Engineering Optimization
Vol. 00, No. 00, September 2007, 1–28

Hybridizing Evolutionary Strategies with Continuation Methods for Solving Multi-Objective Problems

Oliver Schütze[1], Carlos A. Coello Coello[2], Sanaz Mostaghim[3]
El-ghazali Talbi[1] and Michael Dellnitz[4]
[1]**(corresponding author)**
INRIA Futurs, LIFL, CNRS Bât M3, Cité Scientifique
59655 Villeneuve d'Ascq, FRANCE
email: {`schuetze,talbi`}`@lifl.fr`
[2]CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México D.F. 07300, MEXICO
email: `ccoello@cs.cinvestav.mx`
[3] AIFB Institute
University of Karlsruhe, GERMANY
email: `mostaghim@aifb.uni-karlsruhe.de`
[4] University of Paderborn
Faculty for Computer Science, Electrical Engineering and Mathematics
Institute for Mathematics
D-33098 Paderborn, Warburger Strasse 100, GERMANY
email: `dellnitz@upb.de`

(November 2006)

In this paper, two techniques for the numerical treatment of multi-objective optimization problems—a continuation method and a particle swarm optimizer—are combined in order to join their particular advantages. Continuation methods can be applied very efficiently to perform the search along the Pareto set, even for high-dimensional models, but are of local nature. In contrast, many multi-objective particle swarm optimizers tend to have slow convergence, but instead, accomplish the "global task" exceedingly. In this paper, an algorithm which combines these two techniques is proposed, some convergence results for continuous models are provided, possible realizations are discussed, and finally some numerical results are presented indicating the strength of the novel approach.

1 Introduction

Multi-objective optimization is a research field that has attracted a significant amount of interest in the last few years (Ehrgott 2005, Miettinen 1999). One of the emergent sub-areas within multi-objective optimization has been the use of metaheuristics, which is a discipline that has experienced a significant growth in the last ten years (Coello et al. 2002, Deb 2001, Ehrgott and Gandibleux 2004, Tan et al. 2005).

However, despite the considerable interest raised by solving multi-objective optimization problems using metaheuristics, certain research topics have remained only scarcely studied. One of them is the hybridization between mathematical programming techniques (or concepts) and metaheuristics. This sort of hybridization is precisely the focus of this paper, in which a continuation method is combined with a particle swarm optimizer (Coello Coello et al. 2004).

The remainder of this paper is organized as follows. Section 2 presents the previous related work. Section 3 contains the basic concepts requried to understand the rest of the paper. A novel approach is described in Section 4, including a proof of convergence, and a discussion of some of its possible realizations and its computational cost. Section 5 contains some numerical results of the proposed hybrid. The assessment of the results is presented in Section 6. Finally, Section 7 contains some conclusions as well as possible paths for future research.

2 Related Work

There are relatively few attempts to combine mathematical programming techniques and concepts with multi-objective evolutionary algorithms. For example, some researchers have attempted to use gradient information to define search directions of stochastic search methods (e.g., (Lahanas et al. 2003, Brown and Smith 2003, Bosman and de Jong 2005, 2006)) by adapting ideas from the mathematical programming literature (Fliege and Fux Svaiter 2000, Schäffler et al. 2002)).

The same idea of using local information (but without using gradients) for generating promising search directions has been exploited in the so-called memetic algorithms, whose application in multi-objective optimization has become more extended within the last few years (Knowles and Corne 2000, Gandibleux et al. 2001, Knowles and Corne 2005).

Other researchers have combined multilevel subdivision techniques (Dellnitz et al. 2005) with multi-objective evolutionary algorithms (Coello et al. 2002), in order to increase performance (Schütze et al. 2003). The idea is to subdivide the search space in order to perform local search in each subspace.

Continuation methods (Hillermeier 2001, Schütze 2004) have been used in the mathematical programming literature as a mechanism for recovering the portions of the Pareto front that are normally produced when generating it with traditional multi-objective optimization techniques (e.g., gradient-based). The underlying assumption of these methods is the fact that, under mild conditions, the Pareto front of a continuous problem, is a piecewise continuous $(k-1)$-dimensional manifold (k is the number of objectives). Other researchers have recently exploited this same concept to devise algorithms that exploit this regularity and are, therefore, more efficient (see for example (Zhou et al. 2006)).

One approach to combine multi-objective path following methods with evolutionary strategies can be found in Harada et al. (2007). Although the preliminary results presented are very promising, the algorithm can get computationally expensive in particular for higher dimensional models due to its uniform sampling approach. Additionally, this method is restricted to the bi-objective case.

Finally, regarding convergence of multi-objective evolutionary algorithms towards the Pareto set, little theoretical work still exists, and this is mostly focused on discrete models (Rudolph 1998, Rudolph and Agapie 2000, Laumanns et al. 2002).

3 Background

In this section the required background for the algorithm described in Section 4 is summarized in brief: the concept of Pareto optimality and some theoretical background, the Recovering algorithm as well as Particle Swarm Optimization techniques.

3.1 *Multi-Objective Optimization*

In a *multi–objective optimization problem* (MOP) the task is to simultaneously optimize k *objective functions* $f_1, \ldots, f_k : \mathbb{R}^n \to \mathbb{R}$. More precisely, a general unconstrained MOP can be stated as follows:

$$\min_{x \in \mathbb{R}^n} \{F(x)\}, \tag{MOP}$$

where the function F is defined as the vector of the objective functions

$$F : \mathbb{R}^n \to \mathbb{R}^k, \qquad F(x) = (f_1(x), \ldots, f_k(x)).$$

Obviously, it has to be defined what is meant by finding the minimum of a vector valued function in (MOP). For this, the following definition is stated

Definition 3.1 *(a) Let* $v, w \in \mathbb{R}^k$*. Then the vector* v *is* less than w *(*$v <_p w$*), if* $v_i < w_i$ *for all* $i \in \{1, \dots, k\}$*. The relation* $\leq_p$ *is defined in an analogous way.*

(b) A vector $v \in \mathbb{R}^k$ *is* dominated *by a vector* $w \in \mathbb{R}^k$ *if* $w \leq_p v$ *and* $v \neq w$ *(i.e. there exists a* $j \in \{1, \dots, k\}$ *such that* $w_j < v_j$*).*

Since $\leq_p$ just defines a partial order on $\mathbb{R}^n$, one cannot proceed as in the classical scalar case. In fact, one cannot expect to find isolated stationary points. Rather one has to find the set of "optimal compromises" and – following Pareto (V. Pareto (1964 (first edition in 1896))) – these are defined in the following way.

Definition 3.2 *(a) Consider the multi–objective optimization problem (MOP). Then a point* $\bar{x} \in R$ *is called* (globally) Pareto optimal *or a* (global) Pareto point *if there is no* $y \in R$ *such that*

$$F(y) \neq F(\bar{x}) \quad \text{and} \quad F(y) \leq_p F(\bar{x}). \tag{3.1}$$

(b) A point $\bar{x} \in R$ *is a* local Pareto point*, if there is a neighborhood* $U(\bar{x}) \subset R$ *of* $\bar{x}$ *such that there is no* $y \in U(\bar{x})$ *satisfying (3.1).*

(c) A point $x \in \mathbb{R}^n$ *is* weakly Pareto optimal *if there does not exist another point* $y \in \mathbb{R}^n$ *such that* $F(y) <_p F(x)$*.*

Fundamental for most of the methods for the numerical treatment of MOPs is the following theorem of Kuhn and Tucker (1951) which states a necessary condition for Pareto optimality for MOPs.

