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.

2 Previous Work on Multiobjective Structural Optimization

The first application of multiobjective optimization concepts in structural mechanics appeared in a 1968 paper by Krokosky [1]. In this early paper, Krokosky adopted a random search technique to find the best trade-off correlating the different objectives in terms of the a priori chosen design parameters. Since then, multiobjective optimization has attracted a lot of attention among structural engineers, and several surveys are available in the literature [2, 3, 4, 5].

To facilitate the study and comparison of the most important mathematical programming and GA-based multiobjective optimization techniques, the authors developed MOSES (Multiobjective Optimization of Systems in the Engineering Sciences), which is intended to serve as a common platform to test any new and/or existing multiobjective optimization technique. MOSES was written in GNU C and runs under Unix. For details on its implementation see Coello [4].

3 Basic Concepts

Multiobjective optimization (also called multicriteria optimization, multiperformance or vector optimization) can be defined as the problem of finding [6]:

> a vector of decision variables which satisfies constraints and optimizes a vector function whose elements 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 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 $i = 1, 2, \ldots, k$

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

where $\bar{x}^*$ is a desirable solution. However, we normally never have a situation like this, in which all the $f_i(\bar{x})$ have a minimum in the feasible region $\mathcal{F}$ at a common point x^*.

3.1 Pareto Optimum

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-dominated* solutions.

3.2 Min-Max Optimum

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. The optimum is defined as:

$$\bar{x}^* = min \left[max \ \{z_i^{'}(\bar{x}), z_i^{''}(\bar{x})\} \right] \tag{9}$$

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.

4 Osyczka's Multicriterion Optimization System

Osyczka's system contains several multiobjective optimization methods [7]:

(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:

$$f(\bar{x}) = \sum_{i=1}^{k} \left(\frac{f_i^0 - f_i(\bar{x})}{f_i^0} \right)^p \tag{10}$$

is used as the global function. 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 using the equation

$$\bar{x}^* = min \left[max\{w_i z_i^{'}(\bar{x}), w_i z_i^{''}(\bar{x})\} \right] \tag{11}$$

(4) Pure weighting method : The equation

$$min \sum_{i=1}^{k} w_i f_i(\bar{x}) \tag{12}$$

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

$$\sum_{i=1}^{k} w_i = 1 \tag{13}$$

(5) Normalized weighting method : $\bar{f}(\bar{x})$ is used in equation (4).

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 using an iterative or random search method.

5 Multiobjective Optimization using GAs

The notion of genetic search in a multicriteria problem dates back to the late 1960s, in which Rosenberg's [8] 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 [9]. 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 are the following (for a detailed survey of this subject see Coello [5]):

1. **VEGA**: David Schaffer [10] modified the selection operator of a Simple Genetic Algorithm (SGA) 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.

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 [11]. Another version of the algorithm reported by Fourman [12] consisted of randomly selecting the objective to be used at each generation.

3. **Weighted Sum**: Hajela and Lin [13] 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. Also, a mating restriction mechanism was imposed, to avoid members within a radius σ_{mat} to cross.

4. **Multiple Objective Genetic Algorithm**: Fonseca and Fleming [14] 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 [14]:

$$rank(x_i, t) = 1 + p_i^{(t)} \tag{14}$$

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.

5. **Non-dominated Sorting Genetic Algorithm**: The Non-dominated Sorting Genetic Algorithm (NSGA) was proposed by Srinivas and Deb [15], and is based on several layers of classifications of the individuals. Before the selection is performed, the population is ranked on the basis of non-domination: 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.

6. **Niched Pareto GA**: Horn and Nafpliotis [16] 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 [17]. 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 [9].

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

The basic algorithm proposed by the authors is the following [4] (see Figure 1):

1. The initial population is generated randomly, but 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. The procedure adopted in this case is death penalty (i.e., if a chromosome encodes an infeasible solution, it is destroyed and replaced by a newly generated string). However, a penalty function or any other constraint-handling approach can also be used.

