New Ideas in Applying Scatter Search to Multiobjective Optimization

Antonio J. Nebro, Francisco Luna, Enrique Alba

Departamento de Lenguajes y Ciencias de la Computación
E.T.S. Ingeniería Informática
Campus de Teatinos, 29071 Málaga (SPAIN)
{antonio,flv,eat}@lcc.uma.es

Abstract. This paper elaborates on new ideas of a scatter search algorithm for solving multiobjective problems. Our approach adapts the well-known scatter search template for single objective optimization to the multiobjective field. The result is a simple and new metaheuristic called SSMO, which incorporates typical concepts from the multiobjective optimization domain such as Pareto dominance, crowding, and Pareto ranking. We evaluate SSMO with both constrained and unconstrained problems and compare it against NSGA-II. Preliminary results indicate that scatter search is a promising approach for multiobjective optimization.

1 Introduction

Most optimization problems in the real world involve the optimization of more than one function, which in turn can require a significant computational time to be evaluated. This feature and the fact that the search space tends to be very large in multiobjective problems (MOPs) make deterministic techniques difficult to apply in order to obtain the Pareto-optimal solutions of MOPs. As a consequence, stochastic techniques have been widely proposed and applied in this domain. Among them, evolutionary algorithms have been investigated by many authors, and some of the most well-known algorithms for solving MOPs belong to this class (e.g. NSGA-II [1], PAES [2], SPEA-2 [3], and micro-GA [4]).

Many evolutionary algorithms for solving MOPs are some kind of genetic algorithm. This implies they use the concepts of population, crossover, mutation, and similar genetic operators (an exception is PAES, which is an (1+1) evolution strategy). We are interested in studying the application of scatter search, another kind of population-based evolutionary algorithm, to solve MOPs. Scatter search has proved to be very effective for solving a diverse set of single objective optimization problems from both classical and real world settings [5], but little attention has been paid to its use in multiobjective optimization (existing works almost reduce to [6–8]).

Scatter search is based on using a small population known as the reference set, whose individuals are combined to construct new solutions which, in contrast to other evolutionary algorithms, are obtained in a systematic way (i.e., stochastic

procedures such as crossover and mutation are not used). Furthermore, these solutions can be improved by applying a local search method. The reference set is initialized from an initial population composed of dispersed solutions, and it is updated taking into account the results of the local search improvement.

The *scatter search template* presented in [9] has served as the main reference for most of the scatter search implementations to date. The template consists of five methods: diversification generation, improvement, reference set update, subset generation, and solution combination. This template is used in [10] to design a scatter search procedure for single objective optimization problems with continuous bounded variables. In this paper, we have taken this implementation as the basis of a scatter search algorithm for multiobjective optimization, trying to modify it as little as possible with the idea of getting a simple algorithm. We have named this algorithm SSMO (Scatter Search for Multiobjective Optimization). Our main goal is to identify and study new issues that can affect the performance of the algorithm for MOPs.

The contributions of our work can be summarized as follows:

- We propose a scatter search algorithm for solving constrained as well as unconstrained MOPs. The algorithm is based on incorporating the concepts of Pareto dominance, ranking, and crowding, and they are applied to define the improvement and reference set update methods of the scatter search algorithm.
- Two strategies for building the reference set are studied. The first one uses ranking and crowding to carry out a sorting of the population to obtain the best individuals, while the second strategy is based on applying a clustering technique to get a set of centroids of the individuals with best rank.
- The algorithm is evaluated using a benchmark of constrained plus unconstrained MOPs, and it is compared against the NSGA-II algorithm.

The remaining of the paper is organized as follows. In Section 2, we discuss related works concerning multiobjective optimization and scatter search. In Section 3, we describe our proposal. Experimental results are presented in Section 4. Finally, in Section 5 we give some conclusions and lines for future research.

2 Related Work

The application of scatter search to multiobjective optimization has received little attention until recently. We analyze here the proposals presented in [6], [7], and [8]. We use the following terminology: P is the initial set, k is the number of objective functions, and the reference set is composed of $p + q$ individuals, which are obtained by selecting the best p solutions of P, while the remaining q individuals are selected from both P and the current reference set by using a mechanism promoting diversity.

MOSS [6] is an algorithm that proposes a tabu/scatter search hybrid method for solving nonlinear multiobjective optimization problems. Tabu search is used in the diversification generation method to obtain a diverse approximation to

the Pareto-optimal set of solutions; it is also applied to rebuild the reference set after each iteration of the scatter search algorithm. To measure the quality of the solutions, MOSS uses a weighted sum approach. This algorithm is compared against NSGA-II, SPEA-2, and PESA on a set of unconstrained test functions.

Similarly to MOSS, SSPMO [7] is a scatter search algorithm which includes tabu search, although they differ in the use of different tabu search algorithms. SSPMO obtains a part of the reference set by selecting the best solutions of the initial set P for each of the k objective functions. The rest of the reference set is obtained by using the usual approach of selecting the remaining solutions in P that maximize the distance to the solutions already in the reference set. In contrast to MOSS, the set P is updated with solutions generated in the scatter search main loop. SSPMO is evaluated by using a benchmark of unconstrained test functions.

