Mathematical Methods of Operations Research manuscript No.
(will be inserted by the editor)

FEMOEA: a Fast and Efficient Multi-Objective Evolutionary Algorithm

J.L. Redondo · J.Fernández · P.M. Ortigosa

Received: date / Accepted: date

Abstract A multi-objective evolutionary algorithm which can be applied to many nonlinear multi-objective optimization problems is proposed. Its aim is to quickly obtain a fixed size set approximating the complete Pareto-front. It adapts ideas from different multi-objective optimization evolutionary algorithms, but also incorporates new devices. In particular, the search in the space is carried out on promising areas (hyperspheres) determined by a radius value, which decreases as the optimization procedure evolves. This mechanism helps to maintain a balance between exploration and exploitation of the search space. Additionally, a new local search method which accelerates the convergence of the population towards the optimal Pareto-front, has been incorporated. It is an extension of the local optimizer SASS and improves a given solution along a search direction (no gradient information is used). Finally, a termination criteria has also been proposed, which stops the algorithm if during three consecutive iterations the changes experimented in the candidate Pareto-front are negligible (in terms of the objective function values). To know how far two sets are from each other, a modification of the well-known Hausdorff distance is proposed. In order to analyze the algorithm performance, it has been compared to the reference algorithms NSGA-II and SPEA2 and the state-of-the-art

This work has been funded by grants from the Spanish Ministry of Economy and Competitiveness (ECO2011-24927 and TIN2012-37483-C03-03), Junta de Andalucía (P10-TIC-6002, P11-TIC7176 and P12-TIC301), Fundación Séneca (The Agency of Science and Technology of the Region of Murcia, 15254/PI/10), in part financed by the European Regional Development Fund (ERDF). Juana López Redondo is a fellow of the Spanish 'Ramón y Cajal' contract program, co-financed by the European Social Fund.

J.L. Redondo and P.M. Ortigosa
Dpt. of Informatics, University of Almería, Agrifood Campus of International Excellence, ceiA3, Almería, Spain
E-mail: jlredondo@ual.es, ortigosa@ual.es

J. Fernández
Dpt. of Statistics and Operations Research, University of Murcia, Murcia, Spain
E-mail: josefdez@um.es, tlf: +34 868884186, fax: +34 868884181 (Corresponding author)

algorithms MOEA/D and SMS-EMOA. Several quality indicators have been considered, namely, hypervolume, average distance, additive epsilon indicator, spread and spacing. According to the computational results performed, the new algorithm, named FEMOEA, outperforms the other algorithms.

Keywords Nonlinear multi-objective optimization · evolutionary algorithm · quality indicators · computational study

1 Introduction

Multiobjective optimization problems are ubiquitous. Many real-life problems require taking several conflicting points of view into account. In fact, although the origins of multi-objective optimization literature are linked to utility theory, game theory, linear production theory and economics (see [31]), we now can find applications in many and diverse fields, such as portfolio optimization [22], jury selection [51], airline operations [23], radiation therapy [39], manpower planning [54] or reservoir management [1], among others. In [65], White mentions more than 500 applications between 1955 and 1986. Classical references on multi-objective optimization are the books [6,12,56,67,68]. Other more recent books are [20,27,44].

In this paper, we deal with the general nonlinear multi-objective optimization problem (MOP), which can be formulated as follows:

$$\begin{array}{ll} \min \; \{f_1(y), \ldots, f_m(y)\} \\ \text{s.t.} \;\; y \in S \subseteq \mathbb{R}^n \end{array} \tag{1}$$

where $f_1, \ldots, f_m : \mathbb{R}^n \longrightarrow \mathbb{R}$ are m real-valued functions. Let us denote by $f(y) = (f_1(y), \ldots, f_m(y))$ the vector of objective functions and by $Z = f(S)$ the image of the feasible region.

When dealing with multi-objective problems we need to clarify what 'solving' a problem means. In the following some widely known definitions are provided to explain the concept of solution of (1).

Definition 1 A feasible vector $y^* \in S$ is said to be *efficient* iff there does not exist another feasible vector $y \in S$ such that $f_l(y) \leq f_l(y^*)$ for all $l = 1, \ldots, m$, and $f_j(y) < f_j(y^*)$ for at least one index j ($j \in \{1, \ldots, m\}$). The set S_E of all the efficient points is called the *efficient set* or *Pareto-set*. If y_1 and y_2 are two feasible points and $f_l(y_1) \leq f_l(y_2)$ for all $l = 1, \ldots, m$, with at least one of the inequalities being strict, then we say that y_1 *dominates* y_2.

Efficiency is defined in the decision space. The corresponding definition in the criterion space is as follows:

Definition 2 An objective vector $z^* = f(y^*) \in Z$ is said to be *non-dominated* iff y^* is efficient. The set Z_N of all non-dominated vectors is called the *non-dominated set* or *Pareto-front*. If y_1 and y_2 are two feasible points and y_1 dominates y_2, then we say that $f(y_1)$ dominates $f(y_2)$.

Ideally, solving (1) means obtaining the whole efficient set, that is, all the points which are efficient, and its corresponding Pareto-front. However, for a majority of MOPs, it is not easy to obtain an exact description of the efficient set or Pareto-front, since those sets typically include an infinite number of points (usually a continuum set). To the extent of our knowledge, only two exact general methods, namely, two interval branch-and-bound methods (see [25,26]) have been proposed in literature which obtain an enclosure of those sets up to a pre-specified precision. Specifically, they offer a list of boxes (multi-dimensional intervals) whose union contains the complete efficient set (and their images the corresponding Pareto-front) as a solution. However, they are time consuming. Furthermore, they have large memory requirements, so that only small instances can be solved with them.

Contrarily, the use of (meta)heuristics may allow to obtain 'good approximations' of the Pareto-front, even for problems with more variables and objectives. By a good approximation we mean a discrete set of points covering the complete Pareto-front and evenly distributed over it. There is a plethora of methods with that purpose in literature. These include extensions of simulated annealing [14,46,52,58,61], tabu search [32,33,37,40], scatter search [13, 47,62], ant systems [18,42,43,53] or particle swarm optimization [11,41,49], among others, to multi-objective programming. However, most of them are designed to deal with combinatorial MOPs (some exceptions are [37,47,53]).

Nonetheless, the most common approaches utilized in literature to cope with (1) is the use of multi-objective evolutionary algorithms (MOEAs). This is due to their ability to find multiple efficient solutions in one single simulation run. The numerous proposed variants have been surveyed, for instance, in [7, 21,28,36,60]. For a more complete review of all the topics related to MOEAs see the book [10]. For an excellent selection of applications of MOEAs to real-world problems we refer the reader to [9].

