Covering Pareto Sets by Multilevel Evolutionary Subdivision Techniques

Oliver Schütze[1], Sanaz Mostaghim[2], Michael Dellnitz[1], and Jürgen Teich[2]

[1] Department of Computer Science, Electrical Engineering and Mathematics
http://math-www.uni-paderborn.de/~agdellnitz
[2] Department of Computer Science, Electrical Engineering and Mathematics
http://www-date.uni-paderborn.de
University of Paderborn
Paderborn, Germany

Abstract. We present new hierarchical set oriented methods for the numerical solution of multi-objective optimization problems. These methods are based on a generation of collections of subdomains (boxes) in parameter space which cover the entire set of Pareto points. In the course of the subdivision procedure these coverings get tighter until a desired granularity of the covering is reached. For the evaluation of these boxes we make use of evolutionary algorithms. We propose two particular strategies and discuss combinations of those which lead to a better algorithmic performance. Finally we illustrate the efficiency of our methods by several examples.

1 Introduction

In the optimization of technical devices or economical processes one frequently has the goal to minimize simultaneously several conflicting objectives. Such objectives can be, for instance, cost and energy. Thus, in applications one is typically confronted with *multi-objective optimization problems* (MOPs) and one has to find the set of optimal trade-offs, the so-called *Pareto set*.
In this paper we propose several new methods for the numerical computation of Pareto sets of MOPs. Similar to [**?**,**?**] we use multilevel subdivision techniques for the solution of these problems. However, in contrast to this previous work we now combine the hierarchical search with multi-objective evolutionary algorithms (MOEAs) which are used for the evaluation of subdomains of parameter space. In this way the robustness of our algorithms is significantly increased, and this is of particular interest for higher dimensional problems. Simultaneously the number of function calls can be reduced.
In a second step we discuss combinations of these algorithms in order to improve the total performance. We also describe in which way these methods can be coupled with "standard" MOEAs for the solution of MOPs. In the final section we illustrate the efficiency of these algorithms by a couple of examples.
An outline of the paper is as follows: in Section 2 we summarize the necessary background for the algorithms which are described in Section 3. The computational results are presented in Section 4. We conclude the paper with a summary in Section 5.

2 Background

In this section we briefly summarize the background for the algorithms which are described in Section 3.

2.1 Multi-objective Optimization

In an MOP several objective functions are to be minimized or maximized at the same time. In the following, we state the MOP in its general form:

minimize $(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x}))$
subject to $\boldsymbol{x} \in Q$.

Here $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2, \ldots, m$, are conflicting objective functions that we want to minimize simultaneously. The *decision vectors* $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$ belong to the feasible region $Q \subset \mathbb{R}^n$. We assume Q to be compact and that its boundary can be defined by constraint functions.
We denote the image of the feasible region by $Z \subset \mathbb{R}^m$ and call it the *feasible objective region.* That is,

$$Z = \{F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})) : \boldsymbol{x} \in Q\}.$$

The elements of Z are called *objective vectors.*
A decision vector $\boldsymbol{x}_1 \in Q$ is said to *dominate* a decision vector $\boldsymbol{x}_2 \in Q$ if the following two conditions are satisfied:

(i) $f_i(\boldsymbol{x}_1) \leq f_i(\boldsymbol{x}_2)$ for all $i = 1, \cdots, m$, and
(ii) $f_j(\boldsymbol{x}_1) < f_j(\boldsymbol{x}_2)$ for at least one $j = 1, \cdots, m$.

A decision vector $\boldsymbol{x}_1 \in Q$ is called *Pareto-optimal* (relative to Q) if there is no $\boldsymbol{x}_2 \in Q$ dominating $\boldsymbol{x}_1$. Finally, an objective vector is also called *Pareto-optimal* if a corresponding decision vector is Pareto-optimal.

2.2 The Subdivision Algorithm

We now briefly introduce set oriented hierarchical subdivision techniques. For the details we refer to [**?**],[**?**] and [**?**]. The common aim of these algorithms is the computation of invariant sets of dynamical systems of the form

$$\boldsymbol{x}_{j+1} = g(\boldsymbol{x}_j), \qquad j = 0, 1, ... \tag{2.1}$$

where $g : \mathbb{R}^n \to \mathbb{R}^n$ is continuous. (Recall that a set $A \subset \mathbb{R}^n$ is *invariant* if $g(A) = A$.) The computation starts with a (big) compact set (box[3]) in parameter space. By a repeated bisection and selection of boxes the box coverings $\mathcal{B}_k$ of the invariant sets get tighter until the desired granularity of this outer approximation is reached. In computer implementations the selection process is based on an application of the underlying dynamical system (**??**) on a finite set of test