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 (see below for details). The number of weight combinations is usually small (no more than 15 in the case of the experiments reported in this paper) and can be generated using a deterministic procedure in which each weight ranges from certain initial value to a final value using a user-defined increment. Notice that the use of a file containing weight combinations makes unnecessary to encode such weights in the chromosome itself as another decision variable.

3. The fitness value of each chromosome is computed according to equation (9). In general, the fitness function has the form:

$$fitness_i = \sum_{i=1}^{n} w_i min \left[max\{z_i^{'}(\bar{x}), z_i^{''}(\bar{x})\} \right] \tag{15}$$

   Notice that the variation of the weights will allow us to explore different parts of the Pareto front, and that this approach works both with convex and nonconvex search spaces [4].

   Since this expression 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 (see next section). Alternatively, an estimated set of values close to the desired goals can be provided by the user. These goals can underestimate or overestimate the ideal vector, as long as they lie on the feasible region.

4. The crossover and mutation operators were modified to ensure that they produced only feasible solutions. Whenever a child encoded an infeasible solution, it was replaced by one of its parents (randomly chosen). The best solution found was kept through generations until a better one emerged in a further stage of the search process (elitism).

5. No sharing is required in this case, since each process spawned deals with a single point of the Pareto front, and we do not have to avoid global convergence of the population towards such point. However, in case we wish to generate with the same process a set of points instead of only one (requiring a single GA run), we may use of a sharing function of the form:

$$\phi(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^{\alpha}, & d_{ij} < \sigma_{share} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

   where normally $\alpha = 1$, d_{ij} is a metric indicative of the distance between designs i and j, and σ_{share} is the sharing parameter which controls the extent of sharing allowed (a value between 0.01 and 0.1 is normally used). The fitness of a design i would then be modified as:

$$f_{s_i} = \frac{f_i}{\sum_{j=1}^{M} \phi(d_{ij})} \tag{17}$$

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

The main reason why we did not take this approach (like in Hajela and Lin's work [13]) is because the definition of σ_{share} is subject to intensive experimentation and its choice has a dramatic impact on the performance of the technique.

6. After the n processes are terminated (n=number of weight combinations provided by the user), a final file is generated containing the non-dominated solutions found. This file is formed by picking up the best solution from each of the processes spawned in step 2, and will contain the min-max optimum solutions to the problem (equivalent to the Pareto set).

7. Notice that the 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.

The procedure described above is not only very easy to implement, but is also very efficient (computationally speaking) if we use a distributed system, because the processes can be assigned to different processors and be run in parallel.

GA-based methods that use Pareto-based techniques need to check for non-dominated solutions, and that is a process that requires $k \times m^2$ operations, where k is the number of objectives and m is the population size. This is therefore, an expensive process (computationally speaking).

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 of the difficulties found to adjust its parameters (i.e., population size, maximum number of generations, mutation and crossover rate) [18]. 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.

In some previous work [4, 19], we have successfully used 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 [4]. 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 that 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 [20], 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 [21] and it exhibits its best behavior with a floating point representation in numerical optimization problems.

7 Structural Optimization using Genetic Algorithms

Goldberg and Samtani [22] 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 [23] 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 [24] 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 [25] used the GA for discrete optimization of generalized trusses. Schoenauer and Xanthakis [26] 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 [27] described a design optimization tool based on genetic search which inspired the development of MOSES [4]. 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 [13] 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 [28] 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 [29], 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 [30] 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 [31] also used a GA for structural optimization of large scale structures on massively parallel supercomputers.

Rajan [32] 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.

Yeh [33] used a hybrid genetic algorithm to optimize truss structures. This work focuses on the efficiency of the optimization process using a GA, rather than in the results obtained (which are, nevertheless better than those obtained by a simple GA), and the search space is considered as discrete to exploit the search capabilities of the GA.

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 [25] shown in Figure 2. 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.

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

Consider the 200-bar plane truss taken from Belegundu [34], shown in Figure 3 (taken from Belegundu [34]). 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.