In this paper, a new Fast and Efficient Multi-Objective Evolutionary Algorithm (FEMOEA), aimed at obtaining a good fixed size approximation of the Pareto-front is presented. To help FEMOEA to accelerate its convergence towards the optimal Pareto-front, two new devices have been incorporated: a new improving method, where no differentiability is assumed, and a new stopping rule. These two contributions can be included in any MOEA. Furthermore, it adapts some concepts from other evolutionary algorithms (EAs) devised to cope with single-objective optimization problems. It also includes ideas from other typical MOEAs, as the use of the crowding distance as a way to compute the estimation of the density of solutions during the selection procedure (see [16]). It is known that the crowding mechanism only works well for bi-objective problems [17]. However, we use this operator in this work because we want to focus on solving optimization problems with few objectives, usually only two. The interest in this type of problems comes from the fact that, in practice, it is only for those bi- (or tri-)objective problems that obtaining the whole efficient set and its corresponding Pareto-front makes sense. For problems with more objectives those sets are usually big, representing them in a understandable way for the decision-maker is difficult, and even if it was possible, the amount of information will be so big for the decision-maker that he/she will not be able to handle it in an appropriate way. For that type of problems other multi-objective optimization techniques which somehow incorporate preferences for the objectives in order to reduce the set of efficient solutions to be shown to the decision maker, may be a better choice. For instance, in interactive methods the decision-maker can direct the search for efficient points to the areas of his/her interest. On the other hand, there are many real world problems where only two (maybe three) objectives are considered. As an example, the authors are familiar with continuous location theory, an area of operations research where only bi-objective problems have been considered so far (see [25,26] and the references therein). Location theory is a prolific research area, to the point that it has now its own entry (90B85) in the *Mathematics Subject Classification* used by *Mathematical Reviews* and *Zentralblatt für Mathematic.* In fact, many optimization methods are evaluated by making use of location problems [57], mainly due to their high complexity and required computational effort. FEMOEA was indeed initially designed to cope with a competitive facility location and design problem, with an improving method which made use of differentiability (see [2,50]).It proved in [50] to be superior to other state-of-the-art MOEAs. The aim of this paper is to investigate whether FEMOEA is still a competitive algorithm when applied to general MOPs with two or three objectives. For this study, a derivative-free improving method has been implemented, to make FEMOEA more widely applicable.

FEMOEA could also be used to cope with problems with more than three objectives, but the selection procedure should then be based on other measures of the density of solutions different from the crowding distance.

A comprehensive computational study is carried out in this paper to compare FEMOEA with the well-known NSGA-II [16] and SPEA2 [72] algorithms, which have become the reference algorithms in the multi-objective evolutionary computation community. Additionally, two other state-of-the-art algorithms have been included in the comparison: the algorithms MOEA/D [69] and SMS-EMOA [3], which have proved to be very competitive in different studies. The implementation of those four algorithms in the platform j-Metal [19] has been used for the evaluation. Following the existing performance indicators in literature, the comparisons have been accomplished in terms of effectiveness, i.e. in terms of *quality* of the obtained approximations of the Pareto-front. The *modus operandi* has been to provide the algorithms with a budget in the number of function evaluations and to obtain a fixed size approximation of the Pareto-front.

A set of 20 benchmark problems with 2 or 3 objectives and 2 or 3 variables, and another set with 10 problems with 2 or 3 objectives and 30 variables have been solved, and different *quality indicators* have been analyzed. The results show that FEMOEA reaches, on average, better results than the other algorithms.

The rest of the paper is organized as follows. Our new algorithm FEMOEA is introduced in the following section, where special subsections are devoted to the new improving method and stopping rule. Section 3 presents the computational study. The test problems and the implementations employed are given first. Next, the quality indicators used in the comparative study are presented. In addition, some results proving the usefulness of the improving method are provided along with the comparison with NSGA-II, SPEA2, MOEA/D and SMS-EMOA algorithms. In Section 4, our main conclusions are summarized and the research issues that we believe to be worth exploring in the future are highlighted.

2 Description of FEMOEA

FEMOEA is an evolutionary algorithm devised to cope with nonlinear multi-objective problems. Its main objective is to provide a good fixed size approximation of the Pareto-front, i.e., a fixed number of well-distributed and non-dominated solutions. To this aim, it combines ideas from typical algorithms for solving general multi-objective optimization problems: an *external archive* is utilized to store *preferable* non-dominated solutions [38,47], and the crowded comparison operator is applied to guide the algorithm towards a uniformly spread Pareto-front approximation [16]. Additionally, it also inherits some concepts from other evolutionary algorithms devised to cope with single-objective optimization problems, namely from UEGO algorithm [48]. In particular, it has adopted the concept of a decreasing radius, as a mechanism of maintaining a balance between exploration and exploitation of the search space. In addition, FEMOEA incorporates new mechanisms which help to accelerate the optimization process (*efficiency*) and improve the quality (*effectiveness*) of the solutions. The 'termination criteria' or the 'improving method' are two of those specific contributions.

The most important concept in FEMOEA is that of individual. An individual is a solution of the decision space, but it has associated a radius which determines the subregion of the search space (a hypersphere) *covered* by that individual. The main aim of the radius is to focus the searching operators on the corresponding hyperspheres. At any iteration of the algorithm, an individual can be created and its assigned radius only depends on this iteration number. The radius is a monotonous function that decreases as the optimization process moves forward, i.e., as the number of iterations increases. At each stage of the algorithm, several individuals with different radii (created at different iterations) can coexist simultaneously. The use of different radii throughout the optimization process allows, on the one hand, to identify regions in the search space with high quality solutions and, on the other hand, not to waste too much time on regions of the search space which are either already explored or do not provide high quality solutions [5]. This idea of a decreasing radius is a legacy of UEGO [48].

Apart from the radius, an individual has two attributes which are related to the criterion space: the non-domination rank (d_{rank}) and the crowding distance (c_{dist}), see [16]. The non-domination rank indicates the number of individuals which dominate that particular individual. In this sense, a zero value means

Algorithm 1 Crowding distance assignment(P)

1: $p = |P|$
2: **for** $i = 1$ to p
3: $\quad c^i_{dist} = 0$
4: **for** $l = 1$ to m
5: $\quad P$=sort(P, l) *Sort using the l-th objective function value*
6: $\quad c^1_{dist} = c^p_{dist} = \infty$ *In this way, boundary points are always selected*
7: $\quad$ **for** $i = 2$ to $p - 1$
8: $\quad\quad c^i_{dist} = c^i_{dist} + \left(\frac{f_l^{i-1} - f_l^{i+1}}{f_l^{max} - f_l^{min}}\right)^2$
9: **for** $i = 1$ to p
10: $\quad c^i_{dist} = \sqrt{c^i_{dist}}$

that such an individual is not dominated by any of the remaining ones in the current population. Regarding the second attribute, the crowding distance is an estimation of the density of solutions surrounding a particular solution in a population. In [16], Deb et al. proposed an algorithm which calculated the crowding distance of each point in a population P. In that paper, the crowding distance was computed using the rectangular distance. However, in FEMOEA, the Euclidean distance has been considered, since it represents the crowding better than the rectangular one. For the sake of completeness, the algorithm proposed in [16] is depicted (see Algorithm 1). Notice that steps 8 to 10 have been modified to consider the Euclidean distance.

In Algorithm 1, f_l^i refers to the l-th objective function value of the i-th point in the set P, and f_l^{max} and f_l^{min} refer to the maximum and minimum objective function values of the l-th objective function, respectively.

FEMOEA works with two lists of individuals, whose maximum size M is the same for both lists and it is a given input parameter. Parameter M refers to the desired number of solutions in the final Pareto-front. The first list, named *population_list*, is composed of M diverse individuals with different attributes, i.e. various radii, non-domination ranks and crowding distances. FEMOEA is in fact a method for managing this list (i.e. creating, deleting and improving individuals). The second list, called *external_list*, can be understood as a deposit to keep non-dominated solutions. Notice that the number of non-dominated points may be fewer than M during the early stages of the optimization algorithm and hence, the *external_list* may contain fewer elements than the desired ones. In fact, it cannot be guaranteed that M non-dominated solutions have been found once the termination criteria have been satisfied, although this has always been the case in our computational experiments. When this is not the case, the *external_list* is then completed up to M elements with the most *preferable* solutions.