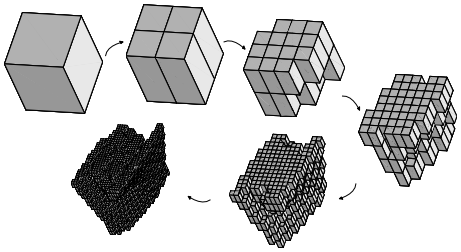


Fig. 1. Schematic illustration of the subdivision algorithm for the approximation of invariant sets.

points within each box.
The general structure of the subdivision algorithm is as follows:

Algorithm `DS-Subdivision`

1.) **Subdivision** Construct from $\mathcal{B}_{k-1}$ a new system $\hat{\mathcal{B}}_k$ of subsets such that

$$\bigcup_{B \in \hat{\mathcal{B}}_k} B = \bigcup_{B \in \mathcal{B}_{k-1}} B$$

and

$$\operatorname{diam}^4(\hat{\mathcal{B}}_k) = \theta_k \operatorname{diam}(\mathcal{B}_{k-1}),$$

where $0 < \theta_{min} \leq \theta_k \leq \theta_{max} < 1$.

2.) **Selection**

Define the new collection $\mathcal{B}_k$ by

$$\mathcal{B}_k = \left\{ B \in \hat{\mathcal{B}}_k : \text{there exists } \hat{B} \in \hat{\mathcal{B}}_k \text{ such that } g^{-1}(B) \cap \hat{B} \neq \emptyset \right\}.$$

In [**?**] this algorithm was modified and adapted to the context of global zero finding, and in [**?**] these methods were applied to the context of multi-objective optimization. The common idea in these papers is to interpret iteration schemes as dynamical systems. For instance, using the descent direction for MOPs with differentiable objectives as proposed in [**?**] one may formulate a dynamical system which has as an invariant set all the points where the Kuhn-Tucker condition holds (see [**?**]). In other words, the set of points which are locally nondominated – including points on the boundary of Q – becomes an invariant set of this specific dynamical system. Thus, it remains to compare images of boxes in Z in order to find the (global) Pareto set. The so-called `sampling algorithm` reads as follows:

---

[3] A n-dimensional box can be represented by a center $c \in \mathbb{R}^n$ and a radius $r \in \mathbb{R}^n$. Thus $B = B_{c,r} = \{x \in \mathbb{R}^n : c_i - r_i \leq x_i \leq c_i + r_i \; \forall i = 1, .., n\}$.

[4] $\operatorname{diam}(B_{c,r}) = 2\|r\|_2$ and $\operatorname{diam}(\mathcal{B}) = \max_{B \in \mathcal{B}} \operatorname{diam}(B)$.

`Sampling Algorithm`
1.) **Subdivision**
as in algorithm `DS-Subdivision`.
2.) **Selection**
for all $B \in \hat{\mathcal{B}}_k$
choose a set of test points P_B
$N :=$ nondominated points of $\bigcup_{B \in \hat{\mathcal{B}}_k} P_B$
$\mathcal{B}_k := \left\{ B \in \hat{\mathcal{B}}_k : \exists p \in P_B \cap N \right\}$

By using this algorithm – in some realizations in combination with `DS-Subdivision` – it is possible to detect the entire set of (global) Pareto points, including points on the boundary of the feasible region Q. Observe that these subdivision techniques possess naturally a "smoothing" property which is the basis for the combination with MOEAs: a box is kept if it contains *at least* one "good" point. This smoothing property is illustrated in Example A below.
`DS-Subdivision` works particularly well in the case when both the number of objective functions and the number of parameters is not too big. Otherwise the number of boxes created in the subdivision procedure is getting too large. In this article we propose the combination of `DS-Subdivision` with MOEAs in order to overcome this problem (see Section 3).

EXAMPLE A *We consider an MOP* $F = (f_1, f_2) : \mathbb{R}^3 \to \mathbb{R}^2$ *taken from [?].*

$$f_1(\boldsymbol{x}) = \sum_{j=1}^{3} x_j, \quad f_2(\boldsymbol{x}) = 1 - \prod_{j=1}^{3}(1 - p_j(x_j))$$

