SSPMO: A Scatter Tabu Search Procedure for Non-Linear Multiobjective Optimization

JULIÁN MOLINA[1], MANUEL LAGUNA[2], RAFAEL MARTÍ[3] AND RAFAEL CABALLERO[1]

[1] Universidad de Málaga, Spain, {julian.molina, rafael.caballero}@uma.es
[2] University of Colorado at Boulder, USA, laguna@colorado.edu
[3] Universitat de València, Spain, Rafael.marti@uv.es

Latest revision: March 17, 2005

Abstract — We describe the development and testing of a metaheuristic procedure, based on the scatter search methodology, for the problem of approximating the efficient frontier of nonlinear multiobjective optimization problems with continuous variables. Recent applications of scatter search have shown its merit as a global optimization technique for single-objective problems. However, the application of scatter search to multiobjective optimization problems has not been fully explored in the literature. We test the proposed procedure on a suite of problems that have been used extensively in multiobjective optimization. Additional tests are performed on instances that are an extension of those considered classic. The tests indicate that our extension of the basic scatter search framework is a viable alternative for multiobjective optimization.

Keywords: Multiobjective Metaheuristics, Non-Linear Multiobjective Optimization, Evolutionary Multiobjective Optimization.

1. Introduction

Most decision problems in the real world require the simultaneous optimization of more than one criterion. That is, decision makers are not able to base their decisions on a single criterion in order to identify one or more attractive courses of action. Moreover, real decision problems are such that often conflicting criteria are used to evaluate alternative solutions. Multiobjective Programming is the branch of Mathematical Programming that deals with problems for which more than one objective function is required to evaluate the merit of alternative decisions. Formally, the problem is stated as follows:

$$\begin{array}{ll} \text{Max} & \left(f_1(x), f_1(x), \ldots, f_p(x)\right) \\ \text{s.t.} & x \in X \end{array}$$

where

$x = (x_1, x_2, \ldots, x_n)$ are the decision variables

X is the set of feasible solutions

$y_i = f_i(x)$ is the i^{th} objective function (or decision criterion)

$Y = f(X)$ is the objective function space

Comparing alternative solutions to a given problem is one of the first difficulties encountered when moving from single-objective optimization to multiobjective optimization. In other words, how do we know that one solution is better than another when performance is measured by more than one objective function value? Given that the merit of a solution is represented by a vector of objective function values, it is not always possible to determine when a vector is strictly larger or smaller than another. Generally, there is no complete order in *Y*. Multiobjective optimization introduces the concept of efficiency, developed by Vilfredo Pareto in 1896. Essentially, efficiency means that a solution to a multiobjective function is such that no single objective can be improved without deteriorating another one. Assuming that we want to maximize all objective functions, the Pareto order can be defined mathematically as:

> Given two points $y, y' \in Y$, we say that y is preferred to y' if $y_i \geq y'_i \ \forall i = 1, \ldots, p$ and there exists at least one $j \in \{1, \ldots, p\}$ such that $y_j > y'_j$.

Clearly, the Pareto order is a partial one and therefore it is not possible to select a solution that is preferred over all other solutions, that is, it is not possible to find "an optimal solution". Hence, the concept of optimality must be generalized in such a way that more than one solution can be considered ideal (optimal) in problems with multiple objectives. This generalization follows from the definition of the Pareto order and is known as Pareto efficiency:

> A solution $x^* \in X$ is efficient if there is no other solution $x \in X$ such that $f(x)$ is preferred to $f(x^*)$ according to the Pareto order. That is, $x^* \in X$ is efficient if there is no solution $x \in X$ such that $f_i(x) \geq f_i(x^*)\ \forall i = 1,\ldots,p$ and at least one $j \in \{1,\ldots,p\}$ such that $f_j(x) > f_j(x^*)$.

Most multiobjective programming techniques focus on finding the set of efficient points (E) for a given problem or, in the case of heuristic procedures, an approximation of the efficient set ($\hat{E}$). In this paper, we describe the development and testing of a metaheuristic procedure for multiobjective optimization problems for which $f_i(x)$, for $i = 1, \ldots, p$, are nonlinear functions and x are continuous and bounded variables. Since our approach is not exact, our goal is to search for the best $\hat{E}$. Metaheuristics have been applied to this problem, so before discussing our proposed procedure, we review the approaches that are relevant to our current investigations.

2. Metaheuristics for Multiobjective Optimization

Most exact techniques for multiobjective optimization deal with continuous and discrete linear problems. The application of these techniques to complex multiobjective optimization problems has been considered impractical. We refer to complex problems to those with nonlinearities in the objective function and/or constraints, with a large number of discrete variables or uncertainty in key data. To deal with these difficulties, which are not unusual in real settings, researchers and practitioners have relied on metaheuristic procedures, as indicated in the survey articles by Ehrgott and Gandibleux (2000) and Jones, Mirrazavi and Tamiz (2002).

Evolutionary algorithms are the most visible and common metaheuristic technique in the realm of multiobjective optimization. These procedures, known as MOEA (multiobjective evolutionary algorithms), dominate the multiobjective optimization research whose focal point is not the development of exact procedures[1]. VEGA (Vector Evaluated Genetic Algorithm) by Schaffer (1985) has become a classic method in the MOEA literature. Since the development of VEGA, many other alternative procedures have been developed. One of the most complete references for MOEA is the book by Coello, Van Veldhuizen and Lamont (2002). Chapter 4 (MOEA Testing and Analysis) of this book provides a brief description of 11 MOEA procedures, although experimental results for only 4 of these MOEA variants (MOGA, MOMGA, NPGA and NSGA) are presented. The testing was done over a set of 6 unconstrained nonlinear multiobjective problems with bounded continuous variables. These procedures plus one more (SPEA) were recommended as the best MOEA implementations in Van Veldhuizen and Lamont (2000), where brief descriptions can be found. Zitzler, Laumanns and Thiele (2001) developed SPEA2, an updated version of SPEA that consists of adding (1) a fine-grained fitness assignment strategy, (2) a density estimation technique (which we also use in our work as a solution quality measurement) and (3) an enhanced archive truncation method.

Most tabu search (Glover and Laguna, 1997) applications to multiobjective optimization employ the so-called *independent sampling technique*. This technique is based on aggregating the objective functions by assigning a weight to each of them. Each sample consists of solving the single-objective optimization problem that results from applying a given set of weights. To obtain an approximation of the efficient frontier ($\hat{E}$) the procedure must be run as many times as the desired number of points, using different weight values. The performance of implementations based on independent sampling deteriorates as the need for generating more efficient solutions increases, since this is directly proportional to the number of times that the procedure must be executed. Tabu search implementations based on independent sampling are due to Hertz, et al. (1994), Dahl, et al. (1995), Gandibleux, et al. (1997), Hansen (1997), Ben Abdelaziz, et al. (1999), Alves and Climaco (2000), and Gandibleux and Freville (2000). For the most part, these procedures do not use advanced tabu search strategies for diversification and intensification.