Definition 3 A solution i is *preferable* to a solution i', $i \succ i'$, if

$d^i_{rank} < d^{i'}_{rank}$, or
$d^i_{rank} = d^{i'}_{rank}$ and $c^i_{dist} > c^{i'}_{dist}$.

The previous relation is known as *crowded comparison operator* (see [16]). To accelerate the selection process, both lists are always sorted according to the crowded comparison operator, i.e. in ascending order according to non-domination rank, and in descending order of the crowding distance when several elements share the same non-domination rank.

In FEMOEA, each individual is intended to occupy an efficient solution. For this purpose, FEMOEA directs the individuals during the searching process towards the most suitable regions. Notice, therefore, that a particular individual is not a fixed part of the search domain, but it can move through the space as the search proceeds. 'Individuals-management' is one of the core parts of FEMOEA. It consists of procedures for creating and selecting individuals during the whole optimization process. Additionally, FEMOEA includes an improving method, which has been logically separated from the individual-management. In this sense, FEMOEA can be considered a *memetic* algorithm [45]. Furthermore, this means that FEMOEA can easily be adapted to solve any multi-objective problem, only adapting the improving technique (applications to competitive facility location problems can be found in [24,50]).

2.1 The algorithm

A global description of FEMOEA is given in Algorithm 2. In the following, the different key stages in the algorithm are described:

Algorithm 2 Algorithm FEMOEA

```
Init_individual_lists
while termination criteria are not satisfied
  Create_new_individual(evals)
  if (length(population_list) > M)
    Select_individual(population_list)
  Improve_individual(population_list)
  Update_external_list
  if length(external_list) > M
    Select_individual(external_list)
  Improve_individual(external_list)
if length(external_list) < M
  Compose_pareto
```

- *Init_individual_lists:* In this procedure, as many individuals as the parameter M indicates are randomly created. The radius associated to them will be the one associated at level 1. Such a radius should be greater than or equal to the diameter of the search space, so that the whole search area will be covered by each individual. The *population_list* is initialized from this set of individuals, while the *external_list* will consist only of the non-dominated individuals.

After this procedure, the FEMOEA main loop starts, which basically consists of three procedures: creating, improving and selecting individuals. This loop is executed until a stopping condition is fulfilled, namely, until a considerable improvement is not obtained in three consecutive Pareto-fronts (placed in *external_list*) or a number of maximum levels is achieved. The number of levels (cycles or generations) will be given by the input parameter L.

- *Create_new_individual(evals):* For every individual in the *population_list*, $evals/2$ random trial points in the area defined by its radius are created. $evals$ refers to the budget of function evaluations available for each existing individual for creating a new offspring. After a preliminary computational study, this value has ben set to $evals = 20$.
  Furthermore, for each new random candidate solution, the *closest point* (in the objective space) in the *external_list* is calculated. Then, a new random point is computed in the segment joining the candidate solution with its closest point. Notice that the intermediate point can be placed outside the area covered by the original individual. If the intermediate point dominates the candidate solution, then it will be included in the *population_list* as a new individual. On the contrary, if the candidate solution is the one which dominates the other, it will be the one inserted in the *population_list*. Additionally, if the two points are *indeterminate* (not one dominates the other), then both will be inserted as new individuals. The radius assigned to each new individual is the one associated with the current level i. The radius of an individual created at level i, R_i, is given by a decreasing exponential function, where R_L and R_1 are the given (input parameters) smallest and largest radii. For a detailed description of how to compute the radius at each level of the algorithm see [48]. It is interesting to remark that a location in the search space can be covered by different individuals with different radii. Individuals with small radii examine a relatively small area, their motion in the space is slower, but they are able to differentiate between efficient solutions which are very close. On the contrary, individuals with large radii study a somewhat bigger region, they may move great distances and discover new promising areas, which may be analysed conscientiously in later stages of the algorithm.
  Additionally, both the non-domination rank and the crowding distance associated to each new individual are computed. The *population_list* is then sorted according to the crowding comparison operator.
- *Select_individual(list):* If *list* reaches its maximum allowable capacity, a decision has to be made to determine which individuals should be kept and not removed. The selection strategy used in this work is based on the *crowded comparison operator* [16]. Then, the most preferable individuals will be selected, i.e. between two individuals with different non-domination rank we prefer the one with the lower rank. Otherwise, we prefer the point which is located in a region with the fewest number of points (i.e., the highest crowding distance). In this process, each time an individual is re-

moved, the *crowded comparison operator* associated to the adjacent points is updated before removing the next individual.
- *Improve_individual(list):* This procedure applies the improving method to all the individuals on the *list*. As can be observed in Algorithm 2, this technique is applied to both the *population_list* and the *external_list* at different stages of the optimization process, i.e., steps 6 and 10, respectively. Once all the individuals in the input *list* have invoked the improving method, the improved individuals are reordered according to the crowded comparison operator.
  Subsection 2.2 describes the improving method proposed in this paper.
- *Update_external_list*: New non-dominated points may be generated during the previous process. In Step 7 of Algorithm 2, the *external_list* is updated by copying the non-dominated solutions of the *population_list* to it. Of course, this implies that the points on the *external_list* dominated by the new ones have to be removed, and a reordering of the remaining ones according to the new values of the crowded comparison operator has to be performed.
- *Compose_pareto*: The solution provided by the algorithm must include M solutions since it is a requirement imposed by the user. If the number of solutions on the *external_list* reaches this value, the Pareto-set presented as the final solution will be the one kept on that list. Notice that, on the *external_list*, the non-dominated solutions which are better spread during the optimization process have been stored. However, it may happen that the number of non-dominated solutions found by the algorithm is smaller than M. In such a case, a joint list will be composed considering all the elements on the *population_list* and the *external_list*, and the M most preferable solutions among them will be offered as a result by the algorithm.

2.2 The improving method

Most MOEAs include a mutation operator that alters the individuals of the population from its initial state. This can result in entirely new solutions being added to the population. With these new solution values, a multi-objective algorithm may be able to increase the population diversity and the probability to escape from local optima, helping then to push the population towards the true Pareto-set. However, the mutation operators are usually slow, requiring many function evaluations for convergence.

In this work, an improving method is suggested to accelerate the convergence of the population towards the optimal Pareto-front. Basically, the local method improves a given solution by making changes of different sizes along a search direction. In fact, the designed improving method is an extension of the local optimizer SASS, initially proposed by Solis and Wets in [55] to cope with single-objective optimization problems. Here, it has been adapted to work on multi-objective optimization problems. The proposed algorithm will be called

MO_SASS throughout this paper. The way the heuristic MO_SASS works is described in Algorithm 3.

The algorithm MO_SASS can be applied to an arbitrary multi-objective optimization problem over a bounded subset of $\mathbb{R}^n$, although internally it assumes, as the original SASS does, that the range in which each variable is allowed to vary is the interval $[0, 1]$. The new points are generated using a Gaussian perturbation $\xi \in \mathbb{R}^n$ over the search point y and a normalized bias term $b \in \mathbb{R}^n$ to direct the search. In this way, given y, a first trial point, $y + \xi$ is considered, and if it dominates y, then $y + \xi$ replaces the initial point, but maintaining the same radius value. Otherwise, if y and $y+\xi$ are indeterminate solutions, then $y + \xi$ is compared pairwise to the points on the *external_list*. If it is dominated by any point from such a list, it is discarded; otherwise, it is stored in the *external_list*. Notice that, as a consequence of this inclusion, there may be dominated solutions in the *external_list*. In such a case, those solutions are removed.

