MOSES : A Multiobjective Optimization Tool for Engineering Design

Carlos A. Coello Coello & Alan D. Christiansen

Department of Computer Science, Tulane University, New Orleans, LA 70118, USA

Abstract: *In this paper, we introduce a multiobjective optimization tool called MOSES (Multiobjective Optimization of Systems in the Engineering Sciences). This tool is a convenient testbed for analyzing the performance of new and existing multicriteria optimization techniques, and it is an effective engineering design tool. Two new multiobjective optimization techniques based on the genetic algorithm (GA) are introduced, and two engineering design problems are solved using them. These methods are based in the concept of min-max optimum, and can produce the Pareto set and the best trade-off among the objectives. The results produced by these approaches are compared to those produced with other mathematical programming techniques and GA-based approaches, showing the new techniques' capability to generate better trade-offs than the approaches previously reported in the literature.*

Keywords*:* multiobjective optimization, genetic algorithms, design optimization, min-max optimization, multicriteria optimization, artificial intelligence.

1 Introduction

Much work has been done in engineering optimization during the last few years, but the trend has been to deal with ideal and unrealistic problems, rather than with real-world applications. One of the reasons for this has been that for many years only single-objective functions were considered. As we know, this is not a realistic assumption, since most real-world problems have several (possible conflicting) objectives. This situation has led designers to make decisions and trade-offs based on their experience, instead of using some well-defined optimality criterion.

Over the years, more than 20 mathematical programming techniques have been developed to deal with multiple objectives. However, the main focus of these approaches is to produce a *single* trade-off based on some notion of optimality, rather than producing several possible alternatives from which the designer may choose. More recently, the genetic algorithm (GA), an artificial intelligence search technique based on the mechanics of natural selection, has been found to be effective on some scalar optimization problems. In order to extend the GA to deal with multiple objectives, the structure of the GA has been modified to handle a vector fitness function.

This paper will review some of the previous work in multiobjective optimization using genetic algorithms, and two new approaches, proposed by the authors, will be introduced. Also, MOSES (Multiobjective Optimization of Systems in the Engineering Sciences), a system developed as a testbed for multiobjective optimization techniques by the authors, will be described together with two examples of its use. The new approaches, based on the notion of min-max optimum, are able to generate the Pareto set and better trade-offs than any of the other techniques included in MOSES. Interestingly, the GA-based engine included in MOSES has been able to generate better ideal vectors than traditional mathematical programming techniques, by using alphabets of cardinality higher than two, contradicting the common notion that the GA is not very appropriate as a numerical optimization tool.

2 General Concepts

Here we define some concepts that will be used in this paper.

2.1 Statement of the Problem

Multiobjective optimization (also called multicriteria optimization, multiperformance or vector optimization) can be defined as the problem of finding [49]:

> 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:

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

the p equality constraints

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

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** X and any point $\bar{x}$ in X defines a **feasible solution**. The vector function $\bar{f}(\bar{x})$ is a function which maps the set X in the set F which represents all possible values of the objective functions. The k components of the vector $\bar{f}(\bar{x})$ represent the non-commensurable criteria 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).

2.2 Ideal Vector

Let us assume that we find the minimum (or maximum) of each of the objective functions $f_i(\bar{x})$ separately. Assuming that they can be found, let

$$\bar{x}^{0(i)} = [x_1^{0(i)}, x_2^{0(i)}, \ldots, x_n^{0(i)}]^T \tag{4}$$

be a vector of variables which optimizes (either minimizes or maximizes) the ith objective function $f_i(x)$. In other words, the vector $\bar{x}^{0(i)} \in X$ is such that

$$f_i(\bar{x}^{0(i)}) = \underset{x \in X}{opt} \quad f_i(\bar{x}) \tag{5}$$

In general, there will be a unified criterion with respect to "opt". Most authors prefer to treat it as a minimum. In that case, $f_i(\bar{x}^{0(i)})$ or simply f_i^0 (more convenient notation) will denote the minimum value of the ith function. Hence, the vector $\bar{f}^0 = [f_1^0, f_2^0, \ldots, f_k^0]^T$ is ideal for a multiobjective optimization problem, and the point in R^n which determined this vector is the ideal (utopical) solution, and is called the **ideal vector**. Although this solution is generally not feasible, this is an important definition that will be used later.

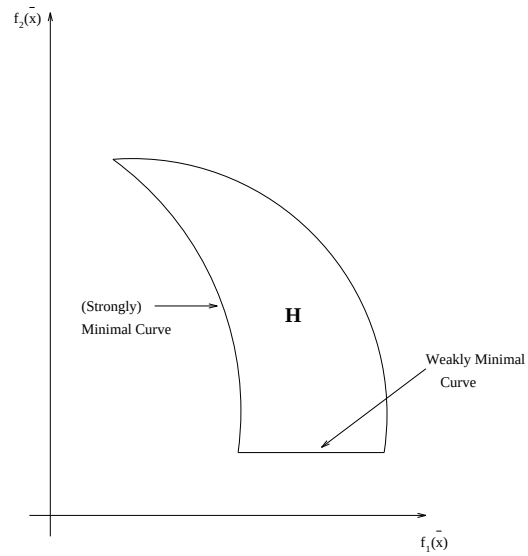

Figure 1: Weakly and strongly non-dominated curves on the biobjective case.

2.3 Pareto Optimum

The concept of **Pareto optimum** was formulated by Vilfredo Pareto in 1896 [50], and constitutes by itself the origin of research in multiobjective optimization. We say that a point $\bar{x}^* \in X$ is **Pareto optimal** if for every $\bar{x} \in X$ either,

$$\bigwedge_{i \in I} (f_i(\bar{x}) = f_i(\bar{x}^*)) \tag{6}$$

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

$$f_i(\bar{x}) > f_i(\bar{x}^*) \tag{7}$$

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.

2.4 Non-dominated Solutions

A point $\bar{x}^* \in X$ is a **weakly non-dominated solution** if there is no $\bar{x} \in X$ such that $f_i(\bar{x}) < f_i(\bar{x}^*)$, for $i = 1, \ldots, n$.

A point $\bar{x}^* \in X$ is a **strongly non-dominated solution** if there is no $\bar{x} \in X$ such that $f_i(\bar{x}) \leq f_i(\bar{x}^*)$, for $i = 1, \ldots, n$ and for at least one value of i, $f(\bar{x}) < f(\bar{x}^*)$.

Thus, if $\bar{x}^*$ is strongly non-dominated, it is also weakly non-dominated, but the converse is not necessarily true. Non-dominated solutions for the biobjective case can readily be represented graphically by passing into the objective function space $\{f_1(\bar{x}), f_2(\bar{x})\}$. To the locus of strongly non-dominated points corresponds the so-called **minimal curve**, and to the locus of weakly non-dominated points, the **weakly minimal curve** [2]. These two curves are sketched in Figure 1 (taken from Duckstein [14]). We will use X^p to denote the set of noninferior or nondominated solutions, and F^p to denote the map of X^p in the space of objectives. The set X^p is, of course, determined from the set F^p which satisfies (6) and (7).

In engineering optimization, strongly non-dominated solutions are sought, and the qualification "strongly" is generally omitted [14]. Let H be the set, called the pay-off set and shown in Figure 1, defined by:

$$H = (\bar{a}|\bar{a} = (a_i) \in R^n, \bar{x} \in X \; such \; that \;\; a_i = f_i(\bar{x}) \; for \; every \; i) \tag{8}$$

Then, as proved by Szidarovszky and Duckstein [62], if H is non-empty closed and for every i

$$max \; \{a_i|(a_i) \in H\} < \infty, \tag{9}$$

then H has at least one strongly non-dominated solution. Thus, a large class of multiobjective optimization problems in engineering design may be expected to possess at least one non-dominated solution, and usually the problem is, as we mentioned before, that there is a great number of possible solutions to choose from, and this may cause difficulties both in generating the solution set and in handling the results [39] [35].

2.5 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 [48] and Solich [48]. It has been further developed by Osyczka [46] [47], Rao [52] and Tseng and Lu [63].

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

$$z_i^{'}(\bar{x}) = \frac{|f_i(\bar{x}) - f_i^0|}{|f_i^0|} \tag{10}$$

or from

$$z_i^{''}(\bar{x}) = \frac{|f_i(\bar{x}) - f_i^0)|}{|f_i(\bar{x})|} \tag{11}$$

It should be clear that for (10) and (11) we have to assume that for every $i \in I$ and for every $\bar{x} \in X$, $f_i(\bar{x}) \neq 0$.

If all the objective functions are going to be minimized, then equation (10) defines function relative increments, whereas if all of them are going to be maximized, it defines relative decrements. Equation (11) 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 R^k. The components of the vector $z(\bar{x})$ will be evaluated from the formula

$$\forall_{i \in I}(z_i(\bar{x}) = max \; \{z_i^{'}(\bar{x}), z_i^{''}(\bar{x})\} \tag{12}$$

Now we define the min-max optimum as follows [48]:

A point $\bar{x}^* \in X$ is min-max optimal, if for every $\bar{x} \in X$ the following recurrence formula is satisfied:

<u>Step 1:</u>

$$v_1(\bar{x}^*) = \min_{x \in X} \; \max_{i \in I} \; \{z_i(\bar{x})\} \tag{13}$$

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 $X_1 \subset X$ which satisfies Step 1, then

<u>Step 2:</u>

$$v_2(\bar{x}^*) = \min_{x \in X_1} \; \max_{i \in I, \, i \notin I_1} \; \{z_i(\bar{x})\} \tag{14}$$

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 $X_{r-1} \subset X$ which satisfies step $r-1$ then

<u>Step r:</u>

$$v_r(\bar{x}^*) = \min_{x \in X_{r-1}} \; \max_{i \in I, \, i \notin I_{r-1}} \; \{z_i(\bar{x})\} \tag{15}$$

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.

