Covering Pareto-optimal Fronts by Subswarms in Multi-objective Particle Swarm Optimization

Sanaz Mostaghim
Electrical Engineering Department,
University of Paderborn,
Germany
mostaghim@date.upb.de

Jürgen Teich
Computer Science Department,
Friedrich-Alexander-University, Erlangen,
Germany
teich@informatik.uni-erlangen.de

***Abstract*— Covering the whole set of Pareto-optimal solutions is a desired task of multi-objective optimization methods. Because in general it is not possible to determine this set, a restricted amount of solutions are typically delivered in the output to decision makers. In this paper, we propose a new method using multi-objective particle swarm optimization to cover the Pareto-optimal front. The method works in two phases. In phase 1 the goal is to obtain a good approximation of the Pareto-front. In a second run subswarms are generated to cover the Pareto-front. The method is evaluated using different test functions and compared with an existing covering method using a real world example in antenna design.**

I. Introduction

A Multi-objective Optimization problem (MOP) is solved, when all its Pareto-optimal solutions are found. Indeed, this is the goal of Multi-objective Optimization (MO) to find a set of optimal solutions in one simulation run, in contrast to classical optimization methods e.g., weighting sum methods. However, it is impossible to find the whole set of Pareto-optimal solutions of a continuous front. But, because the decision makers need only a restricted amount of well distributed solutions along the Pareto-optimal front, the task of MO methods is changed to find a relatively small amount of solutions. In elitist Multi-objective Evolutionary Algorithm (MOEA) and Multi-objective Particle Swarm Optimization (MOPSO) methods, the elite solutions are transfered by an archive to the next generation. The archive of the last generation is the output of the method, the size of which should be restricted. Indeed, the size of the archive is always kept constant. Restricting the size of the archive also has influences on the diversity of solutions and the computational time. Therefore, the results obtained by most of MOEA and MOPSO methods have restricted amount of solutions in the output, by keeping a good diversity along the Pareto-optimal front. Diversity of output solutions is studied by applying methods like niching, clustering or truncation by several researchers [2], [14], [15]. These techniques often need a high computational time and at last we have a restricted number of solutions in the output [14], [15]. On the other hand, finding a large set of Pareto-optimal solutions is possible, if we run the MO methods for a large number of generations. This also needs a high computational time, with this difference that the decision maker is free to select some solutions from the whole Pareto-optimal front.

One way in finding a relatively large set of approximated Pareto-optimal solutions is to apply covering techniques. Indeed, by covering we find a finite set of solutions which are very close to each other. Covering the Pareto-optimal front is studied by Hybrid MOEA [13]. In [13], the non-dominated solutions are found by MOEA and then a recovering method is applied to cover the Pareto-optimal front. In this case, the recovering method is a combination of MOEA and a subdivision method [3]. Also, the ϵ-MOEA method is theoretically able to cover the approximated Pareto-optimal front in the case of using small values of ϵ [8], [9]. However, it also needs a high computational time to complete the covering.

In this paper, we address the covering of the Pareto-optimal front by applying MOPSO. MOPSO methods have the property that the particles move towards the Pareto-optimal front during generations. By running a MOPSO with a restricted archive size, it is possible to find a well distributed set of non-dominated solutions very close to the Pareto-optimal front [10]. Here, we use this knowledge and propose another MOPSO (called *covering MOPSO*) to cover the gaps between the non-dominated solutions. The particles in the population of the covering MOPSO are divided into subswarms after one generation by using the Sigma method in [10] and these subswarms take the responsibility to recover the Pareto-optimal front. This method is tested on different test functions and compared with the Hybrid MOEA covering technique in [13] for a real world application in antenna design.

This paper has the following structure. Definitions of MOP are outlined at the end of this section. Section II gives a brief background on covering the Pareto-front by MOEA and Hybrid MOEA methods. In Section III, the proposed method in the Paper is studied. Section IV is dedicated to the experiments on some selected test functions and in Section V a real world example is studied. The Paper is then concluded in Section VI.

A. Definitions

In the following, we state the multi-objective optimization problem in its general form:

$$\text{minimize} \qquad \vec{y} = \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \cdots, f_m(\vec{x}))$$

0-7803-8515-2/04/$20.00 ©2004 IEEE

The *decision vectors* (parameters) $\vec{x}$ belong to the feasible region denoted $S \subset \Re^n$. We denote the image of the feasible region by Z and call it a feasible objective region. The elements of Z are called objective vectors and they consist of objective values $\vec{f}(\vec{x})$.

A decision vector $\vec{x}_1 \in S$ is said to *dominate* a decision vector $\vec{x}_2 \in S$ (denoted $\vec{x}_1 \prec \vec{x}_2$), iff the decision vector $\vec{x}_1$ is not worse than $\vec{x}_2$ in all objectives and strictly better than $\vec{x}_2$ in at least one objective.

