Multiobjective Optimization of Trusses using Genetic Algorithms

Carlos A. Coello Coello† Alan D. Christiansen‡

† Engineering Design Centre, University of Plymouth, Plymouth, PL4 8AA, UK*
‡ Department of Computer Science, Tulane University, New Orleans, LA 70118, USA

Abstract: *In this paper we propose the use of the genetic algorithm (GA) as a tool to solve multiobjective optimization problems in structures. Using the concept of min-max optimum, a new GA-based multiobjective optimization technique is proposed and two truss design problems are solved using it. The results produced by this new approach are compared to those produced by other mathematical programming techniques and GA-based approaches, proving that this technique generates better trade-offs and that the genetic algorithm can be used as a reliable numerical optimization tool.*

Keywords*:* genetic algorithms, multiobjective optimization, vector optimization, multicriteria optimization, structural optimization, truss optimization

1 Introduction

In most real-world problems, several goals must be satisfied simultaneously in order to obtain an optimal solution. The multiple objectives are typically conflicting and non-commensurable, and must be satisfied simultaneously. For example, we might want to be able to minimize the total weight of a truss while minimizing its maximum deflection and maximizing its maximum allowable stress. The common approach in this sort of problem is to choose one objective (for example, the weight of the structure) and incorporate the other objectives as constraints. This approach has the disadvantage of limiting the choices available to the designer, making the optimization process a rather difficult task.

Another common approach is the combination of all the objectives into a single objective function. This technique has the drawback of modelling the original problem in an inadequate manner, generating solutions that will require a further sensitivity analysis to become reasonably useful to the designer.

A more appropriate approach to deal with multiple objectives is to use techniques that were originally designed for that purpose in the field of Operations Research. Work in that area started a century ago, and many approaches have been refined and commonly applied in economics and control theory.

This paper addresses the importance of multiobjective structural optimization and reviews some of the basic concepts and part of the most relevant work in this area. Also, we discuss the suitability of a heuristic technique inspired by the mechanics of natural selection (the genetic algorithm, or GA) to solve multiobjective optimization problems. We also introduce a new method, based on the concept of min-max optimum. The new method is compared with other GA-based multiobjective optimization methods and some mathematical programming techniques. We show that the new method is capable of finding better trade-offs among the competing objectives. Our approach is tested on two well-known truss optimization problems. We perform these tests with a computer program called MOSES, which was developed by the authors to experiment with new and existing multiobjective optimization algorithms.

*This work was done while the author was at Tulane University.

2 Previous Work on Multiobjective Structural Optimization

The first application of multiobjective optimization concepts in structural mechanics appeared in a 1968 paper by Krokosky [35]. In this early paper, Krokosky devised a method that accommodates the designer's preference of the design requirements. In his approach, Krokosky required the most desirable and least desirable values of each of the decision variables and the different levels of desirability of various combinations of such decision variables provided through a ranking matrix. Krokosky adopted a random search technique to find the best trade-off correlating the different objectives (i.e., desired values of the decision variables) in terms of the a priori chosen design parameters. This technique happens to be computationally inefficient and impractical because it is sometimes very difficult, or even impossible, to get all the requirements in terms of one quantity [45]. Krokosky's approach was later applied to the optimal design of sandwich panels [55]. Rao et al. [45] used utility theory to overcome some of the drawbacks of Krokosky's approach when dealing with optimal material choice problems.

Stadler [58] noted the scientific application of the concept of Pareto optimality to problems of natural structural shapes. He used this concept to compute optimal initial shapes of uniform shallow arches. Rao et al. [47, 48] showed significant work in multiobjective structural optimization with uncertain parameters. Rao was one of the first to point out the importance of incorporating concepts from game theory into structural optimization, and used several mathematical programming techniques such as global criterion, utility function, goal programming, goal attainment, bounded objective function and lexicographic methods, to solve multiobjective structural optimization problems. A more extended analysis of the use of game theory as a design tool may be found in [61], although no applications are included.

Carmichael [7] proposed the use of the ε-constraint method to the multiobjective optimum design of trusses. A more formal treatment of the subject wass given by Koski and Silvennoinen [33], who proposed a numerical method that generates the Pareto optimal set of an isostatic truss, based on the exact solution of bicriterion subproblems. An extension to this work was published later [34] by the same authors. In this latter paper the authors proposed the scalarization of the original multiobjective optimization problem by using norm methods and the reduction of the dimension of the problem by a partial weighting method. They used trusses (both isostatic and hyperstatic) to illustrate these methods.

Fu and Frangopol [19] formulated a multiobjective structural optimization technique based on structural reliability theory. This approach was illustrated solving a hyperstatic truss and a frame system. El-Sayed et al. [12] used linear goal-programming techniques with successive linearizations to solve nonlinear structural optimization problems. Their application was a three bar truss with uncertainties in both load magnitude and direction.

Hajela and Shih [29] presented a slight variation of the global criterion approach, used in conjunction with a branch and bound algorithm, to solve multiobjective optimization problems that involve a mix of continuous, discrete and integer design variables. A simply supported I-beam and a composite laminated beam were included to exemplify their approach. Another variant of the global criterion approach was suggested by Saravanos and Chamis [52] to design lightweight, low-cost composite structures. Tseng and Lu [60] applied goal programming, compromise programming and the surrogate trade-off method to the selection of system parameters and large scale structural design optimization problems.

Grandhi et al. [25] presented a reliability-based decision criterion approach for multiobjective optimization of structures with a large number of design variables and constraints. Lounis and Cohn [38] used a projected Lagrangian algorithm to transform the multiobjective optimization of prestressed concrete structures into a single-objective optimization problem.

Finally, the book by Eschenauer et al. [13] is a very valuable guide to some of the most relevant work in multiobjective design optimization in the last few years. Good surveys on multiobjective structural optimization may be found in Stadler [59], Duckstein [11] and Coello [10].

3 Basic Concepts

Multiobjective optimization (also called multicriteria optimization, multiperformance or vector optimization) can be defined as the problem of finding [41]:

a vector of decision variables which satisfies constraints and optimizes a vector function whose elements

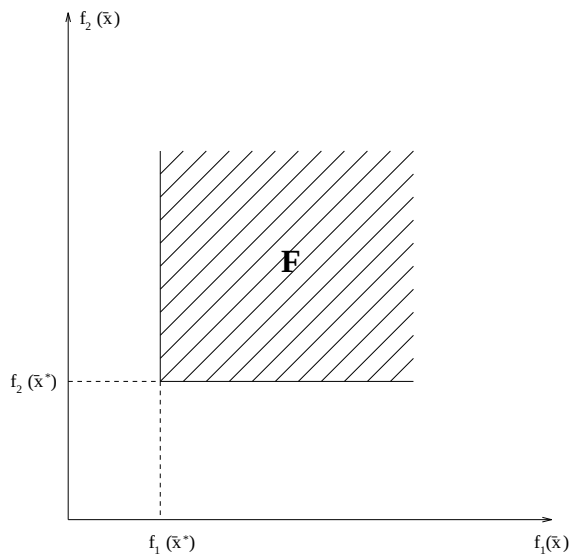

Figure 1: Ideal solution in which all our functions have their minimum at a common point.

represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the designer.

Formally, we can state it as follows:

Find the vector $\bar{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which will satisfy the m inequality constraints:

$$g_i(\bar{x}) \geq 0 \quad i = 1, 2, \ldots, m \tag{1}$$

the p equality constraints

$$h_i(\bar{x}) = 0 \quad i = 1, 2, \ldots, p \tag{2}$$

and optimize the vector function

$$\bar{f}(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}), \ldots, f_k(\bar{x})]^T \tag{3}$$

where $\bar{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables.

In other words, we wish to determine from among the set of all numbers which satisfy (1) and (2) the particular set $x_1^*, x_2^*, \ldots, x_k^*$ which yields the optimum values of all the objective functions.

The constraints given by (1) and (2) define the *feasible region* $\mathcal{F}$ and any point $\bar{x}$ in $\mathcal{F}$ defines a *feasible solution*. The vector function $\bar{f}(\bar{x})$ is a function which maps the set $\mathcal{F}$ in the set $\mathcal{X}$ which represents all possible values of the objective functions. The k components of the vector $\bar{f}(\bar{x})$ represent the *non-commensurable* criteria[1] which must be considered. The constraints $g_i(\bar{x})$ and $h_i(\bar{x})$ represent the restriction imposed on the decision variables. The vector $\bar{x}^*$ will be reserved to denote the optimal solutions (normally there will be more than one).

The problem is that the meaning of *optimum* is not well defined in this context, since we rarely have an $\bar{x}^*$ such that for all $i = 1, 2, \ldots, k$

$$\bigwedge_{x \in \mathcal{F}} \quad (f_i(\bar{x}^*) \leq f_i(\bar{x})) \tag{4}$$

[1] *Non-commensurable* means that the values of the objective functions are expressed in different units.

If this was the case, then $\bar{x}^*$ would be a desirable solution, but we normally never have a situation like this, in which all the $f_i(\bar{x})$ have a minimum in $\mathcal{F}$ at a common point x^*. An example of this ideal situation is shown in Figure 1. However, since this situation rarely happens in real-world problems, then we have to establish a certain criteria to determine what would be considered as an "optimal" solution.