Theorem 3.1 *Let* x^* *be a Pareto point of (MOP). Then there exists a vector* $\alpha \in \mathbb{R}^k$ *with* $\alpha_i \geq 0, i = 1, \dots, k,$ *and* $\sum_{i=1}^k \alpha_i = 1$ *such that*

$$\sum_{i=1}^{k} \alpha_i \nabla f_i(x^*) = 0. \tag{3.2}$$

Obviously, (3.2) is not a sufficient condition for (local) Pareto optimality. On the other hand points satisfying (3.2) are certainly 'Pareto candidates' and thus, following Miettinen (1999), their relevance is now emphasized by the following

Definition 3.3 *A point* $x \in \mathbb{R}^n$ *is called a* substationary point *or* Karush–Kuhn–Tucker point[1] *(KKT–point) if there exist scalars* $\alpha_1, \ldots, \alpha_k \geq 0$ *with* $\sum_{i=1}^k \alpha_i = 1$ *such that (3.2) is satisfied.*

Having stated Theorem 3.1, one is in the position to give a qualitative description of the set of Pareto optimal solutions.
Denote by $\tilde{F} : \mathbb{R}^{n+k} \to \mathbb{R}^{n+1}$ the following auxiliary function:

$$\tilde{F}(x,\alpha) = \begin{pmatrix} \sum\limits_{i=1}^{k} \alpha_i \nabla f_i(x) \\ \sum\limits_{i=1}^{k} \alpha_i - 1 \end{pmatrix}. \qquad (3.3)$$

By Theorem 3.1 it follows that for every substationary point $x^* \in \mathbb{R}^n$ there exists a vector $\alpha^* \in \mathbb{R}^k$ such that

$$\tilde{F}(x^*, \alpha^*) = 0. \qquad (3.4)$$

Hence one expects that the set of KKT-points defines a $(k-1)$-dimensional manifold due to the Implicit Function Theorem. This is indeed the case under certain smoothness assumptions. See (Hillermeier 2001) for a thorough discussion of this topic.

3.2 *Recovering Techniques*

Here a continuation method is described for the computation of KKT-points which utilizes observation (3.4): starting with a given point $\hat{x} \in \tilde{F}^{-1}(0)$ of an MOP, these techniques can be applied to detect further solutions in the neigborhood of $\hat{x}$. In the next step, again further points are computed starting with these new-found solutions, and so on.
One crucial point when using these methods – in particular for MOPs with more than two objectives – is to obtain a 'global picture' of the part of $\tilde{F}^{-1}(0)$ that is covered (in some suitable sense) by the set of solutions which are already computed. As a tool to obtain this as well as to maintain a good spread of the solutions, *boxes* are used, which will be decribed in the following.

Let us assume that every parameter is restricted to a certain range, i.e.

$$a_i \leq x_i \leq b_i, \quad i = 1, \ldots, M, \qquad (3.5)$$

---

[1] Named after the works of Karush (1939) and Kuhn & Tucker (1951) for scalar–valued optimization problems.

where $M = n + k$. The search space thus is given by

$$Q = [a_1, b_1] \times \ldots \times [a_M, b_M] \subset \mathbb{R}^M. \tag{3.6}$$

Every box $B \subset \mathbb{R}^M$ can be represented by a center $c \in \mathbb{R}^M$ and a radius $r \in \mathbb{R}^M_+$ such that

$$B = B(c, r) = \{x \in \mathbb{R}^M \ : \ |x_i - c_i| \leq r_i \ \forall i = 1, \ldots, M\}.$$

The box B can be subdivided with respect to the the j-th coordinate. This division leads to two boxes $B_-(c^-, \hat{r})$ and $B_+(c^+, \hat{r})$, where

$$\hat{r}_i = \left\{ \begin{array}{cl} r_i & \text{for } i \neq j \\ r_i/2 & \text{for } i = j \end{array} \right. , \quad c_i^{\pm} = \left\{ \begin{array}{cl} c_i & \text{for } i \neq j \\ c_i \pm r_i/2 & \text{for } i = j \end{array} \right. .$$

Let $P(Q, 0) := Q$, that is, $P(Q, 0) = B(c^0, r^0)$, where

$$c_i^0 = \frac{a_i + b_i}{2}, \quad r_i^0 = \frac{b_i - a_i}{2}, \quad i = 1, \ldots, M.$$

Denote by $\mathcal{P}(Q, d), d \in \mathbb{N}$, the set of boxes obtained after d subdivision steps starting with $B(c^0, r^0)$, where in each step $i = 1, \ldots, d$ the boxes are subdivided with respect to the j_i-th coordinate, where j_i is varied cyclically. That is, $j_i = ((i - 1) \mod n) + 1$. Note that for every point $y \in Q \backslash \partial Q$ and every subdivision step d there exists exactly one box $B = B(y, d) \in \mathcal{P}(Q, d)$ with center c and radius r such that $c_i - r_i \leq y_i < c_i + r_i, \ \forall i = 1, \ldots, M$. Thus, every set of solutions $\mathcal{S}_B$ leads to a (unique) set of box collections $\mathcal{B}_d(\mathcal{S}_B) := \{B(y, d) \in \mathcal{P}(Q, d) \,|\, y \in \mathcal{S}_B\}$. These collections can easily be stored in a binary tree with depth d (see (Dellnitz and Hohmann 1997)). Note that each set $\mathcal{B}_d$ is completely determined by the tree structure and the initial box $B(c^0, r^0)$. Using this scheme, the memory requirements grow only linearly in the dimension M of the problem.

Having stated these preliminaries, the continuation algorithm for the computation of solution sets $\tilde{F}^{-1}(0)$ of given MOPs can be presented. Starting with a box collection $\mathcal{B} \subset \mathcal{P}(Q, d)$, where every box $B \in \mathcal{B}$ contains a computed solution, the aim is to successively extend $\mathcal{B}$ by further boxes which also contain a part of $\tilde{F}^{-1}(0)$. Hence, one can associate with every box $B \in \mathcal{B}$ an approximated solution $(x^B, \alpha^B) \in \mathbb{R}^M$, i.e. $\tilde{F}(x^B, \alpha^B) \approx 0$.

Given an *archive* A (where all the non-dominated solutions found so far are stored[1]) and an initial box collection $\mathcal{B}$ with an insertion depth d and a subset $\mathcal{B}_m \subset \mathcal{B}$ of boxes, which are marked for further extension, the algorithm reads as follows:

Algorithm Recover.

```
(1) for all marked boxes $B \in \mathcal{B}$:
      (a) unmark box
      (b) compute a set $\{(x_1, \alpha_1), \ldots, (x_l, \alpha_l)\}$ of distinct and well distributed
          points near to $(x^B, \alpha^B)$ and $\tilde{F}^{-1}(0)$.
      (c) for $i = 1, \ldots, l$:
            starting with $(x_i, \alpha_i)$, compute $(x_i^*, \alpha_i^*)$ with $\tilde{F}(x_i^*, \alpha_i^*) \approx 0$.
            update the Archive $A$ by $x_i^*$.
            if $x_i^* \in A$ and $B((x_i^*, \alpha_i^*), d) \notin \mathcal{B}$
               $\mathcal{B} := \mathcal{B} \cup B((x_i^*, \alpha_i^*), d)$
               mark $B((x_i^*, \alpha_i^*), d)$
    Repeat (2) while new boxes are added to $\mathcal{B}$ or until a prescribed maximal
    number of steps is reached.
```

Remarks 3.4 *(a) Due to steps (1b) and (1c) the algorithm stated above is contained in the class of* predictor-corrector methods *(PC methods). The classical realization of these two steps is to linearize the manifold $\tilde{F}^{-1}(0)$ around a given point (x^B, α^B) in order to obtain suitable predictors in step (1b) – done via a QR decomposition of $(\tilde{F}')^T$ – and to use a Gauss-Newton method for (1c). By doing this the most efficient realization is guaranteed (Deuflhard 2004), but, in turn, the second derivatives of all objectives are needed. Alternatively, there exists a version for continuous models (Schütze et al. 2003), where multi-objective evolutionary algorithms are used to search along the efficient set.*

(b) The local nature of the PC methods has to be mentioned: given a set of initial solutions S_0, the PC methods can in general only detect further solutions within the 'known' connected components, i.e. the connected components which contain a point $s \in S_0$ (but are very effective on them). Since the Pareto set can consist of a lot of connected components, the PC methods are in general not able to solve the global problem. Thus, e.g. a combination with a global strategy seems to be advantageous.

Example 3.5 *Consider the MOP consisting of the following three objective*

---

[1] Alternative ways to organize the archive can be found e.g. in Schütze et al. (2006) or Schütze et al. (2007).

functions $f_1, f_2, f_3 : \mathbb{R}^3 \to \mathbb{R}$,

$$\begin{aligned} f_1(x_1, x_2, x_3) &= (x_1 - 1)^4 + (x_2 - 1)^2 + (x_3 - 1)^2, \\ f_2(x_1, x_2, x_3) &= (x_1 + 1)^2 + (x_2 + 1)^4 + (x_3 + 1)^2, \\ f_3(x_1, x_2, x_3) &= (x_1 - 1)^2 + (x_2 + 1)^2 + (x_3 - 1)^4. \end{aligned}$$

Since all the objectives are strictly convex, the Pareto set consists of one connected component. Figure 1 shows the result of an application of the recover algorithm starting with two initial solutions.