The method developed by Caballero, Gandibeux and Molina (2004) is, to the best of our knowledge, the only tabu search implementation that employs Pareto-sampling instead

[1] Jones, Mirrazavi and Tamiz (2002) estimate that about 70% of the metaheuristic applications to multiobjective optimization reported in the literature are evolutionary algorithms, while 24% are based on simulated annealing and only 6% are based on tabu search.

of independent sampling in the context of multiobjective combinatorial optimization. Also, this procedure, known as MOAMP (Multiobjective Metaheuristic using an Adaptive Memory Procedure), is the only implementation that is capable of including any solution visited during the search (if it qualifies) into the final approximation of the efficient frontier. MOAMP separates itself from the other tabu search implementations by looking for efficient points with an intensification process (second phase of this procedure) around an initial set of efficient points (first phase of this procedure). To build this initial set of efficient points, MOAMP carries out a series of linked tabu searches (linked means that the last point of one search becomes the initial point of the next search) where each point visited could be included in the final $\hat{E}$. This is achieved by checking the dominance criteria for each solution around its neighbourhood. Solutions that are not dominated are declared "possibly efficient" and are added to a list used to update $\hat{E}$.

The second phase of MOAMP exploits the proximate optimality principle (POP), which stipulates that good solutions at one level are likely to be found close to good solutions at an adjacent level. The POP concept may be viewed as a heuristic counterpart of the so called Principle of Optimality in dynamic programming (see section 5.5 of Glover and Laguna, 1997). The interpretation of POP within multiobjective optimization is that efficient points are "connected" by a curve inside the efficient set. This is why the second phase of MOAMP intensifies the search around the initial set of efficient points found in the first phase.

Our current interest and the purpose of this paper is to extend the basic scatter search methodology (Glover, Laguna and Martí, 2000) to tackle nonlinear multiobjective optimization problems. Some applications of scatter search to multiobjective optimization problems have been developed in recent years. Gomes da Silva, Clímaco and Figueira (2004) described a scatter search method with surrogate constraints for solving multi-dimensional knapsacks with two criteria. Garcia, et al. (2002) applied scatter search to a multiobjective location problem. Beausoleil (2005) developed MOSS (Multiobjective Scatter Search), a tabu/scatter search hybrid for nonlinear multiobjective optimization problems. We address the differences between MOSS and our own scatter-tabu search hybrid in section 3.3.

3. Scatter Tabu Search Procedure for Multiobjective Optimization (SSPMO)

Our solution method (SSPMO) consists of a scatter/tabu search hybrid that includes two different phases:

- Generation of an initial set of efficient points through various tabu searches
- Combination of solutions and updating of $\hat{E}$ via a scatter search

A detailed description of the single-objective scatter search methodology is not included because this information has been published widely in journal articles, book chapters, and conference proceedings. For a complete description, we refer the interested reader to the book by Laguna and Martí (2003). Hence, it suffices to indicate that a scatter search consists of constructing and then maintaining a reference set (*RefSet*) of solutions (obtained from a larger source set *P*) through the application of five methods: diversification generation, subset generation, combination, improvement, and reference set update.

Scatter search orients its explorations systematically relative to a set of reference points that typically consist of good solutions obtained by prior problem solving efforts, where the criteria for "good" are not restricted to objective function values, and may apply to sub-collections of solutions rather than to a single solution, as in the case of solutions that differ from each other according to certain specifications. The reference set is a collection of both high quality solutions and diverse solutions that are used to generate new solutions by way of applying a combination method. In single-objective optimization, diversity is measured with reference to the solution space (i.e., diversity increases when solutions that have different structural properties are included in the reference set), whereas the aim of multiobjective metaheuristics is to find solutions that are diverse in the objective function space. Deb et al. (2000) emphasize this distinction by stating that the necessary conditions to convert a single-objective evolutionary method into a multiobjective method are both assigning fitness to population members based on non-dominated sorting and preserving diversity among solutions of the same non-dominated front. The following subsections describe how our implementation deals with both conditions within the scatter search framework.

3.1 Initial Phase

Our procedure starts with the application of the first phase of MOAMP (Caballero, Gandibeux and Molina, 2004). This phase consists of linking p+1 tabu searches. The first tabu search starts from an arbitrary point and attempts to find the optimal solution to the problem with the single objective $f_1(x)$. Let x^1 be the last point visited at the end of this search. Then, a tabu search is applied again to find the best solution to the problem with the single-objective $f_2(x)$ using x^1 as the initial solution. This process is repeated until all the single-objective problems associated with the p objectives have been solved. Then, we solve again the problem with the first objective $f_1(x)$ starting from x^p, to finish a cycle around the efficient set. This phase yields the p points that approximate the best solutions to the single-objective problems that result from ignoring all but one objective function. Additional efficient solutions may be found during this phase because all visited points are checked for inclusion in $\hat{E}$.

Still within the MOAMP framework, we launch several tabu searches using a *global criterion method*. In this step, the aim is to minimize a function that measures the distance to the ideal point. The ideal point $f^{\max}$ is that for which each criterion i (for i = 1, …, p) achieves its maximum value $f_i^{\max}$. Similarly, the anti-ideal point $f^{\min}$ is that for which each criterion i (for i = 1, …, p) achieves its minimum value $f_i^{\min}$. We approximate $f^{\max}$ and $f^{\min}$ with the maximum and minimum values, respectively, that each objective achieves in the current $\hat{E}$. Note that with all likelihood each objective achieves its maximum in a different point of the solution space. Hence, the purpose of determining $f^{\max}$ is not searching for an efficient solution in X but for an ideal value in the image space in order to measure the quality of solutions generated during this step. This global criterion method follows the notion of *compromise programming* (Duckstein, 1984; Yu, 1973; Zeleny, 1973). Given that the ideal point consists of the optimal (or best known) values of the individual objective functions, compromise programming assumes that it is logical for the decision maker to prefer a point that is closer to the ideal point over one that is farther away. An essential element of compromise programming is therefore the notion of distance between points in an Euclidian space. The L_q metric ($1 \le q \le \infty$), a generalization of the Euclidian distance (q = 2), is commonly used in compromise programming. A characteristic of the L_q family is that a larger weight is given to the maximum deviation as q increases. L_1 and L_∞ are the only linear metrics out of the entire family. L_∞ is often preferred because it has been shown to lead to balanced efficient solutions. The L_∞ metric results in a min-max global criterion:

$$L_\infty(x) = \max_{i=1,\dots p}\left\{ w_i \left(\frac{f_i^{\max} - f_i(x)}{f_i^{\max} - f_i^{\min}} \right) \right\}$$

The motivation for using this metric for the global criterion tabu searches is that a solution x is efficient for a given set of weights w if it minimizes $L_\infty(x)$. In general, a point that minimizes an L_q distance to $f^{\max}$ is an efficient point. The set of all points obtained in this way is called the *compromise set*. Compromise solutions have the characteristic of providing a good balance among the values of the p objective functions. It has been shown that the best balance is achieved when using the L_∞ metric, and hence our choice. MOAMP's first phase generates random weights until *InitPhase* consecutive searches fail to produce a new efficient point in $\hat{E}$. Through experimentations, we have determined that the final $\hat{E}$ obtained in this initial phase contains a diverse set of efficient points (where diversity is measured in the objective function space).