3.1 Pareto Optimum

The concept of *Pareto optimum* was formulated by Vilfredo Pareto in 1896 [42], and constitutes by itself the origin of research in multiobjective optimization. We say that a point $\bar{x}^* \in \mathcal{F}$ is *Pareto optimal* if for every $\bar{x} \in \mathcal{F}$ either,

$$\bigwedge_{i \in I} (f_i(\bar{x}) = f_i(\bar{x}^*)) \tag{5}$$

or, there is at least one $i \in I$ such that

$$f_i(\bar{x}) > f_i(\bar{x}^*) \tag{6}$$

In words, this definition says that $\bar{x}^*$ is Pareto optimal if there exists no feasible vector $\bar{x}$ which would decrease some criterion without causing a simultaneous increase in at least one criterion. Unfortunately, the Pareto optimum almost always gives not a single solution, but rather a set of solutions called *non-inferior* or *non-dominated* solutions.

3.2 Pareto Front

The minima in the Pareto sense are going to be in the boundary of the design region, or in the locus of the tangent points of the objective functions. This region is called the *Pareto Front*. In general, it is not easy to find an analytical expression of the line or surface that contains these points, and the normal procedure is to compute the points $\mathcal{F}^k$ and their corresponding $f(\mathcal{F}^k)$. When we have a sufficient amount of these, we may proceed to take the final decision.

3.3 Min-Max Optimum

The idea of stating the *min-max optimum* and applying it to multiobjective optimization problems, was taken from game theory, which deals with solving conflicting situations. The min-max approach to a linear model was proposed by Jutler [40] and Solich [40]. It has been further developed by Osyczka [39], Rao [46] and Tseng and Lu [60].

The min-max optimum compares relative deviations from the separately attainable minima. Consider the ith objective function for which the relative deviation can be calculated from

$$z_i^{'}(\bar{x}) = \frac{|f_i(\bar{x}) - f_i^0|}{|f_i^0|} \tag{7}$$

or from

$$z_i^{''}(\bar{x}) = \frac{|f_i(\bar{x}) - f_i^0)|}{|f_i(\bar{x})|} \tag{8}$$

It should be clear that for (7) and (8) we have to assume that for every $i \in I$ and for every $\bar{x} \in \mathcal{F}$, $f_i(\bar{x}) \neq 0$.

If all the objective functions are going to be minimized, then equation (7) defines function relative increments, whereas if all of them are going to be maximized, it defines relative decrements. Equation (8) works conversely.

Let $\bar{z}(\bar{x}) = [z_1(\bar{x}), \ldots, z_i(\bar{x}), \ldots, z_k(\bar{x})]^T$ be a vector of the relative increments which are defined in $\mathcal{R}^k$. The components of the vector $z(\bar{x})$ will be evaluated from the formula

$$\forall_{i \in I}(z_i(\bar{x})) = max\ \{z_i^{'}(\bar{x}), z_i^{''}(\bar{x})\} \tag{9}$$

Now we define the min-max optimum as follows [40]:
A point $\bar{x}^* \in \mathcal{F}$ is min-max optimal, if for every $\bar{x} \in \mathcal{F}$ the following recurrence formula is satisfied:
Step 1:

$$v_1(\bar{x}^*) = \min_{x \in \mathcal{F}} \max_{i \in I} \{z_i(\bar{x})\} \tag{10}$$

and then $I_i = \{i_1\}$, where i_1 is the index for which the value of $z_i(\bar{x})$ is maximal.
If there is a set of solutions $x_1 \subset \mathcal{F}$ which satisfies Step 1, then
Step 2:

$$v_2(\bar{x}^*) = \min_{x \in x_1} \max_{i \in I,\, i \notin I_1} \{z_i(\bar{x})\} \tag{11}$$

and then $I_2 = \{i_1, i_2\}$, where i_2 is the index for which the value of $z_i(x)$ in this step is maximal.
If there is a set of solutions $x_{r-1} \subset \mathcal{F}$ which satisfies step $r-1$ then
Step r:

$$v_r(\bar{x}^*) = \min_{x \in x_{r-1}} \max_{i \in I,\, i \notin I_{r-1}} \{z_i(\bar{x})\} \tag{12}$$

and then $I_r = \{I_{r-1}, i_r\}$, where i_r is the index for which the value of $z_i(\bar{x})$ in the rth step is maximal.
If there is a set of solutions $x_{k-1} \subset \mathcal{F}$ which satisfies Step $k-1$, then
Step k:

$$v_k(\bar{x}^*) = \min_{\bar{x} \in x_{k-1}} z_i(\bar{x}) \max_{i \in I,\, i \notin I_{k-1}} \quad \text{for i} \in \text{I and i} \notin \text{I}_{\text{k}-1} \tag{13}$$

where $v_1(\bar{x}^*), \ldots, v_k(\bar{x})$ is the set of optimal values of fractional deviations ordered non-increasingly.

This optimum can be described in words as follows. Knowing the extremes of the objective functions which can be obtained by solving the optimization problems for each criterion separately, the desirable solution is the one which gives the smallest values of the relative increments of all the objective functions.

The point $\bar{x}^* \in \mathcal{F}$ which satisfies the equations of Steps 1 and 2 may be called the best compromise solution considering all the criteria simultaneously and on equal terms of importance.

4 Mathematical Programming Techniques

Multiobjective optimization is a very well known topic in Operations Research, and many techniques have been developed to deal with such problems. The first step in developing a mathematical programming technique to deal with multiobjective optimization problems is to be able to identify, given a set of feasible solutions to the problem, which of them are Pareto optimal. After that, it will be desirable to agree upon a further concept of absolute optimality in this context, in case the designer desires a single final solution. For the scope of this paper, we will adopt the concept of min-max optimum for that purpose.

Osyczka provides in his book [40] an algorithm based on the contact theorem [37], which can identify Pareto optimal solutions from a given set of feasible solutions. This algorithm was implemented by the authors of this paper and incorporated in MOSES together with another one that finds the optimum in the min-max sense [10].

4.1 Monte Carlo Methods

We also implemented the two Monte Carlo methods used by Osyczka [40] to find the min-max optimum. These methods are called *exploratory* because a point is generated by means of a rule which disregards the results previously obtained. In particular, the Monte Carlo method picks up a certain number of points at random over the estimated range of all the variables of the problem. This is done formally by obtaining the randomly selected value for x_i from the following formula

$$x_i = x_i^a + \delta_i(x_i^b - x_i^a) \quad \text{for } i = 1, 2, \ldots, n \tag{14}$$

where x_i^a is the estimated or given lower limit for x_i, x_i^b is the estimated or given upper limit for x_i, and δ_i is a random number between zero and one. We employed the same random number generator used by the genetic algorithm to implement the FORTRAN function RANF of the original program.

If we want to generate the values of variables for l^a points, we start by generating random numbers δ_i for each point, and then use equation (14) to obtain the values of the variables x_i. After that, we test each generated point for violation and discard it if it is not a feasible solution. If the point is in the feasible region, we evaluate the objective function for that point. The best result is taken as the minimum, and a new set of random numbers is generated for each of l^a points.

The two Monte Carlo methods described by Osyczka [40] to find the min-max optimum were incorporated in MOSES. In the first of them, the search space is explored twice, first searching for the ideal vector $\bar{f}^0$ and then searching for the min-max optimum. In the second method, the space of variables is explored only once, and the Pareto set is generated while searching for the ideal vector $\bar{f}^0$. Then, this set analyzed to check which solution is the min-max optimum.

There are several trade-offs between these two methods. For example, the second method uses less CPU time than the first, because the space of variables is explored only once, but it also requires much more memory since the whole Pareto set has to be stored. Obviously, the designer normally wants to analyze the entire Pareto set in order to take a decision, but as we mentioned before, this set could be too large and the computational resources available could be insufficient for that sake. Osyczka recommends the reduction of this set by introducing constraints of the form

$f_i(\bar{x}) \leq f_i^0$ for $i = 1, 2, \ldots, k$

where values of f_i^0 are chosen by the designer.

The second method should be preferred for problems with a large number of constraints and for discrete programming problems, because in those cases we expect to have a small Pareto set. The main advantage of exploratory methods in general is their flexibility, since they can be applied both to linear and non-linear programming problems. However, they are normally recommended only for cases where a few decision variables are handled because otherwise, they could take too long to find a reasonable good solution.

4.2 Osyczka's Multicriterion Optimization System

This system was developed at the Technical University of Cracow, and its FORTRAN implementation is provided in Osyczka's book [40]. A C translation of that code was incorporated into MOSES, and its contents is summarized next.

Osyczka's system contains several multiobjective optimization methods:

(1) Min-max method : Equation (9) is used to determine the elements of the vector $\bar{z}(\bar{x})$.

(2) Global criterion method : In this method, the equation:

$$f(\bar{x}) = \sum_{i=1}^{k} \left(\frac{f_i^0 - f_i(\bar{x})}{f_i^0} \right)^p \tag{15}$$

is used as the global function. For this formula Boychuk and Ovchinnikov [6] have suggested $p = 1$, and Salukvadze [51] has suggested $p = 2$, but other values of p can also be used. Obviously, the results will differ greatly depending on the value of p chosen. Thus, the selection of the best p is an issue in this method, and it could also be the case that any p could produce an unacceptable solution. We assumed $p = 2$ for our experiments.

(3) Weighting min-max method : This is a combination of the weighting method and the min-max approach that can find the Pareto set of solutions for both convex and non-convex problems. The equation

$$\forall_{i \in I}(z_i(\bar{x}) = max\{w_i z_i^{'}(\bar{x}), w_i z_i^{''}(\bar{x})\}) \tag{16}$$

is used to determine the elements of vector $\bar{z}(\bar{x})$.