A decision vector $\vec{x}_1 \in S$ is called *Pareto-optimal* if there does not exist another $\vec{x}_2 \in S$ that dominates it. An objective vector is called Pareto-optimal if the corresponding decision vector is Pareto-optimal.

The non-dominated set of the entire feasible search space S is the Pareto-optimal set. The Pareto-optimal set in the objective space is called *Pareto-optimal front*.

II. Background

MOPSO methods are one of the utilities in solving different kinds of MOPs. In MOPSO [1], [4], [6], [10], a set of particles are initialized in the decision space at random. To each particle i, a position x_i in the decision space and velocity v_i is assigned. The particles change their positions and move towards the best so far found solutions. The best solutions are indeed, the non-dominated solutions from the last generations, which are kept in the archive[1]. Moving towards the optima is done in the calculations of the velocities as follows:

$$\begin{aligned} v^i_{j,t+1} &= w v^i_{j,t} + c_1 R_1 (p^i_{j,t} - x^i_{j,t}) + c_2 R_2 (p^{i,g}_{j,t} - x^i_{j,t}) \\ x^i_{j,t+1} &= x^i_{j,t} + v^i_{j,t+1} \end{aligned} \quad (1)$$

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 $[0, 1]$. i is the index of a particle, and t denotes the generation index.

According to Equation 1, each particle has to change its position $\vec{x}^i_t$ towards the position of a local guide $\vec{p}^{i,g}_t$ which must be selected from the updated set of non-dominated 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 studied in [1], [4], [6], [10]. $\vec{p}^i$ is a memory for the particle i and keeps the non-dominated (best) position of the particle by comparing the new position $\vec{x}^{\,i}_{t+1}$ in the objective space with $\vec{p}^i_t$ ($\vec{p}^i_t$ is the last non-dominated (best) position of the particle i). For escaping 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:

$$x^i_{j,t} = x^i_{j,t} + R_T \; x^i_{j,t} \quad (2)$$

In Equation (2), $R_T \in [-1, 1]$ is a random value added to the updated 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.

Finding a relatively large set of Pareto-optimal solutions is possible by running the MOPSO for many generations. It is proved by Rudolph [12] that existence of elitism (keeping elite solutions in the archive) is necessary to converge to the Pareto-optimal front in MOEAs. MOPSOs have also the same structure as MOEA, with the differences in fitness assignment, selection and recombination operators. However, the existence of the archive helps the method not to lose the so far found non-dominated solutions and therefore after many generations the Pareto-optimal solutions are gathered in the archive.

In most of MOPSO and MOEA methods, the archive is restricted to a certain size. This is done because of the following reasons:

- Most of MO methods need a high computational time, when the size of the archive increases. This is also studied by [5], [11].
- Diversity of solutions improves, when the archive size is fixed, particularly in MOPSO.
- The computational time for finding the best local guides in MOPSO increases, if we store high number of solutions in the archive.

Therefore, the solutions stored in the archive are just a selection of the Pareto-optimal solutions (if converged to the true Pareto-optimal front). The methods try to keep a good diversity of solutions, so that the decision maker has the possibility to access the whole Pareto-optimal front. However, in some cases if we can find a good cover of the whole Pareto-optimal front, the task of the decision makers would be easier to select the desired solutions. Therefore, covering the Pareto-optimal front in less computational time is also important in MO.

A. Covering by Hybrid MOEA

Convergence to the true set of Pareto-optimal solutions is possible when the whole search space is explored. One possible solution for having a controllable exploration is to use Hybrid MOEA [13]. Indeed, this method is based on iterative division of the search space into subspaces (boxes) in the parameter space. Then the boxes which contain *good* solutions are divided again into boxes. This procedure continues until an acceptable granularity for the Pareto-optimal front is reached. Figure 1 shows an example in dividing a two dimensional parameter space into boxes. The algorithm Hybrid MOEA is outlined in Algorithm 1. Using a multi-level subdivision scheme, the Algorithm produces a sequence of sets $\mathcal{B}_0,\ \mathcal{B}_1,\ \ldots$ where each $\mathcal{B}_k$ consists of elements called *boxes* (B):

$$\operatorname{diam} \mathcal{B}_k = max_{B \in \mathcal{B}_k} \operatorname{diam}(B)$$

The Algorithm starts with a big compact set $\mathcal{B}_0$ and the $\mathcal{B}_k$ is inductively obtained from $\mathcal{B}_{k-1}$ in parameter space. By a repeated bisection and selection of boxes the box coverings get tighter until the desired granularity of this outer approximation is reached. The stopping criteria should be defined by a maximum value of k.

[1]An archive is an external population, in which the so far found non-dominated solutions are kept. The archive members don't dominate each other.