Compared to MOSS and SSPMO, our proposal is also applied for solving MOPs with continuous bounded variables, but we additionally consider constrained MOPs. We use a non-dominating sorting procedure to build the reference set from the initial set P, and a local search based on a mutation operator is used instead of a tabu search to improve the solutions obtained from the reference set. MOSS and SSPMO do not seem to search a Pareto front with a bounded number of solutions, as it is usual in many evolutionary algorithms for solving MOPS, but they search as many solutions as possible; we consider the former goal, and it is achieved by using the set P as a population where all the non-dominated solutions found in the scatter search loop are stored.

In [8] a scatter search algorithm for solving the bi-criteria multi-dimensional knapsack problems is proposed. This algorithm is tailored to solve a specific problem, so the scatter search methods differ significantly of those used in this work.

Concerning evolutionary algorithms, the micro-GA [4] is similar to our proposal in the sense that they two use a small population and a reinitialization process. However, in the micro-GA this population is very small (typically four members), it is obtained by randomly choosing individuals of another population composed of non-variable and variable parts, and crossover and mutation operators are used to generate new individuals. In contrast, in SSMO the size of the reference set ranges between ten to twenty solutions, which are selected from an initial population choosing the best p individuals according to two different strategies, and new individuals are obtained from the reference set by applying a systematic combination procedure. Finally, the micro-GA uses an external archive to store the non-dominated solutions found, while SSMO uses the initial set P already included in the standard scatter search template.

3 Non-Dominated Sorting Scatter Search Algorithm

SSMO is based on the scatter search template proposed in [9] and its application to solve bounded continuous single objective optimization problems [10]. The template consists of the definition of five methods, as depicted in Fig. 1. We

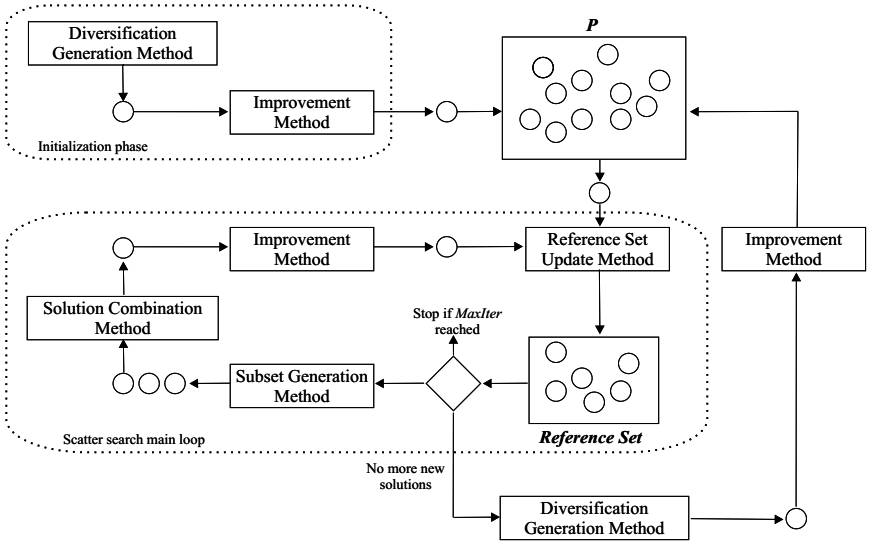

Fig. 1. Outline of the standard scatter search algorithm.

first describe these methods, focusing mainly in the improvement and reference set update procedures, which constitute the basis of our proposal. Then, we detail how the initial population P is managed. Finally, we outline the overall algorithm.

3.1 Scatter Search Methods

Diversification Generation Method This method is basically the same one proposed in [10]. The goal is to obtain an initial set P of diverse solutions. The method consists in dividing the range of each variable in a number of sub-ranges of equal size; then, each solution is obtained in two steps. First, a sub-range is randomly selected, with the probability of selecting a sub-range being inversely proportional to its frequency count (the number of times the sub-range has been selected); second, a value is randomly generated within the selected range.

Improvement Method The idea behind this method is to use a local search algorithm to improve the solutions in the initial set P. In contrast to [10], where a simplex method is used, we have to deal with MOPs which have constraints, so simplex seems not adequate. Instead, we propose an improvement method based on a mutation operator and a Pareto dominance test. We describe the method in Fig. 2.

The improvement method is simple. Taking as an argument an individual, this is repeatedly mutated with the aim of obtaining a better individual. The term "better" is defined here in a similar way as the constrained-dominance approach used in NSGA-II [1]. The constraint violation test checks whether two

```
Individual improvement(Individual originalIndividual, int iter) {
  Individual improvedIndividual
  repeat iter times {
    mutatedIndividual = mutation(originalIndividual)
    if (the problem has constraints) {
      evaluateConstraints(mutatedIndividual)
      best = constraintTest(mutatedIndividual, originalIndividual)
      if (none of them is better than the other one) {
        evaluate(mutatedIndividual)
        best = dominanceTest(mutatedIndividual, originalIndividual)
      } // if
      else if (mutatedIndividual is best)
        evaluate(mutatedIndividual)
    } // if
    else { // the problem has no constraints
      evaluate(mutatedIndividual)
      best = dominanceTest(mutatedIndividual, originalIndividual)
    } // else
    if (mutatedIndividual is best)
      originalIndividual = mutatedIndividual
    else if (originalIndividual is best)
      delete(mutatedIndividual)
    else { // both individuals are non-dominated
      add originalIndividual to P
      originalIndividual = mutatedIndividual
    } // else
  } // repeat
  return originalIndividual
} // improvement
```

