Robust Multi-Objective Optimization in High Dimensional Spaces

André Sülflow, Nicole Drechsler, and Rolf Drechsler

Institute of Computer Science
University of Bremen
28359 Bremen, Germany
{suelflow,nd,drechsle}@informatik.uni-bremen.de

Abstract. [1]In most real world optimization problems several optimization goals have to be considered in parallel. For this reason, there has been a growing interest in Multi-Objective Optimization (MOO) in the past years. Several alternative approaches have been proposed to cope with the occurring problems, e.g. how to compare and rank the different elements. The available techniques produce very good results, but they have mainly been studied for problems of "low dimension", i.e. with less than 10 optimization objectives.
In this paper we study MOO for high dimensional spaces. We first review existing techniques and discuss them in our context. The pros and cons are pointed out. A new relation called *ϵ-Preferred* is presented that extends existing approaches and clearly outperforms these for high dimensions. Experimental results are presented for a very complex industrial scheduling problem, i.e. a utilization planning problem for a hospital. This problem is also well known as *nurse rostering*, and in our application has more than 20 optimization targets. It is solved using an evolutionary approach. The new algorithms based on relation *ϵ-Preferred* do not only yield better results regarding quality, but also enhances the robustness significantly.

1 Introduction

To solve complex optimization problems today, it is often not sufficient to only consider a single optimization criteria. In contrast, many real world problems have several – often contradicting – optimization goals. Thus, in the recent past several techniques for *Multi-Objective Optimization* (MOO) have been proposed.

One of the first approaches in this direction was the use of *Pareto-optimal elements*. This has been discussed in the context of *Evolutionary Algorithms* (EAs) in [1]. The goal is to determine elements from the *Pareto set*. To guide this search, there exist several alternative methods (see e.g. [2, 3]) where the core is a relation that allows to compare different elements. E.g. the relation *Dominate* proposed in [1] can be applied. These methods are well known and have been

[1] Copyright Springer-Verlag, LNCS 4403, pp. 715-726, available at http://www.springerlink.com

studied intensively. But so far these studies mainly consider problems with a small number of optimization criteria, e.g. in [2] comparisons for dimensions two or three are given.

For higher dimensional spaces there only exist a few studies (see e.g. [4–7]). As testcases scalable test functions proposed in [4] are considered. For example, in [3] it is reported that the number of individuals in the Pareto set, i.e. the non-dominated solutions, increase with the number of optimization objectives. Experiments have shown that for 20 objectives the percentage of solutions that cannot be distinguished using relation *Dominates* in random populations is nearly 100%. For this reason, new measures and relations have to be defined that help to automate and guide the optimization process.

In general for higher dimensions *weighted sums* or *aggregation* have been proposed, since they are easy to describe and, on a first sight, scale well. But for high dimensions these techniques reach their limits, since it is hard (or even impossible) to determine good weights or the fitness of the optimal solution is not known in advance, respectively.

In [8, 9] an alternative relation called *Preferred* (originally introduced as *Favour*) has been proposed and applied for five dimensions. Experiments have shown that *Preferred* clearly outperformed relation *Dominates* and an approach based on weighted sums. But in all cases described above the dimensions considered are rather small, i.e. less than 10. However, for complex optimization problems, where especially EAs are frequently used, often a higher number of dimensions occur. Of course, the standard algorithms can also be applied in the case of higher dimensions, but it will be shown in this paper by a detailed discussion and also by experimental studies for an industrial application that other techniques should be applied.