Figure 1 shows a pseudo-code that summarizes the steps in Phase 1 of our procedure.

Input Parameters: *InitPhase*

1. Generate an initial point
2. **While** *number of searches without change* < *InitPhase* **do**
    2.1 Choose one of the *p* functions to optimize (or construct a compromise function)
    2.2 Last point visited is becomes the new initial point
    2.3 Launch a tabu eearch to optimize the selected function starting from the selected initial point (*initial_point*)
    2.4 Check if any change in the number of efficient points found

Choosing the function to be optimized

1. All the objective functions of the problem are selected in sequence (and the first one is selected again after the p^{th} one)

2. For the remaining of the search, a random vector of weights is generated and a compromise function is constructed

Tabu Search

Input Parameters: *TabuIter, TabuTenure*

1. *current_point* = *initial_point*
2. **For** *TabuIter* iterations **do**
    2.1. Generate the neighborhood of *current_point*.
    2.2 If any of the neighbors is non-tabu, feasible and better than the *current_point*, then stop exploring the current neighborhood and choose this point as the next *current_point*. Go to step 2.4.
    2.3 If *current_point* was not dominated by any of its neighbors, then add it to the list of efficient points
    2.4 Choose the non-tabu feasible neighbor with the best objective function value as the next *current_point*
    2.5 The former *current_point* is declared tabu for the next *TabuTenure* iterations

Figure 1. Phase 1 Pseudo-Code

3.2 Scatter Search Phase

The reference set (*RefSet*) is at the core of scatter search implementations. For single-objective problems, the *RefSet* contains a mixture of high-quality and diverse solutions, where quality is measured with reference to the single objective function and diversity is measured (using the Euclidian distances) in the solution space. We have modified the role of *RefSet* to deal with the special characteristics of multiobjective optimization. In particular, solution quality is measured considering p objective functions and solution diversity is measured in the objective function space. Note that since most multiobjective optimization problems consist of conflicting objective functions, it is reasonable to expect that diversity in the solution space will induce diversity in the solution space.

The main search mechanism in this phase of our procedure is the combination of solutions that are currently considered efficient and therefore belong to $\hat{E}$. The solutions to be combined are selected from the reference set, where $RefSet \subset \hat{E}$. *RefSet* consists of b solutions ($b > p$) and is initially constructed as follows:

1. Select the best solution in $\hat{E}$ for each of the p objective functions and add them to *RefSet.* (Note that it is possible, but unlikely, to select fewer than p solutions in this step if a solution happens to be best for more than one objective function.)
2. Select b-p solutions from $\hat{E} \setminus RefSet$ that maximize the distance between them and those solutions already in *RefSet*. Since the solutions are selected sequentially, the distance measure is updated after each selection. Because we look for diversity in the objective function space, distance is measured with a normalized L_∞ metric.

The construction of the initial *RefSet* reveals that in our multiobjective implementation of scatter search $P = \hat{E}$. This is an expanded role for P (when compared to single-objective optimization) because it not only supports the diversification of *RefSet* but also acts as a repository of efficient solutions. The experiments reported in section 4 show that efficient solutions provide enough diversification in the search, making unnecessary the addition of non-efficient solutions to *RefSet*.

A list of solutions that have been selected as reference points is kept to prevent the selection of those solutions in future iterations. Therefore, every solution that is added to *RefSet* is also added to *TabuRefSet*. The size of *TabuRefSet* increases as the search progresses because this memory function is an explicit record of past reference solutions. The motivation for creating and maintaining *TabuRefSet* is that we would like to obtain a final $\hat{E}$ of adequate density. That is, we would like to encourage a uniform generation of points in the efficient frontier and avoid gaps that may be the result of generating too many points in one region while neglecting other regions.

A linear combination method is used to combine reference solutions. All pairs of solutions in *RefSet* are combined and each combination yields four new trial solutions. Let x^i and x^j be the reference solutions being combined, then four trial solutions x^k are obtained with the following line search:

$$x^k = \lambda x^i + (1-\lambda)x^j \qquad \lambda = \text{-}1/3,\ 1/3,\ 2/3 \text{ and } 4/3$$

The selection of this combination method follows the suggestions made by Glover (1994) in connection with non-linear optimization of single-objective problems. This method does not rely in randomization and includes both convex and non-convex combinations of the reference solutions. The rational behind this choice is that it generates two non-convex combinations and two convex ones. Also, two of the trial solutions generated are close to one of the reference solutions and the other two trial solutions are close to the second reference solution. Both of these characteristics make the method appropriate within the philosophy of the scatter search framework.

Each of the new trial solutions is subjected to an improvement method. Tabu search is the mechanism used to improve new trial solutions. This is the same tabu search used in the initial phase (described in section 3.1). The objective function that guides the search for an improved solution is the L_∞ metric with w_i = 1 for i = 1, .., p. The $f_i^{\max}$ and $f_i^{\min}$ are calculated considering only the reference points x^i and x^j whose combination resulted in the trial point x^k currently being improved. We set the weights to 1 because this drives the search to focus on the compromise area that we are trying to explore. Solutions generated during this improvement phase are tested for possible inclusion in $\hat{E}$. Note that any addition to $\hat{E}$ may cause some previously efficient points to become dominated and therefore expelled from the set. Figure 2 is a graphical representation of the improvement phase within the scatter search. By creating the ideal point from the reference points that originated the four new trial solutions, the procedure attempts to find efficient points that "fill the gap" between x^i and x^j.

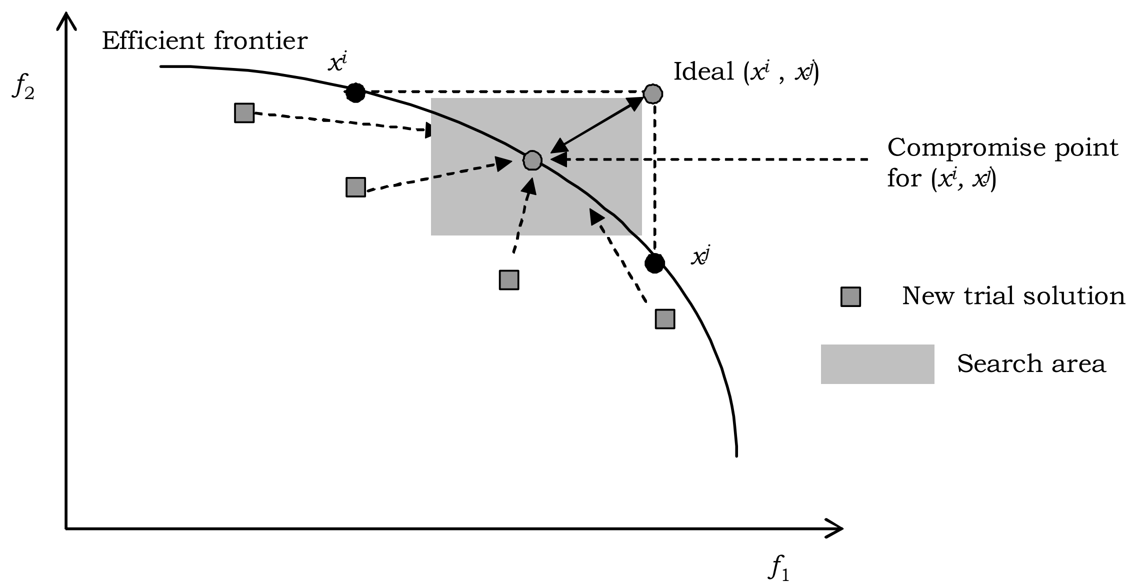

Figure 2. Graphical representation of the improvement method

Once all the solution pairs in *RefSet* are combined and the new trial solutions are improved, the procedure updates the reference set in preparation for the next scatter search iteration. The first step in the updating process is to choose the best solutions according to each of the objective functions taken separately. This is the same first step as in the construction of the initial reference set. The step does not consider whether these efficient points belong to *TabuRefSet*. The remaining *b-p* reference solutions are chosen as follows:

1. For each solution $x \in \hat{E} \setminus TabuRefSet$, a normalized (using the range of each function) L_∞ distance is calculated. The normalization is such that all distances are between 0 and 1. The distance calculations involve x and all the solutions in *TabuRefSet*. Let the minimum of these normalized distances be $L_\infty^{\min}(x)$. This minimum normalized distance is used as the probability that x is declared eligible as a reference solution. This is the probability of being included in the list of eligible solutions *LES*. Hence, the larger the minimum distance between the candidate solution x and all the solutions in *TabuRefSet* the better the

chance for x of being eligible as a reference solution. A uniform random number is generated and if it is less than $L_\infty^{\min}(x)$ then x is declared eligible.

2. From the list *LES*, we choose sequentially the *b-p* solutions with largest minimum distance to *TabuRefSet*. The distance is measured against *TabuRefSet* instead of *RefSet* to move away from areas that have been explored in the past by virtue of combining former reference points. Also, *TabuRefSet* is updated after each selection in order to avoid choosing points that are too close to each other.

It is important to point out that the updating procedure describe above is such that it attempts to find those solutions that are away from the solutions currently in *TabuRefSet*. A probabilistic element is included to add flexibility to this process. The scatter search continues until the mean value of $L_\infty^{\min}(x)$ for the set of eligible solutions *LES* in step 2 above falls below a pre-specified threshold *MeanDist*. The pseudo-code in Figure 3 summarizes the steps in the scatter search phase of our procedure.

Input Parameters: *MeanDist, b*

1. Build the list of eligible solutions *LES* from $\hat{E}$. Compute *MVL*, the mean value of $L_\infty^{\min}(x)$ for *LES*. If *MVL* is less than *MeanDist*, stop. Otherwise, continue to step 2.
2. Select the best solution in *LES* for each of the p objective functions and add them to *RefSet*. Also add these solutions to *TabuRefSet*.
3. Select the *b-p* solutions from *LES**TabuRefSet* that maximize the distance between them and the solutions in *TabuRefSet*. Since the solutions are selected sequentially (and added to *TabuRefSet*), the distance measure to *TabuRefSet* is updated after each selection.
4. Combine pairs of solutions in *RefSet,* where each combination yields four new trial solutions.
5. Improve each of the new trial solutions, using a tabu search guided by a compromise function.

Figure 3. Pseudo-code of the scatter search phase

3.3 Differences between MOSS and SSPMO

MOSS (Beausoleil, 2005) is related to SSPMO in the sense that it applies similar techniques to the same class of problems. The tabu search elements included in MOSS create restrictions that are used to prevent moves toward solutions that are "too close" to previously visited solutions. The definition of distance is adjusted during the search to control the number of solutions that are classified tabu during a given number of iterations. These solutions are not allowed to be combined within the scatter search iterations. A sequential fan candidate list strategy is used to explore solution neighborhoods. A weighted linear function is used to aggregate the objective function values and to provide a way of choosing the best move in a neighborhood. Long term memory is used to encourage diversification by giving incentives to sampling neglected subranges of values within the feasible range of each decision variable.

Although both MOSS and SSPMO are optimization procedures based on a hybridization of scatter and tabu search, there are significant differences between them:

1. MOSS does no utilize a guiding function to direct the tabu searches as SSPMO does. Instead, MOSS employs an aspiration level criterion to avoid moving the search to a dominated point. The guiding functions in SSPMO are both the objective functions in the problem and the L_∞ compromise functions.

2. The update of the *RefSet* in MOSS mimics the updating procedure that is traditional in scatter search implementations for single-objective problems. In particular, a reference point may stay in the *RefSet* for several iterations. In contrast, SSPMO updates the *RefSet* around the best solutions found for each objective function. This is achieved by adding all former reference points to *TabuRefSet*.

3. The subset generation method in MOSS considers the Kramer choice function to create the solution subsets to be combined. In SSPMO, we take into consideration the characteristics of a point according to its $L_\infty^{\min}(x)$ value. This produces a uniform sampling of the efficient frontier and avoids the selection of points for a combination subset that are concentrated in the same area.

4. Computational Experiments

The performance of this solution method depends on the value of three search parameters: *InitPhase*, *b* and *MeanDist*. The *InitPhase* parameter controls the termination of the initial phase of the procedure. In particular, the initial phase terminates after *InitPhase* consecutive searches fail to produce a new efficient point. The most effective value of 3 for *InitPhase* was determined with a series of tuning experiments. The *MeanDist* value controls the termination of the scatter search phase. The parameter value is a threshold of required average distance at which potential reference points must be from the current *RefSet*. Since the distance is normalized to be between 0 and 1, the *MeanDist* value also should be within such a range. *MeanDist* values close to zero make the scatter search phase longer. We employ a value of 0.1 for this parameter because this value provides a good balance between computational effort and solution quality. The tuning experiments (using instances with $p \leq 3$) also showed that the procedure is insensitive to specific values of b in the $(2p, 3p)$ range. Hence, we have selected $b = 2p$ for our computational tests.

We perform tests on a Pentium 4 at 2.4 GHz with three different sets of problems from the literature. MOAMP, MOSS and SPEA2 were re-implemented in order to compare the performance of all the methods on the same computer. The ZDT and DTLZ instances respectively from Zitzler, Deb and Thiele (2000) and Deb, et al. (2002) are well-known, standard problems in the multiobjective optimization literature. We also use instances from Deb (1999) that include non-convex and disjoint efficient frontiers as well as non-uniform distribution of efficient points. A summary of the characteristics of the 27 problems in our test set is shown in Table 1.

Table 1. Characteristics of test problems

Set	*Number of problems*	*Number of variables*	*Number of objectives*
Deb	18	2	2
DTLZ	4	12	3
ZDT	5	10 and 30	2

Detailed description of the instances is not included because this information is available in the published articles associated with each problem set. However, computer code for all the problems in our test set is available from julian.molina@uma.es.

The main goal of our experiments is to show that the proposed procedure is capable of creating better approximations of the efficient frontiers than existing methods for nonlinear multiobjective optimization. Specifically, we are interested in comparing the performance of SSPMO with MOAMP, SPEA2 and MOSS. The performance measures that we employ are:

1. *Number of points*: This refers to the ability of finding efficient points. We assume that the decision maker prefers more rather than fewer efficient points.