(4) Pure weighting method : The equation

$$min \sum_{i=1}^{k} w_i f_i(\bar{x}) \tag{17}$$

is used to determine a preferred solution, where $w_i \geq 0$ are the weighting coefficients representing the relative importance of the objectives. It is usually assumed that

$$\sum_{i=1}^{k} w_i = 1 \tag{18}$$

(5) Normalized weighting method : $\bar{f}(\bar{x})$ is used in equation (4.2).

Since all these methods require the ideal vector, the user is given the choice of providing it, or letting the system to find it automatically. For this purpose, the system includes a single criterion optimization technique:

(i) The flexible tolerance (FT) method : Is a sequential method in which a point is established on the basis of the previously obtained results. Based on this information, the method will know where the minimum is likely to be so that the appropriate search direction may be established. Typical sequential methods, even when they are more efficient and more highly developed than exploratory methods, tend to be designed to solve only continuous convex problems. However, this particular method can deal with non-linear models [30]. A detailed explanation of this algorithm and its implementation may be found in [10].

5 Multiobjective Optimization using GAs

The notion of genetic search in a multicriteria problem dates back to the late 60s, in which Rosenberg's [50] study contained a suggestion that would have led to multicriteria optimization if he had carried it out as presented. His suggestion was to use multiple *properties* (nearness to some specified chemical composition) in his simulation of the genetics and chemistry of a population of single-celled organisms. Since his actual implementation contained only one single property, the multiobjective approach could not be shown in his work, but it was a starting point for researchers interested in this topic.

Genetic algorithms require scalar fitness information to work, which means that when approaching multicriteria problems, we need to perform a scalarization of the objective vectors. One problem is that it is not always possible to derive a global criterion based on the formulation of the problem. In the absence of information, objectives tend to be given equivalent importance, and when we have some understanding of the problem, we can combine them according to the information available, probably assigning more importance to some objectives. Optimizing a combination of the objectives has the advantage of producing a single compromise solution, requiring no further interaction with the decision maker [17]. The problem is, that if the optimal solution cannot be accepted, either because the function used excluded aspects of the problem which were unknown prior to optimization or because we chose an inappropriate setting of the coefficients of the combining function, additional runs may be required until a suitable solution is found. Some of the main approaches proposed in the literature will be summarized in the remainder of this section.

5.1 VEGA

David Schaffer [53] extended Grefenstette's GENESIS program [26] to include multiple objective functions. Schaffer's approach was to use an extension of the Simple Genetic Algorithm (SGA) that he called the Vector Evaluated Genetic Algorithm (VEGA), and that differed of the first only in the way in which selection was performed. This operator was modified so that at each generation a number of sub-populations was generated by performing proportional selection according to each objective function in turn. Thus, for a problem with k objectives, k sub-populations of size N/k each would be generated, assuming a total population size of N. These sub-populations would be shuffled together to obtain a new population of size N, on which the GA would apply the crossover and mutation operators in the usual way. This process is illustrated in Figure 2 (taken from Schaffer [53]).

Schaffer realized that the solutions generated by his system were non-inferior in a local sense, because their non-inferiority is limited to the current population, and while a locally dominated individual is also globally dominated, the converse is not necessarily true [53]. An individual who is not dominated in one generation may become dominated by an individual who emerges in a later generation. Also, he noted that the so-called

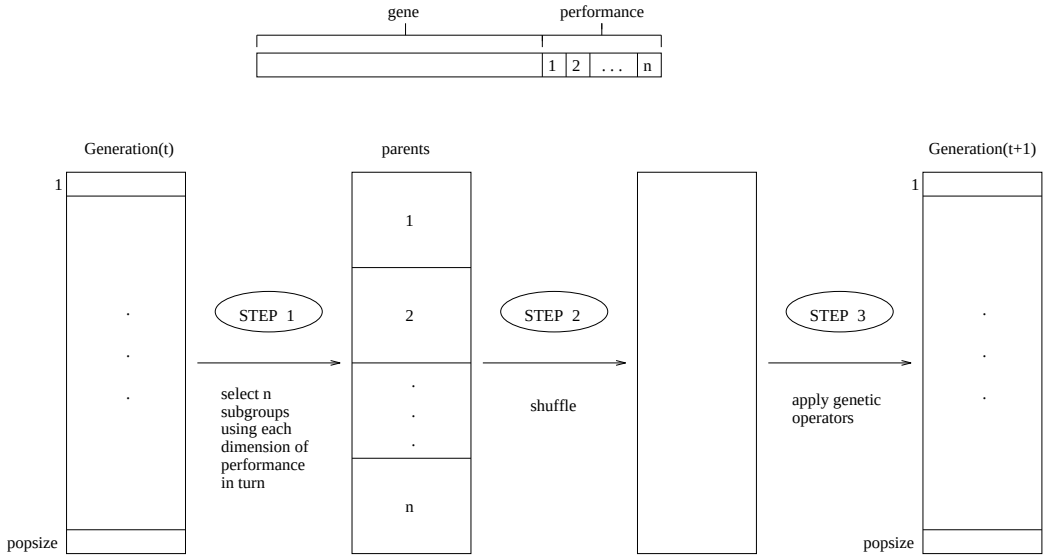

Figure 2: Schematic of VEGA selection.

"speciation" problem could arise from his approach (i.e., we could have the evolution of "species" within the population which excel on different aspects of performance).

This problem arises because this technique selects individuals who excel in one dimension of performance, without looking at the other dimensions. The potential danger doing that is that we could have individuals with "middling" performance in all dimensions, which could be very useful for compromise solutions, but that will not survive under this selection scheme, since they are not in the extreme for any dimension of performance (i.e., they do not produce the best value for any objective function, but only moderately good values for all of them). Speciation is undesirable because it is opposed to our goal of finding a compromise solution.

Schaffer suggested some heuristics to deal with this problem. For example, to use a heuristic selection preference approach for non-dominated individuals in each generation, to protect our "middling" chromosomes. Also, crossbreeding among the "species" could be encouraged by adding some mate selection heuristics instead of using the random mate selection of the traditional GA. Although Schaffer reported some success, Richardson et al. [49] noted that the shuffling and merging of all the sub-populations corresponds to averaging the fitness components associated with each of the objectives. Since Schaffer used proportional fitness assignment, these were in turn proportional to the objectives themselves [17]. Therefore, the resulting expected fitness corresponded to a linear combination of the objectives where the weights depended on the distribution of the population at each generation [49]. As a consequence, different non-dominated individuals were generally assigned different fitness values. This problem becomes more severe when we have a concave trade-off surface because points in concave regions of the trade-off surface cannot be found by optimizing a linear combination of the objectives, no matter what set of weights we use.

5.2 Lexicographic ordering

The basic idea of this technique is that the designer ranks the objectives in order of importance. The optimum solution is then found by minimizing the objective functions, starting with the most important one and proceeding according to the order of importance of the objectives [47]. Fourman [18] suggested a selection scheme based on lexicographic ordering. In a first version of his algorithm, objectives were assigned different priorities by the user and each pair of individuals were compared according to the objective with the highest priority. If this resulted in a tie, the objective with the second highest priority was used, and so on.

A second version of this algorithm, reported to work surprisingly well, consisted of randomly selecting the objective to be used in each comparison. As in VEGA, this corresponds to averaging fitness across fitness

components, each component being weighted by the probability of each objective being chosen to decide each tournament [17]. However, the use of pairwise comparisons makes an important difference with respect to VEGA, since in this case scale information is ignored. Therefore, the population may be able to see as convex a concave trade-off surface, depending on its current distribution, and on the problem itself.

5.3 Weighted Sum

Hajela and Lin [28] included the weights of each objective in the chromosome, and promoted their diversity in the population through fitness sharing. Their goal was to be able to simultaneously generate a family of Pareto optimal designs corresponding to different weighting coefficients in a single run of the GA. Besides using sharing, Hajela and Lin used a vector evaluated approach based on VEGA to achieve their goal. They proposed the use of a utility function of the form:

$$\bar{U} = \sum_{i=1}^{l} W_i \frac{F_i}{F_i^*} \tag{19}$$

where F_i^* are the scaling parameters for the objective criterion, l is the number of objective functions, and W_i are the weighting factors for each objective function F_i.

This approach uses a sharing function of the form:

$$\phi(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{sh}}\right)^{\alpha}, & d_{ij} < \sigma_{sh} \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

where normally $\alpha = 1$, d_{ij} is a metric indicative of the distance between designs i and j, and σ_{sh} is the sharing parameter which controls the extent of sharing allowed (Hajela and Lin [28] chose a number between 0.01 and 0.1 for σ_{sh}. The fitness of a design i is then modified as:

$$f_{s_i} = \frac{f_i}{\sum_{j=1}^{M} \phi(d_{ij})} \tag{21}$$

where M is the number of designs located in vicinity of the i-th design.

Under Hajela's representation, weight combinations are incorporated into the chromosomic string, and a single number represents not the weight itself, but a combination of weights. For example, the number 4 (under floating point representation) could represent the vector $X_w = (0.4, 0.6)$ for a problem with two objective functions. Then, sharing is done on the weights.

Finally, a mating restriction mechanism was imposed, to avoid members within a radius σ_{mat} to cross. The value of $\sigma_{mat} = 0.15$ was suggested by Hajela and Lin in their paper [28].

