued that

$$h_t = g^t \, h_0, \tag{A.11}$$

where $g$ is a coefficient depending on the function optimized (typically $g \approx 0.95, 0.96$). For a random initial population of large size with binary coding, $h_0 = l/2$. Thus, the time to convergence as predicted by that model under the form of average hamming distance is linearly proportional to the string length.

Figure A.1 has been reproduced from [Alander92]. It shows the "processing time" as a function of the population size. The "processing time" is the time until the global optimum is found, averaged over 100 runs. Results are given for different values of $c$, which is the string length ($l$ in our standard notation). It can be seen in Figure A.1 that the "processing time", which is another way of measuring the length of the search, is a linear function of the population size. Another interesting conclusion, that should be compared to the argument of Section 5.4.3, stems from Figure A.1: suppose we use a fixed amount of processing time and look at problems with increasing string length, then the optimal population size would decrease as the string length

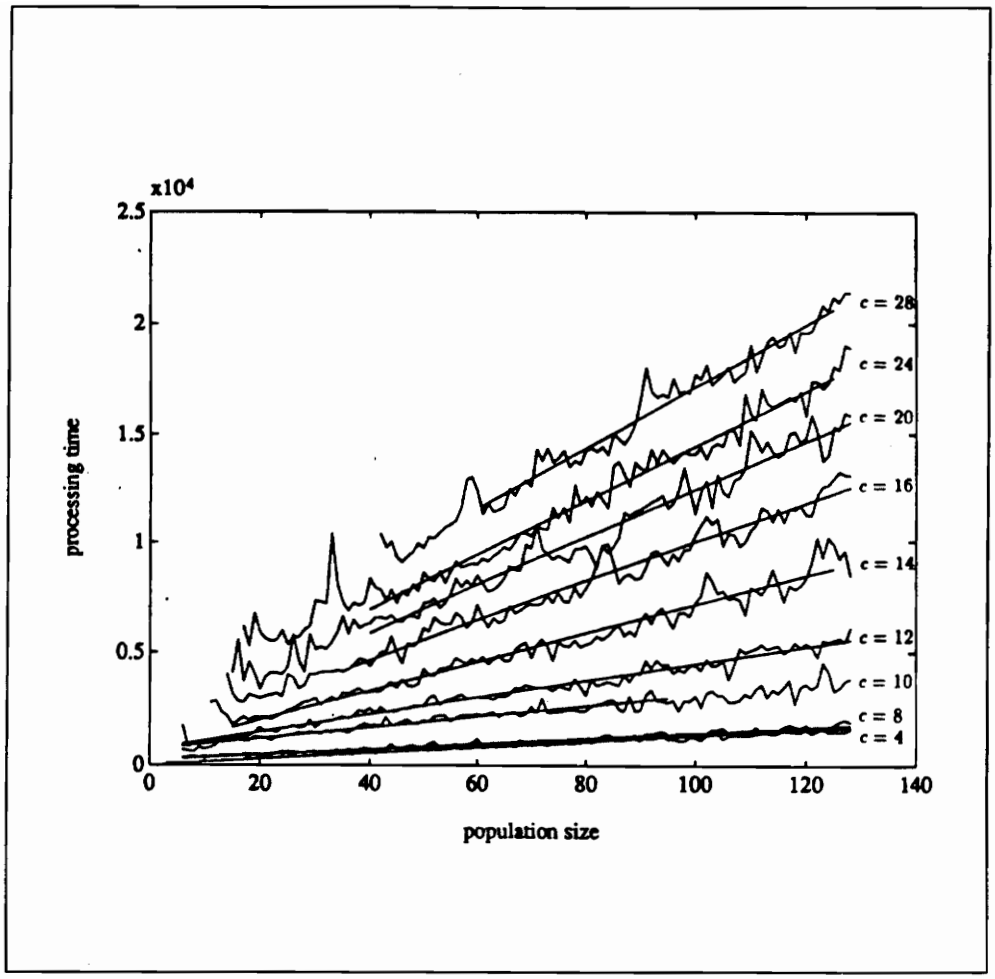



**Figure A.1. Processing time versus population size and string length $c$ (from [Alander92]).**

increases. For example, if we decide to use 0.5E+04 units of processing time, then the optimal population size (that uses up all of the computer resources) is around $m$ =140 for $c$ = 12, $m$ = 75 for $c$ = 14, $m$ = 45 for $c$ = 16, etc.. Of course, because the test problem solved in [Alander92] is easy, it does not make sense to use all of the 0.5E+04 units of processing time for $c$ = 12, 14, 16. However, on a difficult problem like the wing box-beam, a limit on processing time is rapidly reached and the above reasoning is of practical interest.

# Appendix B

# COMPOSITE WING BOX MODEL

Details of the wing box model used in Chapter 5 are given in this Appendix. We start with details of the finite element analysis procedure, and derivation of the internal element stiffness matrix . We then discuss how strains and loads in the structure are calculated from the displacements. Finally, we show how the analytical expressions of buckling due to in-plane normal loads and buckling due to in-plane shear loads are derived.