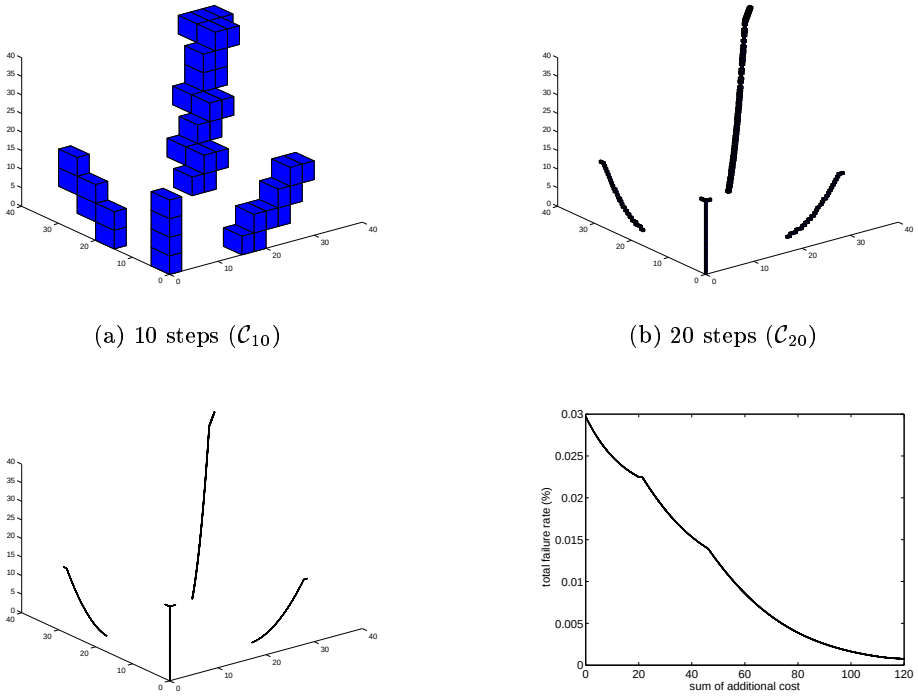
where

$$p_j(\boldsymbol{x}) = \begin{cases} 0.01 \cdot exp(-(\frac{x_j}{20})^{2.5}) & for \quad j = 1, 2 \\ 0.01 \cdot exp(-\frac{x_j}{15}) & for \quad j = 3 \end{cases}$$

In Figure **??** *we present box collections computed by the sampling algorithm. The smoothing property of the box approach becomes apparent because the coverings preserve the symmetry of the function in the first two parameters* $(F(x_1, x_2, x_3) = F(x_2, x_1, x_3)\ \forall \boldsymbol{x} \in \mathbb{R}^3)$.

2.3 MOEAs

Evolutionary algorithms (EAs) are iterative stochastic search methods that are based on the two concepts of generate and evaluate [**?**]. Up to now, there are many multi-objective optimization methods that are based on this idea of EAs. MOEAs have demonstrated the advantage of using population-based search algorithms for solving multi-objective optimization problems. In all of these methods converging to the Pareto-optimal front and maintaining a spread of solutions (diversity) are the most important factors. MOEAs can be divided into two groups. The first group contains the MOEAs that always keep the best solutions (non-dominated solutions) of each generation in an *archive*, and they are called



(a) 10 steps ($\mathcal{C}_{10}$)

(b) 20 steps ($\mathcal{C}_{20}$)

(c) 30 steps ($\mathcal{C}_{30}$)

(d) image set $F(\mathcal{C}_{30})$

Fig. 2. Computation of the Pareto set of a model $F : \mathbb{R}^3 \to \mathbb{R}^2$ using subdivision techniques.

MOEAs with elitism. It is proved by Rudolph [**?**] that in some cases elitism will provide convergence to the Pareto set. In the second group, there is no archive for keeping best solutions and MOEA may loose them during generations. MOEAs with elitism are studied in several methods like Rudolph and Agapie' Elitist GA, Elitist NSGA-II, SPEA, PAES (see e.g. [**?**] for all) and SPEA2 [**?**].

Figure **??** shows the typical structure of a MOEA with elitism, where t denotes the number of the generation, P_t the population, and A_t the archive at generation t. The aim of function *Generate* is to generate new solutions in each iteration t which is done through selection, recombination and mutation. The function *Evaluate* calculates the *fitness* value of each individual in the actual population P_t. Fitness assignment in MOEA is done in different ways such as by Pareto-ranking [**?**], non-dominated sorting [**?**], or by calculating Pareto-strengths [**?**]. Since only the superior solutions must be kept in the archive, it must be updated after each generation. The function *Update* 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

BEGIN
Step 1: $t = 0$;
Step 2: Generate the initial population P_0 and initial archive A_0
Step 3: *Evaluate* P_t
Step 4: $A_{t+1} := Update(P_t, A_t)$
Step 5: $P_{t+1} := Generate(P_t, A_t)$
Step 6: $t = t + 1$
Step 7: Unless a *termination criterion* is met, goto Step 3
END

Fig. 3. Typical structure of an archive-based MOEA.

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. These three phases of an elitist MOEA are iteratively repeated until a termination criterion is met such as a maximum number of generations or when there has been no change in non-dominated solutions found for a given number of generations. The output of an elitist MOEA is the set of non-dominated solutions stored in the final archive. This set is an approximation of the Pareto-set and often called *quality set*. The above algorithm structure is common to most elitist MOEAs. In some of these methods (e.g., Rudolph and Agapie' Elitist GA, NSGA2), in the case of inadequate available space in the archive to store all of the non-dominated solutions, only those non-dominated solutions that are maximally apart from their neighbors are chosen. Therefore a crowding method is executed to select the solutions in less crowded areas. However, the true convergence property cannot be achieved, since an existent Pareto-optimal solution may get replaced by one which is not Pareto-optimal during the crowding selection operation. In some other methods (e.g., SPEA) clustering is done among the archive members, when the size of the archive exceeds. The use of clustering among the archive members guarantees spread among them. However, these algorithms lack a convergence proof, simply because of the same reason as in crowding methods: during the clustering procedure an existent Pareto-optimal archive member may get replaced by a non-Pareto-optimal.