2. *SSC*: This metric suggested by Zitzler and Thiele (1999) measures the size of the space covered (SSC). In other words, SSC measures the volume of the dominated points. Hence, the larger the SSC value the better.

3. *k-distance*: This density estimation technique used by Zitzler, Laumanns and Thiele (2001) in connection with the computational testing of SPEA2 is based on the k^{th} nearest neighbor method of Silverman (1986) . The metric is simply the distance to the k^{th} nearest efficient point. We use $k = 5$ and calculate both the mean and the max of k-distance values. The k-distance value is such that the smaller the better in terms of frontier density.

4. M_1^*: This metric suggested by Zitzler (1999) consists of calculating an average Euclidean distance (L_2) between the estimated efficient frontier and the Pareto-optimal set. Clearly, smaller values of M_1^* indicate better approximations of the Pareto-optimal set.

5. $C(A,B)$: This is known as the coverage of two sets measure (Zizler and Thiele, 1999). $C(A,B)$ represents the proportion of points in the estimated efficient frontier B that are dominated by the efficient points in the estimated frontier A.

In our first experiment, we compare the solutions found with the initial phase of our procedure with the solutions found after the entire procedure has terminated (that is after the application of the scatter search phase). This experiment attempts to measure the contribution of each phase of the proposed method toward the quality of the final approximation of the efficient set. Table 2 shows the results of this experiment, where SSPMO represents the entire procedure and "Initial Phase" represents the solutions obtained after the first phase terminates and before the scatter search initiates.

Table 2. Comparison of SSPMO and Initial Phase. Average and standard deviation values of five performance measures over a set of 27 problems.

Methods	*N. of points*	*k-distance (mean)*	*k-distance (max)*	*SSC*	M_1^*
SSPMO	1622.185	0.006	0.149	0.534	0.061
	1708.762	0.007	0.188	0.287	0.160
Initial Phase	187.370	0.086	0.280	0.314	0.262
	314.395	0.185	0.270	0.295	0.363

For each method (SSPMO and Initial Phase), Table 2 shows the average (first row) and the standard deviation (second row) of the 5 measurements described above. The statistics are calculated over the 27 instances summarized in Table 1. The results in this table indicate that the scatter search phase is a major contributor to the quality of the solutions obtained by the entire SSPMO.

We use the same scheme to compare the four methods that we have implemented. In this case, however, we have added a "Time" column to show the average and standard deviation of the CPU seconds associated with each procedure. Table 3 summarizes the results of our second experiment.

Table 3. Comparison of 4 methods for multiobjective optimization. Average and standard deviation values of six performance measures over a set of 27 problems.

Methods	*N. of points*	*k-distance (mean)*	*k-distance (max)*	*SSC*	M_1^*	*Time*
SSPMO	1622.185	0.006	0.149	0.534	0.061	22.926
	1708.762	0.007	0.188	0.287	0.160	25.859
MOAMP	764.963	0.032	0.144	0.482	0.095	6.111
	1011.204	0.036	0.119	0.291	0.236	10.467
SPEA2	68.889	0.232	0.450	0.507	0.073	12.519
	92.696	0.274	0.327	0.273	0.187	13.791
MOSS	83.111	0.055	0.527	0.498	0.066	237.630
	28.860	0.031	0.347	0.303	0.168	446.714

The results in Table 3 indicate that SSPMO is capable of finding efficient frontiers with a large number of points and high density, as indicated by the small *k*-distance values. The number of points is not an input parameter of SSPMO. SSPMO's termination criteria allow it to find a large number of points while improving on the density of $\hat{E}$. This is not the case for MOAMP, which can construct efficient frontiers with a fair amount of points but at the same time is not capable of identifying gaps that ultimately result in sparse areas of the frontiers. The SSC and M_1^* values also show a superior performance of SSPMO over the competing approaches. Regarding execution time, it is clear that the large number of efficient points that SSPMO is able to generate makes the procedure slower than MOAMP and SPEA2. The tradeoff, however, still favors SSPMO if we consider the time per point in the final efficient set.

Another way of analyzing the results of the same experiment is to count the number of times that each procedure performs best according to each performance measure. Table 4 shows the summary of such computation.

Table 4. Number of "wins" for each method over the 27 problems

Methods	*N. of points*	*k-distance (mean)*	*k-distance (max)*	*SSC*	M_1^*	*Time*
SSPMO	25	22	15	15	5	1
MOAMP	2	5	11	9	15	21
SPEA2	0	0	0	1	5	3
MOSS	0	0	1	2	2	2

The values in Table 4 confirm the merit of SSPMO according to most of the standard performance measures. SSPMO is inferior to MOAMP only in execution time and number of best M_1^* values. It is interesting to note that although MOAMP achieves the best M_1^* values 15 times, the average M_1^* value still favors SSPMO (as shown in Table 3).

The *C*(*A,B*) measure allows us to make a comparison according to the dominance of one efficient frontier over another. Table 5 shows the average *C*(*A,B*) values over the entire set of test problems. The values in Table 5 show that the efficient points generated by SSPMO tend to dominate those generated by other methods. That is, C(SSPMO, -) > C(-, SSPMO) except in the case of C(MOAMP, SSPMO).

Table 5. Coverage of two sets

A/B	SSPMO	MOAMP	SPEA2	MOSS
SSPMO	0.000	0.160	0.261	0.197
MOAMP	0.230	0.000	0.196	0.181
SPEA2	0.106	0.140	0.000	0.113
MOSS	0.115	0.111	0.093	0.000

To conclude this section on computational experiments, we examine the results obtained by SSPMO, MOAMP (the method closest to SSPMO in terms of overall performance) and MOSS (a similar scatter/tabu search hybrid) on two problem instances in the Deb set (see Table 1). We chose these two problems to gain additional insight on the strengths and weakness of each approach. In particular, we would like to address the issue of finding efficient frontiers that are dense and whose points are uniformly distributed. This is possible only by including mechanisms that can detect gaps in the approximation and dynamically adapt the search. First, we examine the second problem in section 4.1 of Deb (1999). The approximations of the efficient frontier obtained by SSPMO, MOAMP and MOSS are shown in Figures 4 and 5.