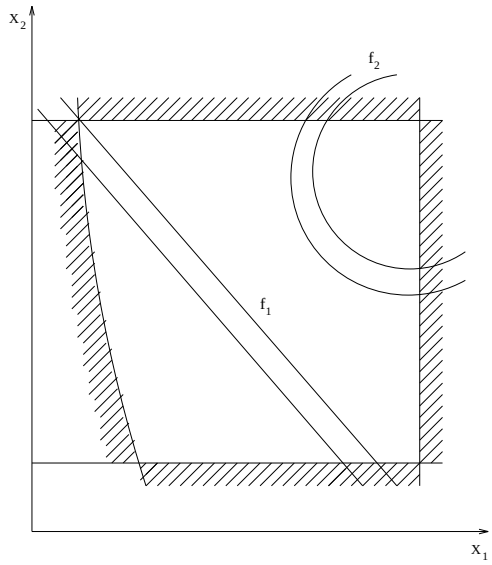

Figure 2: An example of a problem with two variables and two objective functions. The pareto optimal solutions are indicated by the shaded boundaries of the design region.

If there is a set of solutions $X_{k-1} \subset X$ which satisfies Step $k-1$, then
Step k:

$$v_k(\bar{x}^*) = \min_{\bar{x} \in X_{k-1}} z_i(\bar{x}) \quad \max_{i \in I,\, i \notin I_{k-1}} \quad for\ i \in I\ and\ i \notin I_{k-1} \tag{16}$$

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 $\bar{x}^* \in X$ 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. It should be noted that although these equations look quite complicated, in many optimization models, only the first step of this process will be necessary to determine the optimum.

In most cases, there will be several optimal solutions in the Pareto sense, and the designer will have to look at the values of the objective functions corresponding to $F(X^p)$ in order to decide which value seems the most appropriate. This process in which a solution is accepted is called the **decision making** process.

2.6 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. Figure 2 (taken from Hernández [36]) shows these boundaries shaded. This shaded 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 X^p and their corresponding $F(X^p)$. When we have a sufficient amount of these, we may proceed to take the final decision.

In general, we can say that if all the criteria are equally important in a problem, then the optimum in the min-max sense may give us a desirable solution. In all other cases a solution from the set of optimal solutions in the Pareto sense should be chosen.

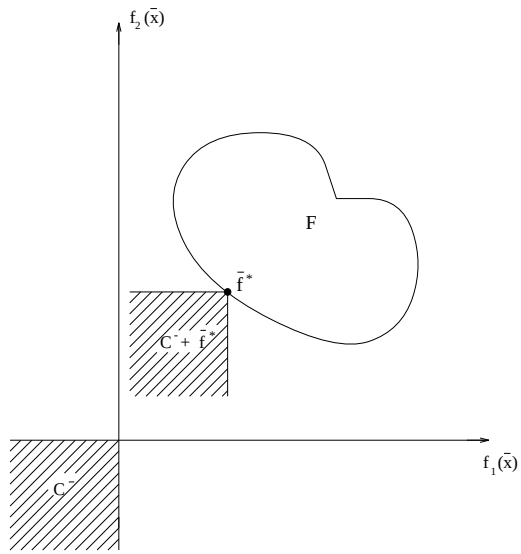

Figure 3: Graphical illustration of the contact theorem.

3 Mathematical Programming Techniques

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 concept of optimality in this context, in case the designer desires a single final solution. For the scope of this work, we will adopt the concept of min-max optimum for that sake. Therefore, we will also provide an algorithm that will give us the min-max optimum from a certain set of solutions.

3.1 Generating Pareto Optimal Solutions

We have seen the concept of Pareto optimality, but we have not yet described any algorithm that can identify the Pareto optimal solutions from a given set of feasible solutions. Osyczka [48] provides an algorithm that is based on the contact theorem, which is one of the main theorems in multiobjective optimization [43].

First, let us define a negative cone [48]. The negative cone in R^k is the set

$$C^- = \{\bar{f} \in R^k | \bar{f} \leq 0\} \tag{17}$$

Thus, the contact theorem is:

A vector f^* is a Pareto optimal solution for the general multiobjective optimization problem if and only if

$$(C^- + \bar{f}^*) \cap F = \{f^*\} \tag{18}$$

A graphical illustration of this theorem for a two criterion problem is shown in Figure 3 (taken from Osyczka [48]).

Consider two solutions $\bar{x}^{(1)}$ and $\bar{x}^{(2)}$ for which we may have two specific cases

$$(1)(C^- + \bar{f}(\bar{x}^{(1)})) \subset (C^- + \bar{f}(\bar{x}^{(2)})) \tag{19}$$

$$(2)(C^- + \bar{f}(\bar{x}^{(1)})) \supset (C^- + \bar{f}(\bar{x}^{(2)})) \tag{20}$$

A graphical illustration of these cases is presented in Figure 4 (taken from Osyczka [48]).

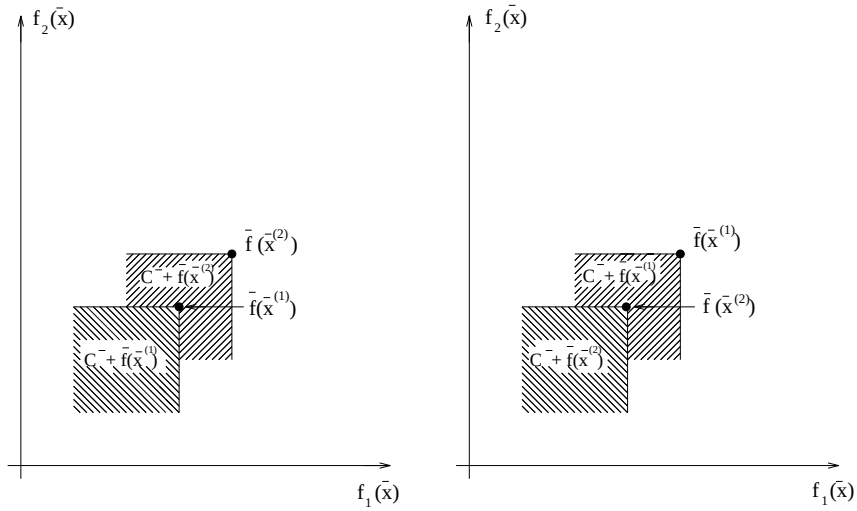

Figure 4: Graphical illustration of equations (19) and (20).

We denote $\bar{x}^{(l)} = [x_1^l, x_2^l, \ldots, x_n^l]^T$ = any given point in X,
$f(\bar{x}^{(l)}) = [f_1(\bar{x}^{(l)}), f_2(\bar{x}^{(l)}), \ldots, f_k(\bar{x}^{(l)})]^T$ = vector of objective functions for the point $\bar{x}^{(l)}$,
$\bar{x}_j^p = [x_{1j}^p, x_{2j}^p, \ldots, x_{nj}^p]^T$= The jth Pareto optimal solution,
$\bar{f}_j^p = [f_{1j}^p, f_{2j}^p, \ldots, f_{kj}^p]^T$=vector of objective functions for the jth Pareto optimal solution.
Now the problem is to choose from any given set of solutions
$L = \{1, 2, \ldots, l, \ldots, l^a\}$, the set of Pareto optimal solutions
$J = \{1, 2, \ldots, j, \ldots, j^a\}$.

The main idea behind the Pareto algorithm is the following. Let $\bar{x}^{(l)}$ be a new solution to be considered. If in the set of Pareto optimal solutions there is a solution x_j^p such that it

(i) satisfies (19) then $\bar{x}^{(l)}$ is substituted for $\bar{x}_j^p$, or

(ii) satisfies (20) then $\bar{x}^{(l)}$ is discarded.

If none of the solutions from the Pareto set satisfies either (19) or (20), then $\bar{x}^{(l)}$ becomes a new Pareto optimal solution.

The steps of the algorithm are the following [48]:

(1) Read k (number of objective functions), n (number of decision variables), l^a (number of solutions available).
(2) Set $f_{i1}^p = \infty$ for $i = 1, 2, \ldots, k$ and $j^a = 1$.
(3) Set $l = 1$.
(4) Read $\bar{x}^{(l)}$ and $f(\bar{x}^{(l)})$.
(5) Set $j = 1$.
(6) If for every $i \in I$ we have $f_i(\bar{x}^{(l)}) < f_{ij}^p$ then substitute $\bar{x}_j^p = \bar{x}^{(l)}$ and $f_j^p = f(\bar{x}^{(l)})$ and go to 10, otherwise go to 7.
(7) If for every $i \in I$ we have $f_i(\bar{x}^{(l)}) > f_{ij}^p$ then go to 10 otherwise go to 8.
(8) Set $j = j + 1$.
(9) If $j > j^a$ then $j^a = j^a + 1$ and $x_{ja}^p = \bar{x}^{(l)}$ and $\bar{f}_j^p = \bar{f}(\bar{x}^{(l)})$ and go to 10, otherwise go to 6.
(10) Set $l = l + 1$.
(11) If $l \leq l^a$, then go to 4, otherwise go to 12.
(12) Print $\bar{x}_j^p$ and $\bar{f}_j^p$ for $j = 1, 2, \ldots, j^a$.

This algorithm is called PARETO by Osyczka [48] and it was translated from FORTRAN to C and incorporated into MOSES.

3.2 The min-max algorithm

This algorithm chooses from any given set of solutions $L = \{1, 2, \ldots, l, \ldots, l^a\}$, the min-max optimal solution as defined in Chapter 1. We assume that the ideal vector $\bar{f}^0$ is given.

The steps of the algorithm are the following [48]:

(1) Read k (number of objective functions), n (number of decision variables), l^a (number of available solutions) ,$\bar{f}^0$ (ideal vector).

(2) Set $v_1^* = \infty$.

(3) Set $l = 1$.

(4) Read $\bar{x}^{(l)}$ and $\bar{f}(\bar{x}^{(l)})$.

(5) Evaluate vector $\bar{z}(\bar{x}^{(l)})$ using formula (12).

(6) If $\bar{z}(\bar{x}^{(l)}) = 0$ then retain this solution as the optimum since there is no better solution, and go to 11, otherwise go to 7.