In this paper we first discuss the existing techniques and point out their main properties. Then, an experimental study shows the weaknesses of the above techniques for higher dimensions. For the experiments an industrial application where a very complex scheduling problem with many constraints occurs is considered. I.e. the *nurse rostering problem* [10], a well-known problem in mixed integer optimization, where a highly constraint schedule for employees in a hospital is generated. In this problem, 25 optimization goals are considered in parallel. As an additional difficulty, there are different types of constraints. Some can be seen as “hard constraints” that are enforced by state laws, while others are “soft constraints” that should be fulfilled as good as possible. It is demonstrated by experiments that for high dimensions the approach using relation *Preferred* outperforms traditional methods based on non-dominated sorting (relation *Dominates*). But relation *Preferred* is not robust for high dimensions and has to be extended accordingly. Therefore, we propose an extension of *Preferred* that also takes the relative difference over all dimensions into account. It considers environments of radius ϵ, where elements outside this region are “punished”. The new relation is called ϵ-*Preferred*. Experimental results show that the new approach results in higher quality and, additionally, gives very robust optimization results.

The paper is structured as follows: In Section 2 previous work is reviewed and properties of the different relations are discussed. An experimental study for a complex scheduling problem is presented in Section 3. This study clearly shows the weaknesses of the existing techniques. Our new approach including experimental evaluations is introduced in Section 4. Finally, in Section 5 the results are summarized and directions for future research are pointed out.

2 Preliminaries

To make the paper self-contained, a brief review of proposed relations for comparing MO solutions is given. In the second part the MOO methods used for the experiments are described.

2.1 Relations

A multi-objective optimization problem is defined as follows: Given a search space Ω, an evaluation function $F : \Omega \rightarrow \mathbb{R}^m$ is defined to calculate the fitness vector of size m: $F(A) : \forall A \in \Omega$. Then the optimization goal is to minimize (or maximize) the elements of $F(A)$. In the following we assume, without loss of generality, that F has to be minimized for all objectives. According to [1] it holds:

Definition 1. *Let* $A, B \in \Omega$. $A \prec_{dominates} B :\Leftrightarrow$
$\exists j : F_j(A) < F_j(B) \wedge \forall i \neq j : F_i(A) \leqslant F_i(B), 1 \leqslant i \leqslant m.$

Based on this, we can describe a *Pareto set* (*non-dominated set*) as χ: $\forall p \in \chi : \nexists q \in \chi : q \prec_{dominates} p$.

As can be seen from the definition above, if two elements $A, B \in \Omega$ are compared with relation *Dominates*, then A dominates B only if it is less or equal to B in all objectives and if it is better in at least one objective. Using relation *Dominates*, a set of elements can be classified into several levels of non-dominated solutions. Thus, first the non-dominated set is computed. Then, disregarding the non-dominated set, the next level of non-dominated elements is found. This is repeated, until all elements have been considered. The resulting procedure is called non-dominated sorting [3].

In comparison relation *Preferred* [8] respects the number of objectives in which A differs from B:

Definition 2. *Let* $A, B \in \Omega$. $A \prec_{preferred} B :\Leftrightarrow$
$|\{i : F_i(A) < F_i(B), 1 \leqslant i \leqslant m\}| > |\{j : F_j(B) < F_j(A), 1 \leqslant j \leqslant m\}|.$

A is then said to be *preferred* to B if A is better than B in a larger number of objectives. The Relation *Preferred* is not transitive. This means it is possible to have cycles in the relation graph of the elements of Ω.

Analogously to non-dominated sorting a set of elements, e.g. a population, can be grouped into several levels by using relation *Preferred* [8].

2.2 Methods

For our analysis three different methods are used. Based on relation *Dominates* are the methods `Dominates` [11] and `NSGA II` [12]. Based on relation *Preferred* the proposed strongly-connected components building algorithm `Preferred` is used [8].

The method `Dominates` is a part of the `Evolving Object Library` [11] used as backbone for our study. This method counts for each individual the number of individuals which are dominated. Thus, if the number is zero, then this individual is in the Pareto-front. The best rating is given to the individuals without any dominators. Then the elements with one dominator follow and so on. Thus, in contrast to non-dominated sorting [1], only the first Pareto-front is built and considered using the method of [11]. Note, that the "distribution" of the elements in the solution space is not taken into account. By this, it might happen that all elements from the same region are favoured, while other regions are not considered.

