Incorporating User Preferences in MOEA/D through the Coevolution of Weights

Martin Pilát
Charles University in Prague
Faculty of Mathematics and Physics
Malostranské náměstí 25
118 00 Prague, Czech Republic
Martin.Pilat@mff.cuni.cz

Roman Neruda
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 271/2
182 07 Prague, Czech Republic
roman@cs.cas.cz

ABSTRACT

The resulting set of solutions obtained by MOEA/D depends on the weights used in the decomposition. In this work, we use this feature to incorporate user preferences into the search. We use co-evolutionary approach to change the weights adaptively during the run of the algorithm. After the user specifies their preferences by assigning binary preference values to the individuals, the co-evolutionary step improves the distribution of weights by creating new (offspring) weights and selecting those that better match the user preferences. The algorithm is tested on a set of benchmark functions with a set of different user preferences.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*heuristic methods*; G.1.6 [**Mathematics of Computing**]: Optimization—*global optimization*

Keywords

evolutionary algorithms, multi-objective optimization, coevolution, MOEA/D

1. INTRODUCTION

Multi-objective evolutionary algorithms have gained a lot of attention in the recent decades and they are considered to be among the best multi-objective optimizers. Currently, decomposition-based multi-objective algorithms [17, 3] are becoming more popular, mainly because they outperform the more traditional domination based algorithms when solving many-objective problems. The decomposition-based algorithm transform the multi-objective problem into a set of single-objective problems by using a set of weights or reference vectors. The single-objective subproblems are then solved all at once.

GECCO '15, July 11 - 15, 2015, Madrid, Spain
© 2015 ACM. ISBN 978-1-4503-3472-3/15/07...$15.00
DOI: http://dx.doi.org/10.1145/2739480.2754801

The weights used by the algorithm directly affect the resulting non-dominated set. Thus, it should be possible to incorporate user preferences into the search by means of altering the weights. In this work, we take this approach and show that co-evolution of weights can be used as a mean to incorporate user preferences into the MOEA/D algorithm [17].

The way the user has to use to specify the preferences can affect the usability of the approach. If the user has to assign specific preference values to each individual in the population it is much more difficult than assigning only binary values (whether the individual is preferred or not). The later case is also more general and allows for interactive evolution – the user is presented the set of solutions found so far and selects those they like. The algorithm uses this selection to guide the search towards the preferred area of the search space. In the present paper, we propose an approach which can deal with such a binary preferences, thus generalizing the ways in which preferences can be specified.

We consider the multi-objective optimization problem defined as the tuple $\langle D, O, f, g\rangle$, where D is the decision space, O is the objective space, usually $O \subseteq \mathbb{R}$, $f = \{f_1, \dots, f_m\}$ is a set of m objective functions $f_i : D \to O$ which should be minimized, and $g = \{g_1, \dots, g_k\}$ represents a set of k constraints $g_i : D \to \mathbb{R}$ with the feasible set being defined as $g_i(x) \leq 0$. The goal of multi-objective optimization is to find an approximation of the Pareto set, i.e. such set of feasible solutions from the decision space that there is no other feasible solution which would be better in all objectives than some solution from the Pareto set. We say that the approximation is well-converged, if the found solutions are close to the Pareto set, and we say that it is well-spread if for each solution in the Pareto set, there is a close solution in the set returned by the algorithm.

The main goal of this work is to demonstrate a co-evolutionary approach to incorporating of binary user preferences into MOEA/D. We describe an extension of MOEA/D with the co-evolution of weights – cwMOEA/D – and illustrate its performance on a set of benchmark problems with different types of Pareto fronts and with two different types of user preferences. In this work, we also investigate, how the changing of the weights affects the convergence speed of the MOEA/D algorithm and how it affects the number of individuals found in the preferred region of the Pareto front. We also discuss how the proposed algorithm could be extended to provide even better results.