(7) Find the maximal values of all the steps of formula (14) for the points $\bar{x}^{(l)}$. These values are denoted v_r for $r = 1, 2, \ldots, k$, and can be evaluated as follows

$$v_1 = \max_{i \in I} \{z_i(x^{(l)})\} \tag{21}$$

and then $I_1 = \{i_1\}$, where i_1 is the index for which the value of $z_i(\bar{x}^{(l)})$ is maximal,

$$v_2 = \max_{i \in I,\, i \notin I_1} \{z_i(\bar{x}^{(l)})\} \tag{22}$$

and the $I_2 = \{i_1, i_2\}$, where i_2 is the index for which the value of $z_i(\bar{x}^{(l)})$ is maximal,

$$v_r = \max_{i \in I,\, i \notin I_{r-1}} \{z_i(\bar{x}^{(l)})\} \tag{23}$$

and then $I_r = \{I_{r-1}, i_r\}$, where i_r is the index for which the value of $z_i(\bar{x}^{(l)})$ is maximal,

$$v_k = z_i(\bar{x}^{(l)}) \quad for\ i \in I\ and\ i \notin I_{k-1} \tag{24}$$

(8) Replace v_r^* by v_r for $r = 1, 2, \ldots, k$ and retain this solution as the optimum if the following function is satisfied

$$v_1 < v_1^* \bigvee_{r \in \{2, \ldots, k\}} ((v_r < v_r^*) \bigwedge_{s \in \{1, \ldots, r\}} (v_s = v_s^*)) \tag{25}$$

where $v_1^*, v_2^*, \ldots, v_k^*$ is the set of optimal values of relative increments ordered non-increasingly.

(9) Set $l = l + 1$.

(10) If $l \leq l^a$ then go to 4, otherwise go to 11.

(11) Print $x^*, l^*, \bar{f}(\bar{x}^*), \bar{z}(\bar{x}^*)$.

This algorithm is called MINMAX by Osyczka [48] and it was also translated from FORTRAN to C and incorporated into MOSES, both in the mathematical programming software and in some of the GA-based approaches.

3.3 Monte Carlo Methods

We also implemented the two Monte Carlo methods used by Osyczka [48] 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 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

$$x_i = x_i^a + \delta_i(x_i^b - x_i^a) \quad \text{for } i = 1, 2, \ldots, n \tag{26}$$

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 (26) 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 [48] to find the min-max optimum are presented next.

3.3.1 Monte Carlo method 1

In this method, the space of variables is explored twice, first searching for the ideal vector $\bar{f}^0$ and then searching for the min-max optimum. The algorithm is the following [48]:

Do steps 1, 2, 3, 4, for $l = 1, 2, \ldots, l^a$

(1) Generate a random point $\bar{x}^{(l)}$.

(2) If the point $\bar{x}^{(l)}$ is not in the feasible region go to 1, otherwise go to 3.

(3) Evaluate $f_i(\bar{x}^{(l)})$ for $i = 1, 2, \ldots, k$.

(4) Replace f_i^0 by $f_i(\bar{x}^{(l)})$ for every i for which $f_i(\bar{x}^{(l)}) < f_i^0$.

Do steps 5, 6, 7, 8, for $l = 1, 2, \ldots, l^a$

(5) Generate a random point $\bar{x}^{(l)}$.

(6) If the point $x^{(l)}$ is not in the feasible region go to 5, otherwise go to 7.

(7) Evaluate $f_i(\bar{x}^{(l)})$ for $i = 1, 2, \ldots, k$.

(8) Call MINMAX to check if the point $\bar{x}^{(l)}$ is the min-max optimum.

3.3.2 Monte Carlo method 2

Here, 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. The algorithm is the following [48]:

Do steps 1, 2, 3, 4, 5, for $l = 1, 2, \ldots, l^a$

(1) Generate a random point $\bar{x}^{(l)}$.

(2) If the point $\bar{x}^{(l)}$ is not in the feasible region go to 1, otherwise to 3.

(3) Evaluate $f_i(\bar{x}^{(l)})$ for $i = 1, 2, \ldots, k$.

(4) Replace f_i^0 by $f_i(\bar{x}^{(l)})$ for every i for which $f_i(\bar{x}^{(l)}) < f_i^0$.

(5) Call PARETO for checking if the point $\bar{x}^{(l)}$ is Pareto optimal.

Do steps 6, 7 for $j = 1, 2, \ldots, j^a$

(6) Evaluate $f_i(\bar{x}_j^p)$ for $i = 1, 2, \ldots, k$.

(7) Call MINMAX for checking if the point $\bar{x}_j^p$ 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.

3.4 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 [48]. A C translation of that code was incorporated into MOSES, and its contents are explained next.

Osyczka's system contains several multiobjective optimization methods:

(1) Min-max method : Equation (12) is used to determine the elements of the vector $\bar{z}(\bar{x})$.

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

$$f(\bar{x}) = \sum_{i=1}^{k} \left(\frac{f_i^0 - f_i(\bar{x})}{f_i^0} \right)^p \tag{27}$$

is used as the global function. For this formula Boychuk and Ovchinnikov [6] have suggested $p = 1$, and Salukvadze [57] 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

$$\forall_{i \in I}(z_i(\bar{x}) = max\{w_i z_i^{'}(\bar{x}), w_i z_i^{''}(\bar{x})\}) \tag{28}$$

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

(4) Pure weighting method : The equation

$$min \sum_{i=1}^{k} w_i f_i(\bar{x}) \tag{29}$$

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

$$\sum_{i=1}^{k} w_i = 1 \tag{30}$$

(5) Normalized weighting method : $\bar{f}(\bar{x})$ is used in equation (3.4).

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. For this purpose, the system includes a single criterion optimization technique:

(i) The flexible tolerance (FT) method : This 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. Normally sequential methods, even when 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 [31]. A detailed explanation of this algorithm and its implementation may be found in [10].

4 Genetic Algorithms

The famous naturalist Charles Darwin defined *Natural Selection* or *Survival of the Fittest* in his book [11] as the *preservation of favorable individual differences and variations, and the destruction of those that are injurious.* In nature, individuals must adapt to their environment in order to survive. This process is called *evolution*, in which those features that make an individual more suited to compete are preserved when it reproduces, and those features that make it weaker are eliminated. Such features are controlled by units called **genes** which form sets called **chromosomes**. Over subsequent generations not only the fittest individuals survive, but also their

fittest genes which are transmitted to their descendants during the sexual recombination process which is called **crossover**.

John H. Holland became interested in the application of natural selection to machine learning, and in the late 60s, while working at the University of Michigan, he developed a technique that allowed computer programs to mimic the process of evolution. Originally, this technique was called **reproductive plans**, but the term **genetic algorithm** became popular after the publication of his book [32] [33].

In 1989, Goldberg published a book [21] that provided a solid scientific basis for this area, and cited no less than 73 successful applications of the genetic algorithm. In the last few years the growing interest on this technique is reflected in a larger number of conferences, a new international journal, and an increasing amount of software and literature devoted to this subject.

Koza [40] provides a good definition of a GA:

> The **genetic algorithm** is a highly parallel mathematical algorithm that transforms a set (**population** of individual mathematical objects (typically fixed-length character strings patterned after chromosome strings), each with an associated **fitness** value, into a new population (i.e., the next **generation**) using operations patterned after the Darwinian principle of reproduction and survival of the fittest and after naturally occurring genetic operations (notably sexual recombination).

Actually, the genetic algorithm derives its behavior from a metaphor of one of the mechanisms of evolution in nature which is called **hard selection** [29]. Under this scheme, only the best available individuals are retained for generating descendants. This contrasts with **soft selection**, which offers a probabilistic mechanism for maintaining individuals to be parents of future progeny despite possessing relatively poorer objective values.

It has been argued [29] that the term **genetic algorithm** (GA) is misleading, since natural selection is only one of the mechanisms of evolution, and it would be more appropriate to call them **hard selection** (HS) algorithms to reflect the fact that they deal with only that particular selection scheme. However, the term is so common today, that a change does not seem feasible, at least in the near future.

A genetic algorithm for a particular problem must have the following five components [45]:

1. A representation for potential solutions to the problem.
2. A way to create an initial population of potential solutions.
3. An evaluation function that plays the role of the environment, rating solutions in terms of their "fitness".
4. Genetic operators that alter the composition of children.
5. Values for various parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators, etc.).

Some of the basic terminology used by the genetic algorithms (GAs) community is the following [29]:

- A **chromosome** is a data structure that holds a "string" of task parameters, or genes. This string may be stored, for example, as a binary bit-string (binary representation) or as an array of integers (floating point o real-coded representation) that represent a floating point number. This chromosome is analogous to the base-4 chromosomes present in our own DNA. Normally, in the GA community, the haploid model of a cell is assumed (one-chromosome individuals). However, diploids have also been used in the past [21].
- A **gene** is a subsection of a chromosome that usually encodes the value of a single parameter.
- An **allele** is the value of a gene. For example, for a binary representation each gene may have an allele of 0 or 1, and for a floating point representation, each gene may have an allele from 0 to 9.
- The **fitness** of an individual is a value that reflects its performance (i.e., how well solves a certain task). A **fitness function** is a mapping of the chromosomes in a population to their corresponding fitness values.

- A **genotype** represents a potential solution to a problem, and is basically the string of values chosen by the user, also called chromosome.
- A **phenotype** is the meaning of a particular chromosome, defined externally by the user.
- Genetic drift is the name given to the changes in gene/allele frequencies in a population over many generations, resulting from chance rather than from selection. It occurs most rapidly in small populations and can lead some alleles to become extinct, thus reducing the genetic variability in the population.
- A **niche** is a group of individuals which have similar fitness. Normally in multiobjective and multimodal optimization, a technique called **sharing** is used to reduce the fitness of those individuals who are in the same niche. This prevents the population from converging to a single solution, so that stable sub-populations can be formed, each one corresponding to a different objective or peak (in a multimodal optimization problem) of the function.

The basic operation of a Genetic Algorithm is illustrated in the following segment of pseudo-code [8]:

```
generate initial population, G(0);
evaluate G(0);
t:=0;
repeat
   t:=t+1;
   generate G(t) using G(t-1);
   evaluate G(t);
until a solution is found
```

First, an initial population is randomly generated. The individuals of this population will be a set of chromosomes or strings of characters (letters and/or numbers) that represent all the possible solutions to the problem. We apply a **fitness function** to each one of these chromosomes in order to measure the quality of the solution encoded by the chromosome. Knowing each chromosome's fitness, a **selection** process takes place to choose the individuals (presumably, the fittest) that will be the parents of the following generation. The most commonly used selection schemes are the following [23]:

- **Proportionate Reproduction:** This term is used generically to describe several selection schemes that choose individuals for birth according to their objective function values f. In these schemes, the probability of selection p of an individual from the ith class in the tth generation is calculated as

$$p_{i,t} = \frac{f_i}{\sum_{j=1}^{k} m_{j,t} f_j} \tag{31}$$

  where k classes exist and the total number of individuals sums to n. Several methods have been suggested for sampling this probability distribution, including Monte Carlo or **roulette wheel** selection [38], **stochastic remainder** selection [5] [7], and **stochastic universal** selection [4] [27].