3.3 *Particle Swarm Optimization Methods*

A particle swarm optimization (PSO) method is a population based optimization technique which can be formulated as follows: a set of N particles may be considered as a population P_t in the generation t. Each particle i has a position and a velocity defined by $\vec{x}^i = \{x_1^i, x_2^i, \cdots, x_n^i\}$ and $\vec{v}^i = \{v_1^i, v_2^i, \cdots, v_n^i\}$ in the variable space S. In generation $t+1$, these are updated as below:

$$\begin{aligned} v_{j,t+1}^i &= w v_{j,t}^i + c_1 R_1 (p_{j,t}^i - x_{j,t}^i) + c_2 R_2 (p_{j,t}^{i,g} - x_{j,t}^i) \\ x_{j,t+1}^i &= x_{j,t}^i + v_{j,t+1}^i \end{aligned} \tag{3.7}$$

where $j = 1, \cdots, n$, w is the inertia weight of the particle, c_1 and c_2 are two positive constants, and R_1 and R_2 are random values in the range $[0, 1]$.

In Equation 3.7, $\vec{p}_t^{\,i,g}$ is the position of the best global particle in the population which guides the particles to move towards the optimum. $\vec{p}_t^{\,i}$ is the best position that particle i could find so far. Indeed, it is like a memory for the particle i and is updated in each generation. In a PSO, the performance of each particle is measured according to a pre-defined fitness function, which is related to the problem to be solved. The inertia weight w is employed to control the impact of the previous history of velocities on the current velocity, thus to influence the trade-off between global and local exploration abilities of the particles (Shi and Eberhart 1998). In order to solve multi-objective optimization problems, Multi-objective Particle Swarm Optimization (MOPSO) is proposed, which works basically the same as PSO methods. The important part in MOPSO is to determine the best global particle $\vec{p}_t^{\,i,g}$ for each particle i of the population. In single-objective PSO, the best global particle is determined easily by selecting the particle which has the best position. Since in multi-objective optimization problems there is a set of Pareto-optimal solutions (all of which are equally good), each particle of the population should select one of the Pareto-optimals as its best global particle, which is called the *best local guide.* Algorithm *1* shows a typical structure of a MOPSO with elitism, where t denotes the generation index, P_t the population, and A_t the *archive* at generation t. In this method, elitism refers to the fact of saving the obtained non-dominated solutions during generations in the archive. In Algorithm *1*, the function $Evaluate$, evaluates the particles in the population P_t and the function $Update(P_t, A_t)$ compares whether members of the current population P_t are non-dominated with respect to the members of the actual archive A_t and how and which of such candidates should be considered for insertion into the archive and which should be removed. Thereby, an archive is called *domination-free* if no two points in the archive do dominate each other. Obviously, during execution of the function $Update$, dominated points must be deleted in order to keep the archive domination-free. Selecting the global best particle for each particle i is done in the $FindBestLocal(A_{t+1}, \vec{x_t}^i)$ function. In this function each particle has to change its position $\vec{x}_t^i$ towards the position of a local guide which must be selected from the updated set of approximated Pareto-optimal solutions stored in the archive A_{t+1}. How to select the local guide from the archive has a great impact on convergence and diversity of the solutions and is a problem that has been studied by several researchers (Coello Coello et al. 2004, Toscano Pulido and Coello Coello 2004, Fieldsend and Singh 2002, Mostaghim and Teich 2003, Fieldsend 2004, Mostaghim 2004, Branke and Mostaghim 2006, Reyes-Sierra and Coello Coello 2006). In Step 4, $\vec{p}^i$ of the particle i is updated, after finding the current position of particle i. $\vec{p}^i$ is like a memory for the particle i and keeps the non-dominated (best) position of the particle by comparing the new position $\vec{x}_{t+1}^i$ in the objective space with $\vec{p}_t^i$ ($\vec{p}_t^i$ is the last non-dominated (best) position of the particle i). In order to avoid the local optima, a turbulence factor is added to the positions of the particles. This is done by adding a random value to the current position of each particle: The particles change their positions during generations until a termination criterion is met. The termination criteria can be e.g., a maximum number of generations.

4 The Evolutionary Recover Algorithm

In this section the *Evolutionary Recover Algorithm* (ERA) is stated, then some convergence results are presented as well as some details for possible realizations of the algorithm.

4.1 *The Algorithm*

Here an algorithm is stated which combines the recovering technique (denoted by *Recover()*) and an evolutionary search strategy (denoted by *Generate()*)

Algorithm *1* MOPSO Algorithm

```
Input: $N$
Output: $A$
  1. Initialization: Initialize population $P_t$, $t = 0$:
  for $i = 1$ to $N$ do
    Initialize $\vec{x}_t{}^i$, $\vec{v}_t{}^i = \vec{0}$ and $\vec{p}_t{}^i = \vec{x}_t{}^i$
  end for
  Initialize the archive $A_t := \{\}$
  2. Evaluate: $Evaluate(P_t)$
  3. Update: $A_{t+1} := Update(P_t, A_t)$
  4. Move: $P_{t+1} := Move(P_t, A_t)$
  for $i = 1$ to $N$ do
    $\vec{p}_t{}^{i,g} := FindBestLocal(A_{t+1}, \vec{x}_t{}^i)$
    for $j = 1$ to $n$ do
      $v^i_{j,t+1} = wv^i_{j,t} + R_1(p^i_{j,t} - x^i_{j,t}) + R_2(p^{i,g}_{j,t} - x^i_{j,t})$
      $x^i_{j,t+1} = x^i_{j,t} + v^i_{j,t+1}$
    end for
    if $\vec{x}^i_{t+1} \prec \vec{p}_t{}^i$ then
      $\vec{p}^i_{t+1} = \vec{x}^i_{t+1}$
    else
      $\vec{p}^i_{t+1} = \vec{p}_t{}^i$
    end if
  end for
  5. Apply Turbulence Factor.
  6. Termination: Unless a termination criterion is met $t = t + 1$ and goto
  Step 2
```

in order to join their particular strengths.
Given a sequence $S_j, j \in \mathbb{N}_0$, of state spaces (see discussion below) and a sequence $d_j, j \in \mathbb{N}_0$, of insertion depths, the Evolutionary Recover Algorithm reads as described in Algorithm *1* (see also Figure 2).

Note that *Generate()* can in principle be any evolutionary multi-objective optimization (EMO) algorithm. The convergence toward the Pareto set as well as the required assumptions on the EMO algorithm will be discussed in the following.

4.2 *Convergence Results*

For convenience of the reader two required definitions are stated in the following before the results are stated.

Algorithm *1* Evolutionary Recover Algorithm

```
**Initialization:**
  $P_0 \subset S_0$ drawn at random
  $A_{-1} := \emptyset$
  $A_0 :=$ non-dominated points of $P_0$

for $j = 0, 1, 2, \dots$ do
  (a) $P_{j+1}$ := Generate ($P_j$)
  (b) $\tilde{A}_j$ :=ArchiveUpdate ($A_j, P_{j+1}$)
  (c) $\mathcal{B}_j := \left\{ B(a, d_j) \in \mathcal{P}(Q, d_j) | a \in \tilde{A}_j \right\}$
  if $d_j \neq d_{j-1}$ then
      mark all boxes $B \in \mathcal{B}_j$
  else
    mark all boxes $B(a, d_j) \in \mathcal{B}_j$ with $a \in A_j \backslash A_{j-1}$ and $B(a, d_j) \notin \mathcal{B}_{j-1}$
  end if
  (d) $A_{j+1}$ := Recover ($\mathcal{B}_j, A_j$)
end for
```

Definition 4.1 *Let $u \in \mathbb{R}^n$ and $A, B \subset \mathbb{R}^n$. The semi-distance dist$(\cdot,\cdot)$ and the* Hausdorff distance $d(\cdot,\cdot)$ *are defined as follows:*

(a) $dist(u, A) := \inf_{v \in A} \|u - v\|$

(b) $dist(B, A) := \sup_{u \in B} dist(u, A)$

(c) $d(A, B) := \max\{dist(A, B), dist(B, A)\}$

Definition 4.2 *Let $X, X_1, X_2, \dots$ be random variables on a probability space (Ω, Σ, μ). If*

$$\lim_{n \to \infty} X_n(\omega) = X(\omega)$$

for μ-almost all $\omega \in \Omega$, it is said that

$$\lim_{n \to \infty} X_n = X \quad \textit{with probability one.}$$

Now the following result can be stated.