The coefficient values 0.4, 0.2 and 0.5 in steps 18, 24 and 26, used for updating the bias term b are retained from Solis and Wets's results [55]. No attempt has been made in this work to fine-tune those parameters. The standard deviation σ specifies the size of the sphere that most likely contains the perturbation vector, whereas the bias term b locates the center of the sphere based on directions of past successes. The size of the standard deviation of the normalized perturbation ξ_{aux} is controlled by the repeated number of successes, *scnt*, or failures, $fcnt$. A success occurs when the new point dominates the initial one. The contraction (*ct*) and expansion (*ex*) constants are set by the user.

As for the individual radius, it was mentioned that an individual has a radius associated to it which determines the subregion of the search space covered by that individual, in such a way that any single step taken by the improving method in a given individual is no longer than the radius of the individual. Since in MO_SASS the standard deviation σ specifies the size of the sphere that most likely contains the normalized perturbation vector, its upper bound σ_{ub} should have the same value than the normalized radius of the caller individual. That is why the parameter σ_{ub} is also considered an input argument in MO_SASS.

It is worth mentioning that the use of MO_SASS allows, on the one hand, to push y towards the true Pareto-set (steps 14-15 and 20-21 in Algorithm 3) and, on the other hand, to study its surrounding area to obtain indeterminate solutions (steps 17-18 and 23-24 in Algorithm 3). The inclusion of indeterminate points in the *external_list* may improve the quality of the final Pareto-front, but it increases the computational effort (the more elements on the list, the more computing time required to order it). Notice that MO_SASS is called (through the *Improving_method*) by FEMOEA to improve both the *population_list* and the *external_list*, which may involve a large number of indeterminate points. Looking for a compromise between quality in the final Pareto-front and computational effort, *Improving_method* does not execute steps 17-18 and 23-24 when improving the *external_list*. The input parameter *bel* tells MO_SASS

Algorithm 3 Algorithm MO_SASS(y, σ_{ub}, bel)

Set $ic = 1$, $y^{(ic)} = y$, $b^{(ic)} = 0$, $scnt = 0$, $fcnt = 0$, $\sigma^{(0)} = \sigma_{ub}$, $\sigma_{lb} = \max\{\sigma_{ub}/1000, 10^{-5}\}$
Fix ex, ct, $Scnt$, $Fcnt$, $Maxfcnt$, ic_{max}
while $ic < ic_{max}$ and $fcnt < Maxfcnt$
  $\sigma^{(ic)} = \sigma^{(ic-1)}$
  if $scnt > Scnt$
    $\sigma^{(ic)} = ex \cdot \sigma^{(ic-1)}$
  if $fcnt > Fcnt$
    $\sigma^{(ic)} = ct \cdot \sigma^{(ic-1)}$
  if $\sigma^{(ic)} < \sigma_{lb}$
    $\sigma^{(ic)} = \sigma_{ub}$ and $b^{(ic)} = 0$
  if $\sigma^{(ic)} > \sigma_{ub}$
    $\sigma^{(ic)} = \sigma_{ub}$
  Generate a multivariate Gaussian random vector $\xi^{(ic)}_{aux} = N(b^{(ic)}, \sigma^{(ic)} I)$
  if $y^{(ic)} + \xi^{(ic)}$ dominates $y^{(ic)}$
    $y^{(ic+1)} = y^{(ic)} + \xi^{(ic)}$; $scnt = scnt + 1, fcnt = 0$
  else
    if $bel = 0$ and $y^{(ic)} + \xi^{(ic)}$ is not dominated by any point on the *external_list*
      Include $y^{(ic)} + \xi^{(ic)}$ in *external_list*; $scnt = 0, fcnt = fcnt+1$; $b^{(ic+1)} = 0.4\xi^{(ic)}_{aux} + 0.2b^{(ic)}$
    else
      if $y^{(ic)} - \xi^{(ic)}$ dominates $y^{(ic)}$
        $y^{(ic+1)} = y^{(ic)} - \xi^{(ic)}$; $scnt = scnt + 1, fcnt = 0$
      else
        if $bel = 0$ and $y^{(ic)} - \xi^{(ic)}$ is not dominated by any point on the *external_list*
          Include $y^{(ic)} - \xi^{(ic)}$ in *external_list*; $scnt = 0, fcnt = fcnt + 1$; $b^{(ic+1)} = b^{(ic)} - 0.4\xi^{(ic)}_{aux}$
        else
          $b^{(ic+1)} = 0.5b^{(ic)}$, $fcnt = fcnt + 1$, $scnt = 0$
  $ic = ic + 1$
Return $y^{(ic)}$

whether the solution y belongs to the *population_list* ($bel = 0$) or the *external_list* ($bel = 1$).

The stopping rules are determined by the maximum number of iterations ($ic_{\max}$) and by the maximum number of consecutive failures ($Maxfcnt$). After a preliminary computational study, they have been set to 400 and 20, respectively.

In order to study whether the introduction of MO_SASS into FEMOEA really helps to accelerate its convergence towards the optimal Pareto-front or, on the contrary, the obtained results are the consequence of randomness, another improving method has been designed. Algorithm 4 sketches its main structure. Basically, this method behaves like MO_SASS, although it tries to improve the initial solution y by making random changes, instead of following a search direction. The actions carried out when indeterminate solutions are obtained as well as the termination criteria, are the same as the ones described for MO_SASS.

Another improving method which makes use of the differentiability of the objective functions can be found in [50].

Algorithm 4 Algorithm Random1(y, σ_{ub}, bel)

Set $ic = 1$, $y^{(ic)} = y$, $scnt = 0$, $fcnt = 0$
Fix $Scnt$, $Fcnt$, $Maxfcnt$, ic_{max}
while $ic < ic_{max}$ and $fcnt < Maxfcnt$
    Generate a random vector $y^{(ic)}_{ran}$ in the area defined by the species radius σ_{ub}
    if $y^{(ic)}_{ran}$ dominates $y^{(ic)}$
        $y^{(ic+1)} = y^{(ic)}_{ran}$; $scnt = scnt + 1, fcnt = 0$
    else
        if $bel = 0$ and $y^{(ic)}_{ran}$ is not dominated by any point on the *external_list*
            Include $y^{(ic)}_{ran}$ in *external_list*; $scnt = 0, fcnt = fcnt + 1$
        else
            $scnt = 0, fcnt = fcnt + 1$
    $ic = ic + 1$
Return $y^{(ic)}$

2.3 The stopping rule

The termination criterion of most MOEAs in literature is only based on a number of function evaluations, i.e. the algorithms usually stop when a maximum is achieved [16,47,72]. However, although this stopping rule can be suitable to compare algorithms in terms of efficiency, it can be counterproductive for practical purposes. The number of function evaluations required to obtain good approximations of the Pareto-front is not known in advance and depends on the particular instance to be solved. Hence, whatever the number we choose, it may be too small for some problems and too high for others, to obtain the quality desired by the user.