- **Ranking Selection:** In this scheme, proposed by Baker [3] the population is sorted from best to worst, and each individual is copied as many times as it can, according to a non-increasing assignment function, and then proportionate selection is performed according to that assignment.
- **Tournament Selection:** The population is shuffled and then is divided into groups of k elements from which the best individual (i.e., the fittest) will be chosen. This process has to be repeated k times because on each iteration only m parents are selected, where

$$m = \frac{population\ size}{k}$$

  For example, if we use binary tournament selection ($k = 2$), then we have to shuffle the population twice, since in each stage half of the parents required will be selected. The interesting property of this selection

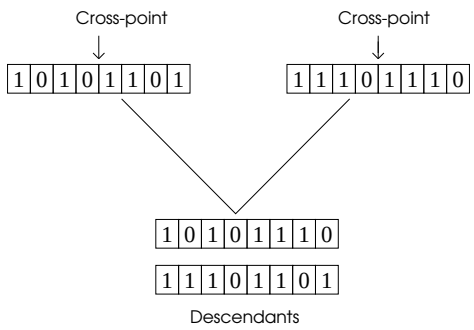

Figure 5: Use of a single-point crossover between two chromosomes. Notice that each pair of chromosomes produces two descendants for the next generation. The cross-point may be located at the string boundaries, in which case the crossover has no effect and the parents remain intact for the next generation.

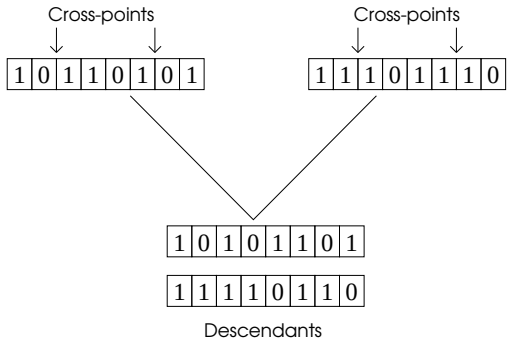

Figure 6: Use of a two-point crossover between two chromosomes. In this case the genes at the extremes are kept, and those in the middle part are exchanged. If one of the two cross-points happens to be at the string boundaries, a single-point crossover will be performed, and if both are at the string boundaries, the parents remain intact for the next generation.

scheme is that we can guarantee multiple copies of the fittest individual among the parents of the next generation.

After being selected, **crossover** takes place. During this stage, the genetic material of a pair of individuals is exchanged in order to create the population of the next generation. The two main ways of performing crossover are called single-point and two-point crossover. When a **single-point** crossover scheme is used, a position of the chromosome is randomly selected as the crossover point as indicated in Figure 5. When a **two-point** crossover scheme is used, two positions of the chromosome are randomly selected as indicated in Figure 6.

Mutation is another important genetic operator that randomly changes a gene of a chromosome. If we use a binary representation, a mutation changes a 0 to 1 and vice-versa. This operator allows the introduction of new chromosomic material to the population and, from the theoretical perspective, it assures that, given any population, the entire search space is connected [8].

If we knew the final solution in advance, it would be trivial to determine how to stop a genetic algorithm. However, as this is not normally the case, we have to use one of the two following criteria to stop the GA: either give a fixed number of generations in advance, or verify when the population has stabilized (i.e., all or most of the individuals have the same fitness).

GAs differ from traditional search techniques in several ways [8]:

1	0	1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0

Representation of the number 35.5072 using binary encoding

3	5	5	0	7	2

Representation of the number 35.5072 using floating point encoding

Figure 7: Representing the same number using binary and floating point encodings.

- GAs do not require problem specific knowledge to carry out a search.
- GAs use stochastic instead of deterministic operators and appear to be robust in noisy environments.
- In evaluating a population of n strings, the GA implicitly estimates the average fitnesses of all schemas that are present in the population, and increasing or decreasing their representation. This simultaneous implicit evaluation of large number of schemas in a population of n strings is known as *implicit parallelism*. This ability makes them less susceptible to local maxima and noise.

The traditional representation used by the genetic algorithms community is the binary scheme according to which a chromosome is a string the form $\langle b_1, b_2, \ldots, b_m \rangle$, where $b_1, b_2, \ldots, b_m$ are called **alleles** (either zeros or ones). Since the binary alphabet offers the maximum number of schemata per bit of information of any coding [21], its use has became very popular among scientists. This coding also facilitates theoretical analysis of the technique and allows elegant genetic operators. However, since the "implicit parallelism" property of GAs does not depend on using bit strings [45] it is worthwhile to experiment with larger alphabets, and even with new genetic operators. In particular, for optimization problems in which the parameters to be adjusted are continuous, a floating point representation scheme seems a logical choice. According to this representation, a chromosome is a string of the form $\langle d_1, d_2, \ldots, d_m \rangle$, where $d_1, d_2, \ldots, d_m$ are digits (numbers between zero and nine). Consider the examples shown in Figure 7, in which the same value is represented using binary and floating point encoding.

The term "floating" may seem misleading since the position of the implied decimal point is at a fixed position, and the term "fixed point representation" seems more appropriate. However, the reason that the term "floating point" is preferred is because in this representation each variable (representing a parameter to be optimized) may have the point at any position along the string. This means that even when the point is fixed for each gene, is not necessarily fixed along the chromosome. Therefore, some variables could have a precision of 3 decimal places, while others are integers, and still they could all be represented with the same string. The term **real-coded** GAs is also used in the literature [22] [65].

Floating point representation is faster and easier to implement, and provides a higher precision than its binary counterpart, particularly in large domains, where binary strings would be prohibitively long. One of the advantages of floating point representation is that it has the property that two points close to each other in the representation space must also be close in the problem space, and vice versa [45]. This is not generally true in the binary approach, where the distance in a representation is normally defined by the number of different bit positions.

Goldberg [22] has presented a theory of convergence for real-coded or floating-point GAs, and also real numbers and other alphabets have been proposed [65], particularly for numerical optimization, in a resemblance of the power of evolutionary strategies [59] in this domain. As Eshelman and Schaffer [16] point out, many researchers

in the GA community have agreed to use real-coded genetic algorithms for numerical optimization despite of the fact that there are theoretical arguments that seem to show that small alphabets should be more effective than large alphabets. Practitioners, on the other hand, have shown that real-coded genes work better in practice [12]. A few attempts have been made to develop a theoretical defense of this representation scheme, from which the recent work by Eshelman and Schaffer deserves special attention [16]. One of the main abilities of real-coded GAs is their capacity to exploit the gradualness of functions of continuous variables (where gradualness is taken to mean that small changes in the variables correspond to small changes in the function) [16] [65].

5 Multiobjective Optimization using GAs

Goldberg [21] indicates that the notion of genetic search in a multicriteria problem dates back to the late 60s, in which Rosenberg's [56] 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 [18]. 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.

5.1 Use of aggregating functions

Several attempts have been made to combine the objective functions in different ways, as Fonseca and Fleming report [18]. One general approach involves the use of aggregating functions. Several attempts are reported in the literature, and will be described next.

5.1.1 Weighted sum approach

Jakob et al. [37] assign weights that estimate the importance of each objective. The problem with this approach is precisely how to determine such weights when we do not have enough information about the problem. In this case, the optimal point obtained will be a function of the coefficients used to combine the objectives. We normally use a simple linear combination of the objectives and we can generate the trade-off surface (the term "trade-off" in this context refers to the fact that we are trading a value of one objective function for a value of another function or other functions) by varying the weights. This approach is very simple and easy to implement, but it has the disadvantage of missing concave portions of the trade-off curve [55].

5.1.2 Reduction to a single objective

Ritzel and Wayland [55] suggest coding the GA in such a way that all the objectives, except for one, are constant (constrained to a single value), and the remaining objective becomes the fitness function for the GA. Then, through a process of running the GA numerous times with different values of the constrained objectives, a trade-off surface can be developed. The obvious drawback of this approach is that it is time-consuming, and the coding of the objective functions may be difficult or even impossible for certain problems.

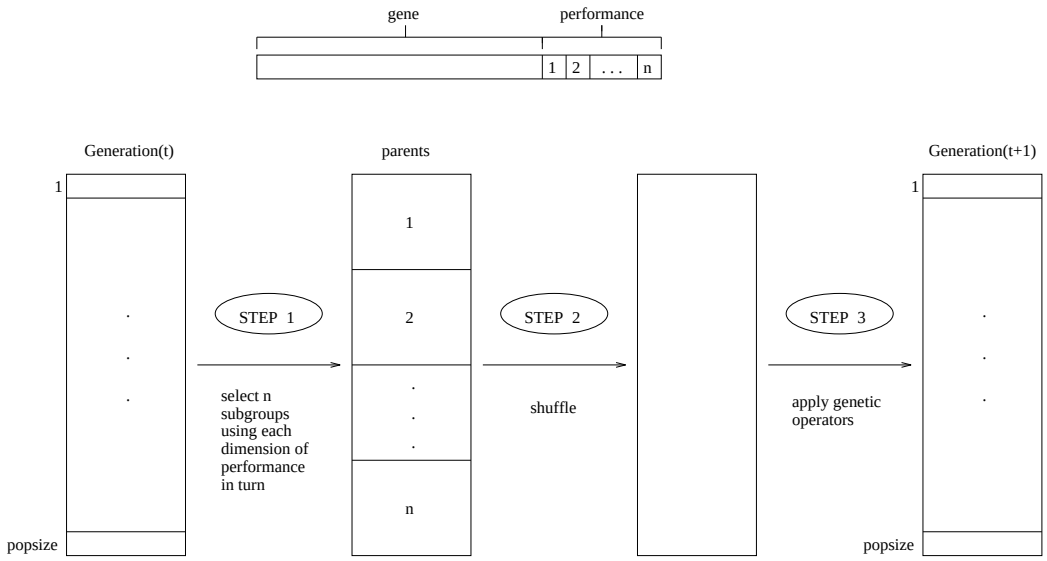

Figure 8: Schematic of VEGA selection.