Fig. 2. Pseudocode describing the improvement method.

individuals are feasible or not. If one of them is feasible and the other one is not, or both are infeasible but one of them has an smaller overall constraint violation, the test returns the winner. Otherwise, a dominance test is performed to decide whether one of the individuals dominates the other one. If the original individual wins, the mutated one is deleted; if the mutated individual wins, it replaces the original one; finally, if they are both non-dominated, the original individual is moved into the initial set P and the mutated individual becomes the new original one.

We can point out several features of the proposed improvement method. First, mutated individuals are only evaluated if they are going to replace the original individual. Second, in the case of finding several non-dominated solutions in the procedure, they are inserted into P, which could eventually fill. The strategy we propose to deal with this issue is explained in Section 3.2. Finally, we can adjust the improvement effort by tuning the parameter *iter*.

```
referenceSetUpdate(bool build) {
  if (build) { // build a new reference set
    select the p best individuals of P
    build the RefSet1 with these p individuals
    compute Euclidean distances in P to obtain q individuals
    build the RefSet2 with these q individuals
  } // if
  else { // update the reference set
    for (each new solution s) {
      test to insert s in RefSet1
      if (test fails)
        test to insert s en RefSet2
          if (test fails)
            delete s
    } // for
  } // else
} // referenceSetUpdate
```

Fig. 3. Pseudocode describing the reference set update method.

Reference Set Update Method The reference set, *RefSet*, is a collection of both high quality solutions and diverse solutions that are used to generate new individuals by applying the solution combination method. The set itself is composed of two subsets $RefSet_1$ and $RefSet_2$ of size p and q, respectively. The first subset contains the best quality solutions in P, while the second subset should be filled with solutions promoting diversity. In [7] the $RefSet_2$ is constructed by selecting from P those individuals whose minimum Euclidean distance to the $RefSet_1$ is the highest. We keep the same strategy for building the $RefSet_2$, but, as is usual in the multiobjective optimization domain, we have to define the concept of "best individual" to build the $RefSet_1$. On the other hand, the reference set update method is used to generate the reference set, but also to update it with the new solutions obtained in the scatter search main loop (see Fig. 1). A scheme of this method is included in Fig. 3.

To select the best p individuals in P we propose the following two strategies:

1. The first approach is to carry out a non-dominated sorting of P. However, as there will typically be several individuals per rank, some kind of niching value can be assigned to them to decide which are the most promising solutions. We have used the crowding distance used in NSGA-II, but other kind of niching measurement are valid (e.g., the density applied in SPEA-2). Thus, the $RefSet_1$ is composed of the best ranked individuals, being the individuals with the same rank ordered by its crowding distance.
2. The second strategy consists in ranking the population P and then applying a clustering algorithm, such as k-means or a minimum spanning tree method, to obtain p centroids of the set composed of individuals with best rank. The centroids are individuals which are representative of the set of solutions they derive, so they can be promising elements to compose the $RefSet_1$. We

```
// Test to update the RefSet1 with individual s
bool dominated = false
for (each solution r in RefSet1)
  if (s dominates r)
    remove r from RefSet1
  else (if r dominates s)
    dominated = true
if (not dominated)
  if (RefSet1 not full)
    add s to RefSet1
  else
    add s to P
else // the individual s is dominated
  // test to update the RefSet2 with individual s
  ...
```

Fig. 4. Pseudocode describing the test to add new individuals to $RefSet_1$.

use the Euclidean distance as the metric to assess the similarity among the individuals.

Once the reference set is completed, its solutions are combined to obtain new solutions which, after applying the improvement method to them, are checked against those belonging to the reference set. According to the scatter search template, a new solution can become a member of the reference set if either one of the following conditions is satisfied:

- The new individual has better objective function value than the individual with the worst objective value in $RefSet_1$.
- The new individual has a better distance value to the reference set than the individual with the worst distance value in $RefSet_2$.

While the second condition holds in the case of multiobjective optimization, we have again the decide about the concept of best individual concerning the first condition. To determine whether a new solution is better than another one in $RefSet_1$ (i.e., the test to insert a new individual s in $RefSet_1$, as it appears in Fig. 3) we cannot use a ranking procedure because the size of this population usually is small (typically the size of the whole reference set is 20 or less). Our approach is to compare each new solution i to the individuals in $RefSet_1$ using a dominance test. This test is included in Fig. 4. (For the sake of simplicity, we do not consider here constraints in the MOP. The procedure to deal with constraints is as explained in the improvement method in Fig. 2.)

Let us note that a new individual does not replace another one in $RefSet_1$. Instead, it is inserted into that set if it is non-dominated by $RefSet_1$ and this is not full; otherwise, it is sent to the set P. This way, we try to keep all non-dominated solutions found by using P as a kind of archive of non-dominated solutions. As we mentioned in the improvement method subsection, this can lead the set P to fill. This issue is considered in Section 3.2.