To avoid this concentration on a small part of the search space, the `NSGA II` algorithm has been proposed [12]. The idea of `NSGA II` is as follows: The individuals in a population are classified by non-dominated sorting [1]. Then the algorithm for computing the crowding distance is used to ensure that the Pareto-front is widely spread. This also helps to preserve a diversity in the set of possible solutions, e.g. in the population in the case of an EA.

The algorithm `Preferred` from [8] builds all strongly-connected components in the relation graph that result from the pairwise comparison of all individuals of the population. All individuals in the same component get the same fitness (ranking value). Then all components are hierarchically ordered, followed by an assignment of ranking values. For more details see [8].

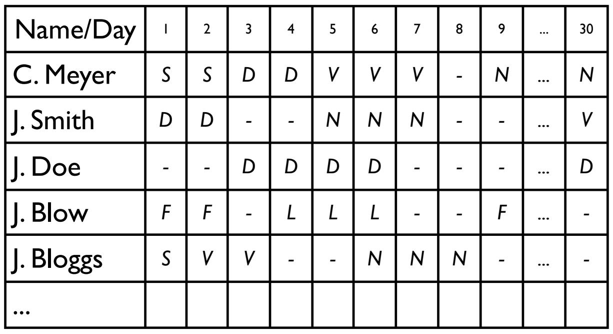
3 Application of Models

While previous methods and algorithms have been successfully applied in many fields ranging from graph problems to circuit design, all the studies have in common that MOO problems of low dimension have been considered, i.e. typically only three or four. The situation changes, if higher dimensions are the main focus. In this section we first introduce a very complex industrial scheduling problem, i.e. a utilization planning problem for a hospital. The problem is taken from a real-world environment of a hospital in Austria. An experimental study follows, using the techniques introduced in Section 2.2.

3.1 Utilization Planing Problem

The problem of utilization planing, i.e. the *nurse rostering problem* [10], is very complex and cannot be described here in all details. We briefly highlight the main aspects to give an idea of the underlying optimization problem.

The problem is to determine a schedule for the employees at a hospital. In the experiments schedules for ten persons for a planning period of 30 days have



Name/Day	1	2	3	4	5	6	7	8	9	...	30
C. Meyer	S	S	D	D	V	V	V	-	N	...	N
J. Smith	D	D	-	-	N	N	N	-	-	...	V
J. Doe	-	-	D	D	D	D	-	-	-	...	D
J. Blow	F	F	-	L	L	L	-	-	F	...	-
J. Bloggs	S	V	V	-	-	N	N	N	-	...	-
...											

Fig. 1. Example nurse rostering schedule

to be computed. The computation of the fitness can be roughly categorized in three main areas:

1. Rules resulting from ergonomics, e.g. having regular shifts
2. Restrictions by law, e.g. maximal hours of work per day or maximal working days per month
3. Rules of the nurse station, e.g. sufficient nurses per shift

Some of these constraints are “hard” in the sense that they have to be fulfilled, while others are “soft”, i.e. they improve the fitness, but a violation does not invalidate the schedule. Altogether 25 optimization objectives are influencing the fitness function. Each one might have a different influence, e.g. some are linear while others are exponential regarding the influence.

Example 1. To give a better understanding of the algorithm a sketch of a schedule is given in Figure 1. Depending on the grade of training the optimization algorithm assigns exactly one shift to a person per day. In this example all given shifts are marked with a letter. These letters have the following meaning: Day shift (D), Night shift (N), Late shift (L), Vacation (V), Free shift (F), Stand-by shift (S), No shift (-).

For more details see [10].

3.2 Implementation

The core of the optimization is directly based on a schedule similar to Figure 1. In the optimization algorithm one mutation and two recombination operators are used. They are applied with a probability of 50% for the mutation operator and 25% for each recombination operator. The mutation operator sets a legal shift for a randomly chosen person and day. Both recombination operators exchange a block of shifts, specified by a random choice of employees and days. Since the focus of this paper is not on the optimization technique, i.e. the EA approach,

Table 1. Fitness for generation 3000

Algorithm	$AVG_{\Psi,3000}$	$AVG_{\Psi,3000}$ in percent	σ_{3000}	$\sigma_{0..3000}$
Dominates	429805	100%	11%	7%
NSGA II	421840	98%	13%	7%
Preferred	196842	46%	67%	88%

the details are left out. In contrast, it is emphasized that the approach presented in this paper is applicable for other MOO techniques as well.