Theorem 4.3 *Let an MOP $F : Q \subset \mathbb{R}^n \to \mathbb{R}^k$ be given, where $Q = [a_1, b_1] \times \dots \times [a_n, b_n] \subset \mathbb{R}^n, a_i, b_i \in \mathbb{R}$, $a_i \leq b_i$, and F is continuous.*

Further let

$$\forall j \in \mathbb{N} \text{ and } \forall B \in \mathcal{P}(Q,j): \qquad P\left(\exists l_B \in \mathbb{N} \,:\, P_{l_B} \cap B \cap Q \neq \emptyset\right) = 1. \quad (4.1)$$

Then an application of the Algorithm 1, where all obtained non-dominated points are kept in the archive, i.e. ArchiveUpdate$(A,P) := \{x \in A \cup P : y \not\prec x \ \forall y \in A \cup P\}$*, leads to a sequence of archives* $\{A_i\}_{i\in\mathbb{N}}$*, such that*

$$\lim_{i\to\infty} \operatorname{dist}(F(P_Q), F(A_i)) = 0 \quad \textit{with probability one},$$

where P_Q *denotes the Pareto set of* $F\big|_Q$*.*

Proof Let $x \in P_Q$. Since (4.1) holds, for every $i \in \mathbb{N}$ there exists a point $x_i \in B(x,i) \cap Q$ such that there is with probability one a $j_i \in \mathbb{N}$ with $x_i \in P_{j_i}$. Hence there exists a point $d_i \in A_{j_i}$ with $F(d_i) \leq_p F(x_i)$. By construction of the archives, for all $N > j_i$ there is a point $d_i^N \in A_N$ with

$$F(d_i^N) \leq_p F(d_i). \qquad (4.2)$$

Since $\lim_{i\to\infty} x_i = x$ and F is continuous it follows that $\lim_{i\to\infty} F(x_i) = F(x)$. Further, since $x \in P_Q$ one can deduce that

$$\lim_{i\to\infty} F(d_i) = F(x). \qquad (4.3)$$

Combining (4.2) and (4.3) it follows that

$$\lim_{i\to\infty} \operatorname{dist}(F(x), F(A_i)) = 0 \quad \text{with probability one},$$

and the proof is done.
□

The next example shows that weak Pareto points which are not properly Pareto optimal can cause problems for the convergence of the A_j's toward the Pareto set if the domain of the MOP is continuous:
Consider the bicriteria optimization problem which is illustrated in Figure 3. Once the weak Pareto point x_1 is added to the archive, this point will only be discarded when x_2 is taken into account, since x_2 is the only point which dominates x_1.
Thus, one can only obtain convergence toward the Pareto set in the probabilistic sense if the MOP contains no weak Pareto point outside the Pareto set.

Theorem 4.4 *In addition to the assumptions of Theorem 4.3 assume that there is no weak Pareto point in* $Q \backslash P_Q$.
Then the algorithm described above generates a sequence of archives $\{A_i\}_{i \in \mathbb{N}}$*, such that*

$$\lim_{i \to \infty} d(F(P_Q), F(A_i)) = 0 \quad \textit{with probability one,}$$

where $d(\cdot, \cdot)$ *denotes the Hausdorff distance.*

Proof Using Theorem 4.3 it remains to show that

$$\lim_{i \to \infty} \operatorname{dist}(F(A_i), F(P_Q)) = 0 \quad \text{with probability one.} \tag{4.4}$$

To see this let $x \in Q \backslash P_Q$. Since x is no weak Pareto point there exists a point $p \in P_Q$ such that $F(p) <_p F(x)$. Since F is continuous and P_Q is a closed set there exists a neighborhood $\mathcal{U}(p)$ of p with

$$F(y) <_p F(x) \quad \forall y \in \mathcal{U}(p).$$

Further there exists a $j_p \in \mathbb{N}$ with $B(p, j_p) \subset \mathcal{U}(p)$. Since (4.1) holds, there exists with probability one a point $d \in B(p, j_p)$ and an index $j \in \mathbb{N}$ with $d \in P_j$. By this it follows that

$$x \notin A_N \quad \forall N \geq j \quad \text{with probability one,}$$

and the proof is complete. □

4.3 *Realization and Discussion*

In the following some comments on possible realizations of the ERA are made, in particular on the interaction of the algorithms, followed by some discussion on the computational cost.

Granularity. In order to obtain convergence one certainly needs a (monotonically increasing) sequence $d_j, j \in \mathbb{N}_0$, with $d_j \to \infty$ for $j \to \infty$ of insertion depths as well as a suitable sequence of state spaces $S_j, j \in \mathbb{N}_0$, in order to fulfill (4.1). In case an EMO algorithm is used for the process *Generate()*, this property should easily be provided if a suitable mutation strategy is applied and if the family of (finite) state spaces is e.g. characterized by

(i) $S_j \subset Q$,
(ii) $S_j \supset S_{j-1}$, if $j \geq 1$, and

(iii) $\forall B \in \mathcal{P}(Q, j) : \ \exists\, x \in S_j \cap B$.

However, in practice it turned out that satisfactory results could also be obtained by using a fixed granuarity, i.e. with a constant insertion depth d and a constant domain S. In case a MOPSO algorithm or any real-coded EMO algorithm is used one can simply assume that $S = Q$.

Switching from Generate() to Recover(). One problem when combining the two different methods is that their underlying domains are not equal. To be more precise, having computed a promising point $x_0 \in \mathbb{R}^n$ by *Generate()* the question that arises is how to continue the search via *Recover()* starting from a 'matching' point $(\hat{x}, \hat{\alpha}) \in \mathbb{R}^{n+k}$. One way to obtain such a point is to perform the following two steps: *(a)* to find a suitable point $\alpha_0 \in \mathbb{R}^k$ and *(b)* to compute a desired point $(\hat{x}, \hat{\alpha}) \in \tilde{F}^{-1}(0)$ starting with (x_0, α_0).
It was observed by the authors of the current work that it is not advisable for the realization of *(a)* to take an arbitrary vector α and to project (x, α) to $\tilde{F}^{-1}(0)$ (e.g., via a Newton's method) since it can happen that (x, α) is projected to the 'wrong' connected component of the solution set (due to the wrong choice of α). Alternatively, much better results have been obtained when α_0 was chosen as a solution of the following (low dimensional) scalar optimization problem, which is motivated by Theorem 3.1:

$$\min_{\alpha \in \mathbb{R}^k} \|F(x, \alpha)\| \tag{4.5}$$

subject to

$$\begin{aligned} \sum_{i=1}^{k} \alpha_i - 1 = 0 \\ \alpha \in [0, 1]. \end{aligned} \tag{4.6}$$

When to stop Generate(). This is a classical problem when running an EMO algorithm (and others), in particular when this algorithm has to be executed several times within another procedure. Of course, one can stop the algorithm when an assigned running time has elapsed. Alternatively, one can use the following stopping criteria: *Generate()* stops when a prescribed number of 'promising' points have been found, i.e. points $p \in \mathbb{R}^n$ with

(i) p is not dominated by any point in the current archive A_j, and
(ii) $B(p, d_j) \notin \mathcal{B}_{j-1}$,

or – following the above discussion – when a maximal number of function calls is reached.

Distribution of Solutions. It is known that one problem of the application of many EMO algorithms (i.e., when elitism is adopted) is to achieve a satisfying distribution of the entries of the archive (see (Rudolph and Agapie 2000) or (Laumanns et al. 2001)). The authors of this paper have observed that this problem was significantly reduced in many test cases by using the boxes as the data structure for the representation of the parts of the solution set which are already 'detected'. By this, the solutions can, for instance, not fall into clusters. See also Example 5.1 in the next section.