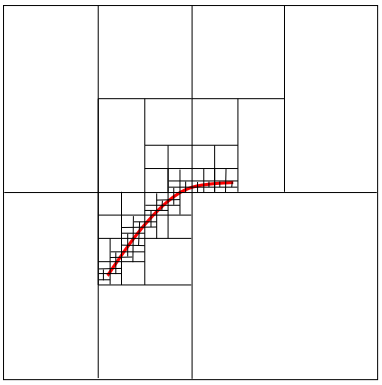


Fig. 1. Example of application of a subdivision method (in two dimensional parameter space)

Algorithm 1 Hybrid MOEA Algorithm

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

where $0 < \theta_{min} \leq \theta_k \leq \theta_{max} < 1$.
2. Pre-optimization
forall boxes B in $\hat{\mathcal{B}}_k$: $P_B = MOEA(B)$
3. Selection
$N :=$ non-dominated points of $\bigcup_{B\in\hat{\mathcal{B}}_k} P_B$
$\mathcal{B}_k := \left\{B \in \hat{\mathcal{B}}_k : P_B \cap N \neq \emptyset\right\}$

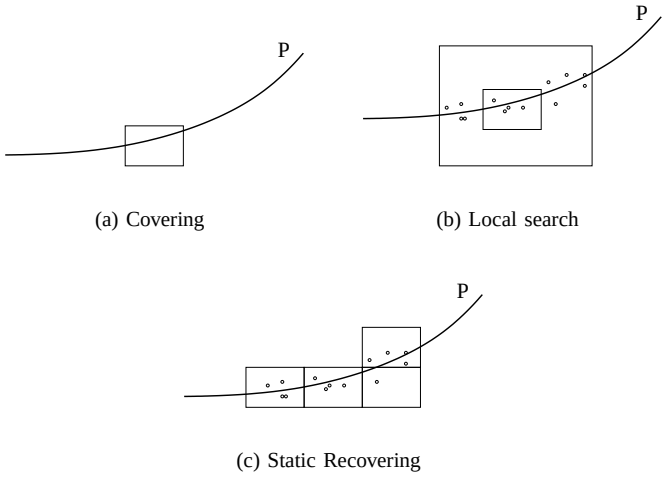
The Pre-optimization process in Step 2 is done in each box as follows. Several test points are defined in each box. These test points are indeed the initial population of a "short" MOEA[2]. The MOEA should be run for a short time in a box B. The box B is kept if it contains at least one solution in N, namely the set of non-dominated solutions of the total set of test points (Step 2, Algorithm 1). Then, the non-dominated solutions among all the test points in the whole set of boxes are stored in the set N. The boxes, which contain at least one of the non-dominated solutions in the set N can survive for further subdivision.

By using this algorithm, it is possible to detect the entire set of true Pareto-optimal solutions. In the subdivision technique, a box is kept if it contains *at least* one non-dominated point.

Using this Algorithm, it may be the case that in the course of the subdivision procedure boxes get lost although they contain points belonging to the Pareto-optimal set. To avoid gaps in the Pareto-optimal front, a strategy called *Static Recovering* is proposed in [13]. The idea is to run the Hybrid MOEA and obtain the possible optimal solutions. Then the Static Recovering is applied on the solutions to fill the gaps.

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

The Static Recovering [13] strategy works as follows. The input to this Algorithm is a set of boxes, which contain parts of the Pareto-front. Static Recovering extends the given box collection step by step along the Pareto-front until no more boxes are added. Figure 2 (b) shows the first step in Static Recovering. The size of each given box is extended by a factor λ. Then a MOEA is run in the extended box. So the extended box can now be divided into small boxes like in the subdivision method and perform the local search in recovering the area in the extended box. This is shown in Figure 2 (c).



(a) Covering (b) Local search

(c) Static Recovering

Fig. 2. Static Recovering in Hybrid MOEA [13]. P denotes the Pareto-optimal set that is to be covered.

a) Discussion: The Hybrid MOEA together with Static Recovering covers the Pareto-optimal front for a given granularity of boxes. This is an advantage of using the Hybrid MOEA, however it needs a high computational time for a large number of parameters. The Static Recovering fills the gaps between the non-dominated solutions on the Pareto-optimal front. But it just makes some local improvements on the obtained solutions. If the obtained non-dominated front is on the local optima, the Static Recovering is not able to improve it to the global front.

III. Covering by MOPSO

As it has been mentioned, it is possible to cover the Pareto-optimal front with MOPSO or MOEA methods. We omit the restriction on the archive size and let the MO method be run for a very high number of generations. Of course this needs a high computational time. MOPSO methods have the advantage that the particles move towards the front. Therefore, it is not important how big is the parameter space. The particles will be near the Pareto-front after several generations. It remains just the convergence and obtaining the diversity of solutions.