Subset Generation Method This method generates subsets of individuals, which will be used for creating new solutions with the solution combination method. Several kinds of subsets are possible [10]. We restrict this method to generate all pairwise combinations of solutions in the reference set.

Solution Combination Method The idea of this method is to find linear combinations of reference solutions. Again, we use the same method proposed in [10], where each pair of solutions x_1 and x_2 can lead to two, three, or four new solutions, depending on whether x_1 and x_2 belong to $RefSet_1$ or $RefSet_2$.

3.2 Managing the Initial Population

Prior to describing the full algorithm of SSMO, we still need to define a procedure to manage the set P. In particular, when new non-dominated solutions are found by the improvement and reference set update methods, they can be inserted in P, which can eventually fill. This issue is also important because diversity can be improved depending on the applied strategy.

Our approach is to allow the set P to grow until a certain limit. Therefore, if P is intended to store up to t individuals, we extend this limit to, for example, $2t$. When a new individual is going to be added to P, a test checking whether there is already an individual with the same objective function values is executed. If the test is successful, the individual is deleted; otherwise, it is inserted into P. If P has reached to its limit, a cutoff procedure is invoked.

The cutoff procedure performs a ranking of P and then it removes all individuals except those with the best rank (i.e., the individuals with rank equal to 0). If after the removing the size is greater than t, the crowding distance of the individuals is calculated, P is ordered according to this value, and the solutions falling into positions beyond t are removed.

3.3 Outline of SSMO

Once the five methods of the scatter search have been proposed and a procedure to manage the population P has been defined, we are now ready to give an overall view of the technique. The outline depicted in Fig. 5 shows that the SSMO algorithm is simple.

Initially, the diversification generation method is invoked to generate s initial solutions, and each of them is passed to the improvement method. The result is the initial set P. Then, a number of iterations is performed (the outer loop in Fig. 5). In each iteration, the reference set is built, the subset generation method is invoked, and the main loop of the scatter search algorithm is executed until there are no new solutions. Then, the individuals in $RefSet_1$ are inserted into P. The number of iterations can be fixed, or it can depend on other conditions; here, we have used as stop condition the computation of a preprogrammed number of fitness evaluations (see next section). Finally, the cutoff procedure is invoked to remove dominated solutions from P.

```
construct the initial set P
// outer loop
until (stop condition) {
  referenceSetUpdate(build=true)
  subsetGeneration()
  // scatter search main loop
  while (new subsets are generated) {
    combination()
    for (each combinated individual) {
      improvement() ;
      referenceSetUpdate(build=false)
    } // for
    subsetGeneration()
  } // while
  add RefSet1 to P
} // until
cutoff()
```

Fig. 5. Outline of the SSMO algorithm.

4 Computational Results

This section is devoted to the evaluation of SSMO. We have chosen several test problems taken from the specialized literature, and we have analyzed the results taking as a reference those obtained with NSGA-II.

Given that SSMO is a real-coded evolutionary algorithm, we have used the real-coded NSGA-II with the parameter settings suggested in [1]. A crossover probability of $p_c = 0.9$ and a mutation probability $p_m = 1/n$ (where n is the number of decision variables) are used. The operators for crossover and mutation are simulated binary crossover (SBX) and polynomial mutation, with distribution indexes of $\eta_c = 20$ and $\eta_m = 20$, respectively. The population size is 100 individuals, and the algorithm is run for 250 iterations.

For SSMO we have chosen a reasonable set of values, and we have not made any effort to find the best parameter settings. The size of P is 100, and it can grow up to 200 individuals. The mutation operator used in the improvement method is the same as in NSGA-II, polynomial mutation, with the same value of η_m. The size of the $RefSet_1$ and $RefSet_2$ is 10 in both sets. The number of iterations in the improvement method has a value of $iter = 10$. The algorithm is run until 25000 function evaluations are computed.

SSMO is written in C++. We have compiled the software with GCC V3.2 and optimization level -O3, and the experiments have been executed in a Pentium 4 at 2.8GHz with 512 MB of RAM, running Suse Linux 8.1 (kernel 2.4.19).

4.1 Test Problems

We have selected both constrained and unconstrained problems that have been used in studies in this area. Given that they are widely known, we do not include

Table 1. Unconstrained test functions.