In this work, a new stopping rule based on the well-known Hausdorff distance is proposed. Informally, it measures how far two sets are from each other. Mathematically, the modified Hausdorff distance hd used is given by

$$hd(F_1, F_2) = \frac{\frac{\sum_{a \in F_1} min\{d(a,b) : b \in F_2\}}{max\{d(a,a') : a,a' \in F_1\}} + \frac{\sum_{b \in F_2} min\{d(a,b) : a \in F_1\}}{max\{d(b,b') : b,b' \in F_2\}}}{2},$$

where F_1 and F_2 are two given discrete sets and $d(\cdot,\cdot)$ is a distance function (we have used the Euclidean distance).

The termination criteria proposed in this work establishes that the algorithm will finish if during three consecutive iterations, the changes experimented in the candidate Pareto-front are negligible (in terms of the objective function values), for a given tolerance tol (for this work, $tol = 10^{-7}$), i.e. the algorithm stops at iteration t provided

$$hd(\textit{Pareto-front}^t, \textit{Pareto-front}^{t-1}) < tol, \text{ and}$$
$$hd(\textit{Pareto-front}^{t-1}, \textit{Pareto-front}^{t-2}) < tol.$$

Notice that this stopping criterion allows the algorithm to terminate whenever a good approximation of the Pareto-front is obtained, or even if the algorithm is trapped, unable to converge to a better Pareto-front. Notice that, due to the way new individuals are created and improved, it is rather unlikely that two consecutive approximations of the Pareto-front have a neglibible hd distance, unless they provide the best possible output of the algorithm. Still, we require this condition for two consecutive pairs of iterations. This stopping rule allow the algorithm to save considerable CPU time in some instances.

As a safeguard, a second termination criterion, which can be based on the maximum number of function evaluations allowed, the maximum number of iterations permitted or even the maximum execution time admitted to provide the solution should be defined. For the particular case of FEMOEA, we have fixed a maximum number of iterations. This maximum value is represented by the input parameter L.

2.4 Input parameters

Five input parameters must be provided by the user:

- M: The number of solutions which must compose the final Pareto-front.
- L: The maximum number of levels (or iterations).
- R_1 and R_L: The radii of the individuals that are associated to the minimum and maximum level, respectively.
- tol: The tolerance associated with the termination criterion.

Notice that the only parameters which really need to be fine tuned are R_L and L. The remaining ones are either a determination of the user based on his/her experience, requirements or needs (as occurs with the value of M and tol), or a parameter associated to the particular problem to be handled (R_1).

3 Computational studies

All the computational results in this paper have been carried out on a 4-core processor HP ProLiant ML330 G6 to 2.00GHz and 7.8GB memory (using one of its cores).

3.1 Test problems

A thorough study has been conducted to investigate the performance of the analyzed algorithms. Two sets of benchmark problems have been used. The first one (Set 1 in what follows) is composed of 20 problems, 18 of them are bi-objective problems, and the other 2 are tri-objective problems. It includes the so-called ZDT family of problems [71] (in particular, functions ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6) and the DTLZ suite [17] (in particular, DTLZ1, DTLZ2, DTLZ3, DTLZ5, DTLZ6 and DTLZ7), all with $m = 2$ objective functions and $n = 2$ variables, as well as other 9 test problems from literature, namely, Viennet and Viennet2 [10] ($m = 3, n = 2$), Deb ($m = 2, n = 2$), Kursawe ($m = 2, n = 3$) and Poloni ($m = 2, n = 2$) (see [35]), and Deb1

($m = 2, n = 2$), Fonseca ($m = 2, n = 3$), Qv ($m = 2, n = 2$) and Schaffer ($m = 2, n = 1$) (see [10,15,35,72,71]).

The second set of problems (Set 2 in what follows) is composed of the first 10 unconstrained test instances employed in the algorithm contest in the Congress on Evolutionary Computation 2009 (CEC09) [70]. Contrary to Set 1, where the number of variables in the problems is small, all the problems in Set 2 have 30 variables; 7 of the instances are bi-objective problems, and the other 3 are tri-objective problems.

3.2 The algorithms and their implementations

In order to study its performance, FEMOEA has been compared to the reference algorithms NSGA-II and SPEA2, and to the state-of-the-art algorithms MOEA/D and SMS-EMOA. The implementations provided by the framework jMetal [19] have been used.

jMetal is an object-oriented Java-based framework aimed at the development, experimentation, and study of metaheuristics for solving multi-objective optimization problems. jMetal provides a rich set of classes which can be used as the building blocks of multi-objective metaheuristics; taking advantage of code-reusing, the algorithms share the same base components, such as implementations of genetic operators and density estimators. jMetal includes several multi-objective metaheuristics and many problems usually included in performance studies. Additionally, it also provides quality indicators to performance assessing as well as a set of utilities that help in carrying out experimental studies.

The versions of MOEA/D, SMS-EMOA, NSGA-II and SPEA2 obtained from jMetal are in Java. The algorithm FEMOEA has been implemented in C++.

The parameter setting for NSGA-II is the one used by Deb in [16]. The operators for crossover and mutation are SBX and polynomial mutation, with distribution indexes of $\eta_c = 20$ and $\eta_m = 20$, respectively. A crossover probability of $p_c = 0.9$ and a mutation probability $p_{mut} = 1/n$ (where n is the number of decision variables) are used. Regarding SPEA2, the crossover and mutation operators are the same as those used in NSGA-II, using the same values concerning their application probabilities and distribution indexes. The parameters used for MOEA/D are the ones proposed in [69], and those of SMS-EMOA the ones suggested in [3]. Regarding FEMOEA, we found that a good parameter setting is: $L = 30$, $R_L = 0.005$ and $tol = 10^{-7}$. The parameter R_1 coincides with the diameter of the search space.

For all the algorithms, the number M of points in the Pareto-front has been set to 100 for bi-objective problems and 300 for tri-objective instances. The same number of function evaluations employed in average (considering all the runs) by FEMOEA for a given problem was used in each of the runs of the other algorithms.

3.3 Effectiveness measures

Effectiveness measures the accuracy and convergence of obtained solutions (the quality of the final Pareto-fronts). Several indicators used in literature have been utilized. To quantify effectiveness, we have proceeded as in literature and all the algorithms have been run considering the same number of function evaluations. Since the analyzed algorithms are heuristics, every particular instance has been run a given number of times (100 in our computational studies) for each algorithm, and average values for every performance indicator have been computed.

For the assessment and comparison of Pareto-set approximations, the dominant method in literature, the *quality indicator* method, has been used. It maps each Pareto-front approximation to a number, and performs an analytical study on the resulting numbers. Before detailing the quality indicators employed, some definitions are needed.

Definition 4 A feasible vector $y^* \in S$ is said to be *weakly efficient* iff there does not exist another feasible vector $y \in S$ such that $f_l(y) \leq f_l(y^*)$ for all $l = 1, \ldots, m$. If y_1 and y_2 are two feasible points and $f_l(y_1) \leq f_l(y_2), l = 1, \ldots, m$, then we say that y_1 *weakly dominates* y_2, and will be denoted by $y_1 \preceq y_2$.

Definition 5 We say that set A *weakly dominates* set B, $A \preceq B$, provided that every point $y_2 \in B$ is weakly dominated by at least one point $y_1 \in A$.

The corresponding definitions apply in the criterion space.

In the following we will give the formulae of several unary quality indicators. A general formal definition follows.

Definition 6 A unary *quality indicator* is a function $\mathcal{I} : \Omega \to \mathbb{R}$ which assigns each Pareto-front approximation set $PF_{ap} \in \Omega$ a real value $\mathcal{I}(PF_{ap})$.