## *B.1 Finite Element Analysis*

### B.1.1 General Organization

| | |
|---|---|
| **PRE-PROCESSING** | - Definition of the mesh, of the material, of the boundary conditions.<br>- Calculation of the element stiffness matrix $[k^e]$. |
| **PROCESSING** | - Assembly, $[K] = \sum [k^e]$.<br>- Calculation of the global displacements $\{U\}$ by solving $[K]\{U\} = \{F\}$. |
| **POST-PROCESSING** | - Calculation of the strains $\varepsilon$ at each node of each element.<br>- Calculation of the principal strains $\varepsilon_{ij}$ in the material direction, at each node, in each layer, in each element.<br>- Calculation of the distributed loads $N_x$, $N_y$, $N_{xy}$ on each edge of each element. |

**Figure B.1. Finite Element Analysis.**

The organization of the finite element procedure is sketched out in Figure B.1. The mesh, the material and the boundary conditions are data here and were described in Chapter 5. The element stiffness matrices $[k]^e$ are calculated from the stacking sequence of each panel (cf. the next subsection). Assembly of the element stiffness matrices $[k]^e$ yields the global stiffness matrix $[K]$. The vector of nodal forces $\{F\}$ being given, the system

$$[K]\,\{U\} \;=\; \{F\} \tag{B.1}$$

is solved for the global displacements $\{U\}$ using Gaussian elimination [Kreiszig83].

## B.1.2 Element Stiffness Matrix

We now outline the calculation of the element stiffness matrix $[k]^e$ for the rectangular finite-element shown on Figure B.2.

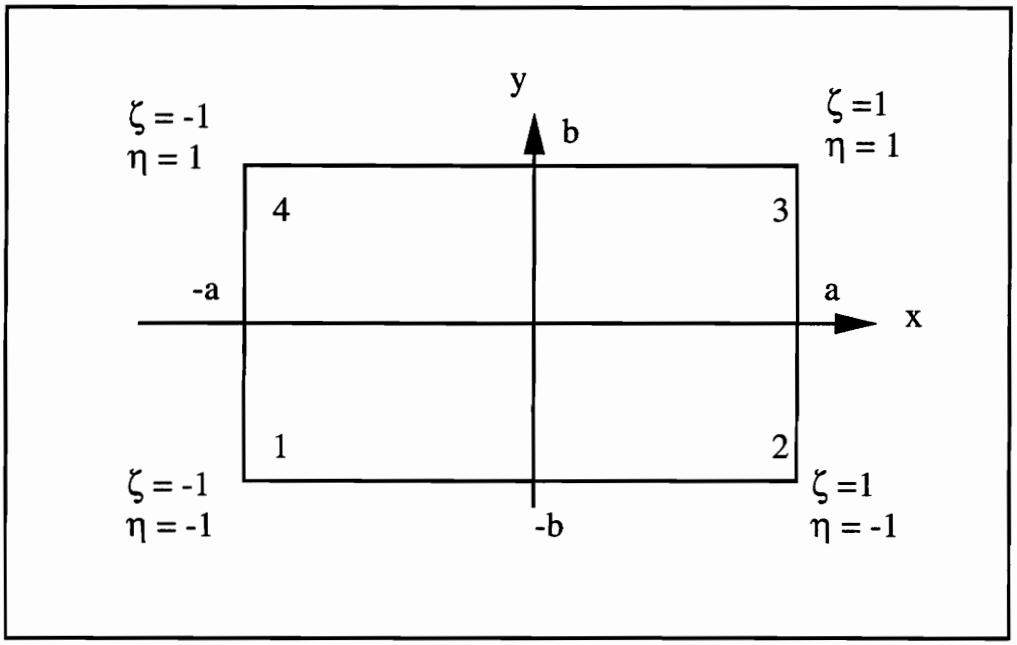



**Figure B.2. 8 degrees of freedom membrane rectangular element.**

We further asume that the composite plate under consideration is in a plane stress state, balanced and symmetric (no extensional-flexural coupling).

The principal of Minimum Total Potential Energy yields the following expression of the element stiffness matrix [Batoz90],

$$[k]^e = \int_V [B]^T[H][B]\, dV \ , \tag{B.2}$$

where $[B]$ is the strain-displacement matrix and $[H]$ is the transformed reduced stiffness matrix. $[B]$ is independent of through the thickness variables, and the integration of $[H]$ through the thickness yields the extensional stiffness matrix $[A]$ from the Classical Lamination Theory,

$$[k]^e = \int_S [B]^T[A][B]\, dx\, dy \ . \tag{B.3}$$

Because we have balanced and symmetric laminates, normal and shear stresses are decoupled, i.e, coefficients $A_{16}$ and $A_{26}$ are zero. The $[B]$ matrix can be decomposed as

$$[B] = [B_1]\,[B_2]\,[B_3]\,[B_4] \ , \tag{B.4}$$

where the $[B_i]$'s are 3x2 submatrices given as

$$[B_i] = \begin{pmatrix} \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{pmatrix} \ , \tag{B.5}$$

and the $N_i$'s are interpolation functions for the displacement field in the element. These are given as

$$N_i(x,y) = \frac{1}{4ab}(x_i + x)(y_i + y)\ \xi_i\eta_i \ , \quad i = 1, ..., 4, \tag{B.6}$$

where $i$ refers to the elemental node number as shown on Figure B.2. For this type of element, all calculations can be carried out analytically. Performing the matrix product yields

$$[B]^T[A][B] = \begin{pmatrix} [C_{11}] & \cdots & [C_{14}] \\ \cdots & \cdots & \cdots \\ C_{41} & \cdots & [C_{44}] \end{pmatrix}, \tag{B.7}$$

where,

$$[C_{ij}] = \begin{pmatrix} A_{11}\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + A_{66}\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} & A_{12}\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} + A_{66}\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} \\ A_{12}\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} + A_{66}\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} & A_{22}\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + A_{66}\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} \end{pmatrix}. \tag{B.8}$$