5.1.3 Goal attainment

Wilson and Macleod [64] used this method to solve an optimization problem. In this method, a vector of weights relating the relative under- or over-attainment of the desired goals must be elicited from the decision maker in addition to the goal vector. By varying the weights, we can generate the set of noninferior solutions, even for nonconvex problems [9]. In the case of underattainment of the desired goals, a smaller weighting coefficient is associated with a more important objective. For overattainment of the desired goals, a smaller weighting coefficient is associated with a less important objective [53].

5.1.4 Use of Penalty Functions

The basic idea of this approach is to "punish" the fitness value of a chromosome whenever the solution produced violates some of the constraints imposed by the problem. Theoretically, the penalty decreases when the value of the penalty function coefficient is increased and convergence is achieved by increasing the penalty function coefficient to infinity [1]. However, a large value for the penalty function coefficient causes ill conditioning in the optimization process and results in numerical instability or slow convergence. Furthermore, since the value of the penalty function coefficient is unknown, much experimentation is required to find an appropriate value. The augmented Lagrangian method has been suggested by Adeli and Cheng [1] to deal with this problem. They integrate the penalty function method with the primal-dual method, which is based on sequential minimization of the Lagrangian function. Instead of only a single penalty function coefficient, in this approach two parameters associated with each constraint are used, and there is no need for the Lagrangian multipliers go to infinity to ensure convergence. Nevertheless, the problem that remains is that penalty functions are generally problem dependent, and therefore difficult to establish.

5.2 Non-Pareto approaches

To overcome the difficulties involved in the aggregating approach, much work has been devoted to the development of alternative approaches based on ranking [51]. We will examine next some of the most popular **non-Pareto approaches**.

5.2.1 VEGA

David Schaffer [58] extended Grefenstette's GENESIS program [25] 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 from GENESIS 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 8 (taken from Schaffer [58]). 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 [58]. 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 "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. [54] 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 [18]. 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 [54]. 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.

5.2.2 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 [53]. Fourman [19] 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 [18]. 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.

5.2.3 Evolutionary Strategies

Kursawe [41] formulated a multiobjective version of evolutionary strategies [59] (ESs). Selection consisted of as many steps as objective functions had the problem. At each step, one of these objectives was selected randomly according to a probability vector, and used to delete a fraction of the current population. After selection, the

survivors became the parents of the next generation. The map of the trade-off surface was produced from the points evaluated during the run. Since the environment was allowed to change over time, diploid individuals were necessary to keep recessive information stored.

5.2.4 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.

5.3 Pareto-based approaches

We will also review some of the main **Pareto-based approaches**.

5.3.1 Pareto-based fitness assignment

This approach was first proposed by Goldberg [21] to solve the problems of Schaffer's approach. He suggested the use of non-domination ranking and selection to move a population toward the Pareto front in a multiobjective problem. The basic idea is to find the set of strings in the population that are Pareto non-dominated by the rest of the population. These strings are then assigned the highest rank and eliminated from further contention. Another set of Pareto nondominated strings are determined from the remaining population and are assigned the next highest rank. This process continues until the population is suitably ranked. Goldberg also suggested the use of some kind of niching to keep the GA from converging to a single point on the front. A niching mechanism such as sharing [24] would allow the GA to maintain individuals all along the non-dominated frontier. Hilliard et al. [30] used a Pareto optimality ranking method to handle the objectives of minimizing cost and minimizing delay in a scheduling problem. They tentatively concluded that the Pareto optimality ranking method outperformed the VEGA method. The Pareto method was found to be superior to a VEGA by Liepins et al. [42] when applied to a variety of set covering problems. Ritzel et al. [55] also used non-dominated ranking and selection combined with deterministic crowding [44] as the niching mechanism. They applied the GA to a groundwater pollution containment problem in which cost and reliability were the objectives. Though the actual Pareto front was unknown, Ritzel et al. used the best trade-off surface found by a domain-specific algorithm, called MICCP (Mixed Integer Chance Constrained Programming), to compare the performance of the GA. They found that selection according to Pareto non-domination was superior to both VEGA and non-domination with deterministic crowding, at least for finding points near or on the front found by MICCP.

5.3.2 Multiple Objective Genetic Algorithm

Fonseca and Fleming [17] 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 [17]:

$$rank(x_i, t) = 1 + p_i^{(t)} \tag{32}$$

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 [17]:

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.

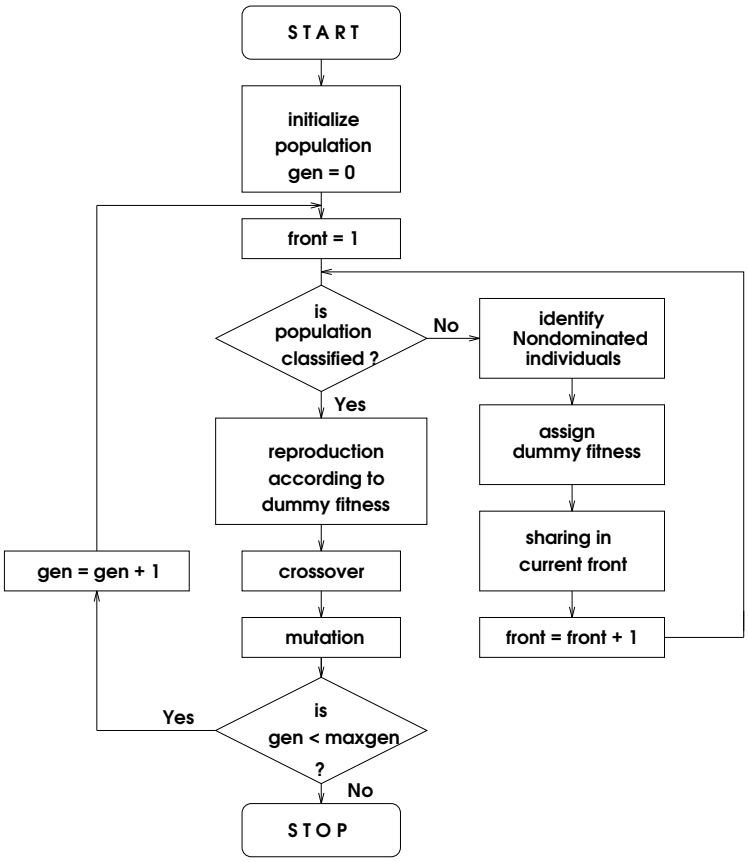

Figure 9: Flowchart of the Nondominated Sorting Genetic Algorithm (NSGA).

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 [23] 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 use 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 [61]. 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.

In this approach, it is possible to evolve only a certain region of the trade-off surface, by combining Pareto dominance with partial preference information in the form of a goal vector. While the basic ranking scheme remains unaltered, as we perform a Pareto comparison of the individuals, then those objectives which already satisfy their goals will not be selected. If we specify fully unattainable goals, then objectives will never be excluded from comparison. Changing the goal values during the search alters the fitness landscape accordingly and allows the decision maker to magnify a particular region of the trade-off surface.

5.3.3 Non-dominated Sorting Genetic Algorithm

The Non-dominated Sorting Genetic Algorithm (NSGA) was proposed by Srinivas and Deb [60], 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 [61], and both maximimization and minimization problems can be handled. Figure 9 (taken from Srinivas and Deb [61]) shows the general flow chart of this approach.

5.3.4 Niched Pareto GA

Horn and Nafpliotis [34] 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 [24]. 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 [18].

The pseudocode for Pareto domination tournaments assuming that all of the objectives are to be maximized is presented below [34]. 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 [18],

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 noted that even when Pareto-based ranking correctly assigns all non-dominated individuals the same fitness, it does not guarantee that the Pareto set will 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 [18]. 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 [18].

Goldberg and Richardson [24] proposed the used of fitness sharing to prevent em genetic drift and to promote the sampling of the whole Pareto set by the population. Fonseca and Fleming [17] 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 [34] 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**.

Another interesting aspect that has been considered in GA-based multiobjective optimization has been the viability of crossover. This is an important issue, because we could have different genetic representations across different regions of the trade-off surface, and therefore we could need to restrict crossover to happen only locally [21]. So far, crossover restrictions have been implemented based on the distance between individuals in the objective domain, either directly [17] or indirectly [28].

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 opportunity 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 A New GA-based Approach Based on Min-Max Selection with Sharing

This is another new approach in which a Min-Max selection strategy replaces the Pareto ranking selection scheme previously reported in the literature, and sharing is used to avoid the GA converging to a single solution. The basic algorithm is the following:

1. The initial population is generated as in the previous approach, ensuring that all the individuals at generation zero encode only valid solutions.

2. By exploring the population at each generation, the local ideal vector is produced. This is done by comparing the values of each objective function in the entire population.

3. The binary tournament selection algorithm is modified, so that instead of comparing the fitnesses of two individuals, we compare their maximal deviations with respect to the local ideal vector. If one dominates the other, then it wins the tournament, and if there is a tie, then sharing is used to decide who is the winner, in a way similar to the NPGA. This means that we count the number of individuals within the niche of each one of the competitors, and the individual with a lower count wins.

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

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

6.2 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) [26]. 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, in addition to 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

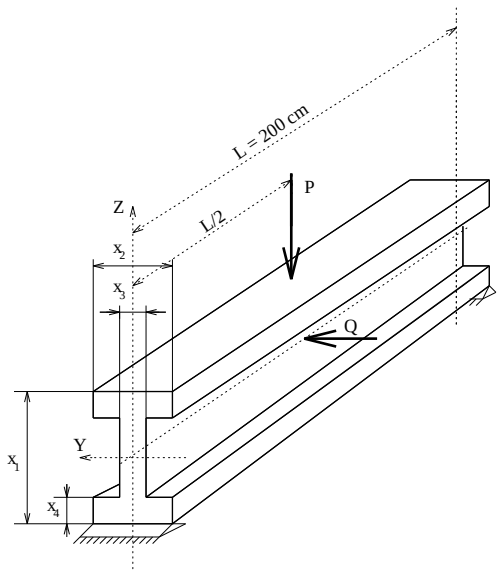

Figure 10: The simply supported I-beam of Example 1.