3 Combination of Subdivision Techniques with MOEAs

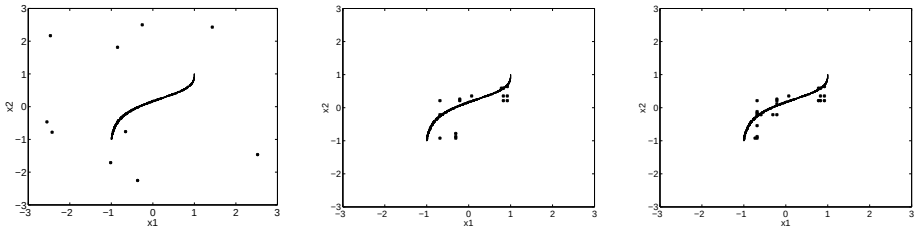
3.1 Basic Idea

The basic idea behind the following algorithms is to view MOEAs as special (set oriented) dynamical systems which have to be combined properly with the subdivision techniques in order to increase the total performance. In particular the

following property of MOEAs will be utilized for the combination of these methods: *MOEAs (typically) generate very quickly some very good approximations of Pareto points.*
We illustrate this by the following example:

$$\begin{aligned} f_1, f_2 &: Q \subset \mathbb{R}^2 \to \mathbb{R} \\ f_1(\boldsymbol{x}) &= (x_1 - 1)^2 + (x_2 - 1)^4 \\ f_2(\boldsymbol{x}) &= (x_1 + 1)^2 + (x_2 + 1)^2 \end{aligned} \tag{3.1}$$

In Figure **??** (a) we show a starting population consisting of 10 randomly chosen points in the domain $Q = [-3, 3] \times [-3, 3]$. The following two figures show the resulting populations after 5 and 10 generations using SPEA (see [**?**]). Here we observe that already after 5 generations there are some individuals close to the Pareto set. This property makes it possible to improve the sampling algo-



(a) initial population (b) generation 5 (c) generation 10

Fig. 4. One advantage of EAs is to find some good solutions quickly. The solid line indicates the actual Pareto set (in parameter space).

rithm described above: instead of using many test points to evaluate a (high-dimensional) box, it is better to take just a few test points as the initial population of a "short" MOEA[5]. The EA only has to run for a short time because a box B is kept if it contains at least one point in N, namely the set of nondominated points of the total set of test points.

3.2 The Algorithms

Here we propose different algorithms with the desire to combine the advantages both of the subdivision techniques and the MOEAs. Practical combinations of these algorithms for the efficient solution of MOPs will be given in the last paragraph.

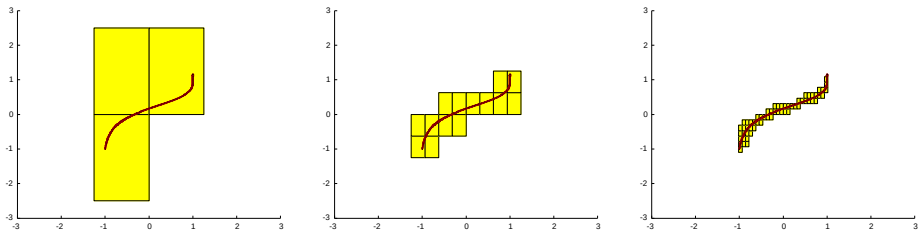
[5] A short MOEA is characterized by a short running time; that means small initial population and few generations.

EA-Subdivision The discussion made above leads directly to the first algorithm: *use the sampling algorithm combined with a "short" MOEA for the evaluation of every box.* That is, we propose a particular choice of test points as follows:

$$P_B := \text{final population of "short" MOEA}$$

The only task of the MOEA is to find as fast as possible one good approximation of a Pareto point relative to the given domain. Therefore, diversity or even clustering are not needed. But special attention should be paid so that the MOEA does not get stuck on local minima. In particular "hill climbers" failed in some situations.

EXAMPLE B *In Figure* **??** *we show the coverings of the set of Pareto points after 4, 8 and 12 subdivision steps. The black "line" indicating the Pareto set is in fact the resulting box collection after 20 subdivision steps. In this example the population size and the number of generations were chosen to be 5.*



(a) 4 steps (b) 8 steps (c) 12 steps

Fig. 5. Application of `EA-Subdivision`.