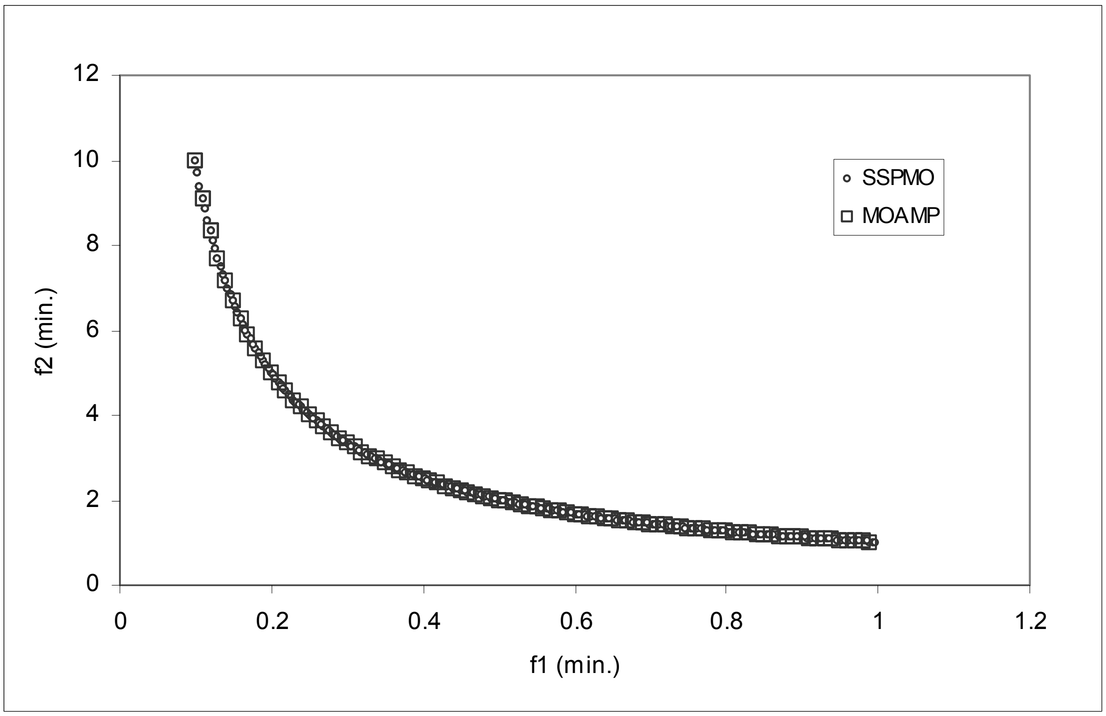

Figure 4. SSPMO and MOAMP approximations of the efficient frontier of the second problem in section 4.1 of Deb (1999)

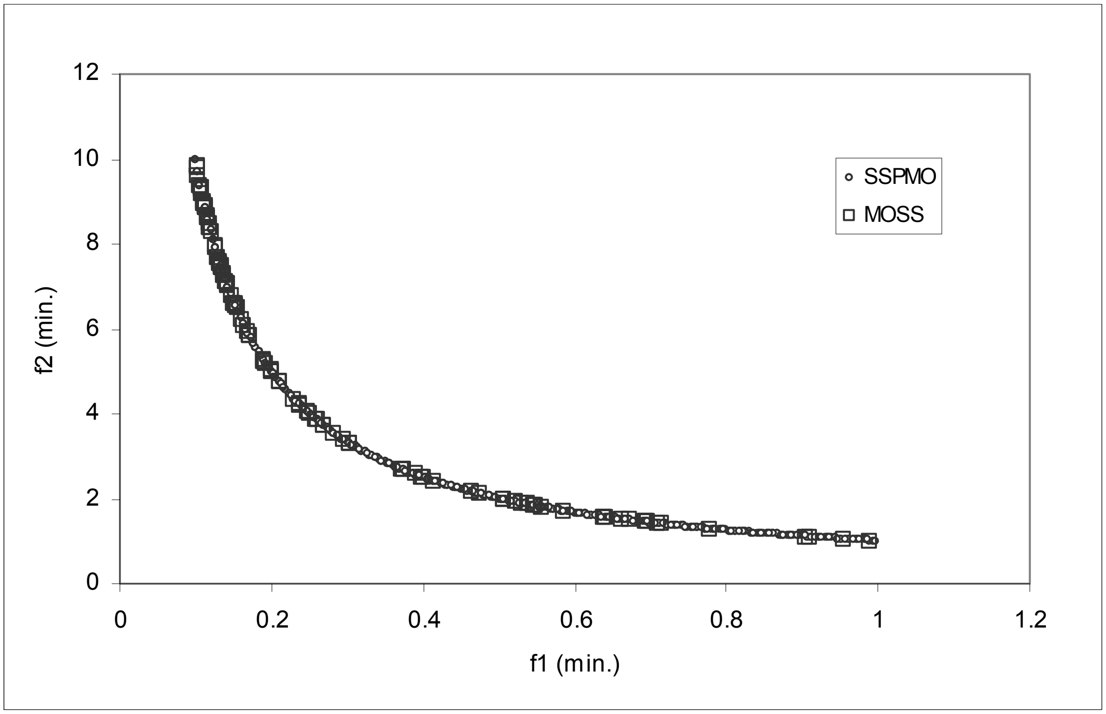

Figure 5. SSPMO and MOSS approximations of the efficient frontier of the second problem in section 4.1 of Deb (1999)

Figure 4 shows that MOAMP is capable of finding solutions along the entire approximation of the efficient frontier but the density is not uniform. In particular, the density of the frontier found by MOAMP decreases in the area that is near the optimum for objective function 1. This unequal density of the frontier produces a *k-distance* measure of 0.0387 for MOAMP while SSPMO achieves a value of 0.0009 for the same measure. Figure 5 shows that MOSS is not capable of identifying neglected areas of the efficient frontier. This produces the gaps that are evident in Figure 5 (in particular toward the best solution found for objective function 2). SSPMO, on the other hand, focuses on finding points along the entire frontier by closing gaps generating a total of 1,660 points versus 99 found by MOSS.

The lack of density of the approximations found by MOAMP and MOSS is more evident in problems with a disjoint efficient frontier. Figure 6 and Figure 7 show the approximations found by SSPMO, MOAMP and MOSS for the problem described in section 5.1.3 of Deb (1999) with parameters $q = 8$ and $\alpha = 2$.

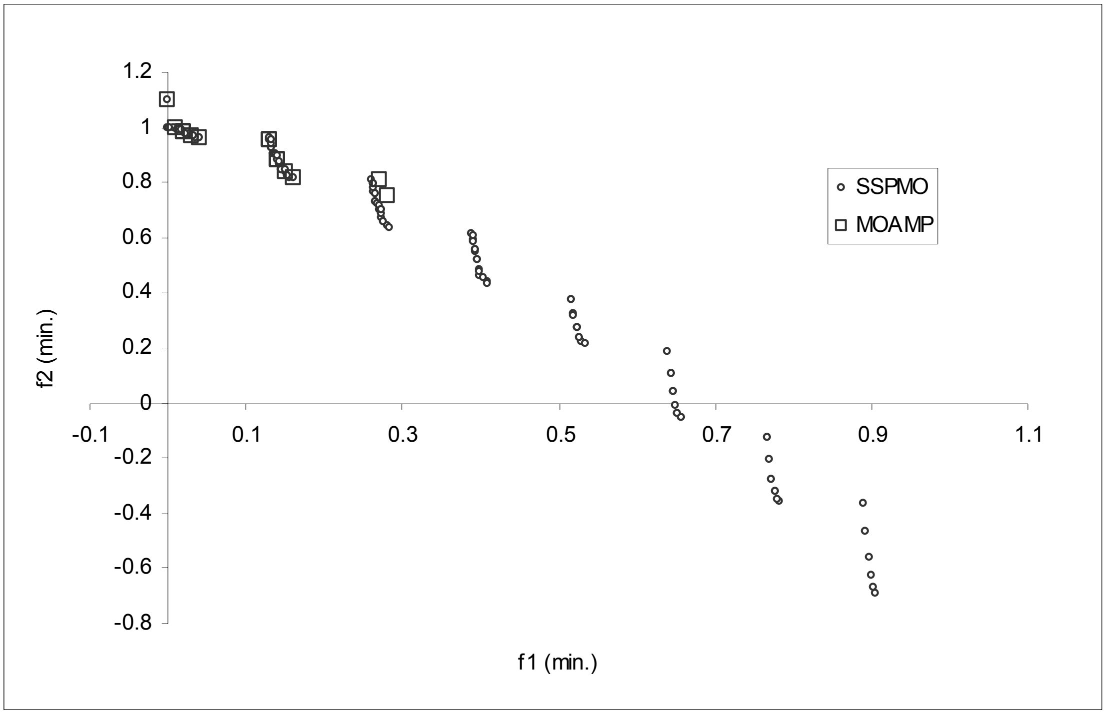

Figure 6. SSPMO and MOAMP approximations of the efficient frontier of the problem in section 5.1.3 of Deb (1999) with parameters q = 8 and α = 2