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 [15], 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 Examples

To illustrate the use of MOSES and the efficiency of the two new techniques proposed, we selected two engineering design examples from the literature [10]. Since it is generally intractable to obtain an analytical representation of the Pareto front, it is usually very difficult to measure the performance of a multiobjective optimization technique. For the purposes of this paper we compared the results only in terms of the best trade-offs that could be achieved. For that sake, we used the expression

$$L_p(f) = \sum_{i=1}^{k} w_i \left| \frac{f_i^0 - f_i(x)}{\rho_i} \right| \tag{33}$$

where k is the number of objectives, $\rho_i = f_i^0$,or $f_i(x)$, depending on which gives the maximum value for $L_p(f)$, and w_i refers to the weight assigned to each objective (if not known, equal weights are assigned to all the objectives).

7.1 Example 1 : Design of an I-beam

The multiobjective optimization problem is formulated as follows [49]:

> Find the dimensions of the beam presented in Figure 10 (taken from Osyczka [49]) which satisfy the geometric and strength constraints and which optimize the following criteria:
>
> 1. cross section area of the beam which for the given length minimizes its volume; and

2. static deflection of the beam for the displacement under the force P.

Both criteria are to be minimized. It should be noted that these criteria are contrary to one another (i.e., the best solution for the first objective function gives the worst solution for the second one and viceversa).

It is assumed that:

1. Permissible bending stress of the beam material $k_g = 16\ kN/cm^2$.
2. Young's Modulus of Elasticity $E = 2 \times 10^4\ kN/cm^2$.
3. Maximal bending forces $P = 600\ kN$ and $Q = 50\ kN$.

The vector of the decision variables is $x = [x_1, x_2, x_3, x_4]^T$. Their values will be given in centimeters. The geometric constraints are:

$$10 \leq x_1 \leq 80, \quad 10 \leq x_2 \leq 50, \quad 0.9 \leq x_3 \leq 5, \quad 0.9 \leq x_4 \leq 5 \tag{34}$$

The strength constraint is

$$\frac{M_y}{W_y} + \frac{M_z}{W_z} \leq k_g \tag{35}$$

where M_y and M_z are maximal bending moments in Y and Z directions respectively; W_y and W_z are section moduli in Y and Z directions respectively. For the forces acting the values of M_y and M_z are $30,000\ kN - cm$ and $2,500\ kN - cm$ respectively. The section moduli can be expressed as follows:

$$W_y = \frac{x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)]}{6x_1} \tag{36}$$

$$W_z = \frac{(x_1 - 2x_4)x_3^3 + 2x_4x_2^3}{6x_2} \tag{37}$$

Thus the strength constraint is:

$$16 - \frac{180,000x_1}{x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)]} - \frac{15,000x_2}{(x_1 - 2x_4)x_3^3 + 2x_4x_2^3} \geq 0 \tag{38}$$

The objective functions can be expressed as follows

1. Cross-section area

$$f_1(x) = 2x_2x_4 + x_3(x_1 - 2x_4)\ cm^2 \tag{39}$$

2. Static deflection

$$f_2(x) = \frac{Pl^3}{48EI}\ cm \tag{40}$$

where I is the moment of inertia which can be calculated from

$$I = \frac{x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)]}{12} \tag{41}$$

After substitution the second objective function is

$$f_2(x) = \frac{60,000}{x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)]} \tag{42}$$

The solution to this problem, as well as Example 2, will be discussed in section 8.

Test No.	v (sfm)	f (ipr)	d (in)	SR (μin)	SI (% undamaged)	TL (min)	MRR (in^3/min)
1	625	0.002	0.050	25	24	150.6	0.75
2	625	0.010	0.050	150	31	108.1	3.75
3	625	0.018	0.050	230	19	89.8	6.75
4	625	0.010	0.097	86	29	80.2	7.28
5	625	0.018	0.097	180	20	29.4	13.10
6	966	0.005	0.078	30	55	32.7	4.52
7	966	0.015	0.078	210	45	24.2	13.56
8	1200	0.010	0.050	95	30	30.5	7.20

Table 1: Machinability data for 390 die cast/carbonide/wet.

7.2 Example 2 : Machining recommendations

The problem is the following [20]:

Machinability tests on 390 die cast aluminum cut with VC-3 carbide cutting tools were conducted over the following ranges of speed, feed rates, and depths of cut:

Cutting speed (v) :	600 sfm to 1200 sfm
Feed rate (f):	0.002 ipr to 0.018 ipr
Depth of cut (d):	0.050 in to 0.100 in

Table 1 (taken from Ghiassi et al. [20]) provides the cutting conditions and associated machining performance measures (criteria). The data in this table were used to develop first-order predicting equations for the performance variables SR, SI, TL, and MRR in terms of the controllable variables v, f and d, in logarithmic transformed coordinates. The following equations represent the least-squares fit to the data; the feed and depth of cut have been multiplied by 1000 to ensure that their logarithms are positive.

$$\begin{aligned} \ln SR &= 7.49 - 0.44 \ln v + 1.16 ln(1000f) - 0.61 \ln(1000d) \\ \ln SI &= -4.13 + 0.92 \ln v - 0.16 \ln(1000f) + 0.43 \ln(1000d) \\ \ln TL &= 21.90 - 1.94 \ln v - 0.30 \ln(1000f) - 1.04 \ln(1000d) \\ \ln MRR &= -11.33 + \ln v + \ln(1000f) + \ln(1000d) \end{aligned} \tag{43}$$

Bounds on the values of the controllable variables are defined below to reflect the ranges over which the machinability tests were run, and bounds on the values of the performance variables.

$$\begin{aligned} 600 \leq v \leq 1200 \ sfm \\ 0.002 \leq f \leq 0.018 \ ipr \\ 0.05 \leq d \leq 0.10 \ in \end{aligned} \tag{44}$$

$$\begin{aligned} SR \leq 75 \ \mu in \\ SI \geq 50\% \ undamaged \\ TL \geq 30 \ min \end{aligned} \tag{45}$$

In order to express the variables in Equations (44) and (45) in the same form as in the performance model, Equation (43), their logarithmic transformations are given in Equations (46) and (47):

$$\begin{aligned} 6.3969 \leq \ln v \leq 7.0901 \\ 0.6931 \leq \ln(1000f) \leq 2.8904 \\ 3.9120 \leq \ln(1000d) \leq 4.6052 \end{aligned} \tag{46}$$

$$
\begin{aligned}
-0.44\ln v + 1.16ln(1000f) - 0.61\ln(1000d) &\leq -3.17 \\
-0.92\ln v + 0.16\ln(1000f) - 0.43\ln(1000d) &\leq -8.04 \\
1.94\ln v + 0.30\ln(1000f) + 1.04\ln(1000d) &\leq 18.50
\end{aligned} \tag{47}
$$

The natural logarithms of SR, TL and SI have been substituted from Equation (43) to obtain Equation (47). The constraint of TL was multiplied by -1 to maintain the same direction of inequality as the other constraints.

To simplify the statement of the problem, let $x_1 = \ln v$, $x_2 = \ln(1000f)$, $x_3 = \ln(1000d)$, and the vector $x = (x_1, x_2, x_3)$. Also, define $z_1(x) = \ln SR$, $z_2(x) = \ln SI$, $z_3(x) = \ln TL$, and $z_4(x) = \ln MRR$. Again, for simplicity, we will refer to the objective functions as z_1, z_2, z_3 and z_4, respectively, and define the vector $z = (-z_1, z_2, z_3, z_4)$. Note that minimizing $z_1 = \ln SR$ is equivalent to maximizing $-z_1$. Accordingly, the components of z [from Equation (1)] may be expressed as:

$$
\begin{aligned}
-z_1 &= -7.49 + 0.44x_1 - 1.16x_2 + 0.61x_3 \\
z_2 &= -4.13 + 0.92x_1 - 0.16x_2 + 0.43x_3 \\
z_3 &= 21.90 - 1.94x_1 - 0.30x_2 - 1.04x_3 \\
z_4 &= -11.331 + x_1 + x_2 + x_3
\end{aligned} \tag{48}
$$

Maximize z, as defined by Equation (48), under the following constraints:

$$
\begin{aligned}
6.3969 &\leq x_1 \leq 7.0901 \\
0.6931 &\leq x_2 \leq 2.8904 \\
3.9120 &\leq x_3 \leq 4.6052
\end{aligned} \tag{49}
$$

$$
\begin{aligned}
-0.44x_1 + 1.16x_2 - 0.61x_3 &\leq -3.1725 \\
-0.92x_1 + 0.16x_2 - 0.43x_3 &\leq -8.0420 \\
1.94x_1 - 0.30x_2 - 1.04x_3 &\leq 18.4988
\end{aligned} \tag{50}
$$

Inequalities (49) and (50) correspond to inequalities (46) and (47), respectively.

8 Comparison of Results

We will compare the ideal vector that each method generates with the best results reported in the Literature. 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).

8.1 Example 1

The ideal vector of this problem was computed using Monte Carlo methods 1 and 2 (generating 100 points), Osyczka's multiobjective optimization system and a GA (with a population of 100 chromosomes running during 50 generations) using binary and floating point representation, with the procedure described before to adjust its parameters. The corresponding results are shown in Table 2, including the best results reported in the literature [49]. 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].

As we can see in Table 3, the two new GA-based approaches proposed by the authors, named GAminmax1 and GAminmax2 respectively, provide the best overall results when a floating point representation is used. The second method slightly improves the result obtained using the first method, although it should be mentioned that the first technique does not require any sort of niching parameters as the second approach. Population size and

Method	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{f}_1$	$\mathbf{f}_2$
Monte Carlo 1	30.84	28.26	3.79	4.06	**188.65**	0.06175
Monte Carlo 1	52.97	44.08	1.99	0.99	555.22	**0.00849**
Min-Max (OS)	74.97	44.97	1.97	1.97	**316.85**	0.01697
Min-Max (OS)	74.99	44.99	1.99	2.06	326.49	**0.01636**
GA (Binary)	66.39	38.63	0.90	0.91	**128.27**	0.05241
GA (Binary)	80.00	50.00	4.99	4.99	848.41	**0.00591**
GA (FP)	61.14	41.14	0.90	0.90	**127.46**	0.06034
GA (FP)	80.00	50.00	5.00	5.00	850.00	**0.00590**
Literature	60.70	49.90	0.90	0.90	**128.47**	0.060
Literature	80.00	50.00	5.00	5.00	850.00	**0.0059**

Table 2: Comparison of results computing the ideal vector of the first example (design of an I-beam). For each method the best results for optimum f_1 and f_2 are shown in **boldface**. OS stands for Osyczka's Multiobjective Optimization System. Every objective is being minimized.

niching parameters play vital roles in numerical optimization with GAs, so it is important to choose appropriate values for them. Some guidelines for this are presented next. Additional details may be found in [10].