Computational Effort. In the following, two rules of thumb for the computational effort of the recovering algorithm are presented. The effort of this algorithm is of course determined by the total number of function evaluations, which in turn mostly depends on the total number of boxes which are generated in the course of the computation.
First, the question will be addressed of how many boxes are added to the collection in the first steps when starting with *one* solution $(x_0, \alpha_0) \in \tilde{F}^{-1}(0)$. If k objectives are under consideration one expects that $\tilde{F}^{-1}(0)$ is – at least locally and under some mild smoothness assumptions – a $(k-1)$-dimensional manifold in n-dimensional space. Since in that case $\tilde{F}^{-1}(0)$ is locally diffeomorph to a $(k-1)$-dimensional cube one can assume that the initial collection $\mathcal{B}_0 := \{B((x_0, \alpha_0), d)\}$ has $3^{k-1} - 1$ neighbor boxes which contain a part of $\tilde{F}^{-1}(0)$ (if d is sufficiently large and thus the boxes are sufficiently small). By the same argument one has to expect to extend $\mathcal{B}_0$ by 5^{k-1} boxes around $\mathcal{B}_0$ after two recovering steps, whereby $5^3 - 3^3$ boxes were added in the last step and will be marked for step three. Thus, starting with one single box which contains a part of $\tilde{F}^{-1}(0)$, one can assume to obtain the following number of boxes in the first (few) iteration steps:

$$\begin{aligned} |\mathcal{B}_0| &= 1 \\ |\mathcal{B}_l| &= (2l+1)^{k-1} \\ &= (2l-1)^{k-1} + \underbrace{(2l+1)^{k-1} - (2l-1)^{k-1}}_{\text{marked boxes}}, \quad l = 1, 2, \ldots \end{aligned} \tag{4.7}$$

Next, the question arises of how many boxes – and thus also how many points in the archive – have to be stored in order to cover the entire Pareto set with a fixed box size. Assume a Box B is given which contains a part of $\tilde{F}^{-1}(0)$. If B is sufficiently small one can expect that after n subdivisions[1] of B approximately

---

[1] Realized via bisection according to one coordinate.

2^{k-1} of the 2^n subboxes of B intersect $\tilde{F}^{-1}(0)$. Denote by $b(Q,d)$ the number of boxes which contain a part of $\tilde{F}^{-1}(0)$ within Q in subdivision depth d. Using the above estimate one obtains

$$b(B,n) \approx 2^{k-1} b(B,0)$$

Hence for one subdivision step (i.e., $d \rightarrow d+1$) one can assume an expansion factor χ defined as

$$\chi \approx \sqrt[n]{2^{k-1}}$$

Using these assumptions and replacing B by Q one gets the following estimate on the number of boxes which are required to cover $\tilde{F}^{-1}(0)$:

$$\begin{aligned} &b(Q,0) = 1 \\ &b(Q,d) \approx \sqrt[n]{2^{k-1}}\, b(Q,d-1) = (\sqrt[n]{2^{k-1}})^d, \quad d = 1,2,\ldots \end{aligned} \tag{4.8}$$

This estimate, though very rough, has matched quite well in most computations with the actual number of boxes which were needed to cover the Pareto set. The approximation of $b(Q,d)$ can, for instance, be used for an *a priori* adjustment of the insertion depth d.

5 Numerical Results

In this section the efficiency of ERA is illustrated on three different test functions. For the 'EMO part' (i.e., *Generate()*) the MOPSO presented in (Coello Coello et al. 2004) has been chosen.

EXAMPLE 5.1 *First, the following MOP is being considered for demonstrational purposes:*

$$\begin{aligned} f_1(x_1,x_2) &= (x_1-1)^2 + (x_2-1)^4, \\ f_2(x_1,x_2) &= (x_1+1)^2 + (x_2+1)^2. \end{aligned} \tag{5.1}$$

Figure 4 shows both the box collections as well as the entries of the archives (in image space) for some iteration steps for this (trivial) example. Here, the number of entries in the archive which are contained in a box B were restricted, and thus the solutions in the archive A_i do not separate into clusters. It can be observed that in this case both convergence and diversity of the solutions is obtained.

Example 5.2 *Next, the MOP known as* ZDT3 *(Zitzler et al. 2000) is considered:*

$$\begin{aligned} & f_1, f_2 : \mathbb{R}^n \to \mathbb{R} \\ & f_1(x) = x_1 \\ & f_2(x) = g(x_2, \ldots, x_n) \cdot h(f_1(x), g(x)) + 1, \end{aligned} \tag{5.2}$$

where

$$\begin{aligned} g(x_2, \ldots, x_n) &= 1 + 9(\sum_{i=2}^{n} x_i)/(n-1), \quad \textit{and} \\ h(f_1, g) &= 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1). \end{aligned} \tag{5.3}$$

For the domain $n = 30$ and $Q = [1,2]^{30}$ have been chosen. This test function contains a disconnected Pareto-optimal front with a high number of local optima, and has been frequently adopted to validate multi-objective evolutionary algorithms.
Figure 5 shows a numerical result of the recovering algorithm, where a final population of a MOPSO run was taken as the initial solution set.

Example 5.3 *Finally, the following MOP is under consideration:*

$$\begin{aligned} & f_1, f_2, f_3 : \mathbb{R}^n \to \mathbb{R} \\ & \quad f_1(x) = \sum_{j=2}^{n} (x_j - a_j^i)^2 + 10\sin(0.05\|x - a^i\|) \sum_{j=1}^{n} (a_j^i)^2, \\ & \quad f_i(x) = \sum_{\substack{j=1 \\ j \neq i}}^{n} (x_j - a_j^i)^2 + (x_i - a_i^i)^4, \quad i = 2, 3. \end{aligned} \tag{5.4}$$

where

$$\begin{array}{lcll} a^1 & = & (1,1,1,1,\ldots) & \in \mathbb{R}^n \\ a^2 & = & (-1,-1,-1,-1,\ldots) & \in \mathbb{R}^n \\ a^3 & = & (1,-1,1,-1,\ldots) & \in \mathbb{R}^n \end{array}$$

This is a three objective test function. The Pareto-optimal front contains a very large surface. Finding a large number of Pareto-optimal solutions will be a complicated task for a MOPSO algorithm as it requires a high computational time to store and update the non-dominated set.
Figure 6 shows a computational result for $n = 10$ and $Q = [-3, 5]^{10}$.

6 Performance Assessment

Figures 4, 5 and 6 show the results of the MOPSO and ERA methods. It can be observed that the ERA method is able to cover the non-dominated front obtained by MOPSO easily. The MOPSO method, on the other hand, can obtain a relatively small set of non-dominated solutions with a good distribution.
It has to be expected from ERA to cover the non-dominated front and at the same time to improve the convergence of the solutions locally. In order to test this, the results of different methods have been evaluated by two measurements.
The first measure is the C metric (Zitzler and Thiele 1999), which compares the convergence rate of two non-dominated sets A and B:

$$C(A, B) = \frac{|\{\vec{b} \in B | \exists \vec{a} \in A : \vec{a} \preceq \vec{b}\}|}{|B|} \tag{6.5}$$

where $\vec{a} \preceq \vec{b}$ denotes $\vec{a}$ weakly dominates $\vec{b}$. The value of $C(A, B) = 1$ means that all the members of B are weakly dominated by the members of A. One can also conclude that $C(A, B) = 0$ means that none of the members of B is weakly dominated by the members of A. $C(A, B)$ is not equal to $1 - C(B, A)$, and both $C(A, B)$ and $C(B, A)$ must be considered for comparisons.
The S metric (Zitzler and Thiele 1999) measures the hyper-volume of a region made by a non-dominated set A and a *reference point* in the objective space. If the non-dominated set A has a better diversity and convergence than a non-dominated set B, then the hyper-volume computed for the set A is bigger than the hyper-volume of the set B. Here, the hyper-volume is not computed in the way is done in the original S metric (Zitzler and Thiele 1999). Instead, it is suggested to approximate the hyper-volume by building a grid in the objective space as follows: (a) Define a reference point and then build a grid between the origin and the reference point in the objective space. (b) Count the grid points which are dominated by at least one of the non-dominated solutions.
A non-dominated set with a good diversity and convergence dominates more solutions than another set with worse diversity and convergence. Actually, this method is very similar to the hyper-volume method with the difference that here the volume is approximated and therefore the computational time is less

than the original method. This method is also very easy to implement.