In order to integrate Equation (B.3) (using (B.6)), we need to calculate the following integrals:

$$\int_{-a}^{a}\int_{-b}^{b} \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\, dy\, dx = \frac{\xi_i\xi_j\eta_i\eta_j}{4ab} y_i\, y_j + \frac{b}{12a}\xi_i\xi_j\eta_i\eta_j, \tag{B.9}$$

$$\int_{-a}^{a}\int_{-b}^{b}\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\,dy\,dx \;=\; \frac{\xi_i\xi_j\eta_i\eta_j}{4ab}x_i\,x_j+\frac{a}{12b}\xi_i\xi_j\eta_i\eta_j, \tag{B.10}$$

$$\int_{-a}^{a}\int_{-b}^{b}\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y}\,dy\,dx \;=\; \frac{\xi_i\xi_j\eta_i\eta_j}{4ab}y_i\,x_j. \tag{B.11}$$

From Equations (B.3), (B.7), (B.8), and (B.9) to (B.11), the element stiffness for a plane stress rectangular finite element with 8 degrees of freedom can be written as,

$$[k]^e \;=\; \begin{pmatrix}[k_{11}] & \dots & [k_{14}]\\ \dots & \dots & \dots\\ [k_{41} & \dots & [k_{44}]\end{pmatrix}, \tag{B.12}$$

where the 2x2 $[k_{ij}]$ submatrices are expressed as,

$$[k_{ij}] \;=\; \frac{\xi_i\xi_j\eta_i\eta_j}{4ab}\begin{pmatrix} A_{11}\left(y_i\,y_j+\frac{b^2}{3}\right)+A_{66}\left(x_i\,x_j+\frac{a^2}{3}\right) & A_{12}\,y_i\,x_j+A_{66}\,x_i\,y_j\\ A_{12}\,y_j\,x_i+A_{66}\,x_j\,y_i & A_{22}\left(x_i\,x_j+\frac{a^2}{3}\right)+A_{66}\left(y_i\,y_j+\frac{b^2}{3}\right)\end{pmatrix}. \tag{B.13}$$

Equations (B.12) and (B.13) define the element stiffness matrix.

The validity of the element stiffness matrix was checked, first of all by recovering, from eqs (B.12) and (B.13), the well known expression of this stiffness matrix for isotropic material (see for instance [Reddy84]). Then, numerical checks were performed by comparing displacements with those obtained using the stiffness matrix from the program *RE_FLEX* ([Batoz90]).

## *B.2 Strains and Distributed In-plane Loads Calculations*

At each node of each element, the strain vector in the x-y global coordinate system $\{\varepsilon\} = [\varepsilon_x\ \varepsilon_y\ \varepsilon_{xy}]^T$ is found using

$$\{\varepsilon\} \;=\; [B]\{u\}, \tag{B.14}$$

where $[B]$ is given by Equations (B.4), (B.5) and (B.6), and $\{u\}$ is the displacement vector obtained by extracting the relevant displacements from the global displacement vector $\{U\}$. Next, strains in the principal material directions are calculated for each layer as

$$\begin{aligned}\varepsilon_1 &= cos^2\theta\varepsilon_x + sin^2\theta\varepsilon_y + cos\theta\ sin\theta\ \varepsilon_{xy},\\ \varepsilon_2 &= sin^2\theta\varepsilon_x + cos^2\theta\varepsilon_y - cos\theta\ sin\theta\ \varepsilon_{xy},\\ \gamma_{12} &= -2\cos\theta\,sin\theta\varepsilon_x + 2\cos\theta\,sin\theta\varepsilon_y + (cos^2\theta - sin^2\theta)\varepsilon_{xy},\end{aligned} \tag{B.15}$$

where index 1 corresponds to the fiber direction, index 2 stands for the direction perpendicular to the fibers and $\theta$ is the fiber direction in the layer.

From the Classical Lamination Theory, the vector of stress resultants $\{N\} = [N_x\ N_y\ N_{xy}]^T$ and the strain vector $\{\varepsilon\}$ are related by

$$\{N\} = [A]\{\varepsilon\}, \tag{B.16}$$

where $[A]$ is the extensional stiffness matrix.

Note that Equation (B.16) holds at each node of each element. The final values of $N_x, N_y$ and $N_{xy}$ are obtained by averaging their values at the four element nodes.

## *B.3 Buckling Models*

### BUCKLING DUE TO NORMAL IN-PLANE LOADS

For a symmetrically laminated balanced orthotropic laminate of dimensions $a$ and $b$ (see Figure 5.4), with simply supported boundaries under in-plane loads $N_x$, $N_y$ and $N_{xy}$, the differential equation governing buckling [Haftka92a] is

$$D_{11}\frac{\partial^4 w}{\partial x^4} + 2(D_{12} + 2D_{66})\frac{\partial^4 w}{\partial x^2 \partial y^2} + D_{22}\frac{\partial^4 w}{\partial y^4} = \lambda\left(N_x\frac{\partial^2 w}{\partial x^2} + N_y\frac{\partial^2 w}{\partial y^2} + 2N_{xy}\frac{\partial^2 w}{\partial x \partial y}\right). \tag{B.17}$$

In the above equation, $w$ stands for the out-of-plane displacement function, the $D_{ij}$'s are the flexural stiffness coefficients, and $\lambda$ is the buckling load factor. The product of the actual load by the buckling load factor $\lambda$ gives the load at which buckling first occurs.