From our experience in numerical optimization, we advise the use of populations of at least as many individuals as the length of the chromosome, and to use twice that amount when possible (i.e., when the CPU time required for the analysis is not too high). As we have stated before [10], floating point representation produces better results and in a shorter period of time, because the chromosomic strings required under this representation scheme are always shorter and easier to decode than their binary counterparts. For this problem, for instance, the chromosome length is 49 using binary representation, but only 16 using floating point representation.

The niche size of our second approach is computed following the work by Deb and Goldberg [13] in which sharing is done over the parameters. The principle to derive such an estimate is to assume that each niche is enclosed in a p-dimensional hypersphere of radius σ_{share} such that each sphere encloses $\frac{1}{q}$ of the volume of the space, where q is the number of peaks in the solution space. The radius of a hypersphere containing the entire space is calculated as [13]

$$r = \frac{1}{2}\sqrt{\sum_{k=1}^{p}(x_{k,max} - x_{k,min})^2} \tag{51}$$

and the volume is calculated as $V = cr^p$ with c a constant. Dividing this volume in q parts and recognizing that the hypervolume has the same form regardless of size, σ_{share} may be calculated as:

$$\sigma_{share} = \frac{r}{\sqrt[p]{q}} \tag{52}$$

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 approaches ensure 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 problem, the results for GALC (see Table 3) show the contrary even for this simple bi-objective problem. Finally, it is interesting to notice how some simple approaches, like Lexicographic ordering (in which an objective is randomly selected at each turn) work remarkably well with problems that have few objectives, like this. Our first approach requires the ideal vector (computed previously with another GA), and a set of weights to delineate the Pareto front. For this example, ten weights were used (all combinations of two, from 0.1 to 0.9). Our second approach computes the ideal vector during run-time, but it requires niching parameters to avoid convergence to a single solution, as indicated before.

Method	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{f}_1$	$\mathbf{f}_2$	$\mathbf{L}_p(f)$
Ideal Vector					127.46	0.0059	0.000000
Monte Carlo 1	77.57	20.59	3.47	3.88	401.77	0.0159	3.837581
Monte Carlo 2	75.01	31.02	1.76	2.48	277.09	0.0198	3.525181
Min-max (OS)	75.06	44.99	1.99	1.99	320.55	0.0167	3.350946
GCM (OS)	75.06	44.99	1.99	1.99	320.55	0.0167	3.350946
WMM (OS)	75.06	44.99	1.99	1.99	320.55	0.0167	3.350946
PMM (OS)	74.97	44.97	1.97	1.97	316.85	0.0170	3.360191
NMM (OS)	74.99	44.99	1.99	2.06	326.49	0.0164	3.332501
GALC (B)	80.00	50.00	0.92	3.98	463.99	0.0083	3.042494
GALC (FP)	80.00	50.00	0.90	3.80	445.55	0.0086	2.953417
Lexicographic (B)	80.00	45.25	0.98	2.73	319.95	0.0124	2.614093
Lexicographic (FP)	80.00	50.00	0.90	2.26	293.74	0.0134	2.572228
VEGA (B)	80.00	50.00	0.94	2.24	295.59	0.0134	2.589958
VEGA (FP)	80.00	23.33	3.52	5.00	479.49	0.0116	3.736026
NSGA (B)	80.00	44.28	2.39	4.35	555.19	0.0080	3.714160
NSGA (FP)	80.00	50.00	5.00	1.18	506.56	0.0132	4.210141
MOGA (B)	80.00	46.48	1.29	2.70	347.27	0.0119	2.743380
MOGA (FP)	80.00	30.38	0.90	3.53	279.95	0.0146	2.668388
NPGA (B)	78.75	36.69	1.40	3.71	372.17	0.0117	2.909137
NPGA (FP)	78.52	29.36	2.51	2.74	344.44	0.0160	3.409632
Hajela (B)	80.00	50.00	0.90	4.72	535.48	0.0072	3.418376
Hajela (FP)	80.00	50.00	1.92	5.00	634.05	0.0066	4.090622
GAminmax1 (B)	80.00	40.58	0.92	3.02	312.77	0.0127	2.603628
GAminmax1 (FP)	80.00	50.00	0.90	2.43	310.33	0.0126	2.568096
GAminmax2 (B)	80.00	49.59	1.12	2.33	315.36	0.0129	2.653479
GAminmax2 (FP)	80.00	50.00	0.90	2.35	303.06	0.0129	2.567664

Table 3: Comparison of the best overall solution found by each one of the methods included in MOSES for the first example (design of an I-beam). 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 0.5 (equal weight for both objectives). Each objective is being minimized.

Method	v (sfm)	f (ipr)	d (in)	SR (μin)	SI (% udmg)	TL (min)	MRR (in^3/min)
Monte Carlo 1	1105.94	0.0020	0.0695	**14.03**	56.15	39.67	1.87
Monte Carlo 1	1105.94	0.0020	0.0695	14.03	**56.15**	39.67	1.87
Monte Carlo 1	1014.64	0.0026	0.0739	18.78	51.18	**40.76**	2.35
Monte Carlo 1	949.56	0.00330	0.0925	22.16	51.06	34.22	**3.48**
Min-Max (OS)	1200.00	0.0016	0.0496	**13.02**	54.18	51.14	1.18
Min-Max (OS)	1200.00	0.0016	0.0496	13.02	**54.18**	51.14	1.18
Min-Max (OS)	1200.00	0.0016	0.0496	13.02	54.18	**51.14**	1.18
Min-Max (OS)	1200.00	0.0026	0.0499	21.95	50.49	44.43	**1.86**
GA (Binary)	1045.64	0.0020	0.100	**11.31**	62.52	30.42	2.51
GA (Binary)	1200.00	0.0020	0.0783	12.36	**63.88**	30.03	2.25
GA (Binary)	1020.42	0.0020	0.0628	15.18	50.05	**51.74**	1.54
GA (Binary)	1074.73	0.0035	0.0819	24.16	53.80	30.01	**3.70**
GA (FP)	1053.00	0.0020	0.1000	**11.28**	62.93	30.01	2.53
GA (FP)	1053.00	0.0020	0.1000	11.28	**62.93**	30.01	2.53
GA (FP)	1134.11	0.0020	0.0500	16.66	50.01	**53.43**	1.36
GA (FP)	952.98	0.0042	0.0960	28.57	50.09	30.42	**4.61**
Literature	1048.0	0.0020	0.1000	**11.30**	62.65	30.29	2.51
Literature	1200.0	0.002	0.0776	12.43	**63.64**	30.31	2.23
Literature	840.0	0.002	0.1000	12.46	51.11	**46.52**	2.02
Literature	944.0	0.004	0.1000	25.60	51.16	30.38	**4.40**

Table 4: Comparison of results computing the ideal vector of the second example (machining recommendations). For each method the best results for each objective function are shown in **boldface**. OS stands for Osyczka's Multiobjective Optimization System. SR is being minimized, and SI, TL and MRR are being maximized.

8.2 Example 2

The second example (machinign recommendations) is very interesting, because it has four objective functions, but the ideal vector is practically achievable [10]. In this case we also used a population size of 100 and the GA was executed for 50 generations. For the Monte Carlo methods and the Osyczka multiobjective optimization system, we generated 100 initial points.

The ideal vector is displayed on Table 4. Again, a combination of the results produced by the GA under binary and floating point representations provides an ideal vector better than the solution previously reported in the literature.

As we can see in Table 5, the two new GA-based approaches proposed by the authors provide again the best overall results when a floating point representation is used. The second method is practically able to find the ideal vector, which is normally only targetable with much larger populations [10]. The chromosome length required for this example is 40 under binary representation and 13 using floating point representation.

The Lexicographic method, like most of the other GA-based techniques, performs very poorly in this problem, indicating, as we mentioned before, its incapability to keep its good performance when the number of objective functions is increased. As expected, a simple linear combination of objectives provides an extremely poor solution. Hajela's approach (very similar to our weighted min-max technique) gets a very good solution, but unfortunately this good behavior is rather erratical [10]. NPGA, on the other hand, is normally very good to find the Pareto front and to keep the GA from converging to a single solution, but it is not as good for finding the best trade-off because it never manipulates the real objective function values within the GA.

Through these results we can see that the second approach proposed by the authors provides remarkable solutions when a proper niche size is used. However, a disadvantage of this method is that it is normally intended to produce a single solution, since it turns out to be quite difficult to keep the GA from converging to a unique value after 50 generations. This is fine for many engineering applications, but it could be undesirable in problems in which several possible trade-offs exist with equivalent advantages each, and the designer wants to be able to see them all to decide which one to choose. For that sake, the first method proposed by the authors works a lot better, since it uses weights to let the designer determine the relative importance of the objectives. Normally, just a few weights are necessary (in this example we used, for instance, 15 weights combinations) to get a reasonably good trade-off, and it is generally the case that each of this weight combinations will produce a different non-dominated solution. Therefore, using our first approach, the designer is able to see a good portion of the Pareto front, so that he/she can decide which solution to adopt based on his/her own needs. Although this fist approach emphasizes the generation of the Pareto front, the best trade-off that it finds is still better than those found by the rest of the techniques included in MOSES (see Tables 3 and 5).

9 Conclusions

We have proposed two multiobjective optimization methods based on the min-max optimization approach. The first approach is very robust because it transforms the multiobjective optimization problem into several single objective optimization problems, and it works very well independently of the representation scheme used. However, a floating point representation seems to work better for numerical optimization applications with any of the two approaches proposed. The main drawbacks of the first min-max approach proposed is that it requires the ideal vector and a set of weights to delineate the Pareto set. Nevertheless, when the ideal vector is not known, a set of target (desirable) values for each objective can be provided instead. Also, finding proper weights is normally an easy task, since not many of them are required to get reasonably good results.