5.4 Multiple Objective Genetic Algorithm

Fonseca and Fleming [16] have proposed a scheme in which the rank of a certain individual corresponds to the number of chromosomes in the current population by which it is dominated. Consider, for example, an individual x_i at generation t, which is dominated by $p_i^{(t)}$ individuals in the current generation. Its current position in the individuals' rank can be given by [16]:

$$rank(x_i, t) = 1 + p_i^{(t)} \tag{22}$$

All non-dominated individuals are assigned rank 1, while dominated ones are penalized according to the population density of the corresponding region of the trade-off surface.

Fitness assignment is performed in the following way [16]:

1. Sort population according to rank.
2. Assign fitness to individuals by interpolating from the best (rank 1) to the worst (rank $n^* \leq N$) in the way proposed by Goldberg [21], according to some function, usually linear, but not necessarily.

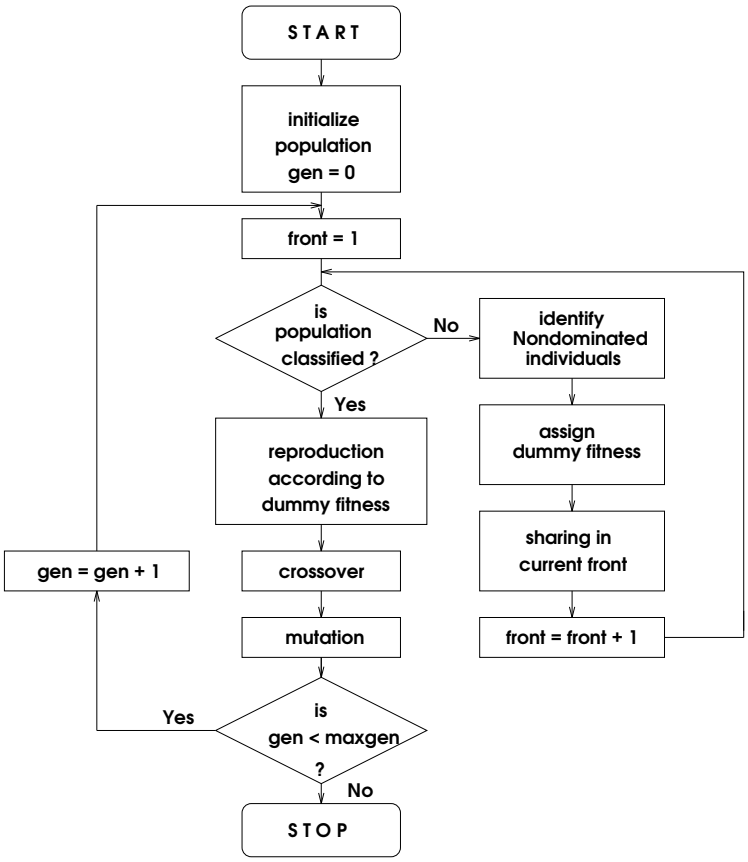

Figure 3: Flowchart of the Nondominated Sorting Genetic Algorithm (NSGA).

3. Average the fitnesses of individuals with the same rank, so that all of them will be sampled at the same rate. This procedure keeps the global population fitness constant while maintaining appropriate selective pressure, as defined by the function used.

As Goldberg and Deb [22] point out, this type of blocked fitness assignment is likely to produce a large selection pressure that might produce premature convergence. To avoid that, Fonseca and Fleming used a niche-formation method to distribute the population over the Pareto-optimal region, but instead of performing sharing on the parameter values, they have used sharing on objective function values [57]. This maintains diversity in the objective function values, but may not maintain diversity in the parameter set, which is an important issue for a decision maker. Furthermore, this approach may not be able to find multiple solutions in problems where different Pareto-optimal points correspond to the same objective function value.

5.5 Non-dominated Sorting Genetic Algorithm

The Non-dominated Sorting Genetic Algorithm (NSGA) was proposed by Srinivas and Deb [56], and is based on several layers of classifications of the individuals. Before the selection is performed, the population is ranked on the basis of nondomination: all nondominated individuals are classified into one category (with a dummy fitness value, which is proportional to the population size, to provide an equal reproductive potential for these individuals). To maintain the diversity of the population, these classified individuals are shared with their dummy fitness values. Then this group of classified individuals is ignored and another layer of nondominated individuals is considered. The process continues until all individuals in the population are classified. A stochastic remainder proportionate selection was used for this approach. Since individuals in the first front have the maximum fitness

value, they always get more copies than the rest of the population. This allows to search for nondominated regions, and results in quick convergence of the population toward such regions. Sharing, by its part, helps to distribute it over this region. The efficiency of NSGA lies in the way multiple objectives are reduced to a dummy fitness function using a nondominated sorting procedure. With this approach, any number of objectives can be solved [57], and both maximimization and minimization problems can be handled. Figure 3 (taken from Srinivas and Deb [57]) shows the general flow chart of this approach.

5.6 Niched Pareto GA

Horn and Nafpliotis [31] proposed a tournament selection scheme based on Pareto dominance. Instead of limiting the comparison to two individuals, a number of other individuals in the population was used to help determine dominance. When both competitors were either dominated or non-dominated (i.e., there was a tie), the result of the tournament was decided through fitness sharing [23]. Population sizes considerably larger than usual were used so that the noise of the selection method could be tolerated by the emerging niches in the population [17].

The pseudocode for Pareto domination tournaments assuming that all of the objectives are to be maximized is presented below [31]. S is an array of the N individuals in the current population, *random_pop_index* is an array holding the N indices of S, in a random order, and t_{dom} is the size of the comparison set.

```
function selection /* Returns an individual from the current population S */
begin
    shuffle(random_pop_index); /* Re-randomize random index array */
    candidate_1 = random_pop_index[1];
    candidate_2 = random_pop_index[2];
    candidate_1_dominated = false;
    candidate_2_dominated = false;
    for comparison_set_index = 3 to t_dom + 3 do
    /* Select t_dom individuals randomly from S */
        begin
            comparison_individual = random_pop_index[comparison_set_index];
            if S[comparison_individual] dominates S[candidate_1]
                then candidate_1_dominated = true;
            if S[comparison_individual] dominates S[candidate_2]
                then candidate_2_dominated = true;
        end /* end for loop */
    if ( candidate_1_dominated AND ¬ candidate_2_dominated )
        then return candidate_2;
    else if ( ¬ candidate_1_dominated AND candidate_2_dominated )
        then return candidate_1;
    else
        do sharing;

end
```

Scaling of the objectives determines the convexity of the trade-off surface, so that if we use a non-linear rescaling, the objective values may convert a concave surface into a convex one, and vice-versa. Pareto-ranking is blind to the convexity of the trade-off surface, but this does not mean that it always precludes speciation [17], since this can still occur if certain regions of the trade-off region are simply easier to find than others. However, Pareto-ranking eliminates sensitivity to the possible non-convexity of the trade-off surface, and also it encourages the production of compromise solutions.

It should be noticed that even when Pareto-based ranking correctly assigns all non-dominated individuals the same fitness, does not guarantee that the Pareto set be uniformly sampled, since finite populations will tend to

converge to only one optimum when several equivalent optima are present, due to stochastic errors in the selection process [17]. This phenomenon, which is known as *genetic drift*, has been observed in both natural and artificial evolution, and can also occur in Pareto-based GA optimization [17].

Goldberg and Richardson [23] proposed the used of fitness sharing to prevent *genetic drift* and to promote the sampling of the whole Pareto set by the population. Fonseca and Fleming [16] implemented fitness sharing in the objective domain and provided theory for estimating the necessary niche sizes based on the properties of the Pareto set. Horn and Nafpliotis [31] also arrived at a form of fitness sharing in the objective domain, and suggested the use of a metric combining both the objective and the decision variable domains, leading to what they called *nested sharing*.

6 A New GA-based Approach Based on a Weighted Min-Max Strategy

This is really a variant of Hajela's idea, in which a few changes were introduced by the authors [10]:

1. The initial population is generated in such a way that all their individuals constitute feasible solutions. This can be ensured by checking that none of the constraints is violated by the solution vector encoded by the corresponding chromosome.

2. The user should provide a vector of weights, which are used to spawn as many processes as weight combinations are provided (normally this number will be reasonably small). Each process is really a separate genetic algorithm in which the given weight combination is used in conjunction with a min-max approach to generate a single solution. Notice that in this case the weights do not have to be encoded in the chromosome as in Hajela's approach.

3. After the n processes are terminated (n=number of weight combinations provided by the user), a final file is generated containing the Pareto set, which is formed by picking up the best solution from each of the processes spawned in the previous step.

4. Since this approach requires knowing the ideal vector, the user is given the choice to provide such values directly (in case he/she knows them) or to use another genetic algorithm to generate it. This additional program works in a similar manner, spawning k processes (k=number of objective functions), where each process corresponds to a genetic algorithm responsible for a single objective function. When all the processes terminate, there will be a file containing the ideal vector, which turns out to be simply the best values produced by each one of the spawned processes.

5. The crossover and mutation operators were modified to ensure that they produced only feasible solutions. Whenever a child encodes an infeasible solution, it is replaced by one of its parents.

6. Notice that the Pareto solutions produced by this method are guaranteed to be feasible, as opposed to the other GA-based methods in which there could be convergence towards a non-feasible solution.

6.1 The GA optimizer for single-objective problems

Using the GA itself as an optimizer for single-objective problems is a controversial topic, mainly because the difficulties found to adjust its parameters (i.e., population size, maximum number of generations, mutation and crossover rate) [27]. Since one of the goals of this work is to be able to produce a reliable design optimization system, this is a natural problem to face. In practice, GA parameters are empirically adjusted in a trial and error process that could take quite a long time in some cases.