For simply supported boundary conditions, we assume a transverse displacement pattern of the form,

$$w(x,y) = \sum_{n=1}^{N}\sum_{m=1}^{M} W_{nm}\, sin\frac{m\pi x}{a}\, sin\frac{n\pi y}{b}, \tag{B.18}$$

where the MxN possible values of m and n represent possible buckling modes that are characterized by a number of half-waves in the $x$ and $y$ directions. This form of displacements satisfies the

boundary conditions exactly. Substituting Equation (B.18) into Equation (B.17), and neglecting the shear loading $N_{xy}$ gives the following expression for the buckling load factor,

$$\frac{\lambda}{\pi^2} = \frac{D_{11}\left(\frac{m}{a}\right)^4 + 2(D_{12}+2D_{66})\left(\frac{m}{a}\right)^2\left(\frac{n}{b}\right)^2 + D_{22}\left(\frac{n}{b}\right)^4}{N_x\left(\frac{m}{a}\right)^2 + N_y\left(\frac{n}{b}\right)^2}. \tag{B.19}$$

The critical buckling load factor due to normal in-plane loads is then found by minimizing $\lambda$ over different possible conbinations of m and n,

$$\begin{aligned} \lambda_n &= min(\lambda), \\ \text{with } & m = 1,..,M, \\ \text{and } & n = 1,..,N. \end{aligned} \tag{B.20}$$

For the present implementation, a truncated series with $M = 5$ and $N = 5$ was used.

**BUCKLING DUE TO SHEAR IN-PLANE LOADS**

The previous model presented a closed-form expression of the critical buckling load $\lambda_n$due to normal in-plane loads. We now turn to plate instability under the effect of uniform in-plane shear load. The plate considered is assumed to be symmetric, balanced and orthotropic. Normal-shear, extensional, extensional-flexural and bending-twisting couplings are thus removed. Furthermore, the plate is considered of infinite length.

There are exact solutions available for plates infinite in length in the x direction [Whitney85]. These exact solutions provide the limiting case for long plates. Figure 5.5 shows the infinite strip under consideration. The strip is simply supported at the edges $y = \pm b/2$, i.e.,

$$w = M_y = -D_{12}\frac{\partial^2 w}{\partial x^2} - D_{22}\frac{\partial^2 w}{\partial y^2} = 0. \tag{B.21}$$

Whitney [Whitney85] suggests a solution in the form,

$$w = f(y)e^{2i\xi x/b}, \tag{B.22}$$

where $i^2 = -1$ and $\xi$ characterizes the length between successive buckling waves in the plate. Substituting this relationship into Equation (B.17) that describes buckling, with $N_x$ and $N_y$ set to 0 and $S = \lambda N_{xy}$, we obtain the following differential equation,

$$D_{11}\left(\frac{2\xi}{b}\right)^4 f - 2(D_{12}+2D_{66})\left(\frac{2\xi}{b}\right)^2\frac{d^2 f}{dy^2} + D_{22}\frac{d^4 f}{dy^4} - iS\left(\frac{2\xi}{b}\right)\frac{df}{dy} = 0. \tag{B.23}$$

The solution to Equation (B.23) can be written in the form,

$$f(y) = Ae^{2ir_1y/b} + Be^{2ir_2y/b} + Ce^{2ir_3y/b} + De^{2ir_4y/b}, \tag{B.24}$$

where $r_1, r_2, r_3$ and $r_4$ are roots of the characteristic equation

$$D_{22}r^4 + 2(D_{12} + 2D_{66})\xi^2 r^2 + D_{11}\xi^4 + 2S\xi r\left(\frac{b}{2}\right)^2 = 0. \tag{B.25}$$

The roots $r_1, r_2, r_3$ and $r_4$ may be real or complex. Combining Equations (B.22) and (B.24), the solution takes the form,

$$w = e^{2i\xi x/b}\left(Ae^{2ir_1y/b} + Be^{2ir_2y/b} + Ce^{2ir_3y/b} + De^{2ir_4y/b}\right). \tag{B.26}$$

Combining Equation (B.26) with the boundary conditions (B.21), we obtain four homogeneous equations with four unknowns $A, B, C, D$. In order to obtain a non zero solution, the determinant of the coefficient matrix must vanish. The solution is found to depend on $\xi$, and the critical buckling load corresponds to the value of $\xi$ which yields the lowest value of $S$. Solutions for $S_{critical}$ are presented in [Whitney85] for a complete range of $\Gamma$, where

$$\Gamma = \frac{\sqrt{D_{11}D_{22}}}{(D_{12} + 2D_{66})}. \tag{B.27}$$

For $1 \leq \Gamma \leq \infty$, the critical buckling load due to in-plane shear $\lambda_S$ is

$$\lambda_S = \frac{4\beta_1(D_{11}D_{22}^3)^{1/4}}{N_{xy}b^2}, \tag{B.28}$$

and, for $0 \leq \Gamma \leq 1$,

$$\lambda_S = \frac{4\beta_1\sqrt{D_{22}(D_{12} + 2D_{66})}}{N_{xy}b^2}, \tag{B.29}$$

where values for $\beta_1$ can be found in table 5.1.