Problem	Objective functions	Variable bounds	n
Schaffer	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$	$-10^5 \leq x \leq 10^5$	1
Fonseca	$f_1(\boldsymbol{x}) = 1 - e^{-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2}$ $f_2(\boldsymbol{x}) = 1 - e^{-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2}$	$-4 \leq x_i \leq 4$	3
Kursawe	$f_1(\boldsymbol{x}) = \sum_{i=1}^{n-1}(-10e^{(-0.2*\sqrt{x_i^2+x_{i+1}^2})})$ $f_2(\boldsymbol{x}) = \sum_{i=1}^{n}(\lvert x_i\rvert^a + 5\sin(x_i)^b)$	$-5 \leq x_i \leq 5$	3
ZDT1	$f_1(\boldsymbol{x}) = x_1$ $f_2(\boldsymbol{x}) = g(x)[1 - \sqrt{x_1/g(x)}]$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$0 \leq x_i \leq 1$	30
ZDT2	$f_1(\boldsymbol{x}) = x_1$ $f_2(\boldsymbol{x}) = g(x)[1 - (x_1/g(x))^2]$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$0 \leq x_i \leq 1$	30
ZDT3	$f_1(\boldsymbol{x}) = x_1$ $f_2(\boldsymbol{x}) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)}\sin(10\pi x_1)]$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$0 \leq x_i \leq 1$	30
ZDT4	$f_1(\boldsymbol{x}) = x_1$ $f_2(\boldsymbol{x}) = g(x)[1 - (x_1/g(x))^2]$ $g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10\cos(4\pi x_i)])$	$0 \leq x_1 \leq 1$ $-5 \leq x_i \leq 5$ $i = 2, \ldots, n$	10
ZDT6	$f_1(\boldsymbol{x}) = 1 - e^{-4x_1}\sin^6(6\pi x_1)$ $f_2(\boldsymbol{x}) = g(x)[1 - (f_1(\boldsymbol{x})/g(x))^2]$ $g(x) = 1 + 9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}$	$0 \leq x_i \leq 1$	10

full details of them here for space constraints. They can be found in the cited references and also in books such as [11] and [12].

The selected unconstrained problems include the studies of Schaffer [13], Fonseca [14], and Kursawe [15], as well as the problems ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6, which are defined in [16]. Their formulation is provided in Table 1. The constrained problems are Osyczka2 [17], Tanaka [18] (which are respectively known as MOP-C2 and and MOP-C4 in [11]), Srinivas [19], Constr_Ex [1], and Golinski [20]. They are described in Table 2.

4.2 Performance Metrics

Several metrics have been proposed for measuring the results of Pareto-based multi-objective optimization algorithms. In this work we use the metrics M_1^* [16] and Δ [1]. The former gives the average distance to the Pareto optimal set and the latter is a diversity metric that measures the extent of spread achieved among the obtained solutions. Their formulations are:

$$M_1^* = \frac{1}{|Y'|}\sum_{d' \in Y'} \min\{\|d' - \bar{d}\|^*; \bar{d} \in \bar{Y}\} \tag{1}$$

where Y is the set of all possible objective vectors, $Y' \subseteq Y$ is the set of objective vectors found, and $\bar{Y} \subseteq Y$ is the set of solutions of the Pareto optimal set. This metric ideally should be zero. Δ is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1}|d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}} \tag{2}$$

Table 2. Constrained test functions.

Problem	Objective functions	Constraints	Variable bounds	n
Osyczka2	$f_1(\boldsymbol{x}) = -(25(x_1-2)^2+$ $(x_2-2)^2+$ $(x_3-1)^2(x_4-4)^2+$ $(x_5-1)^2)$ $f_2(\boldsymbol{x}) = x_1^2+x_2^2+$ $x_3^2+x_4^2+x_5^2+x_6^2$	$g_1(\boldsymbol{x}) = 0 \leq x_1+x_2-2$ $g_2(\boldsymbol{x}) = 0 \leq 6-x_1-x_2$ $g_3(\boldsymbol{x}) = 0 \leq 2-x_2+x_1$ $g_4(\boldsymbol{x}) = 0 \leq 2-x_1+3x_2$ $g_5(\boldsymbol{x}) = 0 \leq 4-(x_3-3)^2-x_4$ $g_6(\boldsymbol{x}) = 0 \leq (x_5-3)^3+x_6-4$	$0 \leq x_1, x_2 \leq 10$ $1 \leq x_3, x_5 \leq 5$ $0 \leq x_4 \leq 6$ $0 \leq x_6 \leq 10$	6
Tanaka	$f_1(\boldsymbol{x}) = x_1$ $f_2(\boldsymbol{x}) = x_2$	$g_1(\boldsymbol{x}) = -x_1^2-x_2^2+1+$ $0.1\cos(16\arctan(x_1/x_2)) \leq 0$ $g_2(\boldsymbol{x}) = (x_1-0.5)^2+$ $(x_2-0.5)^2 \leq 0.5$	$-\pi \leq x_i \leq \pi$	2
Constr_Ex	$f_1(x) = x_1$ $f_2(x) = (1+x_2)/x_1$	$g_1(\boldsymbol{x}) = x_2+9x_1 \geq 6$ $g_2(\boldsymbol{x}) = -x_2+9x_1 \geq 1$	$0.1 \leq x_1 \leq 1.0$ $0 \leq x_2 \leq 5$	2
Srinivas	$f_1(\boldsymbol{x}) = (x_1-2)^2+$ $(x_2-1)^2+2$ $f_2(\boldsymbol{x}) = 9x_1-(x_2-1)^2$	$g_1(\boldsymbol{x}) = x_1^2+x_2^2 \leq 225$ $g_2(\boldsymbol{x}) = x_1-3x_2 \leq -10$	$-20 \leq x_i \leq 20$	2
Golinski	$f_1(\boldsymbol{x}) = 0.7854x_1x_2^2(10x_3^2/3+$ $14.933x_3-43.0934)$ $-1.508x_1(x_6^2+x_7^2)+$ $7.477(x_6^3+x_7^3)$ $+0.7854(x_4x_6^2+x_5x_7^2)$ $f_2(\boldsymbol{x}) = \frac{\sqrt{(\frac{745.0x_4}{x_2x_3})^2+1.69*10^7)}}{0.1x_6^3}$	$g_1(\boldsymbol{x}) = \frac{1.0}{x_1x_2^2x_3}-\frac{1.0}{27.0} \leq 0$ $g_2(\boldsymbol{x}) = \frac{1.0}{x_1x_2^2x_3^3}-\frac{1.0}{27.0} \leq 0$ $g_3(\boldsymbol{x}) = \frac{x_4^3}{x_2x_3^2x_6^4}-\frac{1.0}{1.93} \leq 0$ $g_4(\boldsymbol{x}) = \frac{x_5^3}{x_2x_3x_7^4}-\frac{1.0}{1.93} \leq 0$ $g_5(\boldsymbol{x}) = x_2x_3-40 \leq 0$ $g_6(\boldsymbol{x}) = x_1/x_2-12 \leq 0$ $g_7(\boldsymbol{x}) = 5-x_1/x_2 \leq 0$ $g_8(\boldsymbol{x}) = 1.9-x_4+1.5x_6 \leq 0$ $g_9(\boldsymbol{x}) = 1.9-x_5+1.1x_7 \leq 0$ $g_{10}(\boldsymbol{x}) = f_2(x) \leq 1300$ $a = 745.0x_5/x_2x_3$ $b = 1.575*10^8$ $g_{11}(\boldsymbol{x}) = \frac{\sqrt{(a)^2+b)}}{0.1x_7^3} \leq 1100$	$2.6 \leq x_1 \leq 3.6$ $0.7 \leq x_2 \leq 0.8$ $17.0 \leq x_3 \leq 28.0$ $7.3 \leq x_4 \leq 8.3$ $7.3 \leq x_5 \leq 8.3$ $2.9 \leq x_6 \leq 3.9$ $5.0 \leq x_7 \leq 5.5$	7