For several months, we experimented with a very simple methodology, explained below, for a variety of engineering design optimization problems. The results that we obtained led us to think that it was a reasonable choice to use in MOSES. The method is the following:

- Choose a certain value for the random number seed and make it a constant.

- Make constants for the population size and the maximum number of generations (we normally use 100 chromosomes and 50 generations, respectively).
- Loop the mutation and crossover rates from 0.1 to 0.9 at increments of 0.1 (this is actually a nested loop). This implies that 81 runs are necessary. In each step of the loop, the population is not reinitialized.
- For each run, update 2 files. One contains only the final costs, and the other has a summary that includes, besides the cost, the corresponding values of the design parameters and the mutation and crossover rates used.
- When the whole process ends, the file with the costs is sorted in ascending order, and the smallest value is searched for in the other file, returning the corresponding design parameters as the final answer.

So far, we have found much better results using floating point representation with this methodology, and our results show that this is a trend in numerical optimization problems [10]. This approach is actually a dynamic adjustment of parameters, because the population is initialized only once in the process, so that the individuals' fitness continues improving while changing the crossover and mutation rates. Notice that even when we could know the crossover and mutation rates produced the best answer, running the GA once with those parameters will not necessary generate the exact same answer. The reason is that the population at the moment of finding the best result could have been recombined and improved several times, being quite different of the random initial population of a simple GA. This procedure has some resemblance with Eshelman's CHC Adaptive Search Algorithm [14], but in our case we do not use any re-feeding of the population through high mutation values when it has stabilized, nor a highly disruptive recombinator operator that produces offspring that are maximally different from both parents. Our approach uses a conventional two-point crossover and it exhibits its best behavior with a floating point representation in numerical optimization problems.

7 Structural Optimization using Genetic Algorithms

Goldberg and Samtani [24] appear to have first suggested the use of GAs for structural optimization. They considered the use of a GA to optimize a 10-bar plane truss. Jenkins [32] used a straightforward implementation of Goldberg's SGA (Simple Genetic Algorithm) [21] to optimize a trussed-beam roof structure, a three-bar truss and a thin-walled coss-section.

Hajela [29] analyzed the potential of GAs as function optimizers in the context of structural optimization. He discussed encoding, optimal population size, selection, crossover and mutation over binary alphabets, making an important distinction between random search and genetic search. His FORTRAN implementation of a GA was applied to problems with nonconvex search spaces: a two-beam grillage structure, a two-element thin-walled cantilever torsional rod subjected to sinusiodal excitation and the dynamic response of a 10-bar plane truss.

Rajeev and Khrisnamoorthy [44] used the GA for discrete optimization of generalized trusses. Schoenauer and Xanthakis [54] presented a general method of handling constraints in genetic optimization, based on the *Behavioral Memory* paradigm. Instead of requiring the problem-dependent design of either repair operators (projection onto the feasible region) or penalty functions (weighted sum of constraint violations and the objective function), they sampled the feasible region by evolving from an initial random population, successively applying a series of different fitness functions which embodied constraint satisfaction. Only in the final step was the optimization restricted to the feasible region. The success of the whole process was highly dependent on the genetic diversity maintained during the first steps, ensuring a uniform sampling of the feasible region. They applied this scheme to test problems of truss structure optimization: a 10-bar (2D) and a 25-bar (3D) truss. Sharing and restricted mating were used to ensure genetic diversity in these applications.

Lin and Hajela [36] described a design optimization tool based on genetic search which inspired the development of MOSES [10]. This system, called EVOLVE, was able to handle mixes of integer, discrete and continuous design variables. It had automatic encoding/decoding facilities, automatic constraint handling, sharing to prevent convergence of all candidate designs to a single optimum, it varied the granularity of the representation to increase or decrease the precision with which a design space is represented, and used an special directed crossover operator that identifies significant bit positions on a string constraining the crossover to such bit locations.

Another important paper by Hajela and Lin [28] constitutes one of the very few attempts to achieve multiobjective structural optimization using GAs. The goal of the researchers in this work was to generate the Pareto set with a single run of the GA, and a utility function with sharing is used for that sake. A statically loaded 10-bar truss was used to exemplify their approach.

Adeli and Cheng [1] used a GA to minimize the total weight of a space truss subject to stress, displacement and fabricational (availability of cross-sectional areas) constraints. A quadratic penalty function was used to transform this constrained problem into an unconstrained one, and the fitness function was re-scaled because the GA always maximizes and this was a minimization problem. Three space trusses were used to illustrate their approach: a 12-bar truss, a 25-bar truss and a 72-bar truss. In a further paper by the same authors [2], a hybrid GA that integrated the penalty function method with the primal-dual method was proposed. This approach is based on sequential minimization of the Lagrangian method, and eliminated the difficulties of the unpredictability of the penalty function coefficient. Adeli and Kumar [4] proposed a distributed GA for optimization of large structures on a cluster of workstations connected via a local area network (LAN). The GA used a centralized population model in which the master process had global knowledge about the search process, which resulted in a faster convergence toward the optimal solution. A penalty function method and the augmented Lagrangian method were used again to eliminate the original constraints of the problem. A 17-member truss and a 50-story megastructure (848-element space truss) were solved using this approach. In a further paper, Adeli and Kumar [3] also used a GA for structural optimization of large scale structures on massively parallel supercomputers.

Rajan [43] used a GA to design the size, shape and topology of space structures. Discrete and continuous values were used to define the cross-sectional areas of the members. The nodal locations were treated as continuous design variables and the hybrid shape-optimization methodology, previously used with continuum structures, was adapted to handle skeletal structures. Element connectivity and boundary conditions were treated as Boolean design variables in the context of topology design. Rajan used a penalty function as the fitness, and exception handling was considered to deal with unstable structures, absence of deformations in the structure and zero force members. Also, in an effort to avoid recomputing the fitness function, a history of each chromosome was kept so that when duplicates appeared, it was not necessary to recompute its fitness. The examples used in Rajan's paper include a 6-node truss and a 14-node truss.

8 Examples

To introduce our new GA-based multiobjective optimization approach, we will use two truss design problems that are commonly referenced in the literature. On each of these examples, three objectives will be considered: minimize weight, maximum displacement and stress of the structure using the cross-sectional area of each element as the design variables. Such objectives are conflicting in nature, because if we want to reduce the displacement and the stress that an element supports, we have to increase the cross-sectional area, consequently increasing the weight of the structure. These objectives are also non-commensurable, because whereas stress and weight usually have large values, maximum allowable displacement is in general a small value.

9 Example 1 : Design of a 25-bar space truss

Consider the 25-bar space truss taken from Rajeev and Khrisnamoorthy [44] shown in Figure 4. The problem is to find the cross-sectional area of each member of this truss, such that we minimize its weight, the displacement of each free node, and the stress that each member has to support.

Loading conditions are given in Table 1, member groupings are given in Table 2, and node coordinates are given in Table 3. The assumed data are: modulus of elasticity, $E = 1 \times 10^4$ ksi, $\rho = 0.10$ lb/in^3; $\sigma_a = \pm 40$ ksi, $u_a = \pm 0.35$ in.

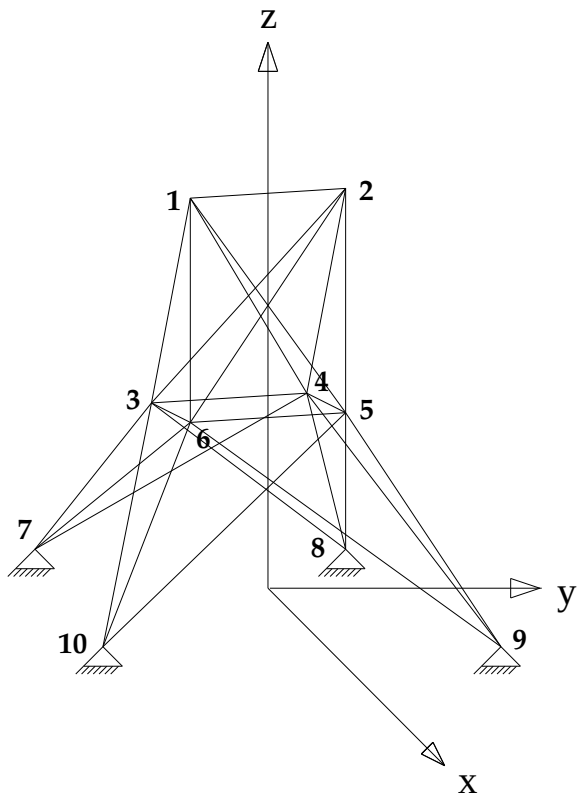

Figure 4: 25-bar space truss used for the first example.

Node	Fx (lbs)	Fy (lbs)	Fz (lbs)
1	1000	-10000	-10000
2	0	-10000	-10000
3	500	0	0
6	600	0	0

Table 1: Loading conditions for the 25-bar space truss shown in Figure 4.

Group Number	Members
1	1-2
2	1-4, 2-3, 1-5, 2-6
3	2-5, 2-4, 1-3, 1-6
4	3-6, 4-5
5	3-4, 5-6
6	3-10, 6-7, 4-9, 5-8
7	3-8, 4-7, 6-9, 5-10
8	3-7, 4-8, 5-9, 6-10

Table 2: Group membership for the 25-bar space truss shown in Figure 4.