As metric to compare the results of the 25 dimensional optimization, a weighted sum approach, which reduces the fitness vector to one dimension, is used. The weighted sum metric is in general defined as follows: $\Psi : \mathbb{R}^m \to \mathbb{R}$ with $\Psi(A) = \sum_{i=1}^{m} w_i \cdot F_i(A)$. The justification of the weights results from the experience of an expert, several months of development in the area of nurse rostering and the given constraints. Note, that a lot of time and experience was necessary to adapt these weights.

To measure the influence of random seeds on the results, the random number generator has been initialized with 10 different values. But they were chosen as constants in each run. The results presented in Section 3.3 and 4.3 give the average value AVG_{Ψ} for these 10 runs. With the best weighted sum of generation g for random seed i denoted as $\Psi_{i,g}$, the average value $AVG_{\Psi,g} : \mathbb{R} \to \mathbb{R}$ of the ten runs $\Psi_{i,g} : 1 \leqslant i \leqslant 10$ is calculated as follows:

$$AVG_{\Psi,g} = \frac{\sum_{i=1}^{10} \Psi_{i,g}}{10}$$

Additionally to the average value, the standard deviation σ_g has been calculated in percent from $AVG_{\Psi,g}$ as follows:

$$\sigma_g = \sqrt{\frac{1}{10-1} \sum_{i=1}^{10} (\frac{\Psi_{i,g}}{AVG_{\Psi,g}} - 1)^2}$$

3.3 Experimental Evaluation

The experiments are based on the standard setting of the EA as applied in the industrial setting (see Section 3.2) . No fine-tuning has been done for any of the algorithms.

The results of all experiments are given for a population size of 10 and a run of 3000 generations. The final average fitness values are shown in Table 1 in column $AVG_{\Psi,3000}$ for the methods `Dominates`, `NSGA II` and `Preferred`, respectively. For comparison, in the next column $AVG_{\Psi,3000}$ is given as percentage normalized for `Dominates`. As can be seen, `Dominates` and `NSGA II` perform almost identical, while `Preferred` gives a reduction of more than 50%.

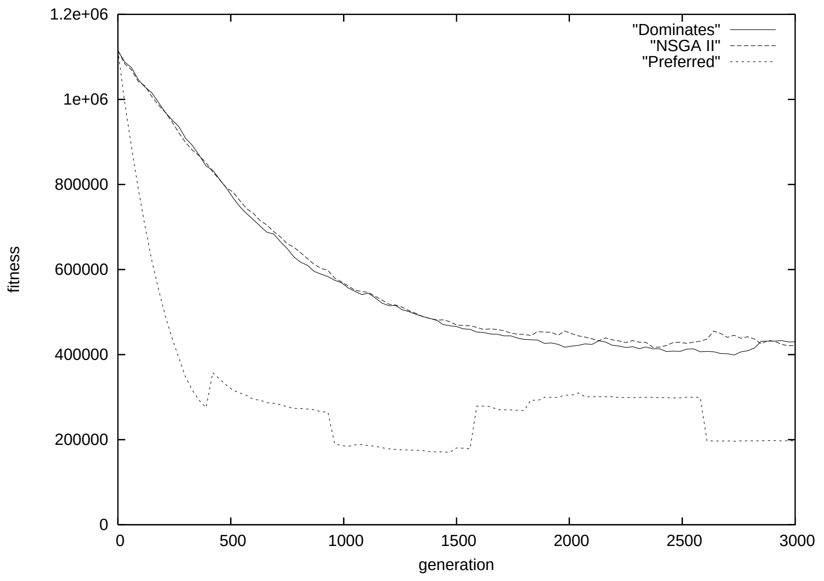


Fig. 2. Fitness for `Dominate`, `NSGA II` and `Preferred`

The next column gives the standard deviation. Here `Dominates` and `NSGA II` have values from 10-15%, while `Preferred` has an "unstable" value of over 60%. The result gets even worse if the average value of the standard deviation from the first to the last generation is considered (see last column). The results of the complete run are shown in Figure 2.