where d_i is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and d_f and d_l are the Euclidean distances to the *extreme* solutions of the exact Pareto front in the objective space (see [1] for the details). The ideal value of Δ is also 0, indicating a perfect spreadout of the solutions along the Pareto front. To calculate these metrics we have used the true Pareto fronts obtained by enumeration of the test problems (excepting the ZDTx family) publicly available at `http://neo.lcc.uma.es/software/esam`.

4.3 Discussion of the Results

The results are summarized in Table 3 (M_1^*) and Table 4 (Δ). For each problem, we have carried out 30 independent runs, and the tables include the mean $\bar{x}$ and standard deviation σ_n, as well as the result of an ANOVA (Analysis of Variance) test with a 5% of significance level (marked as "+" in tables). An ANOVA [21] tests the difference between the means of two or more sets of numeric values.

We have tested two versions of our algorithm. The first version, SSMOv1, selects the best individuals from the reference set using ranking and crowding, while the second version, SSMOv2, uses a clustering procedure to obtain the centroids that will compose the *RefSet1* (see Section 3.1).

Table 3. Mean and standard deviation of the convergence metric $M1^*$.

	NSGA-II		SSMOv1		SSMOv2		
Problem	$\bar{x}$	σ_n	$\bar{x}$	σ_n	$\bar{x}$	σ_n	A
Schaffer	0.0223	0.0011	0.0225	0.0011	0.0225	0.0012	-
Fonseca	0.0025	0.0002	0.0021	0.0002	0.0019	0.0002	+
Kursawe	0.0134	0.0027	0.0531	0.0663	0.0185	0.0036	+
ZDT1	0.0005	6.3e-5	0.0004	5.7e-5	0.0004	4.2e-5	+
ZDT2	0.0004	3.2e-5	0.0004	2.9e-5	0.0004	2.8e-5	+
ZDT3	0.0025	0.0002	0.0016	9.3e-5	0.0020	0.0002	+
ZDT4	0.0044	0.0029	25.6412	12.6266	38.4334	11.1194	+
ZDT6	0.0764	0.0076	0.5954	0.3170	0.2604	0.1773	+
Tanaka	0.0044	0.0003	0.0095	0.0010	0.0096	0.0048	+
Osyczka2	5.4688	8.6247	16.8277	15.2031	13.8621	11.9258	+
Srinivas	0.2570	0.0421	0.2839	0.0489	0.2274	0.0558	+
Constr_Ex	0.0053	0.0003	3.5802	0.4226	3.0383	0.2215	+
Golinski	4.0790	0.7839	35.2460	24.9212	10.9907	11.8089	+

We analyze first the two versions of SSMO. Considering the metric $M1^*$, we can observe that SSMOv2 converges better in 7 out of the 13 problems, while SSMOv1 provides better results only in 3 problems. If we take into account the metric Δ, SSMOv1 and SSMOv2 behave better in seven and six problems, respectively. These observations indicate that the use of centroids for building the reference set is a promising approach for improving the accuracy of the algorithm that can be used as the basis of future developments.