Node	X	Y	Z
1	-37.50	0.00	200.00
2	37.50	0.00	200.00
3	-37.50	37.50	100.00
4	37.50	37.50	100.00
5	37.50	-37.50	100.00
6	-37.50	-37.50	100.00
7	-100.00	100.00	0.00
8	100.00	100.00	0.00
9	100.00	-100.00	0.00
10	-100.00	-100.00	0.00

Table 3: Coordinates of the joints of the 25-bar space truss shown in Figure 4.

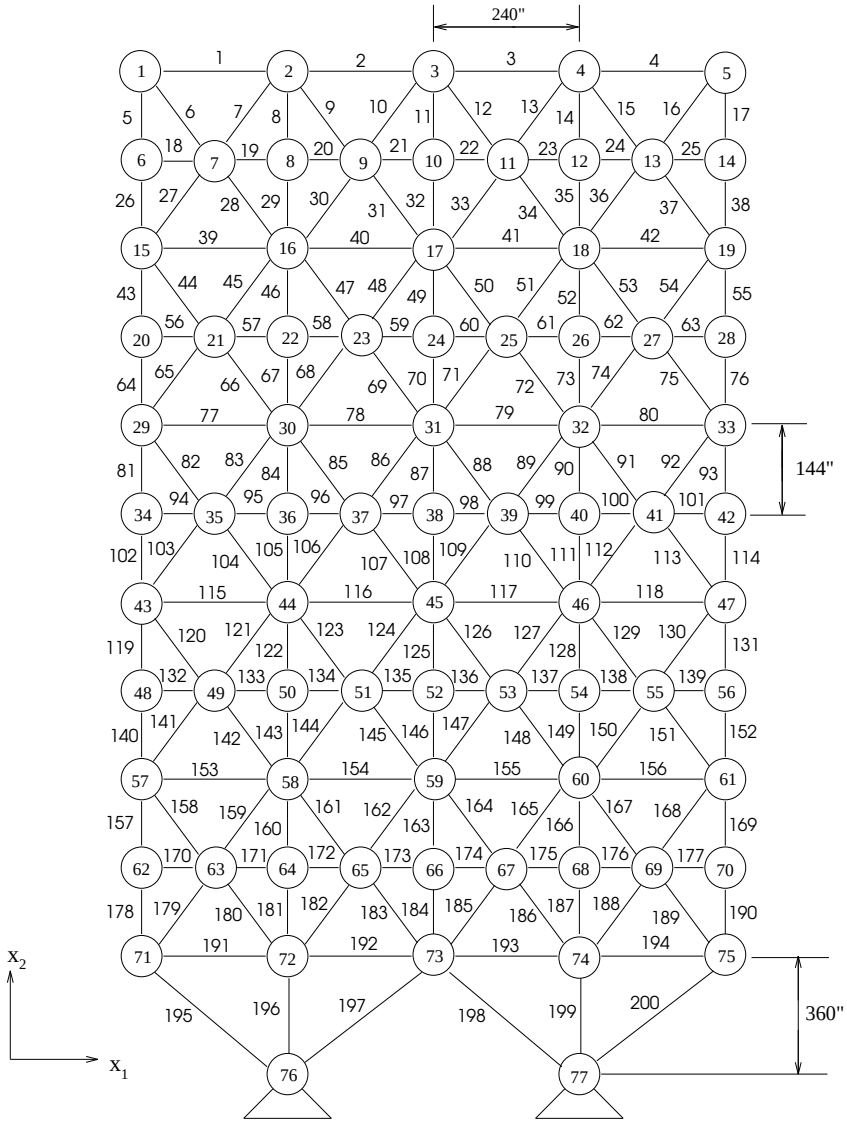

Figure 5: 200-bar plane truss used for the second example.

Group Number	Member Number
1	1,2,3,4
2	5,8,11,14,17
3	19,20,21,22,23,24
4	18,25,56,63,94,101,132,139,170,177
5	26,29,32,35,38
6	6,7,9,10,12,13,15,16,27,28,30,31,33,34,36,37
7	39,40,41,42
8	43,46,49,52,55
9	57,58,59,60,61,62
10	64,67,70,73,76
11	44,45,47,48,50,51,53,54,65,66,68,69,71,72,74,75
12	77,78,79,80
13	81,84,87,90,93
14	95,96,97,98,99,100
15	102,105,108,111,114
16	82,83,85,86,88,89,91,92,103,104,106,107,109,110,112,113
17	115,116,117,118
18	119,122,125,128,131
19	133,134,135,136,137,138
20	140,143,146,149,152
21	120,121,123,124,126,127,129,130,141,142,144,145,147,148,150,151
22	153,154,155,156
23	157,160,163,166,169
24	171,172,173,174,175,176
25	178,181,184,187,190
26	158,159,161,162,164,165,167,168,179,180,182,183,185,186,188,189
27	191,192,193,194
28	195,197,198,200
29	196,199

Table 4: Group membership for the 200-bar plane truss shown in Figure 5.

10 Example 2 : Design of a 200-bar plane truss

Consider the 200-bar plane truss taken from Belegundu [5], shown in Figure 5 (taken from Belegundu [5]). The problem is to find the cross-sectional area of each member of this truss, such that we minimize its weight, the displacement of each free node, and the stress that each member has to support.

There are a total of three loading conditions: (1) 1 kip acting in positive x-direction at node points 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71; (2) 10 kips acting in negative y-direction at node points 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 24, 71, 72, 73, 74, and 75; and (3) loading condition 1 and 2 acting together. The 200 elements of this truss linked to 29 groups. The grouping information is shown in Table 4. The stress in each element is limited to a value of 10 ksi for both tension and compression members. Young's modulus of elasticity = 30,000 ksi, weight density = 0.283×10^{-3} kips/in^3.

Method	f_1	f_2	f_3
Monte Carlo 1	**57144.60**	0.050551	1958.00
Monte Carlo 1	275439.48	**0.003382**	207.27
Monte Carlo 1	232253.56	0.003764	**194.88**
Min-Max (OS)	**1166.98**	0.781186	42028.65
Min-Max (OS)	1359.41	**0.598842**	33872.70
Min-Max (OS)	1359.41	0.598842	**33872.70**
GA (Binary)	**72845.41**	1.544286	87294.85
GA (Binary)	330717.40	**0.002757**	148.303585
GA (Binary)	330717.40	0.002757	**148.303585**
GA (FP)	**468.93**	1.565098	90959.54
GA (FP)	330716.80	**0.002757**	148.303654
GA (FP)	330717.25	0.002757	**148.303598**
Literature	**493.94**	1.285167	79916.70
Literature	493.94	**1.285167**	79916.70
Literature	493.94	1.285167	**79916.70**

Table 5: Comparison of results computing the ideal vector of the first example (design of a 25-bar space truss). For each method the best results for optimum f_1, f_2 and f_3 are shown in **boldface**. OS stands for Osyczka's Multiobjective Optimization System.

11 Comparison of Results

We will compare the ideal vector that each method generates with the best results reported in the literature for two truss-design problems. We used the Monte Carlo methods included in MOSES, together with Osyczka's multiobjective optimization system to obtain the ideal vector. Also, several GA-based approaches will be tested using the same parameters (same population size and same crossover and mutation rates). If niching is required, then the niche size will be computed according to the methodology suggested by the developers of the method (see [10] for details). To perform the analysis required for the examples, we used the matrix factorization method included in Gere and Weaver [20] together with the stiffness method [20] as implemented in [8].

11.1 Example 1

The ideal vector of this problem was computed using Monte Carlo methods 1 and 2 (generating 300 points), Osyczka's multiobjective optimization system and a GA (with a population of 300 chromosomes running during 100 generations) using binary and floating point representation, with the procedure described before to adjust its parameters. The corresponding results are shown in Table 5 including the best results reported in the literature [9]. The results for Monte Carlo Method 2 are the same as for Method 1, and the results presented for the Min-max method are also the basis for computing the best trade-off for all the methods in Osyczka's system. As can be seen from these results, the GA provided the best ideal vector, combining the results produced with both binary and floating point representation, although the second representation scheme provides better results in general [10]. The mathematical programming techniques did not provide any reasonable results in this example, mainly because of the high non-convexity of the search space and the high number of variables involved. It should be noted that the set of results reported by Coello et al. [9] was produced optimizing only the first objective (i.e., the total weight of the truss) in a discrete manner. Assuming continuous variables, the GA-engine for single objective optimization was able to find a lighter truss.