Note that, in the wing box model, buckling loads are calculated for laminates of dimensions 24 in. by 24 in., even though the real dimensions of the plates are 46.5 in. by 29.4 in . This reflects the fact that the model used here does not include stiffeners.

# Appendix C
# WING BOX CONTINUOUS OPTIMIZATION

## *C.1 Continuous Optimization Problem Formulation*

For comparison purpose, a continuous optimization method was used to optimize the wing box. The algorithm used is a successive quadratic programming procedure with finite difference gradient implemented in the IMSL subroutine DNCONF ([IMSL89]), and is itself based on subroutine NLPQL ([Schittkowski86]). Because of the discreteness of the number of plies, stacking sequence problems are intrinsically discrete optimization problems that require transformations in order to be solved using continuous optimization. Here, the stacking sequence of each laminate was fixed, i.e., the $j$th laminate was described by the ply arrangement $[(\theta)_1^j/(\theta)_2^j/.../(\theta)_i^j/.../(\theta)_l^j]_s$, where the angles $\theta$ are in ($0^o$, $45^o$, $90^o$). The thicknesses $T_i^j$ of the $i$th stack of two plies in the $j$th laminate are taken as the continuous variables. Thus, there are $(l \times L)$ continuous design variables, where $l$ is half the number of stacks of two plies per laminate (because of symmetry) and $L$ is the number of different laminates. The continuous optimization problem is formulated as

| | |
|---|---|
| **Minimize** | $\sum_{j=1}^{L} \sum_{i=1}^{l} T_i^j$, the sum of the thicknesses of the upper and lower skin panels, |
| **by changing** | $T_i^j$ , |
| **subject to** | laminates are balanced and symmetric, |
| (strength) | $\lvert\varepsilon_{i\ 1}^j\rvert < \varepsilon_1^{ua}$, $\lvert\varepsilon_{i\ 2}^j\rvert < \varepsilon_2^{ua}$, $\lvert\gamma_{i\ 12}^j\rvert < \gamma_{12}^{ua}$, $i$=1,3, $j$=1,18, |
| (buckling upper skin) | $\lambda_{cs}^j \geq 1$, $\lambda_{cc}^j \geq 1$, $j$=1,9, |
| (buckling lower skin) | $\lambda_{cs}^j \geq 1$, $j$=10,18. |

Note that the 4-ply contiguity constraint was not included because there was no easy way to do so.

In the One Laminate Wing Box problem, only one laminate is considered ($L$=1) and we set the stacking sequence to $[45^o/90^o_2/0^o_2/45^o/90^o_2/0^o_2]_s$, i.e., there are 6 design variables. The stacking sequence is the same in the Two Laminate Wing Box problem, but there are two independent laminates, so the total number of design variables is 12. In the 18 Laminate Wing Box problem, the stacking sequences are $[45^o/90^o_2/0^o_2]_s$ for each of the 18 laminates, for a total of 18×3=54 design variables. All the above problems have 189 constraints (18 for buckling in the upper skin, 9 for buckling in the lower skin, and 18×3×3 for strains). Because the finite element does not account for flexural effects, the strains are the same for all plies of the same orientation in a laminate. Here for example, we only need to check strains in three layers ($0^o$, $90^o$, $+45^o$ or $-45^o$). In an attempt to make the problem easier for the optimizer, we dropped the strain constraints on $\varepsilon_2$ and $\gamma_{12}$ because they have been found to be inactive in the optimal region of the design space. These constraints were then checked for each optimum design.

The sequential quadratic programming algorithm has a low computational cost compared to the GA. However, it is prone to converge to local optima. To limit the risks of false convergence, the search was re-initiated at least 10 times from different starting points.

## *C.2 Rounding of Continuous Optimum Designs*

At the end of the continuous optimization process, the thicknesses $T_i^j$ obtained are not integer multiples of the basic ply thickness $t$. To make the design possible to manufacture, ply thicknesses need to be rounded to a multiple of the basic ply thickness. Rounding an optimum continuous design is difficult, because the design may become infeasible in the process. No strategy, not even rounding all the thicknesses up, is guaranteed to yield a design that does not fail. Figure C.1 for example shows how one constraint on strain and one constraint on buckling vary when a stack thickness is increased, starting from a feasible continuous optimum design. The example corresponds to the 18 Laminate Wing Box problem, the strain constraint is $(\varepsilon_1^{ua}/|\varepsilon_1| - 1)$ in a $0^o$ layer of panel 10, the buckling constraint is $\lambda_{cc}$ in panel 1, and the $0^o$ stack of the 10th panel has its thickness increased. It can be seen that the buckling constraint is violated as the stack thickness is increased.