Here, a new method is proposed to perform the covering in low computational time. The MOPSO methods are able to find solutions with high convergence and diversity [1], [4], [6], [10]. This can be achieved easily for fixed sizes of the archive. The covering is completed in two following steps:

- Initial run: The MOPSO is run with a restricted archive size. We expect relatively well-distributed solutions very close to the Pareto-optimal front. The restriction on the archive can be achieved using the ϵ-dominance strategy presented in [10].
- Covering: The non-dominated solutions obtained from the initial run are considered as the input archive of a new MOPSO called *covering MOPSO*. The covering MOPSO should have more particles in the population and no restriction on the archive size. The particles in the population are divided into subswarms around each non-dominated solution after the first generation. The task of the subswarms is to cover the gaps between the non-dominated solutions obtained from the initial run. No restriction on the archive makes this method faster, because no clustering or truncation method is needed.

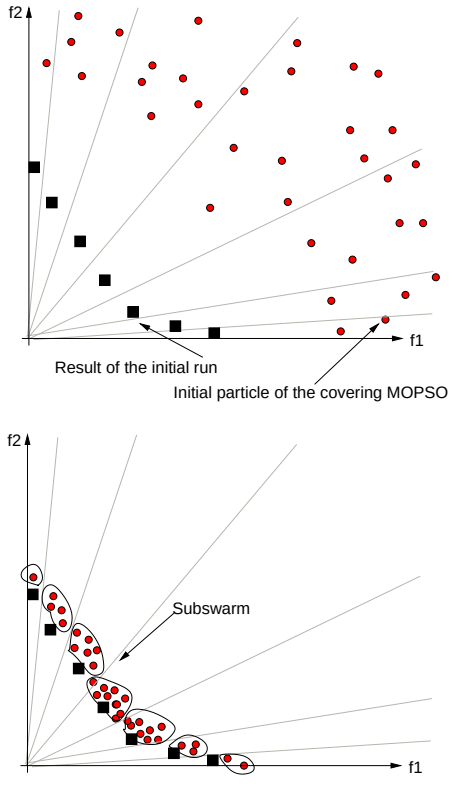


Fig. 3. Covering the approximated Pareto-front by using initial archive members. The initial archive members guide subswarms of solutions around themselves to cover the Pareto-front.

The initial run searches the whole space for obtaining good diversity of solutions. The covering is aimed to send subswarms of particles around the non-dominated solutions on the non-dominated front. Covering the front is then completed by these subswarms, which search just the neighborhood around each non-dominated solution. Figure 3 shows this scenario for covering the Pareto-front by MOPSO. The subswarms are generated by the Sigma method presented in [10] in finding the local best guides. All the particles in a subswarm have one common local best guide, which is in the non-dominated set made by the initial run. The size of each subswarm varies through generations and depends on the size of the initial population of the covering MOPSO.

It should be considered that any knowledge about the solutions helps the methods to find better solutions than before. In the covering process the size of the population has a great impact in covering the front faster. The larger the population size, the more particles gather in a subswarm and the front will be searched faster.

IV. Experiments

The following experiments have been performed on different test functions selected from [2][3] and shown in Table I. These test functions have different points of difficulties, which make the covering process complicated. The MOPSO method uses the Sigma method for finding the local best particles [10].

TABLE I
Test Functions

test	function	
T1	$g(x_2,\cdots,x_n)=1+9(\sum_{i=2}^{n}x_i)/(n-1)$	$x_i\in[0,\ 1]$
	$h(f_1,g)=1-\sqrt{f_1/g}$	$n=30$
	$f_1(x_1)=x_1$	$i=1,\ldots,n$
	$f_2(\vec{x})=g(x_2,\cdots,x_n).h(f_1,g)$	
T3	$g(x_2,\cdots,x_n)=1+9(\sum_{i=2}^{n}x_i)/(n-1)$	$x_i\in[0,\ 1]$
	$h(f_1,g)=1-\sqrt{f_1/g}-(f_1/g)\sin(10\pi f_1)$	$n=30$
	$f_1(x_1)=x_1$	$i=1,\ldots,n$
	$f_2(\vec{x})=g(x_2,\cdots,x_n).h(f_1,g)+1$	
T4	$g(x_2,\cdots,x_n)=1+10(n-1)+$	
	$(\sum_{i=2}^{n}(x_i^2-10cos(4\pi x_i))$	$x_1\in[0,\ 1]$
	$h(f_1,g)=1-\sqrt{f_1/g}$	$x_i\in[-5,\ 5]$
	$f_1(x_1)=x_1$	$n=10$
	$f_2(\vec{x})=g(x_2,\cdots,x_n).h(f_1,g)$	$i=2,\ldots,n$
T6	$g(x_2,\cdots,x_n)=1+9[(\sum_{i=2}^{n}x_i)/9]^{0.25}$	$x_i\in[0,\ 1]$
	$h(f_1,g)=1-\sqrt{x_1/g}$	$n=10$
	$f_1(x_1)=1-exp(-4x_1)sin^6(6\pi x_1)$	$i=1,2,\ldots,n$
	$f_2(\vec{x})=g(x_2,\cdots,x_n).h(f_1,g)$	

In the following, the properties of the Pareto-optimal fronts and the difficulties of the test functions are summarized.