The grid resolution and the reference point will have a great impact on this measurement. In the experiments, a 200×200 grid for the two-objective and a $100 \times 100 \times 100$ grid for the three-objective test functions have been selected. For simplicity, the non-dominated solutions and the grid are normalized to unity.

Table 1 shows the C metric and S metric values. The values of N indicate that ERA is able to find a large set of solutions. Comparing the C metric values computed for Example 5.3, the solutions of ERA dominate 79 percent of the MOPSO solutions whereas MOPSO solutions do not dominate any of the ERA solutions. This means that the ERA method not only finds a large set of solutions, but also locally improves them. Considering the C metric values for Example 5.2, due to differences in terms of the diversities, the solutions have comparable convergence. It can be observed from the S metric values that MOPSO finds solutions with a very good diversity comparable to ERA. Considering both the S and C metric values, MOPSO finds a good diversity, but most of these solutions are getting dominated by ERA solutions. This can be observed in the solutions of the test problem with a larger number of objectives (i.e., Example 5.3).

7 Conclusion and Future Work

In this paper, a novel hybrid algorithm for the computation of the Pareto set of a given multi-objective optimization problem has been proposed. This algorithm combines a continuation method with a multi-objective particle swarm optimizer in order to join their particular advantages. Then, some convergence results have been presented as well as some numerical results to demonstrate the strength of the proposed algorithm.

As part of possible future work, there are several possible ways to extend the techniques presented in this paper. A particularly interesting extension would be to consider hybrid discrete/continuous models since in that case the interplay between local and global search strategies is of particular interest. Another interesting task would be to augment the algorithm described above for the treatment of constrained models.

Acknowledgements. The second author acknowledges support from CONACyT through project no. 45683-Y.

The authors would like to thank Alexander Krüger and Maik Ringkamp for help and fruitful discussions on the contents of this paper.

REFERENCES

Bosman, P. A. and de Jong, E. D. (2005), Exploiting gradient information in numerical multi-objective evolutionary optimization, in *2005 Genetic and Evolutionary Computation Conference (GECCO'2005)*, H.-G. Beyer et al., (Ed.), Vol. 1, ACM Press, New York, USA, pp. 755–762.

Bosman, P. A. and de Jong, E. D. (2006), Combining gradient techniques for numerical multi-objective evolutionary optimization, in *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, M. Keijzer et al., (Ed.), Vol. 1, ACM Press. ISBN 1-59593-186-4, Seattle, Washington, USA, pp. 627–634.

Branke, J. and Mostaghim, S. (2006), About selecting the personal best in multi-objective particle swarm optimization, in *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley and X. Yao, (Eds.), Springer. Lecture Notes in Computer Science Vol. 4193, Reykjavik, Iceland, pp. 523–532.

Brown, M. and Smith, R. E. (2003), Effective use of directional information in multi-objective evolutionary computation, in *Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I*, E. Cantú-Paz. et al., (Ed.), Springer. Lecture Notes in Computer Science Vol. 2723, pp. 778–789.

Coello, C. A. C., Veldhuizen, D. A. V. and Lamont, G. B. (2002), *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers.

Coello Coello, C. A., Toscano Pulido, G. and Salazar Lechuga, M. (2004), Handling multiple objectives with particle swarm optimization, *IEEE Transactions on Evolutionary Computation* **8**(3), 256–279.

Deb, K. (2001), *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley.

Dellnitz, M. and Hohmann, A. (1997), A subdivision algorithm for the computation of unstable manifolds and global attractors, *Numerische Mathematik* **75**, 293–317.

Dellnitz, M., Schütze, O. and Hestermeyer, T. (2005), Covering Pareto sets by multilevel subdivision techniques, *Journal of Optimization Theory and Applications* **124**, 113–155.

Deuflhard, P. (2004), *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms*, Springer.

Ehrgott, M. (2005), *Multicriteria Optimization*, second edn, Springer, Berlin. ISBN 3-540-21398-8.

Ehrgott, M. and Gandibleux, X. (2004), Approximative solution methods for multiobjective combinatorial optimization, *Top* **12**(1), 1–89.

Fieldsend, J. (2004), Multi-objective particle swarm optimisation methods, Technical Report 419, Department of Computer Science, University of Exeter, Exeter, UK.

Fieldsend, J. E. and Singh, S. (2002), A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence, in *Proceedings of the 2002 U.K. Workshop on Computational Intelligence* .

Fliege, J. and Fux Svaiter, B. (2000), Steepest descent methods for multicriteria optimization, *Mathematical Methods of Operations Research* **51**(3), 479–494.

Gandibleux, X., Morita, H. and Katoh, N. (2001), The supported solutions used as a genetic information in a population heuristic, in *First International Conference on Evolutionary Multi-Criterion Optimization*, E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello and D. Corne, (Eds.), Springer-Verlag. Lecture Notes in Computer Science No. 1993, pp. 429–442.

Harada, K., Sakuma, J., Kobayashi, S. and Ono, I. (2007), uniform sampling of local Pareto-optimal solution curves by Pareto path following and its applications in multi-objective GA, in *Genetic and Evolutionary Computation Conference (GECCO-2007)*, pp. 813–820, ACM Press, 2007.

Hillermeier, C. (2001), *Nonlinear Multiobjective Optimization - A Generalized Homotopy Approach*, Birkhäuser.

Karush, W. E. (1939), Minima of functions of several variables with inequalities as side conditions, PhD thesis, University of Chicago.

Knowles, J. and Corne, D. (2000), M-PAES: A memetic Algorithm for multiobjective optimization, in *2000 Congress on Evolutionary Computation*, Vol. 1, IEEE Service Center, Piscataway, New Jersey, pp. 325–332.

Knowles, J. and Corne, D. (2005), Memetic algorithms for multiobjective optimization: issues, methods and prospects, in *Recent Advances in Memetic Algorithms*, W. E. Hart, N. Krasnogor and J. E. Smith, (Eds.), Springer. Studies in Fuzziness and Soft Computing, Vol. 166, pp. 313–352.

Kuhn, H. and Tucker, A. (1951), Nonlinear programming, in *Proceeding of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, J. Neumann, (Ed.), pp. 481–492.

Lahanas, M., Baltas, D. and Giannouli, S. (2003), Global convergence analysis of fast multiobjective gradient based dose optimization algorithms for high-dose-rate brachytherapy, *Physics in Medicine and Biology* **48**(5), 599–617.

Laumanns, M., Thiele, L., Deb, K. and Zitzler, E. (2001), On the convergence and diversity-preservation properties of multi-objective evolutionary algorithms. TIK-Report No. 108, ETH Zürich.

Laumanns, M., Thiele, L., Deb, K. and Zitzler, E. (2002), Combining convergence and diversity in evolutionary multi-objective optimization, *Evolutionary Computation* **10**(3), 263–282.

Miettinen, K. (1999), *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers.

Mostaghim, S. (2004), Multi-Objective Evolutionary Algorithms, Data Structures, Convergence and Diversity, PhD thesis, University of Paderborn.

Mostaghim, S. and Teich, J. (2003), Strategies for finding good local guides in multi-objective particle swarm optimization, in *IEEE 2003 Swarm Intelligence Symposium*, IEEE Press.

Reyes-Sierra, M. and Coello Coello, C. A. (2006), Multi-objective particle swarm optimizers: a survey of the state-of-the-art, *International Journal of Computational Intelligence Research* **2**(3), 287–308.

Rudolph, G. (1998), Finite markov chain results in evolutionary computation: A tour d'horizon, *Fundamenta Informaticae* **35**, 67–89.

Rudolph, G. and Agapie, A. (2000), On a multi-objective evolutionary algorithm and its convergence to the Pareto set, in *2000 Congress on Evolutionary Computation (CEC'2000)*, pp. 1010–1016.

Schäffler, S., Schultz, R. and Weinzierl, K. (2002), A stochastic method for the solution of unconstrained vector optimization problems, *Journal of Optimization Theory and Applications* **114**(1), 209–222.

Schütze, O. (2004), *Set Oriented Methods for Global Optimization*, PhD thesis, University of Paderborn. <http://ubdata.uni-paderborn.de/ediss/17/2004/schuetze/>.

Schütze, O., Laumanns, M., Coello, C. A. C., Dellnitz, M. and Talbi, E.-G. (2006), Convergence of stochastic search algorithms to finite size Pareto set approximations, Research Report 6063, INRIA. <https://hal.inria.fr/inria-00119255>.