Recovering It may be the case that in the course of the subdivision procedure boxes get lost although they contain part of the Pareto set. We now describe two algorithms containing a kind of "healing" process which allows to recover the Pareto set.

Before we can state the algorithms we give some details about the box collections: For theoretical purposes denote $\mathcal{P}_k$ a *complete* partition of the set $Q = B_{\hat{c},\hat{r}}$ into boxes of subdivision size - or *depth*[6] - k, which are generated by successive bisection of Q. Then there exists for every point $\boldsymbol{p} \in Q$ and every depth k exactly one box $B(\boldsymbol{p}, k) \in \mathcal{P}_k$ with center $\boldsymbol{c}$ and radius $\boldsymbol{r}$ such that

$c_i - r_i \leq p_i < c + r_i, \quad \forall i = 1, .., n.$

Let us first consider the case where the covering is not complete but every box contains a part of the Pareto set (like box B_1 in Figure **??**). The aim of the

[6] $\mathcal{P}_k$ and hence every box collection considered here can be identified with a set of leaves of a binary tree of depth k.

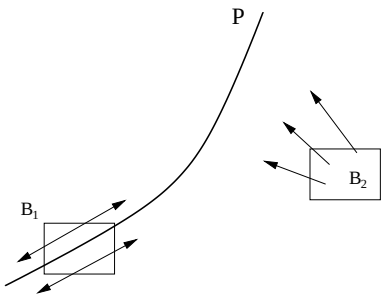


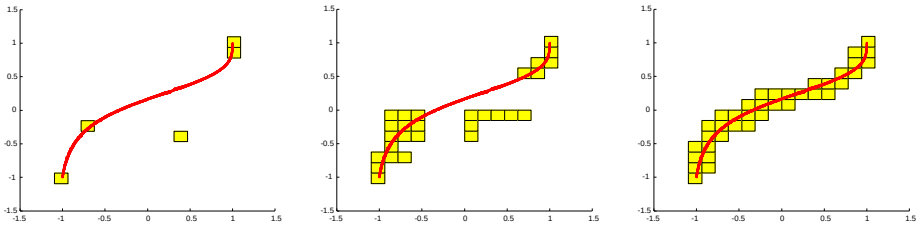
Fig. 6. Different problems for recovering: the algorithms have to cope with the fact that some boxes contain a part of the Pareto set (B_1) but others do not (B_2).

algorithm is to extend the given box collection step by step along the covered parts of the Pareto set until no more boxes are added. In order to find the corresponding neighboring boxes of a given box B with center c and radius r we run a MOEA in the extended box $\hat{B}$ given by center c and radius $\lambda \cdot r$ with $\lambda > 1$, say $\lambda = 3$. Afterwards the box collection is extended by the boxes $B \in \mathcal{P}_k$ which contain points from the resulting population (see Figure **??**). In the first step this is done for all boxes from the box collection, for the following steps this local search has to be done only in the neighborhood of the boxes which were added in the preceding step. With a given box collection $\mathcal{B}_k$ the complete algorithm `StaticRecover` reads as follows:

Algorithm `StaticRecover`

1.) for all $B \in \mathcal{B}_k$
    $B.active := TRUE$
2.) for $i = 1, .., MaxStep$
    $\hat{\mathcal{B}}_k := \mathcal{B}_k$
    for all $\{B \in \mathcal{B}_k : B.active == TRUE\}$
        run MOEA in an extended universe $\hat{B} := (B.c, \lambda \cdot B.r)$
        $P :=$ final population
        $B.active = FALSE$
        for all $p \in P$:
            if $B(p,k) \notin \mathcal{B}_k$
                $\mathcal{B}_k := \mathcal{B}_k \cup B(p,k)$
                $B(p,k).active := TRUE$
    if $\hat{\mathcal{B}}_k == \mathcal{B}_k$ $STOP$

Hence `StaticRecover` only allows the addition of boxes into the given collection. The desired covering of the set of Pareto points cannot get worse, but will improve if the parameters of the algorithm are adjusted properly. On the other hand, `StaticRecover` does not treat adequately the case where a box does not contain some part of the Pareto set but is possibly far away (e.g. box B_2 in Figure **??**). In this case the algorithm would extend the box covering by many undesired regions on their way towards the Pareto set (in particular in higher



(a) given "covering" (b) recover step 1 (c) recover step 2

Fig. 7. Application of `DynamicRecover` on a simple example.

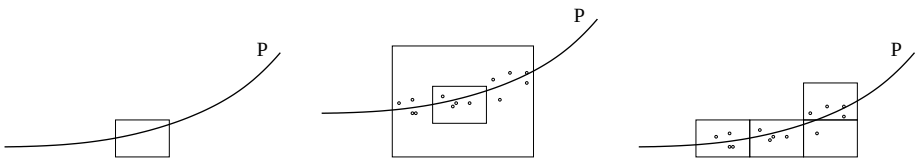
dimensions). Thus, when there are "good" and "bad" boxes like in Figure **??** we propose the following algorithm.