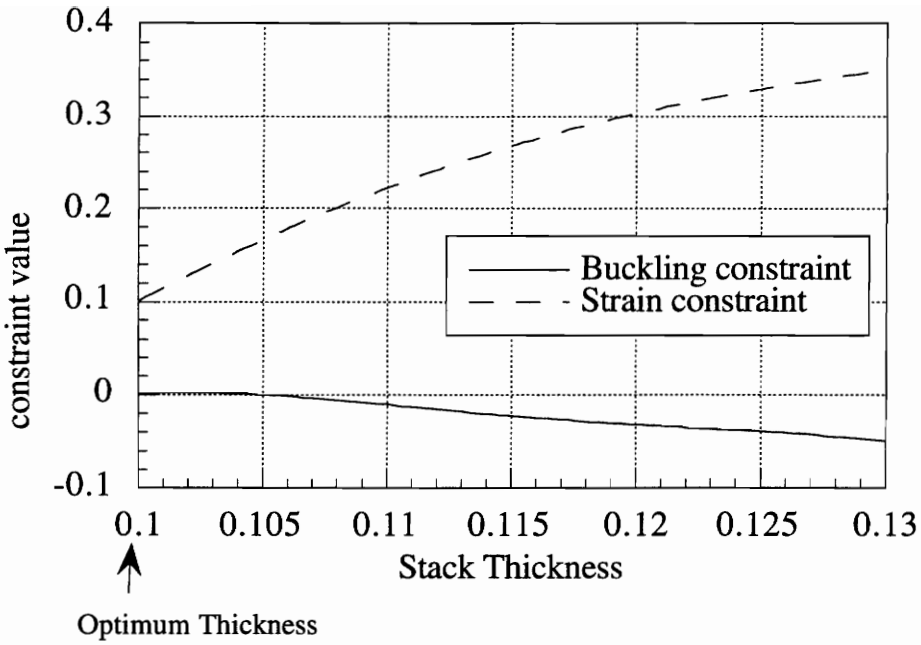



**Figure C.1. Constraint on $\varepsilon_1$ in a $0^o$ layer of panel 10 and constraint on $\lambda_{cc}$ in panel 1 versus thickness of the $0^o$ stack in panel 10, optimum 18 Laminate Wing Box design.**

Many similar phenomena occur in parallel during rounding of an optimum continuous design, making the operation difficult. Therefore, it is necessary to try various combinations of rounding the variables up and down. In the One and Two Laminate Wing Box problems, the number of combinations is $2^6 = 64$ and $2^{12} = 4096$, respectively. These numbers are small enough to make an exhaustive enumeration scheme possible. The enumeration scheme used in this work is sketched in Figure C.2.

For the 18 Laminate Wing Box problem however, there are $2^{54} \approx 1.8.10^{16}$ ways of rounding the design variables up or down, which is too large to try them all. Some heuristic rule is to be used to decide whether to round up or down some of the stack thicknesses, in a process (described below) that we call pre-rounding. Finally, an exhaustive enumeration takes place when no more than 14 thicknesses are left non-rounded ( in our case $2^{14} = 16384$ is still a reasonable number

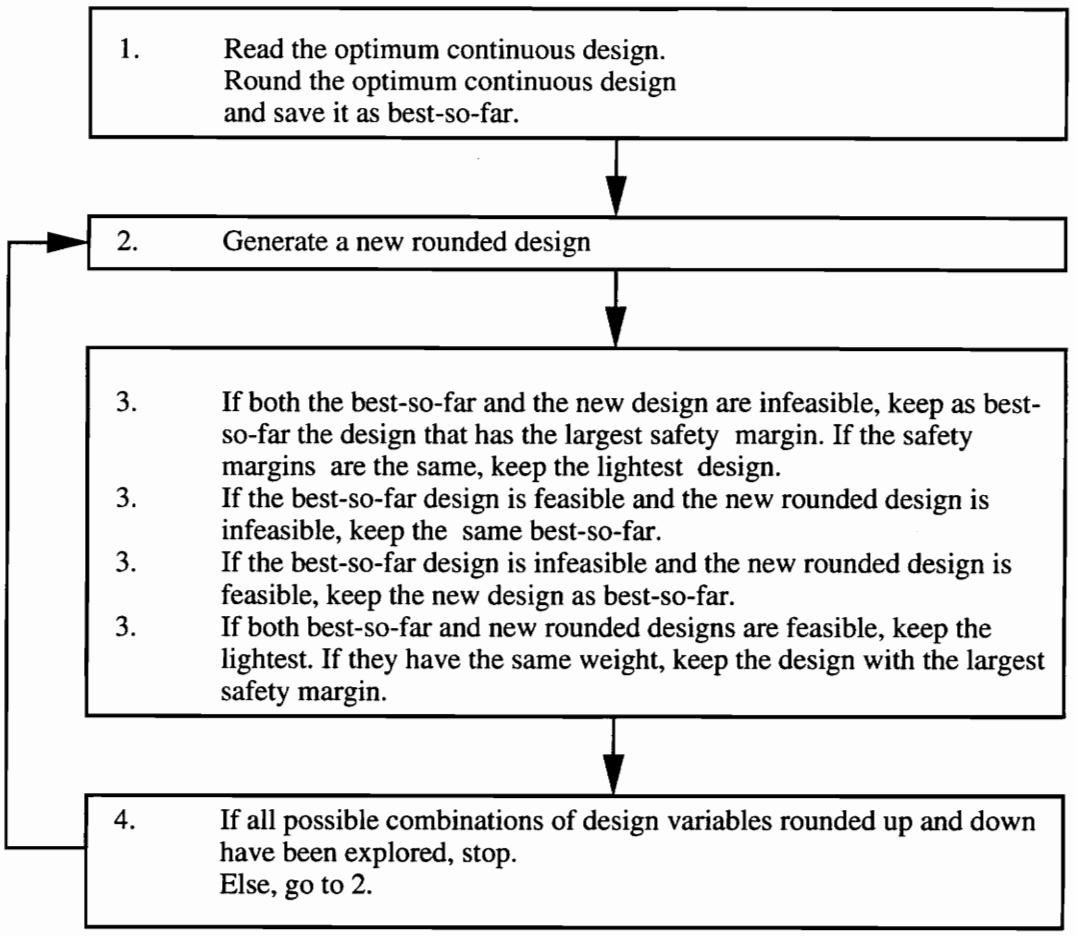



**Figure C.2. Flow chart of the enumeration procedure for rounding.**

of combinations to explore). Obtaining a rounded design that is still feasible is difficult for the 18 Laminate Wing Box problem. We tried the six different pre-rounding strategies that are listed below.