- T1: (2-objective, 30 parameters)
  Pareto-optimal front: Convex, continuous, uniform distribution of solutions
  Difficulty: Large number of parameters
- T3: (2-objective, 30 parameters)
  Pareto-optimal front: Convex, disconnected-continuous
  Difficulties: Large number of parameters, discontinuous front
- T4: (2-objective, 10 parameters)
  Pareto-optimal front: Convex, continuous
  Difficulty: Large number of local Pareto-optimal fronts (21^9)
- T6: (2-objective, 10 parameters)
  Pareto-optimal front: Non-convex, disconnected-continuous, non-uniform density of solutions
  Difficulties: adverse density of solutions, non-convex front and discontinuous front

[3]Test function T6 has a disconnected Pareto-optimal front which is a different version as described in [2].

A. Parameter Setting

The parameters are selected as follows.

- Inertia weight: 0.4
- Turbulence factor: 0.07
- Population size: initial run: 100 for T6, 200 for T1 and T3, 300 for T4, covering: 200
- Number of generations: initial run: 200 for T1 and T3, 2000 for T4 and T6, covering: 500 for T1 and T3, 2000 for T4 and T6
- Archive size: initial run: 50, covering: not restricted

B. Results

Figure 4–7 show the results of covering the T1, T3, T6 and T4 test functions.

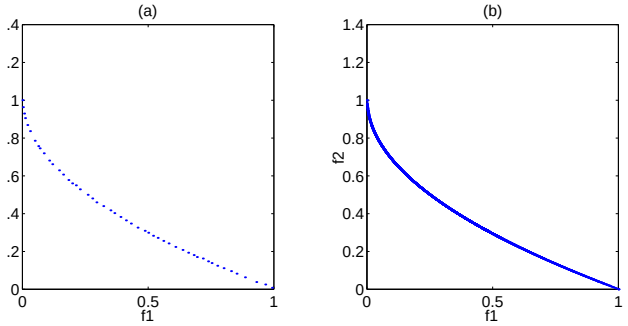


Fig. 4. Solutions of the T1 test function (a) with clustering (a = 50), (b) covered front

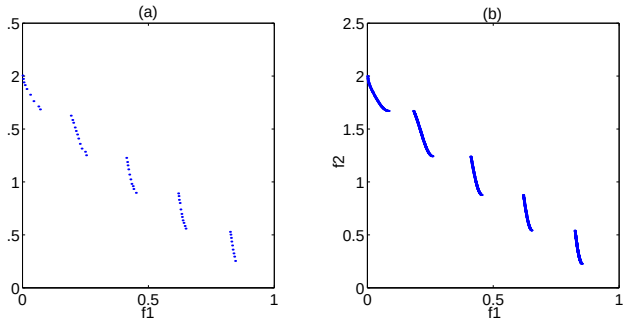


Fig. 5. Solutions of the T3 test function (a) with clustering (a = 50), (b) covered front

Covering the test function T4 is not as simple as the other test functions. This test function has a lot of local Pareto-fronts. By running the covering MOPSO to cover the front, it is observed that the solutions are not already converged to the true Pareto-optimal front. In this case the covering MOPSO improves the convergence of solutions. Indeed, the solutions shown in Figure 7 (1st Run) are not actually on the true Pareto-optimal front, although the MOPSO method has obtained obviously higher converged solutions than the other methods, e.g., [14] (this is also explained in [2]). One reason is the restricted number of generations. We have to notice that the obtained solutions from the first run have good diversity.

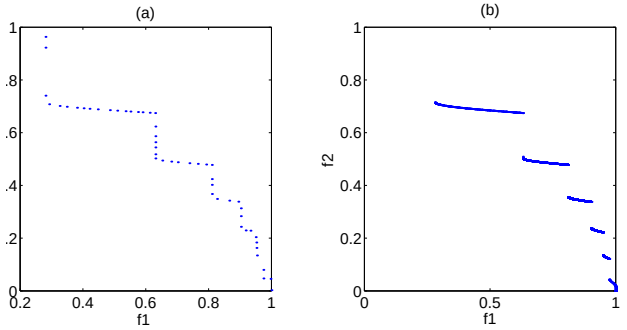


Fig. 6. Solutions of the T6 test function (a) with clustering (a = 50), (b) covered front