As we can see in Table 6, the new GA-based approach proposed by the authors, named GAminmax, provide the best overall results when a floating representation was used. It should be noted that our approach performs hardly over the average when binary representation is used. The reason for its poor performance here, and before (when trying to find the ideal vector) is that the population size does not seem to be large enough to guarantee

Method	f_1	f_2	f_3	$L_p(f)$
Ideal Vector	468.928261	0.002757	148.303585	0.000000
Monte Carlo 1	113293.85	0.006212	363.6076	243.306621
Monte Carlo 2	110264.89	0.006925	394.8020	237.316252
Min-max (OS)	1344.32	0.676830	34793.19	479.969778
GCM (OS)	1359.41	0.598842	33872.70	445.507894
WMM (OS)	1344.32	0.676830	34793.19	479.969778
PMM (OS)	1359.41	0.598842	33872.70	445.50789
NMM (OS)	1359.41	0.598842	33872.70	445.50789
GALC (B)	254696.03	0.003078	178.85	542.46737
GALC (FP)	193849.17	0.003389	202.83	412.98462
Lexicographic (B)	219176.78	0.005791	280.06	468.38825
Lexicographic (FP)	129424.79	0.005412	303.46	277.010455
VEGA (B)	234854.32	0.003473	205.70	500.478800
VEGA (FP)	219453.21	0.003482	202.59	467.617915
NSGA (B)	250615.78	0.002975	171.74	533.680836
NSGA (FP)	226478.31	0.003971	205.53	482.796241
MOGA (B)	85297.74	0.023970	990.87	194.274927
MOGA (FP)	81778.41	0.021254	908.03	185.226194
NPGA (B)	92943.08	0.010969	585.22	203.127877
NPGA (FP)	55812.18	0.029665	1307.44	135.596546
Hajela (B)	99464.52	0.017495	1134.42	223.105294
Hajela (FP)	107947.60	0.007593	421.48	232.796738
GAminmax (B)	85604.05	0.036615	2190.83	207.605910
GAminmax (FP)	16230.99	0.037474	2227.73	60.226711

Table 6: Comparison of the best overall solution found by each one of the methods included in MOSES for the first example (design of a 25-bar space truss). GA-based methods were tried with binary (B) and floating point (FP) representations. The following abbreviations were used: OS = Osyczka's System, GCM = Global Criterion Method (exponent=2.0), WMM (Weighting Min-max), PWM (Pure Weighting Method), NWM (Normalized Weighting Method), GALC = Genetic Algorithm with a linear combination of objectives using scaling. In all cases, weights were assumed equal to 0.33 (equal weight for every objective).

convergence, considering the length of the string which, in this case is of 136 genes [10].

The results obtained for this problem show how easily the mathematical programming techniques can be surpassed by a GA-approach, using the same number of points, though the GA starts with a completely random population (our approach ensures that the initial population contains only feasible individuals, but these solutions are still randomly generated). Although we used the same random numbers generator that the Monte Carlo techniques use, the results are quite different. For those who think that a simple linear combination of objectives should be good enough to deal with multiobjective optimization problems, the results for GALC (see Table 6) show the contrary. Our approach used a set of fifteen weights to compute the ideal vector.

11.2 Example 2

The second example (200-bar plane truss design) presents a larger structure in which the time taken by the analysis becomes a critical issue. The Monte Carlo Methods 1 and 2 were used with 500 points, and the GA also used a population size of 500 (over 100 generations) with binary and floating point representations, with the procedure previously described to adjust its parameters. The corresponding ideal vector is shown in Table 7 including the best results reported in the literature [5]. Notice that the results presented by Belegundu violate 34 constraints of the problem, which means that his solution is not valid. This explains why the GA could not

Method	f_1	f_2	f_3
Monte Carlo 1	**3019191.78**	3.9983	74.99
Monte Carlo 1	5423359.65	**0.522427**	9.5056
Monte Carlo 1	5635985.90	0.556857	**8.7115**
Min-Max (OS)	**617227.10**	3.8398	91.9799
Min-Max (OS)	650984.03	**3.6148**	86.0389
Min-Max (OS)	641862.42	3.6235	**85.8669**
GA (Binary)	**893885.79**	33.6943	957.1493
GA (Binary)	9963295.72	**0.370375**	5.124250
GA (Binary)	9963295.72	0.370375	**5.124250**
GA (FP)	**36167.73**	38.675848	1062.77608
GA (FP)	9961698.96	**0.370376**	5.124382
GA (FP)	9962313.62	0.370377	**5.124336**
Literature	**35162.93**	44.144661	1137.7476
Literature	35162.93	**44.144661**	1137.7476
Literature	35162.93	44.144661	**1137.7476**

Table 7: Comparison of results computing the ideal vector of the second example (design of a 200-bar plane truss). For each method the best results for optimum f_1, f_2 and f_3 are shown in **boldface**. OS stands for Osyczka's Multiobjective Optimization System. All the objectives are being minimized.

achieve such a low weight using floating point representation. In fact, in Belegundu's dissertation [5] he even provides a better solution (with a total weight of 26261.05) but that violates 48 constraints. We chose to include a solution with a higher weight, but a lower number of violations. Nevertheless, the number of constraints violated is still high and the GA could not possibly converge towards such solutions.

In this example, Monte Carlo methods provided results that are better (in general) than the solutions provided by the GA-based techniques, which is remarkable, considering the large size of the search space (see Table 8). This reflects the problems of traditional GA-based techniques to find reasonable trade-offs when the length of the chromosome string is too large (493 genes in this case). Also the high amount of constraints (200 total) makes this problem easier for mathematical programming techniques than for the GA using a penalty function. The performance of Osyzcka's multiobjective optimization system is extremely good, but mainly because the initial guesses provided by the user were quite close to a Pareto solution. The main use of such techniques is precisely in cases in which we have a rough approximation of the solution, or a lot of knowledge about how the solution space looks like is available, and we want to experiment within the boundaries of our partial result. Nevertheless, it should be pointed out that our technique was able to find a better overall result than any other approach (including mathematical programming methods) when a floating point representation was used.

12 Conclusions

We have proposed a new multiobjective optimization method based on the min-max optimization approach. This approach is very robust because it transforms the multiobjective optimization problem into several single objective optimization problems easier and faster to solve. When this approach is used with a floating point representation, the technique seems to work better (i.e., faster and more accurately) than the other approaches considered in this paper. The main drawbacks of our approach are that it requires the ideal vector and a set of weights to delineate the Pareto set. However, our GA-based engine included in MOSES was used to compute the ideal vector, generating results better than those previously reported in the literature. Also, if the ideal vector is not known in advance, a set of goal (desirable) values for each objective can be provided instead. On the other hand, finding proper weights is typically an easy task, since not many of them are required to get reasonably good results. In our applications, for example, no more than fifteen weights were used by our method.

Method	f_1	f_2	f_3	$L_p(f)$
Ideal Vector	36167.73	0.370376	5.124250	0.000000
Monte Carlo 1	4475679.05	0.773293	12.173738	125.211428
Monte Carlo 2	5075790.44	0.561163	9.682691	140.745011
Min-max (OS)	641862.42	3.686123	88.04811	41.881840
GCM (OS)	641862.42	3.623485	85.86687	41.287050
WMM (OS)	641862.42	3.686123	88.04811	41.881840
PMM (OS)	617227.10	3.839831	91.97989	42.382994
NMM (OS)	641862.42	3.623485	85.86687	41.287050
GALC (B)	5388876.23	0.418137	6.623930	148.418423
GALC (FP)	4662186.53	0.426666	6.736934	128.371292
Lexicographic (B)	5106929.51	0.590409	11.446706	142.029185
Lexicographic (FP)	3925963.62	0.538302	9.329877	108.822923
VEGA (B)	5956689.85	0.546020	10.048603	165.131483
VEGA (FP)	4051105.81	0.662998	10.306576	112.810251
NSGA (B)	7020831.93	0.424951	6.678441	193.569332
NSGA (FP)	5369341.87	0.548805	8.334404	148.564917
MOGA (B)	3626863.83	0.489753	8.435848	100.247575
MOGA (FP)	2910316.85	0.722471	12.477403	81.852839
NPGA (B)	4028058.04	3.723483	73.607848	132.789464
NPGA (FP)	4453361.10	0.970288	5.468708	123.817748
Hajela (B)	1924166.83	1.362296	28.222939	59.387070
Hajela (FP)	4291090.29	0.874752	11.478412	120.245983
GAminmax (B)	1508966.74	2.213371	47.516121	53.970163
GAminmax (FP)	686362.29	2.223398	43.910769	30.549494

Table 8: Comparison of the best overall solution found by each one of the methods included in MOSES for the second example (design of a 200-bar plane truss). GA-based methods were tried with binary (B) and floating point (FP) representations. The following abbreviations were used: OS = Osyczka's System, GCM = Global Criterion Method (exponent=2.0), WMM (Weighting Min-max), PWM (Pure Weighting Method), NWM (Normalized Weighting Method), GALC = Genetic Algorithm with a linear combination of objectives using scaling. In all cases, weights were assumed equal to 0.33 (equal weight for every objective).

Our technique ensures that only feasible points are produced at generation zero, and the crossover and mutation operators were modified in such a way that infeasible solutions are never generated by the algorithm. This property makes our approach unique, since none of the other GA-based techniques analyzed considered this important issue. This is mainly because most of the previous work with multiobjective optimization techniques dealt only with unconstrained problems.

Finally, the importance of MOSES as a benchmark for new and existing multiobjective optimization methods should be obvious, since no other similar tools, combining GA-based approaches with mathematical programming techniques, were previously available. Its modular structure allows the easy incorporation of new algorithms without having to modify its main routines. Additional details may be found in [10]. Also, it should be said that the system is a valuable tool, as it is, for engineering design optimization, because of the variety of different approaches that it contains.

13 Future Work

Much additional work remains to be done to improve the performance of our approach. One of our main interests is to be able to compute the ideal vector during run-time, instead of having to give it in advance to the GA. In that respect, we have developed another method that is very promising, but that still has some flaws and does not work properly with problems like the trusses used in this paper in which one of the objectives may strongly guide the search towards the ideal value disregarding the importance of the remaining objectives [10].

It would also be desirable to parallelize the GA and the analysis of the structure, to reduce the computational time required for each iteration. Adeli's approach [4] is an excellent example of the kind of work that can be done in that respect. We also aim to be able to encourage theoreticians to develop a theory of convergence for GAs in multiobjective optimization problems by using concepts from Operations Research such as the min-max optimum.