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 2 Monte Carlo methods included in MOSES (see Coello [4] for details), 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 [4] for details). To perform the analysis required for the examples, we used the matrix factorization method included in Gere and Weaver [35] together with the stiffness method [35] as implemented in [36].

11.1 Example 1

The ideal vector of this problem was computed using the 2 Monte Carlo methods included in MOSES (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 [37]. 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 [4]. 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. [37] 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 convergence, considering the length of the string which, in this case is of 136 genes [4].

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 chromosomes (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 [34]. 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 achieve such a low weight using floating point representation. In fact, in Belegundu's dissertation [34] 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.

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 [4]. 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 [4].

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 [30] 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. In this respect, some important work has been recently done by Rudolph [38] and Van Veldhuizen and Lamont [39], but several issues remain to be solved, such as diverse aspects related to the parallelization of evolutionary multiobjective approaches (e.g., load balancing, impact on Pareto convergence, performance issues, etc.), including new algorithms that are more suitable for parallelization than those currently in use.

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

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

3. 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.

4. C. A. C. 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.

5. C. A. C. Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1999. (Accepted for publication).

6. A. Osyczka. Multicriteria optimization for engineering design. In J. S. Gero, editor, *Design Optimization*, pages 193–227. Academic Press, 1985.

7. A. Osyczka. *Multicriterion Optimization in Engineering with FORTRAN programs.* Ellis Horwood Limited, 1984.

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

9. C. M. Fonseca and P. 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.

10. J. D. 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.

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

12. 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.

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

14. C. M. Fonseca and P. J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In S. 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.

15. 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.

16. 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.

17. D. 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.

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

19. C. A. Coello, F. S. Hernández, and F. A. Farrera. Optimal design of reinforced concrete beams using genetic algorithms. *Expert Systems with Applications. An International Journal*, 12(1):101–108, jan 1997.

20. L. J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann Publishers, San Mateo, California, 1991.

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

22. D. E. Goldberg and M. P. Samtani. Engineering optimization via genetic algorithm. In *Ninth Conference on Electronic Computation*, pages 471–82, New York, N.Y., 1986. ASCE.

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

24. 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.

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

26. M. Schoenauer and S. 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.

27. 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.

28. H. Adeli and N.-T. Cheng. Integrated genetic algorithm for optimization of space structures. *Journal of Aerospace Engineering*, 6(4):315–328, oct 1993.

29. H. Adeli and N.-T. Cheng. Augmented lagrangian genetic algorithm for structural optimization. *Journal of Aerospace Engineering*, 7(1):104–18, jan 1994.

30. H. Adeli and S. Kumar. Distributed genetic algorithm for structural optimization. *Journal of Aerospace Engineering*, 8(3):156–163, jul 1995.

31. H. Adeli and S. Kumar. Concurrent structural optimization on massively parallel supercomputer. *Journal of Structural Engineering*, 121(11):1588–1597, nov 1995.

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

33. I.-C. Yeh. Hybrid Genetic Algorithms for Optimization of Truss Structures. *Microcomputers in Civil Engineering*, 14(3):199–206, May 1999.

34. A. D. Belegundu. *A Study of Mathematical Programming Methods for Structural Optimization*. PhD thesis, University of Iowa, Dept. of Civil and Environmental Engineering, 1982.

35. J. M. Gere and W. Weaver. *Analysis of Framed Structures*. D. Van Nostrand Company, Inc., 1965.

36. C. A. Coello. Análisis de estructuras reticulares por computadora (método de rigideces). Tesis de Licenciatura, 1991. (in Spanish).

37. C. A. Coello, M. Rudnick, and A. 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.

38. G. Rudolph. On a Multi-Objective Evolutionary Algorithm and Its Convergence to the Pareto Set. In *Proceedings of the 5th IEEE Conference on Evolutionary Computation*, pages 511–516, Piscataway, New Jersey, 1998. IEEE Press.

39. D. A. V. Veldhuizen and G. B. Lamont. Evolutionary Computation and Convergence to a Pareto Front. In J. R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228, Stanford University, California, July 1998. Stanford University Bookstore.

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