Algorithm `DynamicRecover`

1.) for all $B \in \mathcal{B}_k$
$B.active := TRUE$
2.) for $i = 1, .., MaxStep$
$\hat{\mathcal{B}}_k := \mathcal{B}_k, \quad \mathcal{B}_k := \emptyset$
for all $B \in \hat{\mathcal{B}}_k$: $B.active == TRUE$
run MOEA in an extended universe $\hat{B} := (B.c, \lambda \cdot B.r)$
$P_B :=$ final population
$P :=$ nondominated points of $\bigcup_{B \in \hat{\mathcal{B}}_k} P_B$
for all $p \in P$:
$\mathcal{B}_k := \mathcal{B}_k \cup B(p,k)$
if $B(p,k) \in \mathcal{B}_k$ $B(p,k).active := FALSE$
else $B(p,k).active := TRUE$
if $\hat{\mathcal{B}}_k == \mathcal{B}_k$ $STOP$

In contrast to `StaticRecover` this algorithm has again the disadvantage that good boxes can be deleted while they have been computed once[7]. But otherwise there would be no chance to sort out the boxes which contain no optimal solution. The speed of the algorithm depends - besides of the MOEA - on the choice of the extension factor λ. A larger value of λ yields faster convergence but lower robustness. In general, the number of generations and the size of the initial population should increase with λ. For this local covering of the part of the Pareto set the MOEA has to preserve diversity. Furthermore the convergence of the MOEA should be good enough in order not to insert too many superfluous boxes.

[7] The usage of an archive seems to be suitable and has to be tested in future work.



(a) covering (b) local search (c) recovering

Fig. 8. Working principle of `StaticRecover`

EXAMPLE C *We again consider the MOP (??). Algorithm* `DynamicRecover` *was applied to a chosen initial box collection (see Figure ??). The algorithm stops after 2 iterations with a total covering of the Pareto set.*

Combination of the Algorithms Here we propose two possible combinations of the algorithms described above which turned out to be practical for the numerical solution of MOPs.
First, if the MOP is "moderate" dimensional we suggest the use of the algorithm `EA-Subdivision` in combination with `StaticRecover`. For most MOPs it is sufficient to use `StaticRecover` only once or twice during the computation, but for more complicated problems it can turn out that both algorithms have to be used in alternation: after some number of iteration steps of `EA-Subdivision`, the number of new boxes added to this box collection by `StaticRecover` gives a feedback on the quality of the computed "covering" and hence the adjustment of the MOEA can be adapted to the next subdivision steps. Eventually the algorithm stops if the desired granularity is reached and no more boxes are added by the recovery step. Here, the global optimization is done by the subdivision techniques while the MOEAs are used for local optimization.
For higher dimensional MOPs we recommend another strategy since the evaluation of the boxes by MOEAs is expensive even if one is only interested in just a few good solutions. Here we suggest to improve the result of a MOEA - where the granularity K can be low - by using `DynamicRecover`. Therefore, the recovery process should extend the box collection which is generated by the final population of the MOEA. Finally, this extended (hopefully complete) covering can be further refined using subdivision techniques. Important for this method is the proper choice of size of the boxes. We suggest to adjust the edge lengths of the boxes to be approximately κK, where κ is a safety factor, say $\kappa = 2$.
Also, in this method the stopping condition is given by the size of the boxes and the number of boxes which are added in a recovery step. But in contrast to the first combination in this method the global optimization is done by the MOEA while the subdivision techniques can only give local improvements.

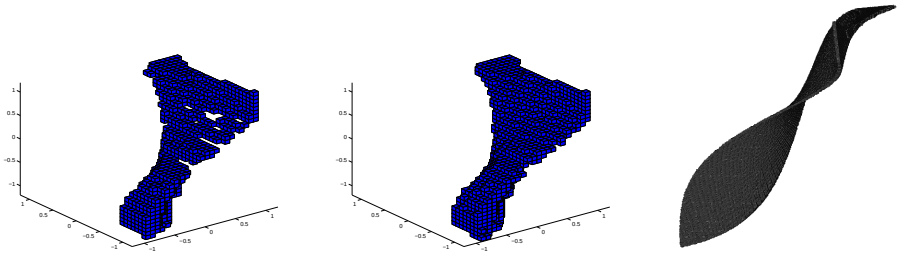
4 Computational Results

4.1 Example 1

The following three objectives have to be minimized at once:

$$\begin{aligned} f_1, f_2, f_3 : Q = [-5,5]^3 &\to \mathbb{R}, \\ f_1(x_1,x_2,x_3) &= (x_1-1)^4 + (x_2-1)^2 + (x_3-1)^2, \\ f_2(x_1,x_2,x_3) &= (x_1+1)^2 + (x_2+1)^4 + (x_3+1)^2, \\ f_3(x_1,x_2,x_3) &= (x_1-1)^2 + (x_2+1)^2 + (x_3-1)^4. \end{aligned}$$

Figure **??** shows the result of a combination of `EA-Subdivision` and `StaticRecover`. To achieve the uncomplete covering of Figure **??** (a) no EA techniques were necessary but only the center point was used to evaluate each box. Figure **??** (c) shows a tight and complete covering of the Pareto set. The picture was produced by the software package GRAPE[8].