Schütze, O., Laumanns, M., Tantar, E., Coello, C. A. C. and Talbi, E.-G. (2007), Convergence of stochastic search algorithms to gap-free Pareto front approximations, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007)*, Springer, pp. 892–899.

Schütze, O., Mostaghim, S., Dellnitz, M. and Teich, J. (2003), Covering Pareto sets by multilevel evolutionary subdivision techniques, in *Evolutionary Multi-Criterion Optimization*, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb and L. Thiele, (Eds.), Lecture Notes in Computer Science.

Shi, Y. and Eberhart, R. C. (1998), Parameter selection in particle swarm optimization, in *Evolutionary Programming VII (EP'98)*, Springer-Verlag, New York, pp. 591–600.

Tan, K., Khor, E. and Lee, T. (2005), *Multiobjective evolutionary algorithms and applications*, Springer-Verlag, London. ISBN 1-85233-836-9.

Toscano Pulido, G. and Coello Coello, C. A. (2004), Using clustering techniques to improve the performance of a particle swarm optimizer, in *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, K. Deb. et al., (Ed.), Springer-Verlag, Lecture Notes in Computer Science Vol. 3102, Seattle, Washington, USA, pp. 225–237.

V. Pareto (1964 (first edition in 1896)), *cours d'economie Politique*, Libraire Droz, Genève.

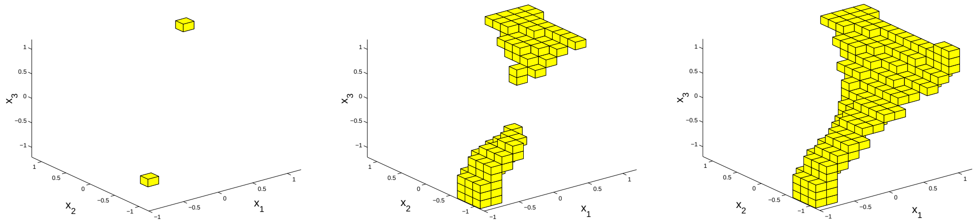
Zhou, A., Zhang, Q., Jin, Y., Sendhoff, B. and Tseng, E. (2006), Modelling the population distribution in multi-objective optimization by generative topographic mapping, in *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley and X. Yao, (Eds.), Springer. Lecture Notes in Computer Science Vol. 4193, Reykjavik, Iceland, pp. 443–452.

Zitzler, E., Deb, K. and Thiele, L. (2000), Comparison of multiobjective evolutionary algorithms: empirical Results, *Evolutionary Computation* **8**(2), 173–195.

Zitzler, E. and Thiele, L. (1999), Multiobjective evolutionary algorithms: A comparative case study and and the strength Pareto approach, *IEEE Transactions on Evolutionary Computation* **3(4)**, 257–271.

Table 1. Evaluation results of Examples 5.2 and 5.3. N, C, S and T indicate the values for the number of obtained non-dominated solutions, C metric, S metric and and computational time (in seconds), respectively.

Test	N_{MOPSO}	N_{ERA}	$C_{(MOPSO,ERA)}$	$C_{(ERA,MOPSO)}$	S_{MOPSO}	S_{ERA}	T_{MOPSO}	T_{ERA}
5.2	50	529	0.39	0.42	5862	6150	12	18
5.3	100	739	0	0.79	968962	958883	25	4



(a) initial box collection (b) 6 recover steps (c) 12 recover steps

Figure 1. Computation of the MOP described in Example 3.5. The figures show the initial box collection and two extensions. The algorithm stops after 12 iteration steps with a perfect covering of the Pareto set.

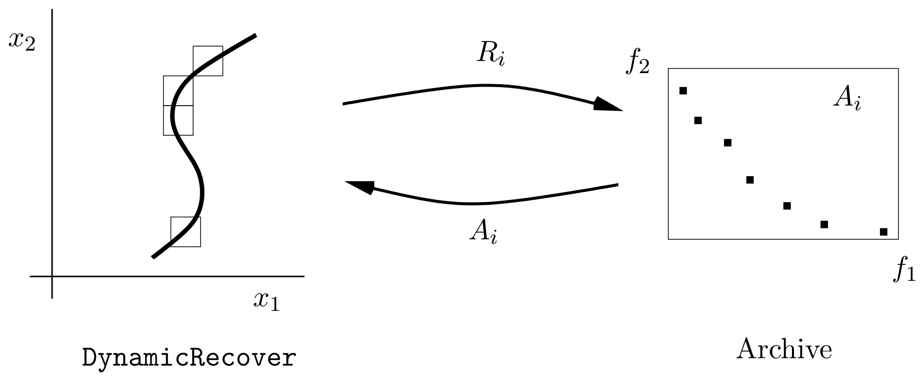


Figure 2. Basic scheme of the Evolutionary Recover Algorithm.

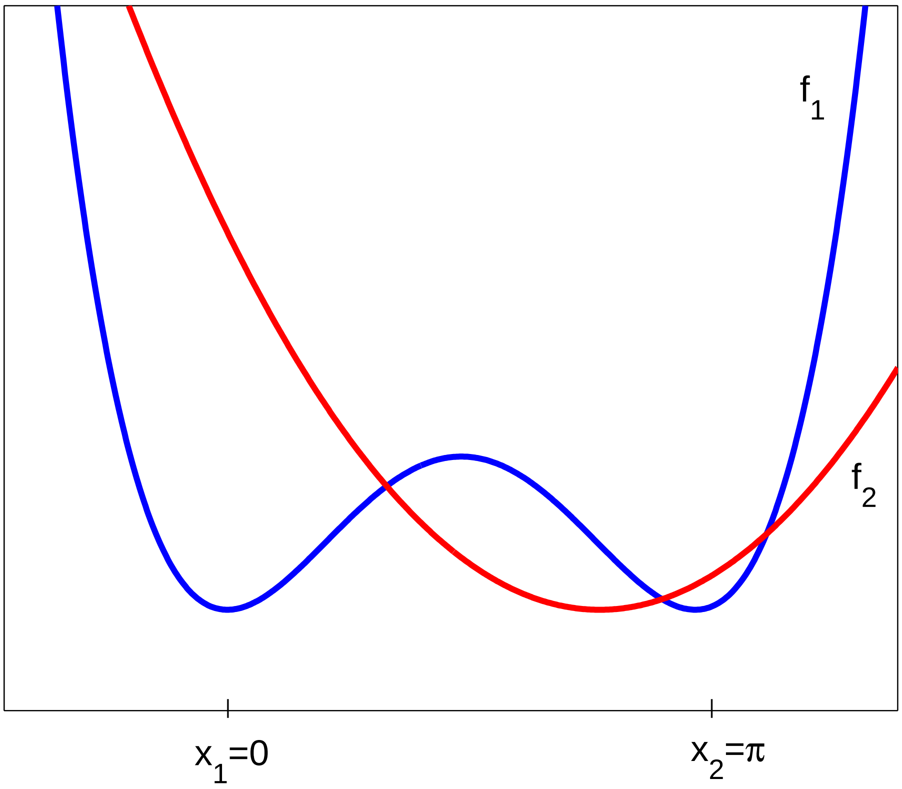


Figure 3. The weak Pareto point x_1 is only dominated by x_2.