**P1** 40 of the stack thicknesses are fixed by rounding up the ones that correspond to more than $X_{up}$ percent of the laminate they belong to, and by rounding down the ones that correspond to less than $X_{down}$ percent of the laminate they belong to. Here values of $X_{up}$=68 percent and $X_{down}$=32 percent were chosen so as to be equidistant from 50 percent and have 40 design variables fixed. Values of $X_{up}$ and $X_{down}$ were chosen to let 40 stack thicknesses not rounded at the end of the pre-rounding step. Values of $X_{up}$ and $X_{down}$ are case dependent.

**P2** Same as P1, but any stack of plies that should have vanished in the process is set to the

minimum stack thickness, i.e., $2t$.

**P3** Same as P2, but the thicknesses of the $45^o$ stacks are always rounded up.

**P4** Same as P3, but the thicknesses of the $(90_2)^o$ stacks are always rounded up.

**P5** 40 of the stack thicknesses are fixed by rounding up the ones that correspond to more than $X_{up}$ percent of the laminate they belong to (no thickness is rounded down). Here, $X_{up}$=10 percent.

**P6** The 40 stack thicknesses that are the closest to their rounded values are scaled up or down.

Pre-rounding procedures P1 to P5 are based on the same principle: the layers that account for an important fraction of the laminate are rounded up, those that account for a small fraction are rounded down. P1 to P5 are increasingly conservative procedures in the sense that more and more stacks are rounded up. Procedure P6 is based on the alternative idea that the continuous optimum design should endure the smallest possible perturbation during pre-rounding. Unlike the other pre-rounding strategies, P5 yielded feasible designs in all of the experiments we performed. P5 is the pre-rounding strategy used in the results sections of this study.

# Appendix D

# LAMINATES AND THE STATIC BUILDING BLOCK HYPOTHESIS

## *D.1 Problem Description*

This appendix describes an experiment made to check some of the conclusions of the Static Building Block Hypothesis (cf. Section 2.3) when applied to laminates. Do the laminate problem and its associated coding satisfy the Static Building Block Hypothesis? Do existing theories on what make a problem GA-hard (Section 2.3.2) explain the differences in price of the genetic search noticed between different load cases?

Our attempt at answering these questions is based on a simplified version of the minimum thickness design problem of Chapter 4. We consider a simply supported plate subjected to normal in-plane loads as shown in Figure 4.1. The same four load cases as the ones of Chapter 4 are used: load case 1, $N_x = 13,000.$ lb/in and $N_y = 1,625.$ lb/in; load case 2, $N_x = 12,500.$ lb/in and $N_y = 3,125.$ lb/in; load case 3, $N_x = 9,800.$ lb/in and $N_y = 4,900.$ lb/in; and a multiple load case where the loadings [$N_x = 12,000.$ lb/in, $N_y = 1,500.$ lb/in], [$N_x = 10,800.$ lb/in, $N_y = 2,700.$ lb/in] and [$N_x = 9,000.$ lb/in, $N_y = 4,500.$ lb/in] are considered simultaneously. Recall from Section 4.4 that load case 1 is the easiest to optimize by GA, followed by load case 2, load case 3 and the multiple load case.

The problem is simplified here so that it is possible to sample all possible chromosomes and calculate schema average performances and variances. For this purpose, the basic ply thickness is increased to $t = 0.02$ in. (instead of 0.005 in.) and we do not consider empty layers (no $E$'s). As a result, the optimal chromosomes are only four digits long. It is then easy to sample the $3^4 = 81$ design points, and calculate average objective function values and variances for each of the $4 \times 3 = 12$ first order schemata. The objective function is calculated from Equation (4.7). The

different parameters of Equation (4.7) take on their default values of Chapter 4, except $P_c$ which is set to 1, so as to remove the ply-contiguity constraint. The contiguity constraint is neglected because the basic ply thickness has been increased with respect to Chapter 4.

## D.2 Results

Tables D.1 to D.4 give the first order schemata average objective functions and variances for the different load cases considered. The optimal design and its objective function are also described at the top of each table.

**Table D.1. First order schemata, load case 1**
**Optimal design:** $[\pm 45/0_6]_s$
**Objective function (O.F.) = 9.51. .**

| schemata | average O.F. | variance of O.F. |
|---|---|---|
| $0_2$*** | 14.84 | 3.82 |
| $\pm 45$*** | 16.32 | 9.34 |
| $90_2$*** | 16.95 | 7.50 |
| *$0_2$** | 14.69 | 4.99 |
| *$\pm 45$** | 16.43 | 8.11 |
| *$90_2$** | 16.99 | 7.00 |
| **$0_2$* | 14.36 | 5.70 |
| **$\pm 45$* | 16.65 | 6.80 |
| **$90_2$* | 17.09 | 6.11 |
| ***$0_2$ | 14.19 | 5.28 |
| ***$\pm 45$ | 16.71 | 6.84 |
| ***$90_2$ | 17.20 | 5.61 |

* : stands for any stack of two plies.