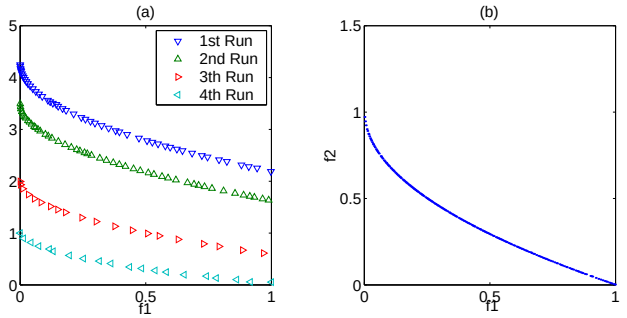


Fig. 7. Solutions of the T4 test function (a) with clustering, obtaining the converged front needs several runs, (b) covered front

Therefore, these solutions are considered as the initial archive of another run and the output of this second run are given as the initial archive of the next run. This procedure is repeated several times until the true Pareto-optimal front is reached[4]. Now the covering process can be started to cover the front. Figures 7 (a) and (b) show different steps in optimizing the test function T4. The reason on repeating the MOPSO instead of running it for a large number of generations is that when there are many local optima, the optimization method needs a lot of computational time to find solutions with high convergence and diversity of solutions simultaneously. The initial archive helps the initial particles to be divided in small groups and search the space faster. The covering result in Figure 7 (b) shows the good potential of MOPSO (Sigma method) in solving very difficult MOPs, where the recorded results of the MOEA method [14] are not on the true Pareto-optimal front.

For evaluation of the covered solutions, Euclidian distance between two neighbor solutions are computed. Table II shows the minimum and maximum obtained distances between the covered solutions. The computational times are also recorded in this table.

Very low values of D_{max} show that the solutions are very close to each other. However, if we run the MOPSO for a larger number of generations than the recorded value, we obtain more solutions and closer to each other. For obtaining faster results we can also apply the ϵMOPSO algorithm [9]. For the initial

[4]This needs a priori knowledge about the true Pareto-optimal front, which is available for this test function.

TABLE II

COMPUTATIONAL TIME OF INITIAL (t_{init}) AND COVERING (t_{cover}) RUNS OF MOPSO IN SECONDS. D_{min} AND D_{max} ARE THE MINIMUM AND MAXIMUM DISTANCES BETWEEN THE NON-DOMINATED SOLUTIONS OF THE APPROXIMATED PARETO-FRONT

test	t_{init}	t_{cover}	D_{min}	D_{max}
T1	103.41	117.93	0.000	0.014
T3	20.70	35.71	0.000	0.009
T4	71.15	300.50	0.0024	0.025
T6	17.08	500.37	0.000	0.007

run, ϵ can be larger than for the covering process.

Further, the covering MOPSO is compared with a simple MOPSO with unrestricted archive size. The simple MOPSO should be able to find a large set of non-dominated solutions after a high number of generations. The following parameters are selected for the simple MOPSO: 6000 generations, population size of 200, inertia weight 0.4 and turbulence factor of 0.1. The method is tested on the test function T1. The computational time for the simple MOPSO is 1779.61 seconds and the minimum and maximum distances between the non-dominated solutions are 0.000 and 0.1 respectively. These results show that not only the simple MOPSO takes much more time than the covering method (8 times), but also there is still a maximum gap of 0.1 between the non-dominated solutions.

V. CASE STUDY

For having a comparison with the previously proposed covering method using the Hybrid MOEA, a real world example on antenna design is studied here. It is a basic problem in antenna design to construct the shape or choose the *feeding* of the antenna to optimize the performance of the antenna. The fixed geometry of the antenna and the wave propagation of the fields generated by currents on the antenna are studied by [7] as a multi-objective optimization problem. The optimization problem is a 2-objective problem: maximize the *radiation efficiency* in a particular direction and minimize the power radiated into other directions. This problem can also be converted to a minimization problem as follows:

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

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

subject to the constraints

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

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

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 $\mathbb{C} \cong \Re^2$, this leads to a model with 12 free parameters. This antenna design problem is studied in [13].

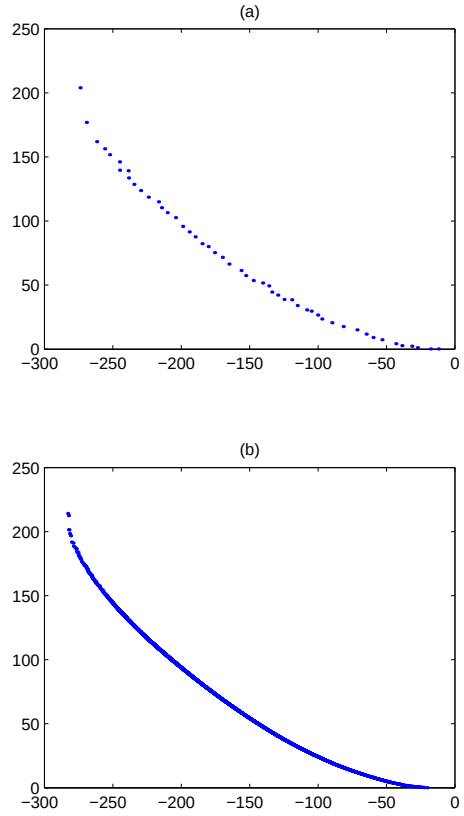


Fig. 8. Antenna design problem, Results of MOPSO (a) archive size: 50 (b) covered Pareto-optimal front (objective space)