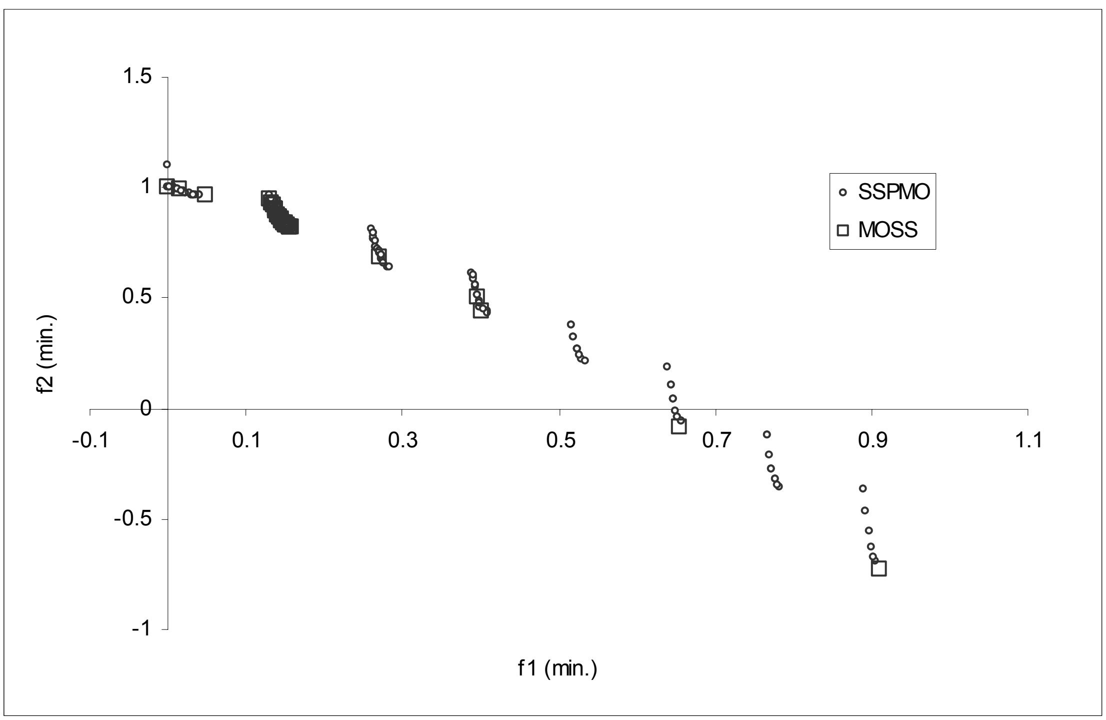

Figure 7. SSPMO and MOSS approximations of the efficient frontier of the problem in section 5.1.3 of Deb (1999) with parameters q = 8 and α = 2

The approximations depicted in Figures 6 and 7 correspond to 128 SSPMO points, 14 MOAMP points and 29 MOSS points. These figures show that SSPMO visits and fills all the disjoint parts of the efficient frontiers because is able to detect the presence of gaps and react accordingly. On the other hand, MOAMP and MOSS are able to achieve good approximations only on the extremes of the efficient frontier because they lack a mechanism to detect areas with low density and redirect the search.

4.1 Additional Analysis of Results

The scatter search phase of SSPMO is designed to address the main problem found in the approximation methods for multiobjective optimization that can be found in the literature. Namely, we attempt to find dense efficient frontiers with a sufficient number of points that are well-distributed. As our experiments show, some of the best methods in the literature either produce approximations with an insufficient number of points (SPEA2) or the approximations contain gaps (MOAMP) or the points are not well-distributed (MOSS). These deficiencies are caused by the following design problems:

1. The size of $\hat{E}$ is pre-specified as an input parameter (SPEA2 and MOSS). This entails that the analyst must estimate the number of points needed for a good approximation. In the case of SPEA2, this is not an easy question to answer. A small number results in sparse approximation. A large number prevents the launching of a diversification phase and results in an inferior distribution of points in the final approximation of the efficient frontier.
2. While diversifying criteria are included in some methods (SPEA2, MOSS and MOAMP), none of these methods include “filling criteria”. That is, these methods include mechanisms to delete or penalize solutions too close to other solutions, but they don’t complement these mechanisms with others that encourage the search to intensify around regions with low density of points.
3. The stopping criteria of the existing methods are not related with density estimations or the presence of gaps (SPEA2, MOSS and MOAMP). This causes the search to either stop before neglected areas are visited or continue even after a well-distributed approximation has been found.

SSPMO addresses these deficiencies by including three strategies: 1) the search focuses on filling the gaps between carefully selected reference points, 2) low density areas are visited by moving the search away from previous reference points, and 3) the search terminates only if a measure designed to identify the largest gap falls below a specified value.

5. Conclusions

We have described the development and implementation of a metaheuristic procedure for the optimization of multiobjective non-linear functions. Our procedure extends the application of scatter search in an innovative way by assigning new roles to the reference and population sets as well as by strategically including tabu search elements. One of the main goals of our effort has been to test the proposed procedure employing all the problem instances available in the literature. The resulting test set consists of 27 instances, with no more than 3 objective functions. Since SSPMO uses the number of objective functions (p) to adjust the size of the reference set, investigating the effects on performance for large p values becomes an open question for future research.

In order to make a valid comparison against competing procedures, we have used several metrics as well as graphical output (when applicable). Our computational experiments show that SSPMO has merit when compared to approximation procedures that are also metaheuristic in nature. Throughout the development of our procedure and the ensuing experimentation we have been able to identify three issues that limit the performance of exiting procedures. The identification of these issues (summarized in section 4.1) have allowed us to not only design a more robust solution procedure but also explain why the procedure performs at a higher level than the ones we have used for comparison.

Acknowledgments

The authors wish to thank three anonymous referees for their thoughtful comments, which helped us to create significantly improved version of this paper. We also like to thank Antonio Barbadillo, whose encouragement made this work possible.

References

Alves, M. J. and J. Climaco (2000) "An Interactive Method for 0-1 Multiobjective Problems Using Simulated Annealing and Tabu Search," *Journal of Heuristics*, vol. 6, no. 3, pp. 385-403.