We now turn to compare our algorithm with NSGA-II. If we analyze the unconstrained problems, we observe that, with the exception of the problems ZDT4 and ZDT6, the two versions of SSMO obtain competitive performance. If we consider that this is just a first approach of using scatter search for MOPs, this sounds quite promising.

Concerning the constrained problems, the two metrics indicate that the results of SSMO are comparable to NSGA-II in terms of diversity, but they are slightly worse in terms of convergence. With respect to the problems Tanaka and Srinivas our two proposals show comparable performance to NSGA-II.

In order to better illustrate the working principles of SSMO, we show a typical simulation result with the problem Kursawe. This problem has three discontinuous regions in the Pareto-optimal front. Fig 6 shows the true Pareto front obtained by enumeration (left) and the solutions obtained by SSMOv2. Next, we show the nondominated solutions of the problem Srinivas. The true Pareto front yielded by the enumerative search appears in Fig 7 (left), while the right part of that figure shows the resulting front from SSMOv2. Finally, in Fig. 8 we show the results of solving the problem ZDT3 with NSGA-II (left) and SSMOv2 (right). This problem has a number of disconnected Pareto-optimal fronts.

We conclude that a more comprehensive study is still necessary to understand the behavior of SSMO. The choice of a size of 10 for both the $RefSet_1$ and $RefSet_2$ intends to keep a balance between intensification and diversification,

Table 4. Mean and standard deviation of the diversity metric Δ.

	NSGA-II		SSMOv1		SSMOv2		
Problem	$\bar{x}$	σ_n	$\bar{x}$	σ_n	$\bar{x}$	σ_n	A
Schaffer	0.4202	0.0264	0.3984	0.0273	0.4323	0.0268	+
Fonseca	0.3756	0.0259	0.3448	0.0324	0.3695	0.0359	+
Kursawe	0.5380	0.0285	0.6720	0.0785	0.5043	0.0542	+
ZDT1	0.5252	0.0333	0.4855	0.0770	0.4871	0.0455	+
ZDT2	0.5149	0.0385	0.5981	0.1660	0.4450	0.0371	+
ZDT3	0.6278	0.0288	1.1508	0.0685	0.7004	0.0582	+
ZDT4	0.4843	0.1750	0.9219	0.0544	0.9766	0.0886	+
ZDT6	0.6078	0.0404	1.3229	0.0604	1.2989	0.1379	+
Tanaka	0.6427	0.0256	0.8694	0.0463	1.0619	0.0931	+
Osyczka2	0.7884	0.0958	1.0817	0.1451	0.9398	0.2197	+
Srinivas	0.3843	0.0353	0.3515	0.0347	0.3964	0.0378	+
Constr_Ex	0.7655	0.0371	0.9070	0.0468	0.8057	0.0228	+
Golinski	0.7516	0.0289	0.9908	0.1682	0.6840	0.1137	+

but most probably other values would enhance the search. On the other hand, our experiments reveal that using a different strategy to update the $RefSet_1$ with new improved individuals allows better performance to be obtained with some difficult problems; for example, in the case of the problem ZDT4, the mean value of the metric $M1^*$ reduces from 38.4334 to 10.4382, and in 15 of the 30 experiments, the value of $M1^*$ is below 0.0038.

5 Conclusions and Future Work

We have proposed a first approximation to the utilization of a scatter search method to solve multiobjective optimization problems. Two variants of our approach have been compared against NSGA-II in thirteen different difficult problems taken from the literature. In the unconstrained test functions used, our

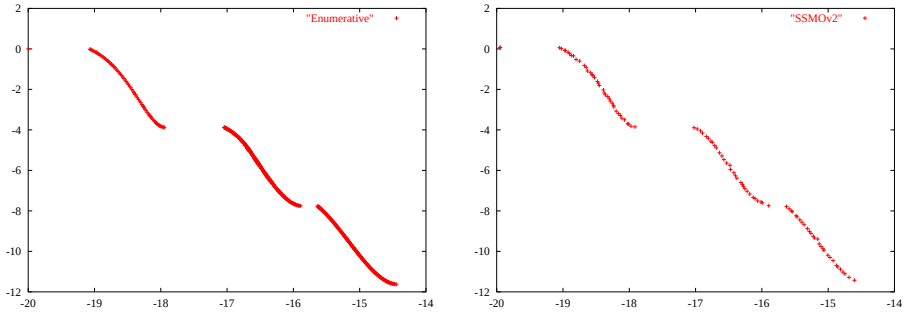

Fig. 6. Exact Pareto front for the problem Kursawe (left) and nondominated solutions obtained with SSMOv2 (right).

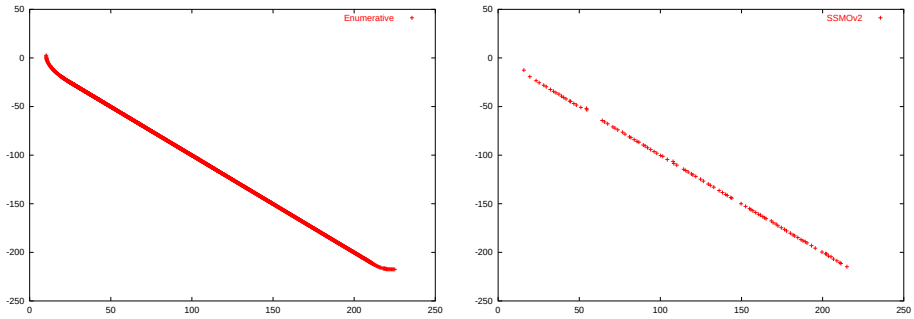

Fig. 7. Exact Pareto front for the problem Srinivas (left) and nondominated solutions obtained with SSMOv2 (right).