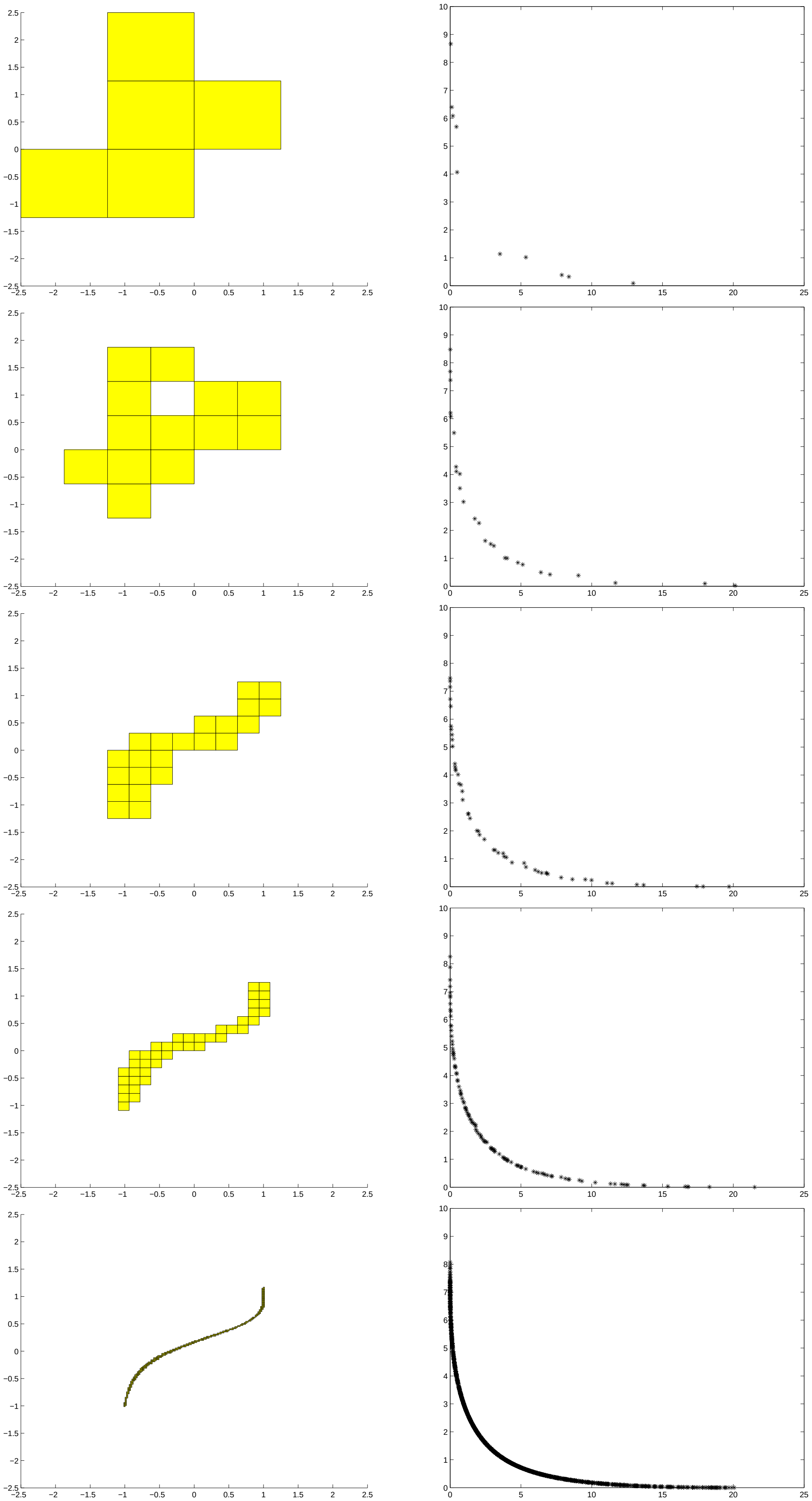


Figure 4. Application of ERA: the figures show box coverings $\mathcal{B}_i$ (left, x_1 is plotted versus x_2) and archives A_i (right, f_1 vs. f_2) of the MOP of the Example 5.1 for $i = 6, 8, 10, 12, 18$.

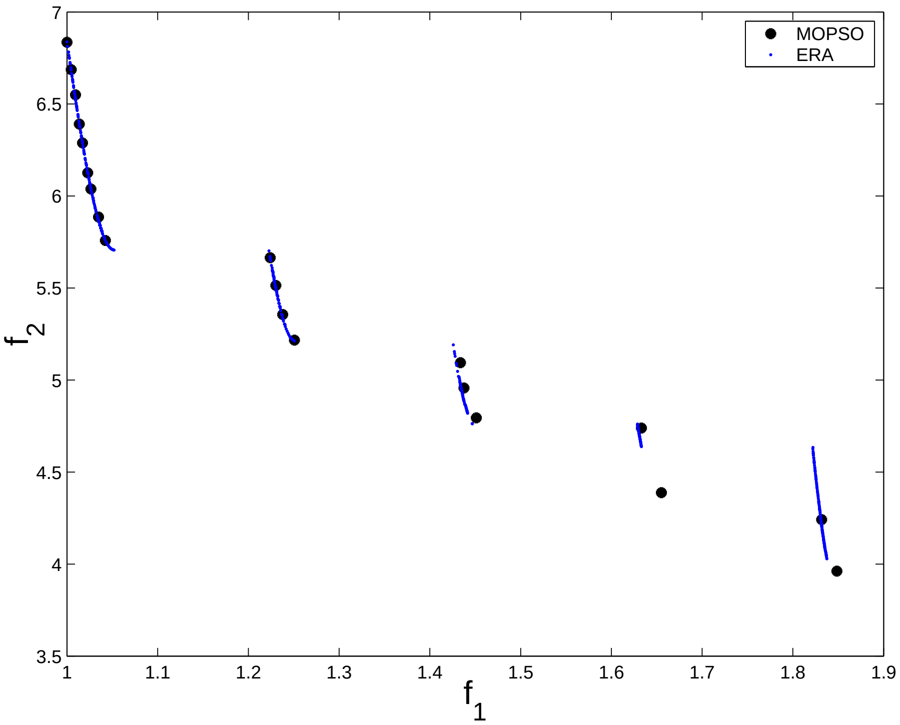


Figure 5. Numerical result for the MOP of Example 5.2.

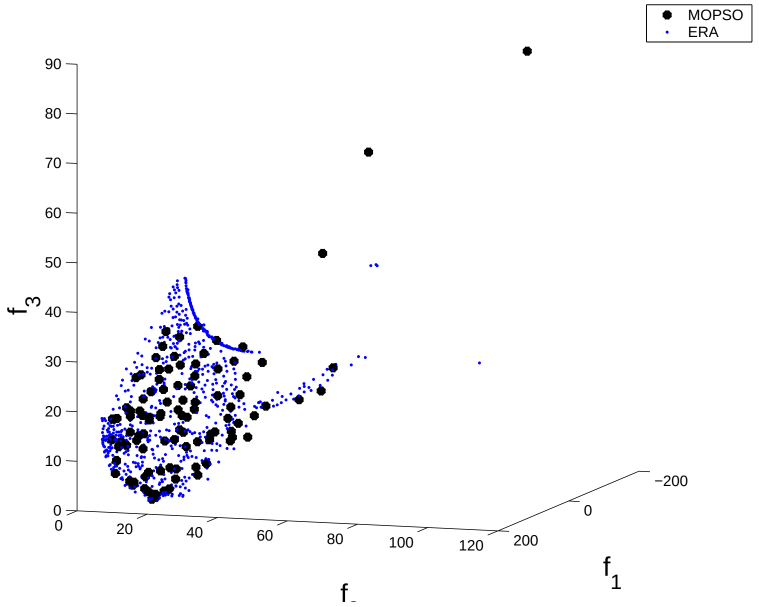
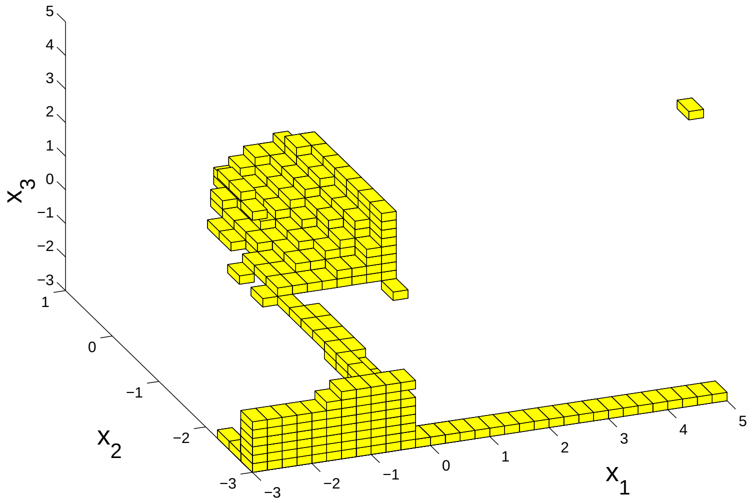


Figure 6. Numerical result for the MOP of Example 5.3 in the image space (above) and the resulting box collection of the ERA (below).


5
4
3
2
1
0
−1
−2
−3
x_3
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
4
5
x_1