Beausoleil, R. P. (2005) “MOSS: Multiobjective Scatter Search Applied to Non-linear Multiple Criteria Optimization,” to appear in the *European Journal of Operational Research.*

Ben Abdelaziz, F., S. Krichen and J. Chaouachi (1999) “A Hybrid Metaheuristic for the Multiobjective Knapsack Problem,” Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, in S. Voss, S. Martello, I. Osman and C. Roucairol (eds.), Kluwer Academic Publishers: Boston, pp. 205-212.

Caballero, R., X. Gandibleux, and J. Molina (2004) “MOAMP- A Generic Multiobjective Metaheuristic using an Adaptive Memory", Technical Report, University of Valenciennes, France.

Coello, C., D. A. Van Veldhuizen and G. B. Lamont (2002) *Evolutionary Algorithms for Solving Multi-Objective Problems,* Kluwer Academic Publishers: Boston, ISBN 0-306-46762-3.

Dahl, G., K. Jörnsten and A. Lokketangen (1995) “A Tabu Search Approach to the Channel Minimization Problem,” Paper presented at ICOTA’95, Chengdu, China.

Darwen P. and X. Yao (1995) “A Dilemma for Fitness Sharing with a Scaling Function,” in *Proc. of the 1995 IEEE International Conference on Evolutionary Computation* (ICEC'95), Perth, Australia, IEEE Press: New York, NY 10017-2394, pp. 166-171.

Deb, K. (1999) “Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems,” *Evolutionary Computation*, vol. 7, no. 3, pp. 205-230.

Deb. K., A. Pratap, S. Agarwal and T. Meyarivan (2000) “A Fast and Elitist Multi-Objective Genetic Algorithm-NSGA-II,” KanGAL Report Number 2000001.

Deb, K., L. Thiele, M. Laumanns and E. Zitzler (2002) “Scalable Multi-Objective Optimization Test Problems,” in Congress on Evolutionary Computation (CEC'2002) Piscataway, New Jersey, IEEE Service Center, vol. 1, pp. 825-830.

Duckestein, L. (1984) “Multiobjective Optimization in Structural Design: The Model Choice Problem,” *New Directions in Optimum Structural Design,* E. Atrek, R. H.

Gallagher, K. M. Ragsdell, and O. C. Zienkiewicz (eds.), John Wiley and Sons, pp. 459-481.

Erhgott, M. and X. Gandibleux (2000) "A Survey and Annotated Bibliography on Multiobjective Combinatorial Optimization," *OR Spektrum*, vol. 22, pp. 425-460.

Fonseca, C. M. and P. J. Fleming (1998) "Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A Unified Formulation," *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, vol. 28, no. 1, pp. 26–37.

Gandibleux, X., N. Mezdaoui and A. Freville (1997) "A Tabu Search Procedure to solve Multiobjective Combinatorial Optimization Problems," *Advances in Multiple Objective and Goal Programming,* R. Caballero, F. Ruiz and R. Steuer (eds.), Lecture Notes in Economics and Mathematical Systems, Springer: Berlin, vol. 455, pp. 291-300.

Gandibleux, X and A. Freville (2000) "Tabu Search Based Procedure for Solving the 0-1 Multiobjective Knapsack Problem: The Two objectives Case," *Journal of Heuristics,* vol. 6, pp. 361-383.

Garcia, F., B. Melian, J. A. Moreno and J. M. Moreno-Vega (2002) "Scatter Search for Multiple Objective p-facility Location problems," Groupe de Travail ROADEF sur la Programmation Mathématique MultiObjectif, 6ème journée de travail.

Glover, F. (1994) "Tabu Search for Non-Linear and Parametric Optimization (with links to Genetic Algorithms)," *Discrete Applied Mathematics*, vol. 49, pp. 231-255.

Glover, F. and M. Laguna (1997) *Tabu Search,* Kluwer Academic Publishers: Boston, ISBN: 0-7923-8187-4, 408 pp.

Glover, F., Laguna, M. and Martí, R. (2000) "Fundamentals of Scatter Search and Path Relinking", Control and Cybernetics, vol. 39, nº 3, pp. 653-684.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley Publishing Company: Reading, Massachusetts.

Goldberg, D. E., Deb, K. and Korb, B. (1990) "Messy genetic algorithms revisited: studies in mixed size and scale", *Complex Systems,* vol. 4, pp. 415-444.

Gomes da Silva, C., J. Clímaco and J. Figueira (2004) "A Scatter Search Method for the Bi-criteria Multi-dimensional {0,1}-Knapsack Problem Using Surrogate Relaxation," *Journal of Mathematical Modelling and Algorithms*, vol. 3, nº3, pp. 183-208.

Hansen, M. P. (1997) "Tabu Search for Multiobjective Optimization: MOTS," 13th International Conference on Mutliple Criteria Decision Making, Cape Town, South Africa.

Hertz, A., B. Jaumard, C. Ribeiro and F. W. Formosinho (1994) "A Multicriteria Tabu Search Approach to Cell Formation Problems in Group Technology with Multiple Objectives," *Recherche Operationelle/Operations Research*, vol. 28, no. 3, pp. 303-328.

Horn, J., N. Nafpliotis, and D. E. Goldberg (1994) "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation*, Z. Michalewicz (ed.), IEEE Press: Piscataway, New Jersey, pp. 82–87.

Jones, D. F., S. K. Mirrazavi and M. Tamiz (2002) "Multiobjective Metaheuristics: An Overview of the Current State of the Art," *European Journal of Operational Research*, vol. 137, pp. 1-9.

Laguna, M. and R. Martí (2003) *Scatter Search: Methodology and Implementations in C,* Kluwer Academic Publishers: Boston, ISBN: 1-4020-7376-3, 312 pp.

Schaffer, J.D. (1985) "Multiple Objective optimization with vector evaluated genetic algorithms", In J.J. Grefenstette (Ed.) Proceedings of the First International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum, pp. 93-100.

Silvermann, B.W. (1986) *Density Estimation for Statistics and Data Analysis,* Chapman and Hall, London.

Srinivas, N. and K. Deb (1994) "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248.

Van Veldhuizen, D. A. (1999) *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, Ph.D. Thesis, AFIT/DS/ENG/99-01, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

Van Veldhuizen, D. A. and G. B. Lamont (2000) "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art," *Evolutionary Computation*, vol. 8, no. 2, pp. 125-147.

Yu, P. L. (1973) "A Class of Solutions for Group Decision Problems," *Management Science*, vol. 19, pp. 936-946.

Zeleny, M. (1973) "Compromise Programming," in *Multiple Criteria Decision Making*, J. L. Cochrane and M. Zeleny (eds.) University of South Carolina Press, Columbia, pp. 262-301.

Zitzler, E. (1999) Evolutionary *Algorithms for Multiobjective Optimization: Methods and Applications*, Ph D. thesis, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland.

Zitzler, E., K. Deb and L. Thiele (2000) "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation Journal*, vol. 8, no. 2, pp. 125-148.

Zitzler, E., M. Laumanns and L. Thiele (2001) "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

Zitzler, E. and L. Thiele (1999) "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271.