In summary the experimental study showed:

- The performance of `Preferred` was significantly better than that of `Dominate` and `NSGA II` for high dimensional MOO.
- Measuring the standard deviation showed that `Preferred` is not very robust, i.e. the algorithms should generate good solutions for each random seed.

3.4 Discussion

In this section the experiments are discussed to explain the observed behavior.

A more detailed analysis of the method `Preferred` showed the reason for the behavior: Relation *Preferred* favours a solution A, if it is better in more dimensions than a solution B. But a "problem" of relation *Preferred* is that the other dimensions, i.e. those where B is better than A, are not considered at all. It might happen that the negative effect of these dimensions is very strong and, as a result "jumps" occur in the weighted sum metric. This effect can, for example, be observed in Figure 2 between generation 450 and 500 or 1600 and 2600.

The main reasons for the weak performance regarding the fitness of relation *Dominates* can be explained as follows: Due to the high number of dimensions it rarely happens that an element is better in all dimensions. In fact, in the EA run 93% of the solutions were not comparable to each other using `Dominates`. Only solutions which are better or equal in all dimensions were distinguishable from other solutions. This also applies to `NSGA II` and is a reason, why the relation *Dominates* based methods `Dominates` and `NSGA II` could not guide to good solutions.

In [3] this problem has been described for up to 20 objectives. There it has been suggested to use a larger population size or to use modified fitness assigning techniques. Both approaches will not work in our application. Increasing the population size means higher run times. Furthermore, this might reduce the number of non-comparable elements. To modify the fitness assignment might help to preserve a good Pareto-front, but not to improve the path to an optimum.

So approaches based on *Preferred* should be used for guiding the search in high dimensional spaces, because it is possible to compare solutions, which are not comparable with relation *Dominates*. *Preferred* uses the number of better objectives as a criterion for the comparison. But as a result the technique suffers from unstable behavior as explained above.

4 Robust MOO

To improve the robustness of the approach based on *Preferred*, an extension called ϵ-*Preferred* is introduced in the following. Before we give a formal definition, the main idea is briefly sketched.

4.1 Overall Idea

One principle in multi-objective optimization is to model the criteria of human decision making. In our application it has been observed that a human planner rejects solutions, if specific limits of the objectives quality are not satisfied. Hence, the idea is to define fitness limits for each dimension. The resulting relation is called ϵ-*Preferred*, where an ϵ-value is defined for each optimization objective. A solution is rejected if it exceeds one or more ϵ-limits.

For a motivation of our idea look at the following example:

Example 2. Consider solutions A and B and a fitness function F for a minimization problem. Let $F(A) = (1, 1, 100)$ and $F(B) = (5, 5, 5)$. Then relation *Preferred* would hold: $A \prec_{preferred} B$

But, dependent on the application considered, solution A is not a satisfying solution, because the third component does not fulfill the planners expectations.

Table 2. Fitness for generation 3000

Algorithm	$AVG_{\Psi,3000}$	$AVG_{\Psi,3000}$ in percent	σ_{3000}	$\sigma_{0..3000}$
Dominates	429805	100%	11%	7%
Preferred	196842	46%	67%	88%
ϵ-Preferred	116594	27%	10%	6,6%

To overcome this problem, a maximum environment ϵ_i is set for each optimization objective $1 \leqslant i \leqslant m$.

As can be seen in Example 2, if we set $\epsilon_3 = 50$, solution A becomes worse than B, because the third component of solution A does not satisfy the given quality limits.