The second technique that we proposed does not require the ideal vector, since it is able to compute it based on the local populations generated. However, to avoid convergence to a single solution, a form of sharing similar to that employed by the NPGA (Niched Pareto Genetic Algorithm) was implemented, but the problem of finding an optimum tournament size was eliminated by using a min-max binary tournament selection strategy. Nevertheless, the niche sharing factor still has to be provided by the user. For that purpose, we have proposed the use of sharing on the parameter values by computing the optimum niche size using the guidelines provided by Goldberg and Deb [13]. Our second method is very fast and reliable except in cases in which it is possible to find solutions that highly

Method	v (sfm)	f (ipr)	d (in)	SR (μin)	SI (% udmg)	TL (min)	MRR (in^3/min)	L_p(f)
Ideal Vector				11.28	63.88	53.43	4.61	0.000000
Monte Carlo 1	914.37	0.0030	0.0969	19.55	51.11	36.12	3.18	1.385445
Monte Carlo 2	1014.64	0.0026	0.0739	18.77	51.18	40.76	2.35	1.279883
Min-max (OS)	1200.00	0.0026	0.0499	21.95	50.49	44.43	1.86	1.889006
GCM (OS)	1200.00	0.0026	0.0499	21.95	50.49	44.43	1.86	1.889006
WMM (OS)	1200.00	0.0026	0.0499	21.95	50.49	44.43	1.86	1.889006
PMM (OS)	1200.00	0.0016	0.0496	13.02	54.18	51.14	1.18	1.856245
NMM (OS)	1200.00	0.0016	0.0496	13.02	54.18	51.14	1.18	1.856245
GALC (B)	1200.00	0.0044	0.0607	36.03	50.47	30.89	3.84	2.944513
GALC (FP)	987.57	0.0020	0.1000	11.60	59.32	33.98	2.37	0.280571
Lexicographic (B)	1200.00	0.0020	0.0657	13.76	59.24	36.05	1.89	0.745638
Lexicographic (FP)	1200.00	0.0020	0.0759	12.60	63.03	31.02	2.19	0.295288
VEGA (B)	1200.00	0.0020	0.0781	12.38	63.81	30.11	2.25	0.230000
VEGA (FP)	1037.39	0.0020	0.1000	11.35	62.07	30.89	2.49	0.064436
NSGA (B)	1200.00	0.0020	0.0691	13.34	60.54	34.20	1.99	0.586649
NSGA (FP)	959.16	0.0020	0.1000	11.75	57.75	35.96	2.30	0.411813
MOGA (B)	1118.34	0.0021	0.0868	12.67	62.10	30.49	2.45	0.184670
MOGA (FP)	1200.00	0.0020	0.0740	12.79	62.35	31.85	2.13	0.371517
NPGA (B)	1075.07	0.0021	0.0939	12.29	61.94	30.33	2.54	0.122571
NPGA (FP)	1094.35	0.0020	0.0927	11.61	63.11	30.13	2.43	0.073782
Hajela (B)	1045.55	0.0020	0.1000	11.31	62.52	30.42	2.51	0.030583
Hajela (FP)	1200.00	0.0034	0.0556	28.19	50.65	36.57	2.72	1.990499
GAminmax1 (B)	1082.18	0.0020	0.0943	11.55	62.92	30.25	2.45	0.063389
GAminmax1 (FP)	1049.50	0.0020	0.1000	11.29	62.73	30.20	2.52	0.014317
GAminmax2 (B)	992.77	0.0020	0.0989	11.65	59.32	34.03	2.36	0.291994
GAminmax2 (FP)	1052.99	0.0020	0.1000	11.28	62.93	30.01	2.53	0.000012

Table 5: Comparison of the best overall solution found by each one of the methods included in MOSES for the second example (machining recommendations). 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 0.25 (equal weight for every objective). SR is being minimized, and SI, TL and MRR are being maximized.

favor more than one objective (namely when some elements of the ideal vector are achievable) but that highly disfavors other objectives [10]. This behavior is common in highly convex search spaces which are unfortunately very common in engineering optimization problems and more work is required to extend this method to deal with such situations.

The two techniques that we developed ensure 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 algorithms. This property makes our approaches 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 multiobjective optimization methods should be obvious, since no other similar tools, combining GA-based approaches with mathematical programming techniques, were previously available. Additional details may be found in [10]. Also, the system is a valuable tool, as it is, for engineering design optimization, because of the variety of different approaches that it contains.

10 Future Work

Much additional work remains to be done, since this is a very broad area of research. For example, it is desirable to do more theoretical work on niches and population sizes for multiobjective optimization problems to verify our empirical results. In that sense, we expect that MOSES may be useful as an experimentation tool for those interested in this area. To talk about convergence in this context seems a rather difficult task, since there is no common agreement on what optimum really means. However, if we use concepts from Operations Research such as the min-max optimum, it should be possible to develop such a theory of convergence for these kinds of problems. Also, 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 work on a theoretical framework for the excellent performance of real-coded GAs so that practice can finally meet theory in numerical optimization problems.

References

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

[2] H. Baier. Uber algorithmen zur emittlung und charakterisierung pareto-optimaler losungen bei entwurfsaufgaben elastischer tragwerke. *ZAMM*, 57(22):318–320, 1977.

[3] J. E. Baker. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 100–111, Hillsdale, New Jersey, 1985. Lawrence Erlbaum.

[4] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In John Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

[5] L. B. Booker. Intelligent behavior as an adaptation to the task environment. Technical Report 243, University of Michigan at Ann Arbor, Ann Arbor, Michigan, 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] A. Brindle. *Genetic Algorithms for Function Optimization.* PhD thesis, Department of Computer Science of the University of Alberta, Alberta, Canada, 1981.

[8] Bill P. Buckles and Frederick E. Petry. *Genetic Algorithms*. Technology Series. IEEE Computer Society Press, 1992.

[9] Y. L. Chen and C. C. Liu. Multiobjective VAR planning using the goal-attainment method. *IEE Proceedings on Generation, Transmission and Distribution*, 141(3):227–232, may 1994.

[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] Charles Darwin. *The Origin of Species by Means of Natural Selection or the Preservation of Favored Races in the Struggle for Life*. The Book League of America, 1929. Originally published in 1859.

[12] Lawrence Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, New York, 1991.

[13] Kalyanmoy Deb and David E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, San Mateo, California, jun 1989. George Mason University, Morgan Kaufmann Publishers.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] M. Ghiassi, R. E. DeVor, M. I. Dessouky, and B. A. Kijowski. An application of multiple criteria decision making principles for planning machining operations. *IIE Transactions*, 16(2):106–114, jun 1984.

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

[22] David E. Goldberg. Real-coded genetic algorithms, virtual alphabets and blocking. Technical Report 90001, University of Illinois at Urbana-Champaign, Urbana, Illinois, sep 1990.

[23] 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.

[24] 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.

[25] 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.

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

[27] J. J. Grefenstette and J. E. Baker. How genetic algorithms work: A critical look at implicit parallelism. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 20–27, San Mateo, California, jun 1989. George Mason University, Morgan Kaufmann Publishers.

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

[29] Joerg Heitkoetter and David Beasley. The hitch-hiker's guide to evolutionary computation (faq in comp.ai.genetic). USENET, sep 1995. (Version 3.3).

[30] M. R. Hilliard, G. E. Liepins, M. Palmer, and G. Rangarajen. The computer as a partner in algorithmic design: Automated discovery of parameters for a multiobjective scheduling heuristic. In R. Sharda, B. L. Golden, E. Wasil, O. Balci, and W. Stewart, editors, *Impacts of Recent Computer Advances on Operations Research*. North-Holland Publishing Company, New York, 1989.

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

[32] John H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Harbor : University of Michigan Press, 1975.

[33] John H. Holland. *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, Massachusetts, 1992.

[34] 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.

[35] C. L. Hwang and A. S. M. Masud. Multiple objective decision-making methods and applications. In *Lecture Notes in Economics and Mathematical Systems*, volume 164. Springer-Verlag, New York, 1979.

[36] Santiago Hernández Ibáñez. *Métodos de Diseño Optimo de Estructuras*. Colegio de Ingenieros de Caminos, Canales y Puertos, 1990. (in Spanish).

[37] W. Jakob, M. Gorges-Schleuter, and C. Blume. Application of genetic algorithms to task planning and learning. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2nd Workshop*, Lecture Notes in Computer Science, pages 291–300, Amsterdam, 1992. North-Holland Publishing Company.

[38] A. K. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[39] K. Koski. Multicriterion optimization in structural design. In E. Atrek, R. H. Gallagher, K. M. Ragsdell, and O. C. Zienkiewicz, editors, *New Directions in Optimum Structural Design*, pages 483–503. John Wiley and Sons, 1984.

[40] John R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. The MIT Press, 1992.

[41] Frank Kursawe. A variant of evolution strategies for vector optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.

[42] G. E. Liepins, M. R. Hilliard, J. Richardson, and M. Palmer. Genetic algorithms application to set covering and travelling salesman problems. In D. E. Brown and C. C. White, editors, *Operations research and Artificial Intelligence: The integration of problem-solving strategies*, pages 29–57. Kluwer Academic, Norwell, Massachusetts, 1990.

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

[44] S. M. Mahfoud. Crowding and preselection revisited. In R. Männer and B. Manderick, editors, *Parallel problem Solving from Nature, 2nd Workshop*, Amsterdam, 1992. North-Holland Publishing Company.

[45] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, second edition, 1992.

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

[47] A. Osyczka. An approach to multicriterion optimization for structural design. In *Proceedings of International Symposium on Optimal Structural Design*. University of Arizona, 1981.

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

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

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

[51] David Powell and Michael M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 424–431. Morgan Kaufmann Publishers, jul 1993.

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

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

[54] 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.

[55] Brian J. Ritzel, J. Wayland Eheart, and S. Ranjithan. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Research*, 30(5):1589–1603, may 1994.

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

[57] 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.

[58] 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.

[59] H. P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley and sons, Great Britain, 1981.

[60] 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.

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

[62] F. Szidarovszky and L. Duckstein. Basic properties of MODM problems. In *Classnotes 82-1*. Department of Systems and Industrial Engineering, University of Arizona, Tucson, Arizona, 1982.

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

[64] P. B. Wilson and M. D. Macleod. Low implementation cost IIR digital filter design using genetic algorithms. In *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, pages 4/1–4/8, Chelmsford, U.K., 1993.

[65] Alden H. Wright. Genetic algorithms for real parameter optimization. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 205–218. Morgan Kaufmann Publishers, San Mateo, California, 1991.