By looking at the Tables D.1 to D.4, it can be seen that in 75% of the cases the higher performance (lower objective function value) first order schemata contain the optimum design.

**Table D.2. First order schemata, load case 2**
**Optimal design: $[(\pm 45)_2/0_4]_s$**
**Objective function (O.F.) = 11.36. .**

| schemata | average O.F. | variance of O.F. |
|---|---|---|
| $0_2$*** | 16.03 | 6.13 |
| $\pm 45$*** | 15.67 | 7.03 |
| $90_2$*** | 16.42 | 6.62 |
| *$0_2$** | 15.54 | 8.66 |
| *$\pm 45$** | 16.08 | 5.01 |
| *$90_2$** | 16.50 | 5.93 |
| **$0_2$* | 14.89 | 7.91 |
| **$\pm 45$* | 16.45 | 5.51 |
| **$90_2$* | 16.79 | 4.51 |
| ***$0_2$ | 14.49 | 6.45 |
| ***$\pm 45$ | 16.51 | 5.34 |
| ***$90_2$ | 17.12 | 4.33 |

* : stands for any stack of two plies.

This shows that laminates essentially satisfy the Static Building Block Hypothesis. In most of the cases, low order building blocks exist that can be discriminated and recombined according to their average performance, building blocks that perform well leading to the optimum. Of course, this is not always true, otherwise optimizing laminates would degenerate to a trivial discrete hill-climbing.

It can also be seen on Tables D.1 to D.4 that, on the average, the third and the multiple load cases that are the most difficult for the GA to optimize are the load cases for which the first order schemata exhibit the largest variances in performance. This confirms that large variance within schemata makes a problem GA-hard (cf. Section 2.3.2).

Load case 3 (Table D.3) clearly shows features of a deceptive problem: the schema $*\pm 45**$ contains the optimum but does not have the best average performance, while the schema $*90_2**$ does not contain the optimum but has the best average performance. In addition, both schemata have a low variance (5.59 and 2.86, respectively, as compared to the variance of $*0_2**$ which is 19.60), which means that the misleading attraction towards $*90_2**$ is a phenomenon that occurs frequently in the populations.

**Table D.3. First order schemata, load case 3**
**Optimal design:** $[90_2/\pm 45/0_4]_s$
**Objective function (O.F.) = 9.01. .**

| schemata | average O.F. | variance of O.F. |
|---|---|---|
| $0_2$*** | 17.52 | 13.69 |
| $\pm 45$*** | 13.78 | 1.82 |
| $90_2$*** | 13.60 | 4.57 |
| *$0_2$** | 16.11 | 19.60 |
| *$\pm 45$** | 14.69 | 5.59 |
| *$90_2$** | 14.10 | 2.86 |
| **$0_2$* | 14.92 | 17.78 |
| **$\pm 45$* | 15.04 | 8.72 |
| **$90_2$* | 14.93 | 3.75 |
| ***$0_2$ | 14.25 | 13.50 |
| ***$\pm 45$ | 15.05 | 8.93 |
| ***$90_2$ | 15.60 | 6.86 |

* : stands for any stack of two plies.

For load case 2, the misleading first order schema $*0_2**$ has a larger variance than $*\pm 45**$ and $*90_2**$. Thus the misleading effect will not occur during the genetic search as often as if the variance of the misleading schema was small. Consequently, load case 2 is easier to solve by GAs than load case 3 is.

Even though schema average performances and variances are clearly related to the difficulty a GA experiences when solving a problem, they do not explain why, in our example, load case 1 was easier than load case 2 and why load case 3 was easier than the multiple load case. For these last cases, different explanations need to be sought. For example, the large number of active constraints in the multiple load case is thought to be a factor of GA-hardness.

**Table D.4. First order schemata, multiple load case**
**Optimal design:** $[90_2/0_6]_s$
**Objective function (O.F.) = 10.35. .**

| schemata | average O.F. | variance of O.F. |
|---|---|---|
| $0_2$*** | 17.21 | 10.63 |
| $\pm 45$*** | 16.04 | 5.64 |
| $90_2$*** | 16.04 | 7.15 |
| *$0_2$** | 16.48 | 13.52 |
| *$\pm 45$** | 16.62 | 4.87 |
| *$90_2$** | 16.19 | 5.87 |
| **$0_2$* | 15.59 | 12.07 |
| **$\pm 45$* | 16.91 | 6.83 |
| **$90_2$* | 16.80 | 4.36 |
| ***$0_2$ | 15.01 | 9.01 |
| ***$\pm 45$ | 17.03 | 6.97 |
| ***$90_2$ | 17.27 | 5.17 |

* : stands for any stack of two plies.

# VITA

The author was born in Sartrouville, France, on the fourth of September 1968. In 1989, he spent six month working for a printing factory in Berlin, Germany. In 1990, he graduated from the Université de Technologie de Compiègne, France, with a Bachelor of Science degree in Material Engineering. He is now working towards his Doctor of Philosophy degree in Aerospace Engineering at Virginia Polytechnic Institute and State University.