4.2 Relation *ϵ-Preferred*

In this section an extension of *Preferred*, denoted as ϵ-*Preferred*, is formally introduced.

Definition 3. *Let $A, B \in \Omega$ and ϵ_i, $1 \leqslant i \leqslant m$*

$$A \prec_{\epsilon-exceed} B \Leftrightarrow$$
$$|\{i : F_i(A) < F_i(B) \wedge |F_i(A) - F_i(B)| > \epsilon_i\}|$$
$$> |\{j : F_j(A) > F_j(B) \wedge |F_j(A) - F_j(B)| > \epsilon_j\}|.$$

The relation ϵ-*exceed* counts how often a solution exceeds the given limits ϵ_i. Then solution A is better than solution B with respect to the limits ϵ_i, if A has less exceedings than B.

Using ϵ-*exceed* the extension ϵ-*Preferred* is defined as follows:

Definition 4. *Given two solutions $A, B \in \Omega$,*
$$A \prec_{\epsilon-preferred} B \Leftrightarrow A \prec_{\epsilon-exceed} B \vee (B \not\prec_{\epsilon-exceed} A \wedge A \prec_{preferred} B)$$

First it is counted how often a solution exceeds the ϵ-limits and the better solution is determined. If both solutions are in the given range, *Preferred* is used for comparison.

By building the relation graph with the newly proposed relation ϵ-*Preferred*, it is possible to create cycles, as relation *Preferred* does, too. For this reason we use the same strongly-connected components building algorithm as suggested in [8].

4.3 Experimental Evaluation

The experimental setting is the same as described in Section 3.3, i.e. we studied the run for 3000 generations. The underlying EA was identical for all approaches and only the MOO relation was changed.

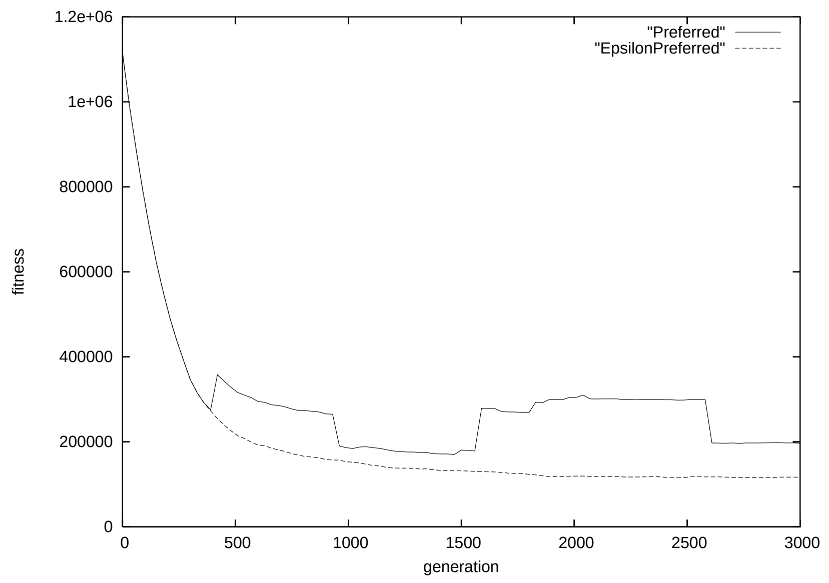


Fig. 3. Fitness comparison for *Preferred* and ϵ-*Preferred*

In the experiments for all dimensions the parameter ϵ is set to 10000. This choice is very conservative and only a weak restriction for the algorithm. But, it can be avoided that the algorithm explores regions of the search space that are not of the planners interest.

The same information as above is now given for ϵ-`Preferred` in Table 2 and Figure 3. Since `Dominates` and `NSGA II` behave almost the same, only `Dominates` is shown in the table. Compared to `Preferred`, the results are further improved by over 30% by using relation ϵ-*Preferred*. But the even more remarkable observation is the robustness of the technique. While it is significantly better than `Preferred`, it is even better than `Dominates`. This can be seen very well in Figure 3, where the fitness over 3000 generations is shown. There are no “jumps” any more.