(a) `EA-subdivision` ($\rightsquigarrow \mathcal{B}^{(1)}$)

(b) `StaticRecover` on $\mathcal{B}^{(1)}$ ($\rightsquigarrow \mathcal{B}^{(2)}$)

(c) `EA-subdivision` on $\mathcal{B}^{(2)}$

Fig. 9. Combination of EA-subdivision and `StaticRecover`

4.2 Example 2

Now we consider the following MOP

$$\begin{aligned} f_1, f_2 : [-5.12, 5.12]^{10} &\to \mathbb{R}, \\ f_1(x) &= \sum_{i=1}^{9} (-10 \mathrm{e}^{-0.2\sqrt{x_i^2 + x_{i+1}^2}}), \\ f_2(x) &= \sum_{i=1}^{10} (|x_i|^{0.8} + 5\sin(x_i)^3). \end{aligned}$$

Figure **??** shows a final population using SPEA (size of initial population: 200; number of generations: 300) and the local improvement by an application of `DynamicRecover` on this result.

[8] http://www.iam.uni-bonn.de/sfb256/grape/

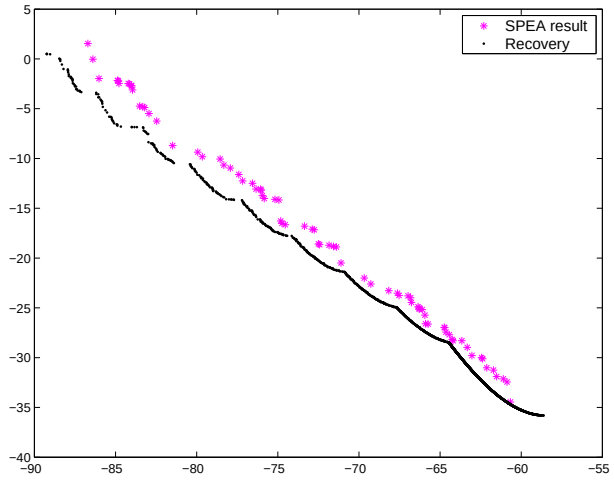


Fig. 10. Local improvement of a MOEA result by using `DynamicRecover`

4.3 Example 3

Finally we consider an MOP which arises in antenna design ([?]):

$$f_1(x_\nu, y_\nu) = -4\pi^2 \left| \sum_{\nu=-n}^{n} (-i)^\nu \mathcal{J}_\nu(\ell)(x_\nu + iy_\nu) \right|^2 ,$$

$$f_2(x_\nu, y_\nu) = \max_{\eta=0,..,5} \left(4\pi^2 \left| \sum_{\nu=-n}^{n} (-i)^\nu \mathcal{J}_\nu(\ell)(x_\nu + iy_\nu) e^{i\nu s_\eta} \right|^2 \right)$$