It is desired that whenever an approximation set A of the Pareto-set is preferable to an approximation set B with respect to weak Pareto dominance, the indicator value for $f(A)$ should be at least as good as the indicator value for $f(B)$. Such indicators are called *Pareto compliant*.

Definition 7 A quality indicator is said to be *Pareto compliant* iff for any pair of approximation sets A, B, $A \preceq B$ implies that the indicator assigns a better (or equal) indicator value to $f(A)$.

Notice that many of the indicators employed in literature are not Pareto compliant. They are usually designed to assess a single aspect of the approximation (its proximity to the true Pareto-front, its spread, its evenness, etc). They are still useful, since they may refine the preference structure of Pareto compliant indicators.

Assume that we want to compare the quality of the outcomes generated by Q stochastic algorithms (in this paper $Q = 5$) for a given problem. For each algorithm q, $q \in \{1, \ldots, Q\}$, e_q runs are performed (in this paper, $e_q = 100$

for all q), generating the approximation sets $PS^q_1, \ldots, PS^q_{e_q}$ (in the decision space). Let us denote by SPS the set of all the approximation sets of the Pareto set, $SPS = \{PS^1_1, \ldots, PS^1_{e_1}, \ldots, PS^Q_1, \ldots, PS^Q_{e_Q}\}$.

In some of the indicators listed below the approximation sets of the Pareto-front need to be compared to the *true* Pareto-front. However, the true Pareto-front is not usually known or cannot be completely obtained. Then, a *reference set* RS which approximates the true Pareto-front is used instead. In our studies, the reference set RS has been obtained as follows. All the approximation sets in SPS are combined, and then, the dominated points are removed from this union. The image of the remaining points forms the reference set.

Additionally, normalized objective values are used to allow different objectives to contribute equally to comparative indicator values. The standard normalization is

$$f_l(y)' = \frac{f_l(y) - z_l^{(\min)}}{z_l^{(\max)} - z_l^{(\min)}},$$

where $z_l^{(\min)}$ (resp. $z_l^{(\max)}$) denotes the minimum (resp. maximum) value of f_l when considering all the solutions in SPS.

The most commonly used quality indicator in literature is *hypervolume* [64, 75]. This Pareto compliant indicator measures the hypervolume of the portion of the criterion space that is weakly dominated by the approximation set. The higher the hypervolume, the better the approximation. In order to measure this quantity, a reference point that is dominated by all points is needed. For a given problem, the same reference point has to be used for all the algorithms and all the runs. In our computational studies, the point whose l-th component is the maximum of all the l-th components of points in $f(SPS)$ is considered. It is an approximation of the Nadir point obtained when considering all the approximations of the Pareto-front together.

Hypervolume can be thought of as a *global* quality indicator, in the sense that it assesses the approximation set as a whole. On the other hand, *proximity* indicators somehow measure the distance between the approximation set and the reference set. In this paper we have used two of those measures, namely, the *average distance* [14] and the *unary additive epsilon indicator* [73]. The former is not Pareto compliant, and is given by

$$D_{av}(f(PS_i)) = \frac{\sum_{b^r \in RS} \min_{a^i \in PS_i} d_\infty(f(a^i), b^r)}{|RS|}$$

where

$$d_\infty(f(a^i), b^r) = \max_{l=1,\ldots,m} \{|f_l(a^i) - b^r_l| / z_l^{(\max)} - z_l^{(\min)}\}.$$

The latter is Pareto compliant and is computed as

$$\begin{aligned} I_{\epsilon+}(f(PS_i)) = \min_{\epsilon \in \mathbb{R}} \{ & \forall b^r \in RS \ \exists a^i \in PS_i : \\ & \frac{f_l(a^i) - z_l^{(\min)}}{z_l^{(\max)} - z_l^{(\min)}} - \epsilon \leq \frac{b^r_l - z_l^{(\min)}}{z_l^{(\max)} - z_l^{(\min)}}) \ \forall l \in \{1 \ldots, m\}\} \end{aligned}$$

and gives the minimum distance by which $f(PS_i)$ needs to be translated in each dimension in objective space such that RS is weakly dominated.

Other two *evenness/diversity* indicators are used in the studies. None of them is Pareto compliant. They are the *spread* [16,47], another well-known indicator, and the *spacing* [59], given by

$$TS(f(PS_i)) = \sqrt{\frac{1}{|PS_i|} \sum_{a^j \in PS_i} \frac{(D_j - \overline{D})^2}{\overline{D}^2}}$$

where $\overline{D} = \sum_{a^j \in PS_i} D_j/|PS_i|$ and D_j is the Euclidean distance in the criterion space between the solution $f(a^j)$ and its nearest solution, i.e.,

$$D_j = \min_{a^i \in PS_i} \{\ell_2((\tfrac{f_1(a^j)-z_1^{(\min)}}{z_1^{(\max)}-z_1^{\min}}, \ldots, \tfrac{f_m(a^j)-z_m^{(\min)}}{z_m^{(\max)}-z_m^{(\min)}}), \\ (\tfrac{f_1(a^i)-z_1^{(\min)}}{z_1^{(\max)}-z_1^{(\min)}}, \ldots, \tfrac{f_m(a^i)-z_m^{(\min)}}{z_m^{(\max)}-z_m^{(\min)}}))\}.$$

It is related to the generational distance [63].

3.4 About the improving method

This section researches, as a first step, whether the designed improving method, i.e. Algorithm 3, really collaborates to approximate the solutions to the optimal Pareto-front or, on the contrary, a simple optimization technique based on random movements (Algorithm 4) is able to obtain similar results. To this aim, only the first set of benchmark problems has been used, and FEMOEA has been executed with both local searching methods. Only a global indicator (the hypervolume), a proximity indicator ($I^1_{\epsilon+}$) and a dispersion indicator (spread) haven been computed.

Since FEMOEA is a heuristic, different runs may provide different solutions. To take this effect into account, FEMOEA with every improving method, has been run 100 times for each test problem, and average values have been computed. In particular, the mean computing time ($Av(T)$) in seconds, the mean number of function evaluations ($Av(eval)$), the mean hypervolume ($Av(hyper)$), the mean $I^1_{\epsilon+}$ indicator ($Av(I^1_{\epsilon+})$) and the mean spread ($Av(Spr)$), have been calculated. Table 1 summarizes those results: it gives the average of the average values for the 20 problems of Set 1. As can be observed, the results obtained by FEMOEA are better when Algorithm 3 is considered as improving method. Notice that the number of consecutive successes is larger when such an algorithm is taken into account, i.e. the number of points which subsequently dominates the caller solution is greater. It means that Algorithm 3 uses more function evaluations, and hence more computing time than Algorithm 4, which usually finishes because the maximum number of failures is reached (see step 3 in Algorithm 4). This fact can be observed when comparing $Av(T)$ and $Av(eval)$ columns.

Table 1 Results obtained by FEMOEA with the different improving methods.

	$Av(T)$	$Av(eval)$	$Av(hyper)$	$Av(I^1_{\epsilon+})$	$Av(Spr)$
Algorithm 3	5.49e+01	5.66e+05	0.5381	0.0048	0.3245
Algorithm 4	4.78e+01	5.33e+05	0.5293	0.0075	0.4341
ModAlg4	2.55e+02	2.44e+06	0.5305	0.0071	0.4315

However, in order to clearly show that the results of FEMOEA are affected by the selected improving method, Algorithm 4 has been modified by omitting Step 11. This allows the method to significantly reduce the number of counted consecutive failures, which obviously increases the number of attempts to achieve non-dominated solutions. Algorithm 4 without step 11 will be called ModAlg4. The results obtained by FEMOEA with ModAlg4 are also shown in Table 1. As can be seen, the Pareto fronts provided by FEMOEA are better when Algorithm 3 is considered, in spite of ModAlg4 executing a larger number of function evaluations.