Influence of Epsilon Values For our experiments above large epsilon values have been used. Only in cases where a solution was “very bad” in one or more dimension it was rejected. In this section we briefly discuss the influence of alternative choices. The experiments are summarized in Table 3. By this, directions for future work are pointed out (see also next section).

Table 3. Fitness cut-out of the 18 most important dimensions

Algorithm	$AVG_{\Psi,3000}$	$AVG_{\Psi,3000}$ in percent	σ_{3000}	$\sigma_{0..3000}$
ϵ-Preferred	45188	100%	16%	11%
ϵ-Preferred-2500	38998	86%	15%	13%
ϵ-Preferred-1000	33018	73%	8%	10%

In a first run, denoted as *ϵ-Preferred-2500*, the epsilon values of 18 out of the 25 optimization objectives were reduced from 10000 to 2500. The 18 dimensions were the ones that the human expert considered "most important". In a second run (*ϵ-Preferred-1000*) we additionally reduced two out of the 18 dimensions to the value 1000 to consider those as "very important" criterias. As can be seen even by these first experiments, the quality could be further improved, while obtaining the same robustness.

If the value of ϵ was too low (e.g. close to zero) for one dimension, then each solution with small deterioration in this direction was immediately rejected without respecting possible improvements in other dimensions. The same problem occurred in the case of relation *Dominates*.

In summary, based on relation *ϵ-Preferred* the quality measured by the fitness value could be significantly improved, while robustness was obtained at the same time.

5 Conclusions and Future Work

With more complex applications, MOO is becoming a more important topic. To compare two solutions relations have to be defined. These have mainly been evaluated on problems of low dimension, i.e. with up to 10 optimization goals.

In this paper a complex industrial scheduling problem has been investigated. The problem has more than 20 optimization goals with different "levels" of importance. Previously proposed relations have been evaluated and discussed in the context of high dimensional MOO. It turned out that the methods either suffer from low quality or low robustness.

A new relation called *ϵ-Preferred* has been suggested and experimentally studied. It was shown that very high quality results could be obtained, i.e. an improvement of more than 30% so far, and in addition the robustness could be improved.

It is focus of current work to develop automatic techniques for determining the epsilon values automatically. In this context also the dynamic reduction during the optimization run will be investigated. First experiments showed that there is still significant potential for improvement.

Acknowledgment

The authors like to thank Peter Bäume from S2-Engineering, Bremen, for support and many helpful discussions.

References

1. D.E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addision-Wesley Publisher Company, Inc., 1989.
2. E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Trans. on Evolutionary Comp.*, 3(4):257–271, 1999.
3. K. Deb. *Multi-objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, New York, 2001.
4. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical Report 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.
5. V. R. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *EMO 2003*, volume 2632 of *Lecture Notes in Computer Science*, pages 376–390, 2003.
6. K. Deb and S. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *GECCO '06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 635–642, 2006.
7. K. Deb, S. Chaudhuri, and K. Miettinen. Towards estimating nadir objective vector using evolutionary approaches. In *GECCO '06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 643–650, 2006.
8. N. Drechsler, R. Drechsler, and B. Becker. Multi-objective optimisation based on relation favour. In *Int'l Conference on Evolutionary Multi-Criterion Optimization*, pages 154–166, 2001.
9. F. Schmiedle, N. Drechsler, D. Große, and R. Drechsler. Priorities in multi-objective optimization for genetic programming. In *GECCO '01: Proceedings of the 6th Annual Conference on Genetic and Evolutionary Computation*, pages 129–136, 2001.
10. E.K. Burke, P. De Causmaecker, G.V. Berghe, and H.V. Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7:441–499, 2004.
11. M. Keijzer, J.J. Merelo, G. Romero, and M. Schoenhauer. Evolving objects: a general purpose evolutionary computation library. *Artificial Evolution*, pages 321–244, 2001.
12. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Comp.*, 6:182–197, 2000.