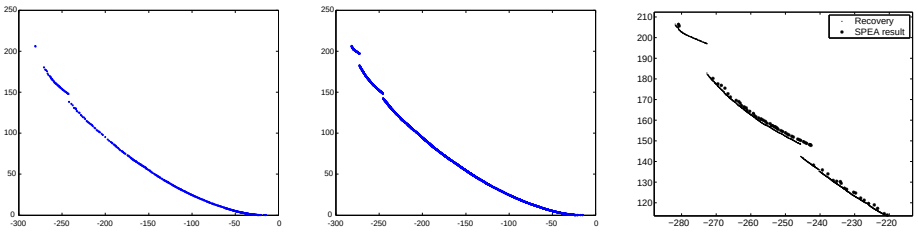
subject to the constraints

$$x_\nu, y_\nu \in \mathbb{R} \quad (\nu \in \mathbb{Z},\ |\nu| \leq n),$$

$$2\pi \sum_{\nu=-n}^{n} (x_\nu^2 + y_\nu^2) \leq 1$$

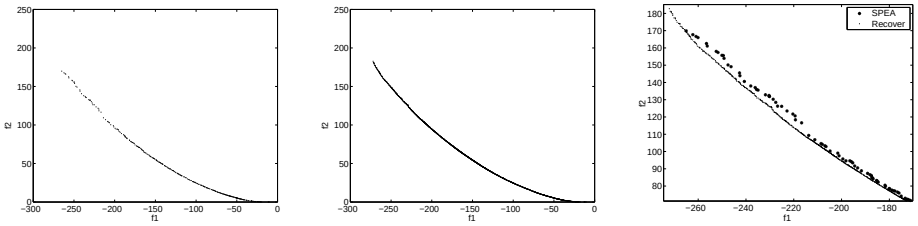
with the specific discretization points $s_\eta = \frac{3}{4}\pi + \eta\frac{\pi}{10}$. Here $\mathcal{J}_\nu$ denotes the Bessel function of ν-th order. We have tested the algorithms for $n = 5$ and $\ell = 10$. Since $\mathcal{J}_\nu(x) = (-1)^\nu \mathcal{J}_{-\nu}(x)$ and $\mathbb{C} \cong \mathbb{R}^2$ this leads to a model with 12 free parameters. First, we have applied the recovery techniques on the result of a SPEA with the following parameter settings: 1000 initial individuals coded to 13 bits, 500 generations and archive size 1000 (see Figure **??**). As in Example 2 also here local improvements can be observed. The total running time was 7.5 hours for the SPEA result and 20 minutes for the application of `StaticRecover` on it. Furthermore, we have taken another SPEA result with 300 initial individuals coded to 7 bits, 300 generations and archive size 300 in order to decrease the computational time. Afterwards we have used the recovering techniques - with larger box sizes due to the lower granularity of the MOEA - and got similar

results (see Figure **??**). The total running time was 20 minutes for the SPEA result and another 15 minutes for the recovering. Therefore, at least here it is possible to use the recovering techniques to speed up the running time while the total performance of the combination of a MOEA and the subdivision techniques is kept.



(a) SPEA result (b) `StaticRecover` (c) comparison in a selected area

Fig. 11. Application of `StaticRecover` on a SPEA result and a comparison in selected area.



(a) SPEA result (b) `StaticRecover` (c) comparison in a selected area

Fig. 12. Application of `StaticRecover` on a SPEA result and a comparison in selected area.

5 Conclusion and Future Work

We have presented robust set oriented algorithms for the numerical solution of MOPs. More precisely, we have presented two different methods. The first one is an improvement of the `sampling algorithm` described in Section 2. Due to the evaluation of the subdomains by MOEAs the speed of the computation of MOPs can be increased while its robustness is kept. This allows to apply the subdivision techniques to higher dimensional MOPs.
The second method is a technique for the local improvement of the diversity of MOEA results, but can also be used to speed up the computational time for the

solution of an MOP.
The general advantages of our algorithms are certainly their robustness, the preservation (or improvement) of the diversity, and the "natural" stopping conditions. On the other hand so far the subdivision techniques are restricted to "moderate" dimensions. Thus, for higher dimensional MOPs recovering techniques should be applied.
In future work we would like to adjust the structure of the MOEAs for the special requirements of the different algorithms. Also other optimization metaheuristics like e.g. *particle swarm optimization* have to be tested with respect to a combination with subdivision strategies.

References

1. D. Corne, M. Dorigo, and F. Glover. *New Ideas in Optimization.* Mc Graw Hill, 1999.
2. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, 2001.
3. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *Parallel Problem Solving from Nature VI (PPSN-VI)*, pages 849–858, 2000.
4. M. Dellnitz, R. Elsässer, T. Hestermeyer, O. Schütze, and S. Sertl. Covering Pareto sets with multilevel subdivision techniques. to appear, 2002.
5. M. Dellnitz and A. Hohmann. The computation of unstable manifolds using subdivision and continuation. In H.W. Broer, S.A. van Gils, I. Hoveijn, and F. Takens, editors, *Nonlinear Dynamical Systems and Chaos*, pages 449–459. Birkhäuser, *PNLDE* 19, 1996.
6. M. Dellnitz and A. Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numerische Mathematik*, 75:293–317, 1997.
7. M. Dellnitz, O. Schütze, and St. Sertl. Finding zeros by multilevel subdivision techniques. 2002.
8. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, Inc., 1989.
9. A. Jüschke and J. Jahn. A bicriterial optimization problem of antenna design. *Comp. Opt. Appl.*, 7:261–276, 1997.
10. H. Kuhn and A. Tucker. Nonlinear programming. *Proc. Berkeley Symp. Math. Statist. Probability, 2nd, (J. Neumann, ed.)*, pages 481–492, 1951.
11. G. Rudolph. On a Multi-Objective Evolutionary Algorithm and Its Convergence to the Pareto Set. In *5th IEEE Conference on Evolutionary Computation*, pages 511–516, 1998.
12. S. Schäffler, R. Schultz, and K. Weinzierl. A stochastic method for the solution of unconstrained vector optimization problems. To appear in J. Opt. Th. Appl., 2002.
13. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. In *Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, 2002.
14. E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and and the strength Pareto approach. *IEEE Trans. on Evolutionary Computation*, 3(4):257–271, 1999.