In what follows, only FEMOEA with Algorithm 3 is used in the comparative studies, and it will be denoted simply by FEMOEA.

3.5 Comparison of the algorithms

In Table 2, the average results obtained by the different algorithms for both sets of benchmark problems, and for all the quality indicators, are given.

Table 2 Average values. MOEA/D, SMS-EMOA, NSGA-II and SPEA2 were run with the same number of functions evaluations as FEMOEA.

	FEMOEA	MOEA/D	SMS-EMOA	NSGA-II	SPEA2
			Hypervolume		
Set 1	0.5381	0.5343	0.5373	0.5356	0.5357
Set 2	0.9041	0.8797	0.8785	0.8897	0.8821
			Average distance		
Set 1	1.0834	1.9625	2.4642	1.5086	1.3152
Set 2	5.1917	6.8059	4.5332	3.9534	4.2271
			$I^1_{\epsilon+}$		
Set 1	0.0048	0.0135	0.0056	0.0050	0.0059
Set 2	0.0251	0.0442	0.0332	0.0288	0.0339
			Spread		
Set 1	0.3245	0.5316	0.3952	0.5165	0.3528
Set 2	1.0568	0.6228	0.9656	0.9190	0.8599
			Spacing		
Set 1	0.5240	1.0919	0.6358	0.7080	0.5451
Set 2	2.2734	0.8915	1.0685	0.9497	0.8692

First of all, we have to say that, regardless of the quality indicator considered, none of the algorithms obtains the best results in all the problems. Also, the algorithm obtaining the best result for more problems for a given quality indicator is not necessarily the one with the best average value.

All this said, we can see that, on average, FEMOEA obtains the best average hypervolume value (highlighted with gray background) for both sets of benchmark problems. For Set 1 SMS-EMOA is the second best, and for Set 2 MOEA/D.

Concerning proximity indicators, FEMOEA obtains the smallest average distance for Set 1, and the smallest $I^1_{\epsilon+}$ for both sets. NSGA-II is the algorithm with the smallest average distance for Set 2. For the average distance, the second best algorithm for both sets is SPEA2, whereas for $I^1_{\epsilon+}$ is NSGA-II.

As for the dispersion indicators, FEMOEA gets the best results for both spread and spacing for Set 1, whereas for Set 2 MOEA/D gets the best results for the spread and SPEA2 for the spacing. For the spacing indicator, NSGA-II is defeated in all the particular problems.

4 Conclusions and future research

In this work, a new multi-objective optimization algorithm, FEMOEA, has been proposed. Furthermore, a new technique (Algorithm 3) to improve the quality of the obtained approximation of the Pareto-front and a new stopping rule to reduce the computational effort have been also presented. These tools, included in FEMOEA, can also be incorporated to any multi-objective optimization algorithm.

FEMOEA has been compared to four algorithms widely referenced in literature, i.e. MOEA/D [69], SMS-EMOA [3], NSGA-II [16] and SPEA2 [72], and using two sets of benchmark problems with 2 or 3 objectives, the first one including problems with up to 3 variables, and the second one with problems with 30 variables. The performance of these four algorithms has been analyzed considering several metrics. More precisely, global indicators (hypervolume), proximity indicators (average distance and $I^1_{\epsilon+}$) and also dispersion indicators (spread and spacing) have been computed. Results have shown that, on average, FEMOEA overcomes all the algorithms in terms of hypervolume and $I^1_{\epsilon+}$ for both sets of benchmark problems. It is also the best algorithm for the other metrics on the first set of problems, although not on the second one.

In the future, FEMOEA will be adapted to solve constrained problems in order to be able to solve any kind of multi-objective optimization problem. Hitherto, little research has been done on the design of methods for the constraint handling in multi-objective optimization (see some examples in [8,16, 30,66]). Therefore, it is an important challenge to research on defining new mechanisms able to solve the different conflicting objectives subject to various constraints.

Another issue that is worth exploring is how to modify FEMOEA to handle problems with more than three objectives. As it has been already pointed out, the selection procedure should then be based on other measures of the density of solutions different from the crowding distance. But which is the best choice for FEMOEA? And are other changes required?

References

1. P. J. Agrell, B. J. Lence, and A. Stam. An interactive multicriteria decision model for multipurpose reservoir management: The shellmouth reservoir. *Journal of Multi-Criteria Decision Analysis*, 7:61–86, 1998.
2. A.G. Arrondo, J.L. Redondo, J. Fernández and P.M. Ortigosa. Parallelization of a non-linear multi-objective optimization algorithm: application to a location problem. *Applied Mathematics and Computation*, 255:114–124, 2015.
3. N. Beume, B. Naujoks and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.
4. S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA - a platform and programming language independent interface for search algorithms. In *Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, pages 494–508. Springer, 2003.
5. C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.
6. V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making Theory and Methodology.* Elsevier Science Publishing Co., New York, 1983.
7. C.A. Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1), 2006.
8. C.A. Coello Coello and A.D. Christiansen. Moses: A multi-objective optimization tool for engineering design. *Engineering Optimization*, 31(3):337–368, 1999.
9. C.A. Coello Coello and G.B. Lamont, editors. *Applications of Multi-Objective Evolutionary Algorithms.* World Scientific, Singapore, 2004.
10. C.A. Coello Coello, G.B. Lamont, and D.A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems.* Genetic and evolutionary computation. Springer, New York, second edition, 2007.
11. C.A. Coello Coello, G. Toscano Pulido, and M. Salazar Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, 2004.
12. J. L. Cohon. *Multiobjective Programming and Planning.* Academic Press, Inc., New York, 1978.
13. A. Corberán, E. Fernández, M. Laguna, and R. Martí. Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the Operational Research Society*, 53(4):427–435, 2002.
14. P. Czyzżak and A. Jaszkiewicz. Pareto simulated annealing-a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47, 1998.
15. K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7:205–230, 1999.
16. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
17. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. In A. Abraham, R. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, chapter 6, pages 105–145. Springer, 2005.
18. K. Doerner, W.J. Gutjahr, R.F. Hartl, C. Strauss, and C. Stummer. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, 131(1-4):79–99, 2004.
19. J.J. Durillo and A.J. Nebro. jMetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.
20. M. Ehrgott. *Multicriteria Optimization.* Springer, Berlin, 2nd edition, 2005.