Finally, it is highly desirable to be able to find more ways of incorporating knowledge about the domain into the GA, as long as it can be automatically assimilated by the algorithm during its execution and does not have to be provided by the user (to preserve its generality). It is also important to follow Eshelman and Schaffer's [15] work on the pursuit of a theoretical framework that explains the excellent performance of real-coded GAs so that practice can finally meet theory in the use of GAs for numerical optimization.

References

[1] Hojjat Adeli and Nai-Tsang Cheng. Integrated genetic algorithm for optimization of space structures. *Journal of Aerospace Engineering*, 6(4):315–328, oct 1993.

[2] Hojjat Adeli and Nai-Tsang Cheng. Augmented lagrangian genetic algorithm for structural optimization. *Journal of Aerospace Engineering*, 7(1):104–18, jan 1994.

[3] Hojjat Adeli and Sanjay Kumar. Concurrent structural optimization on massively parallel supercomputer. *Journal of Structural Engineering*, 121(11):1588–1597, nov 1995.

[4] Hojjat Adeli and Sanjay Kumar. Distributed genetic algorithm for structural optimization. *Journal of Aerospace Engineering*, 8(3):156–163, jul 1995.

[5] Ashok Dhondu Belegundu. *A Study of Mathematical Programming Methods for Structural Optimization.* PhD thesis, University of Iowa, Dept. of Civil and Environmental Engineering, 1982.

[6] L. M. Boychuk and V. O. Ovchinnikov. Principal methods of solution of multicriterial optimization problems (survey). *Soviet Automatic Control*, 6:1–4, 1973.

[7] D. G. Carmichael. Computation of pareto optima in structural design. *International Journal for Numerical Methods in Engineering*, 15:925–952, 1980.

[8] Carlos A. Coello. Análisis de estructuras reticulares por computadora (método de rigideces). Tesis de Licenciatura, 1991. (in Spanish).

[9] Carlos A. Coello, Michael Rudnick, and Alan D. Christiansen. Using genetic algorithms for optimal design of trusses. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*, pages 88–94, New Orleans, LA, nov 1994. IEEE Computer Society Press.

[10] Carlos Artemio Coello Coello. *An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design.* PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA, apr 1996.

[11] L. Duckstein. Multiobjective optimization in structural design: The model choice problem. In E. Atrek, R. H. Gallagher, K. M. Ragsdell, and O. C. Zienkiewicz, editors, *New Directions in Optimum Structural Design*, pages 459–481. John Wiley and Sons, 1984.

[12] M. E. El-Sayed, B. J. Ridgely, and E. Sandgren. Nonlinear structural optimization using goal programming. *Computers and Structures*, 32(1):69–73, 1989.

[13] Hans Eschenauer, Juhani Koski, and Andrzej Osyczka, editors. *Multicriteria Design Optimization. Procedures and Applications.* Springer-Verlag, 1990.

[14] Larry J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in non-traditional genetic recombination. In Gregory E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann Publishers, San Mateo, California, 1991.

[15] Larry J. Eshelman and J. Davis Schaffer. Real-coded genetic algorithms and interval-schemata. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann Publishers, San Mateo, California, 1993.

[16] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[17] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. Technical report, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U. K., 1994.

[18] M. P. Fourman. Compaction of symbolic layout using genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 141–153. Lawrence Erlbaum, 1985.

[19] Gongkang Fu and Dan M. Frangopol. Reliability-based vector optimization of structural systems. *Journal of Structural Engineering*, 116(8):2143–2161, aug 1990.

[20] James M. Gere and William Weaver. *Analysis of Framed Structures.* D. Van Nostrand Company, Inc., 1965.

[21] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, Mass. : Addison-Wesley Publishing Co., 1989.

[22] David E. Goldberg and Kalyanmoy Deb. A comparison of selection schemes used in genetic algorithms. In G.J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, San Mateo, California, 1991.

[23] David E. Goldberg and J. Richardson. Genetic algorithm with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum, 1987.

[24] David E. Goldberg and Manohar P. Samtani. Engineering optimization via genetic algorithm. In *Ninth Conference on Electronic Computation*, pages 471–82, New York, N.Y., 1986. ASCE.

[25] Ramana V. Grandhi, Geetha Bharatram, and Vipperla B. Venkayya. Multiobjective optimization of large-scale structures. *AIAA Journal*, 31(7):1329–1337, jul 1993.

[26] J. J. Grefenstette. GENESIS: A system for using genetic search procedures. In *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pages 161–165, 1984.

[27] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128, 1986.

[28] P. Hajela and C. Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.

[29] P. Hajela and C. J. Shih. Multiobjective optimum design in mixed integer and discrete design variable problems. *AIAA Journal*, 28(4):670–675, apr 1990.

[30] David M. Himmelblau. *Applied Nonlinear Programming.* McGraw-Hill Book Company, New York, 1972.

[31] J. Horn and N. Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[32] W. M. Jenkins. Towards structural optimization via the genetic algorithm. *Computers and Structures*, 40(5):1321–7, 1991.

[33] Juhani Koski and Risto Silvennoinen. Pareto optima of isostatic trusses. *Computer Methods in Applied Mechanics and Engineering*, 31:265–279, 1982.

[34] Juhani Koski and Risto Silvennoinen. Norm methods and partial weighting in multicriterion optimization of structures. *International Journal for Numerical Methods in Engineering*, 24:1101–1121, 1987.

[35] E. M. Krokosky. The ideal multifunctional constructural material. *Journal of the Structural Division, ASCE*, 94:958–981, 1968.

[36] C. Y. Lin and P. Hajela. EVOLVE: A genetic search based optimization code via multiple strategies. In S. Hernández and C. A. Brebbia, editors, *Computer Aided Optimum Design of Structures III. Optimization of Structural Systems and Applications*, pages 639–654. Elsevier Applied Science, 1993.

[37] J. G. Lin. Maximal vectors and multi-objective optimization. *Journal of Optimization Theory and Applications*, 18(1):41–64, jan 1976.

[38] Z. Lounis and M. Z. Cohn. Multiobjective optimization of prestressed concrete structures. *Journal of Structural Engineering*, 119(3):794–808, mar 1993.

[39] A. Osyczka. An approach to multicriterion optimization problems for engineering design. *Computer Methods in Applied Mechanics and Engineering*, 15:309–333, 1978.

[40] Andrzej Osyczka. *Multicriterion Optimization in Engineering with FORTRAN programs.* Ellis Horwood Limited, 1984.

[41] Andrzej Osyczka. Multicriteria optimization for engineering design. In John S. Gero, editor, *Design Optimization*, pages 193–227. Academic Press, 1985.

[42] Vilfredo Pareto. *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.

[43] S. D. Rajan. Sizing, shape, and topology design optimization of trusses using genetic algorithm. *Journal of Structural Engineering*, 121(10):1480–1487, oct 1995.

[44] S. Rajeev and C. S. Krishnamoorthy. Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering*, 118(5):1233–50, may 1992.

[45] Jawalker K. Sridhar Rao, P. N. Murthy, C. V. S. Kameswara Rao, and V. K. Kapur. An algorithm for optimal material choice for multifunctional criteria. *Computers and Structures*, 3:715–724, 1973.

[46] S. Rao. Game theory approach for multiobjective structural optimization. *Computers and Structures*, 25(1):119–127, 1986.

[47] S. S. Rao. Multiobjective optimization in structural design with uncertain parameters and stochastic processes. *AIAA Journal*, 22(11):1670–1678, nov 1984.

[48] S. S. Rao. Multi-objective optimization of fuzzy structural systems. *International Journal for Numerical Methods in Engineering*, 24:1157–1171, 1987.

[49] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some guidelines for genetic algorithms with penalty functions. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 191–197, George Mason University, 1989. Morgan Kaufmann Publishers.

[50] R. S. Rosenberg. *Simulation of genetic populations with biochemical properties.* PhD thesis, University of Michigan, Ann Harbor, Michigan, 1967.

[51] M. E. Salukvadze. On the existence of solution in problems of optimization under vector valued criteria. *Journal of Optimization Theory and Applications*, 12(2):203–217, 1974.

[52] D. A. Saravanos and C. C. Chamis. Multiobjective shape and material optimization of composite structures including damping. *AIAA Journal*, 30(3):805–813, 1992.

[53] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[54] Marc Schoenauer and Spyros Xanthakis. Constrained GA optimization. In *Fifth International Conference on Genetic Algorithms*, pages 573–80, University of Illinois at Urbana-Champaign, jul 1993. Morgan Kauffman Publishers.

[55] C. P. Smolenski and E. W. Krokosky. Optimal multifactor design procedure for sandwich panels. *Journal of the Structural Division, ASCE*, 96:823–837, 1970.

[56] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. Technical report, Department of Mechanical Engineering, Indian Institute of Technology, Kanput, India, 1993.

[57] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, fall 1994.

[58] W. Stadler. Preference optimality and application of pareto optimality. In Marzollo/Leitmann, editor, *Multicriterion Decision Making.* Springer-Verlag, Berlin, Germany, 1975.

[59] W. Stadler. Multicriteria optimization in mechanics (a survey). *Applied Mechanics Review*, 37(3):277–286, 1984.

[60] C. H. Tseng and T. W. Lu. Minimax multiobjective optimization in structural design. *International Journal for Numerical Methods in Engineering*, 30:1213–1228, 1990.

[61] T. L. Vincent. Game theory as a design tool. *Journal of Mechanisms, Transmissions, and Automation in Design*, 105:165–170, jun 1983.