The experiments are done by applying the covering MOPSO and Hybrid MOEA. The selected parameters are set as follows.

- Inertia weight: initial run: 0.75, covering: 0.5
- Turbulence factor: 0.1
- Population size: initial run: 300, covering: 500
- Number of generations: initial run: 500, covering: 1000
- Archive size: initial run: 50, covering: unrestricted

For the initial run, the inertia weight is selected 0.75, which is bigger than for the covering process. The reason is that the high value of inertia weight leads the solutions to high convergence and together with clustering (restricted archive) to a good diversity of solutions. But in the covering process, we need to explore the area around each non-dominated solution. Therefore, the inertia value should be decreased so that the particles may have the chance to search their local neighborhood areas. Figure 8 shows the results of the initial run of MOPSO and the covered Pareto-optimal front. In Figure 8 (b), the top left part of the approximated Pareto-front is not covered completely. This is because of the limited number of generations. The computational times for the initial run and the covering are 160 and 349 seconds respectively.

The Hybrid MOEA is tested on this example as follows. The SPEA method [14] is used as the MOEA. First the problem

is solved by the MOEA method with following parameters. Then the Static Recovering method is applied on it to cover the Pareto-optimal front.

- population size: 1000
- length of individual bitstrings: 13
- number of generation: 500
- archive size : 1000

Figure 9 shows the result of the Hybrid MOEA. The total running time is 20 minutes for the MOEA result and another 15 minutes for the recovering process. The computational time of the method is much higher than our new MOPSO technique. The selected population size is very high. This is needed in

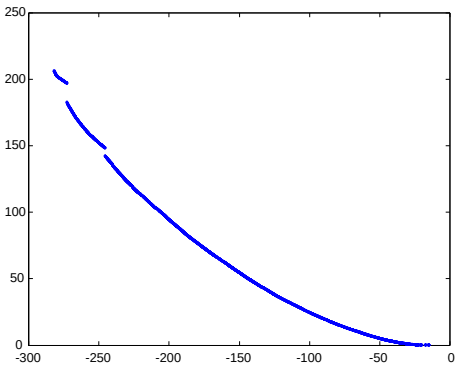


Fig. 9. Result of Hybrid MOEA using Static Recovering technique (objective space) [13]

order to obtain high convergence and diversity of solutions by MOEA. Therefore, the recovering method is able to cover the Pareto-front faster. The archive size is also very high. The existence of the restricted archive is necessary to obtain a good diversity of solutions. The high value of the archive size helps the recovering technique to operate faster to cover the front. For a lower amount of solutions, the Static Recovering needs more computational time to recover the Pareto-optimal front. As it can be observed in Figure 9, there are local improvements. Figure 10 shows the comparisons of selected areas from Figures 8 and 9. Here, the result of a MOEA method is also shown. Indeed, the Hybrid MOEA together with the recovering techniques brings more convergence to the results of the MOEA and therefore covering the Pareto-front is possible.

A. Comparison

Figure 10 shows the results of covered Pareto-optimal front by Hybrid MOEA in [13], MOEA and covered MOPSO. We can observe that for the selected part of the objective space, the MOPSO method can find solutions with even better convergence than the Hybrid MOEA. The quantitative comparison between the solutions are done by applying the Cmetric [14]:

- $C(\text{MOPSO},\ \text{Hybrid MOEA}) = 0.97$
- $C(\text{Hybrid MOEA},\ \text{MOPSO}) = 0.00$

The approximated Pareto-optimal solutions obtained from the covering MOPSO dominate 97% of the covered solutions by the Hybrid MOEA. It means that the covering MOPSO improves the convergence of the solutions, but the Hybrid MOEA makes local improvements around each non-dominated solution. However, it must be emphasized that the MOPSO is able to cover more solutions on the Pareto-optimal front if we run it for a larger number of generations. The minimum