21. M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22:425–460, 2000.
22. M. Ehrgott, K. Klamroth, and S. Schwehm. An MCDM approach to portofolio optimization. *European Journal of Operational Research*, 155:752–770, 2004.
23. M. Ehrgott and D. M. Ryan. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, 11:139–150, 2002.
24. J. Fernández, A.G. Arrondo, J.L. Redondo and P.M.. Ortigosa. A tri-objective model for franchise expansion. In *Proceedings of the XII Global Optimization Workshop,(Mathematical and Applied Global Optimization workshop, MAGO14)*, pages 81–84, Málaga (Spain), September 2014.
25. J. Fernández and B. Tóth. Obtaining an outer approximation of the efficient set of nonlinear biobjective problems. *Journal of Global Optimization*, 38(2):315–331, 2007.
26. J. Fernández and B. Tóth. Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound method. *Computational Optimization and Applications*, 42(3):393–419, 2009.
27. J. Figueira, S. Greco, and M. Ehrgott, editors. *Multiple Criteria Decision Analysis: State of the Art Surveys*. Kluwer, New York, 2005.
28. C. M. Fonseca and P.J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.
29. C. M. Fonseca and P. 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.
30. C.M. Fonseca and P.J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms- part i: A unified formulation. *IEEE Transactions on Systems, Man and Cybernetics.*, 28:26–37, 2009.
31. T. Gal and T. Hanne. On the development and future aspects of vector optimization and MCDM. A tutorial. In J. Climaco, editor, *Multicriteria analysis. Proc. of the XIth Int. Conf. on MCDM*, pages 130–145, Berlin, 1997. Springer-Verlag.
32. X. Gandibleux, N. Mezdaoui, and A. Fréville. A tabu search procedure to solve combinatorial optimisation problems. In R. Caballero, F. Ruiz, and R. E. Steuer, editors, *Advances in Multiple Objective and Goal Programming*, volume 455 of *Lecture Notes in Economics and Mathematical Systems*, pages 291–300. Springer-Verlag, 1997.
33. M. Hansen. Tabu search for multiobjective combinatorial optimization: TAMOCO. *Control and Cybernetics*, 29(3):799–818, 2000.
34. E.M.T. Hendrix and B. G. Tóth. *Introduction to Nonlinear and Global Optimization*. Springer, New York, 2010.
35. S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477 –506, 2006.
36. H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *Proceedings of 2008 IEEE Congress on Evolutionary Computation*, pages 2424–2431, 2008.
37. D. Jaeggi, G. Parks, T. Kipouros, and J. Clarkson. A multi-objective tabu search algorithm for constrained optimisation problems. In C.A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi- a Criterion Optimization. Third International Conference, EMO 2005*, volume 3410 of *Lecture Notes in Computer Science*, pages 490–504. Springer, 2005.
38. J.D. Knowles and D.W. Corne. *The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation*, volume 1, pages 98–105. IEEE Service Center, 1999.
39. K. H. Küfer, A. Scherrer, M. Monz, F. Alonso, H. Trinkaus, T. Bortfeld, and C. Thieke. Intensity-modulated radiotherapy - a large scale multi-criteria programming problem. *OR Spectrum*, 25:223–249, 2003.
40. S. Kulturel-Konak, A.E. Smith, and B.A. Norman. Multi-objective tabu search using a multinomial probability mass function. *European Journal of Operational Research*, 169:918–931, 2006.

41. M. Mahfouf, M.Y. Chen, and D.A. Linkens. Adaptive weighted particle swarm optimisation for multi-objective optimal design of alloy steels. In NewEditor1, editor, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 762–771. Springer-Verlag, 2004.
42. C.E. Mariano and E. Morales. MOAQ: an ant-Q algorithm for multiple objective optimization problems. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Genetic and Evolutionary Computing Conference (GECCO 99)*, volume 1, pages 894–901. Morgan Kaufmann, 1999.
43. P.R. McMullen. An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives. *Artificial Intelligence in Engineering*, 15:309–317, 2001.
44. K. S. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer, Boston, 1998.
45. P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Publishers, Boston MA, 2003.
46. D. Nam and C.H. Park. Pareto-based cost simulated annealing for multiobjective optimization. In L. Wang, K.C. Tan, T. Furuhashi, J.H. Kim, and X. Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, volume 2, pages 522–526. Orchid Country Club, 2002.
47. A.J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J.J. Durillo, and A. Beham. AbYSS: Adapting scatter search to multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 12(4):439–457, 2008.
48. P.M. Ortigosa, I. García, and M. Jelasity. Reliability and performance of UEGO, a clustering-based global optimizer. *Journal of Global Optimization*, 19(3):265–289, 2001.
49. T. Ray and K. Liew. A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, 34(2):141–153, 2002.
50. J.L. Redondo, J. Fernández, J.D. Álvarez, A.G. Arrondo and P.M. Ortigosa. Approximating the Pareto-front of a planar bi-objective competitive facility location and design problem. *Computers and Operations Research*, 62:337–349, 2015.
51. M. J. Schniederjans and E. Hollcroft. A multi-criteria modeling approach to jury selection. *Socio-Economic Planning Sciences*, 39:81–102, 2005.
52. P. Serafini. Simulated annealing for multiple objective optimization problems. In G. Tzeng, H. Wang, and U. Wen, editors, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application*, volume 1, pages 283–292. Springer-Verlag, 1994.
53. P. Shelokar, V. Jayaraman, and B. Kulkarni. Ant algorithm for single and multiobjective reliability optimization problems. *Quality and Reliability Engineering International*, 18(6):497–514, 2002.
54. J. Silverman, R. E. Steuer, and A. W. Whisman. A multi-period, multiple criteria optimization system for manpower planning. *European Journal of Operational Research*, 34:160–170, 1988.
55. F.J. Solis and R.J.B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981.
56. R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Applications*. John Wiley &Sons, New York, 1986.
57. R.G. Strongin and Y.D. Sergeyev. *Global optimization with non-convex constraints: Sequential and parallel algorithms*. Nonconvex optimization and its applications. Kluwer Academic Publishers, Dordrecht, 2000.
58. B. Suman. Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. *Computers &Chemical Engineering*, 28:1849–1871, 2004.
59. K.C. Tan, C.K. Goh, Y.J. Yang, and T.H. Lee. Evolving better population distribution and exploration in evolutionary multi-objective optimization. *European Journal of Operational Research*, 171(2):463–495, 2006.
60. K.C. Tan, T.H. Lee, and E.F. Khor. Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. *Artificial Intelligence Review*, 17:253–290, 2002.
61. E. Ulungu, J. Teghem, P. Fortemps, and D. Tuyttens. MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.

62. J. Vasconcelos, J. Maciel, and R.O. Parreiras. Scatter search techniques applied to electromagnetic problems. *IEEE Transactions on Magnetics*, 41(5):1804–1807, 2005.
63. D.A. Van Veldhuizen and G.B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 1998.
64. L. While, L. Bradstreet, and L. Barone. A Fast Way of Calculating Exact Hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16(1):86–95, February 2012.
65. D. J. White. A bibliography on the applications of mathematical programming multiple-objective methods. *Journal of the Operational Research Society*, 41:669–691, 1990.
66. Y.G. Woldesenbet, G. G. Yen, and B. G. Tessema. Constraint handling in multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 13(3):514–525, June 2009.
67. P. L. Yu. *Multiple-criteria decision making concepts, techniques and extensions*. Plenum Press, New York, 1985.
68. M. Zeleny. *Multiple Criteria Decision Making*. McGraw-Hill, New York, 1982.
69. Q. Zhang, and H. Li. MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.
70. Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical Report CES-487, The School of Computer Science and Electronic Engineering, University of Essex, 2008.
71. E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.
72. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002. International Center for Numerical Methods in Engineering (CIMNE).
73. E. Zitzler, L. Thiele, M. Laumanns, C. M. Foneseca, and V. Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.
74. E. Zitzler, J. Knowles, and L. Thiele. Quality Assessment of Pareto Set Approximations. In Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski, editors, *Multiobjective Optimization. Interactive and Evolutionary Approaches*, pages 373–404. Springer. Lecture Notes in Computer Science Vol. 5252, Berlin, Germany, 2008.
75. E. Zitzler and L. Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.