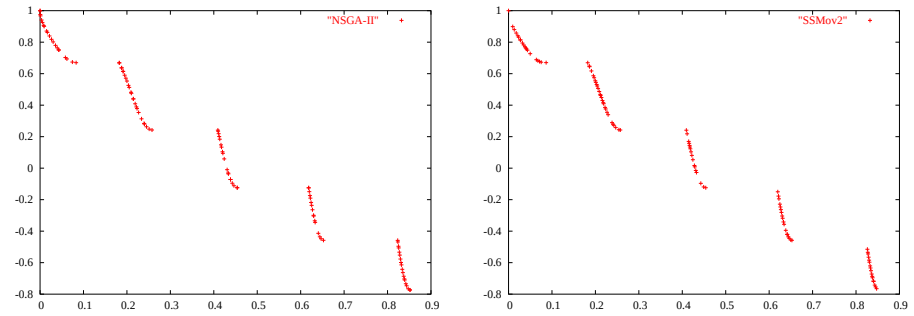

Fig. 8. Nondominated solutions obtained for the problem ZDT3 with NSGA-II (left) and SSMOv2 (right).

algorithm has obtained comparable performance to NSGA-II in six of the eight selected problems. In the unconstrained problems, a more obscure scenario appears, being acccurate for Tanaka and Srinivas, but showing difficulties to solve the problems Constr_Ex, Osyczka2, and Golinski.

A deep study to find the best parameters defining the behavior of SSMO is a matter of future work. In this sense, we also plan to study new strategies for the improvement and reference set update methods, as well as other approaches to store and manage the non-dominated solutions encountered during the execution of the algorithm, such as using an external archive of nondominated solutions.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197
2. Knowles, J., Corne, D.: The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press (1999) 9–105

3. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001)
4. Coello, C.A., Toscano, G.: Multiobjective Optimization Using a Micro-Genetic Algorithm. In: GECCO-2001. (2001) 274–282
5. Glover, F., Laguna, M., Martí, R.: Fundamentals of Scatter Search and Path Relinking. Control and Cybernetics **29** (2000) 653–684
6. Beausoleil, R.P.: MOSS: Multiobjective Scatter Search Applied to Nonlinear Multiple Criteria Optimization. To appear in the European Journal of Operational Research (2004)
7. Caballero, R., Laguna, M., Molina, J., Martí, R.: SSPMO: A Scatter Search Procedure for Non-Linear Multiobjective Optimization. Submitted to INFORMS Journal on Computing (2004)
8. da Silva, C.G., Clímaco, J., Figueira, J.: A Scatter Search Method for the Bi-Criteria Multi-Dimensional {0,1}-Knapsack Problem using Surrogate Relaxation. Journal of Mathematical Modelling and Algorithms **3** (2004) 183–208
9. Glover, F.: A Template for Scatter Search and Path Relinking. In J. K. Hao and E. Lutton and E. Ronald and M. Shoenauer and D. Snyers, ed.: Artificial Evolution. Lecture Notes in Computer Science, Springer Verlag (1997)
10. Glover, F., Laguna, M., Martí, R.: Scatter Search. In Ghosh, A., Tsutsui, S., eds.: Advances in Evolutionary Computing: Theory and Applications. Springer (2003)
11. Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers (2002)
12. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons (2001)
13. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In Grefensttete, J., ed.: First International Conference on Genetic Algorithms, Hillsdale, NJ (1987) 93–100
14. Fonseca, C.M., Flemming, P.J.: Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms - Part II: Application Example. IEEE Transactions on System, Man, and Cybernetics **28** (1998) 38–47
15. Kursawe, F.: A Variant of Evolution Strategies for Vector Optimization. In Schwefel, H., Männer, R., eds.: Parallel Problem Solving for Nature, Berlin, Germany, Springer-Verlag (1990) 193–197
16. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. IEEE Transactions on Evolutionary Computation **8** (2000) 173–195
17. Osyczka, A., Kundo, S.: A New Method to Solve Generalized Multicriteria Optimization Problems Using a Simple Genetic Algorithm. Structural Optimization **10** (1995) 94–99
18. Tanaka, M., Watanabe, H., Furukawa, Y., Tanino, T.: GA-Based Decision Support System for Multicriteria Optimization. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Volume 2. (1995) 1556–1561
19. Srinivas, N., Deb, K.: Multiobjective Function Optimization Using Nondominated Sorting Genetic Algorithms. Evolutionary Computation **2** (1995) 221–248
20. Kurpati, A., Azarm, S., Wu, J.: Constraint Handling Improvements for Multi-Objective Genetic Algorithms. Structural and Multidisciplinary Optimization **23** (2002) 204–213
21. Montgomery, D.C.: Design and Analysis of Experiments. 3 edn. John Wiley, New York (1991)