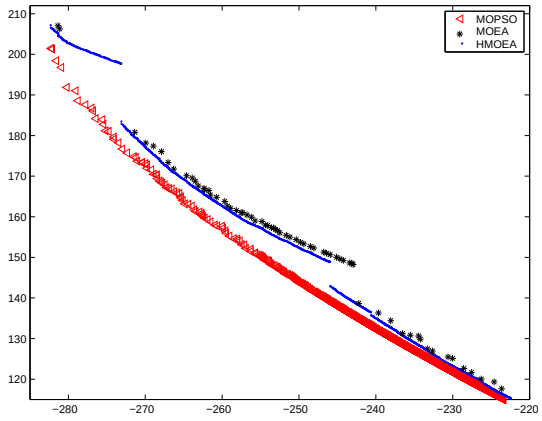


Fig. 10. Comparison of Hybrid MOEA (HMOEA), MOEA and MOPSO for the selected part of the objective space of the antenna design problem (objective space)

and maximum distances between the covered solutions by MOPSO are $D_{min} = 0.000$ and $D_{max} = 4.75$, respectively. The MOPSO covering process can be terminated by assigning a threshold on the maximum distance between the neighbor solutions.

Here, we conclude that the covering MOPSO can cover the approximated Pareto-optimal front much faster than the Hybrid MOEA, while improving the convergence of the solutions.

VI. Conclusion and Future Work

Covering the Pareto-optimal front in multi-objective optimization is studied in this paper. The proposed new method uses the property of moving particles in MOPSO and divides the population of the covering MOPSO into subswarms. The subswarms try to cover the gaps between the non-dominated solutions found in the initial run. The proposed covering method is tested on different test functions. The results show that the approximated Pareto-optimal front can be covered by the particles with high granularity.

It must be emphasized that the proposed method is not just applicable to 2-objective multi-objective optimization problems and can be used for any desired number of objectives. The covering method is also compared with the Hybrid MOEA, which is also a covering technique in MO. The Hybrid MOEA is a possible covering technique which uses a local search around the found solutions. The methods are compared through an example in antenna design. The covering MOPSO covers the approximated front not only is much less computational time, but also with better convergence than the Hybrid MOEA. Indeed, this application shows that the covering MOPSO improves the convergence of the solutions, where the other covering methods e.g., Hybrid MOEA, perform a

local search. Also, it must be emphasized that the Hybrid MOEA is applicable for low number of parameters, where the covering MOPSO can be used for problems with desired number of parameters. The use of ϵMOPSO can also help to win some computational time. This will be performed in future.

Acknowledgments

S. Mostaghim is grateful for stipend from the DFG sponsored Graduiertenkolleg "Application oriented modeling and development of algorithms".

REFERENCES

[1] C. A. Coello Coello and M. S. Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *IEEE Proceedings World Congress on Computational Intelligence*, pages 1051–1056, 2003.

[2] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[3] M. Dellnitz, O. Schtze, and T. Hestermeyer. Covering pareto sets by multilevel subdivision techniques. In *To appear in the Journal of Optimization, Theory and Applications*, 2003.

[4] J. E. Fieldsend and S. Singh. A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In *The 2002 U.K. Workshop on Computational Intelligence*, pages 34–44, 2002.

[5] J.E. Fieldsend, R.M. Everson, and S. Singh. Using unconstrained elite archives for multi-objective optimisation. In *IEEE Transactions on Evolutionary Computation*, 2002.

[6] X. Hu, R. Eberhart, and Y. Shi. Particle swarm with extended memory for multiobjective optimization. In *IEEE Swarm Intelligence Symposium*, pages 193–198, 2003.

[7] A. Jschke, J. Jahn, and A. Kirsch. A bicriterial optimization problem of antenna design. In *Computational Optimization and Applications*, volume 7, pages 261–276, Kluwer Academic Publishers, Netherlands, 1997.

[8] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Archiving with guaranteed convergence and diversity in multi-objective optimization. In *Genetic and Evolutionary Computation Conference (GECCO02)*, pages 439–447, 2002.

[9] S. Mostaghim and J. Teich. The role of e-dominance in multi-objective particle swarm optimization. In *Proc. CEC'03, the Congress on Evolutionary Computation*, Canberra, Australia, December 2003.

[10] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization. In *IEEE Swarm Intelligence Symposium*, pages 26–33, 2003.

[11] S. Mostaghim, J. Teich, and A. Tyagi. Comparison of data structures for storing pareto-sets in MOEAs. In *IEEE Proceedings World Congress on Computational Intelligence*, pages 843–849, 2002.

[12] G. Rudolph. On a Multi-Objective Evolutionary Algorithm and Its Convergence to the Pareto Set. In *Proceedings of the 5th IEEE Conference on Evolutionary Computation*, pages 511–516, IEEE Press, 1998.

[13] O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich. Covering pareto sets by multilevel evolutionary subdivision techniques. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 118–132, 2003.

[14] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Germany, Swiss Federal Institute of Technology (ETH) Zurich, 1999.

[15] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. In *EUROGEN 2001, Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, 2001.