In the next section, we describe the work most related to this contribution in more detail, Section 3 is devoted to the description of the algorithm, in Section 4 we describe the experiments and discuss the performance of the algorithm and, finally, in Section 5 we provide some directions for future research.

2. RELATED WORK

The Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [17] is based on the idea of the decomposition of the multi-objective problem into a set of single-objective subproblems. The decomposition is based on a set of weights each of which corresponds to a single subproblem. Very often, the so called Tchebycheff decomposition is used. Given a weight vector $\lambda = \langle\lambda_1, \ldots, \lambda_n\rangle$, the corresponding subproblem is defined as

$$\text{minimize } g^{te}(x|\lambda, z^*) = \max_{1\leq i\leq m} \lambda_i|f_i(x) - z_i^*|,$$

where z^* is a reference point chosen as the minimum of objective function values found during the evolution. The main advantage of this approach is that it works regardless of the shape of the Pareto front, while other decomposition approaches (like the weighted sum approach) only work for convex Pareto fronts.

Apart from being used for the decomposition, the weights are also used to define neighborhoods of the subproblems. These subproblems are then solved all at once. In each generation, a new individual is generated and evaluated using its own set of weights. It is compared to its parent and in case the offspring is better it replaces the parent. Moreover, it is also compared to other individuals in its neighborhood and is allowed to replace some of them.

Different sets of weights in MOEA/D lead to different resulting sets of solutions found by the algorithm and therefore, it is crucial to set the weights in such a way to provide as good results as possible. Some research has been already done in this direction. Jiang *et al.* [9], for example, proposed a method for symmetric Pareto fronts, which aims at producing such weight vectors, that the hyper-volume of the resulting non-dominated set is maximized. They describe the shape of the Pareto front as $f_1^p + f_2^p = 1$, for some p estimated from the current non-dominated set and seek the optimal set of intersection points by the Simplex Method.

Qi *et al.* [14] created a method for sampling the weight vectors and obtained better spread of solutions. Moreover, the vectors are adjusted later in the process of evolution to obtain even better distribution of the solutions. The adjustment is again based on the geometrical properties of the current non-dominated set.

A slightly different approach to decomposition is used in NSGA-III [3]. The algorithm uses reference points instead of weight vectors to guide the search. The reference points are used inside a secondary sorting criterion. The individuals from the same non-dominated set which are closer to the lines defined by the origin and a reference point are preferred. In an extension of NSGA-III for constrained optimization [8], reference points are removed from the parts of search space which do not satisfy the constraints, and new reference points are generated in those parts of search space which do. The reference points are generated in such a way, to provide equal spread of solutions on the whole Pareto front.

Although the goal of most multi-objective algorithms is to provide the approximation of the whole Pareto front, in practice only an interesting part of the front is often required. Thus, it is important, to provide a way for the user (decision maker) to supply their preferences to the algorithm. If the algorithm is able to respect these preferences, it can focus on interesting parts of the Pareto front and use the computational resources more effectively (e.g. provide more solutions in the preferred area). There have been many attempts to integrate user preferences into multi-objective evolutionary algorithms. Branke [1] in his work describes many different ways how to provide preference information to multi-objective evolutionary algorithm. Among these are: the scaling of objectives, additional constraints, providing a reference point, and limiting the potential trade-offs. Each of these approaches can be useful in different situations, depending on which type of preferences is more natural in the given case.

More specifically, Molina *et al.* [12] created an algorithm based on g-dominance. The algorithm uses a reference point and prefers individuals which either dominated the reference point or are dominated by the point. This leads to the preference of an interesting part of the Pareto front.

Mohammadi *et al.* [11] proposed the R-MEAD2 algorithm. Similarly, to MOEA/D, this algorithm is also based on decomposition, however, R-MEAD2 is guided by a user specified reference point and the goal is to provide Pareto optimal solutions close to this reference point. To this end, the weights are adjusted during the evolution.

Another group of algorithms is based on interactive evolution, such algorithms use the feedback of a decision maker to create a value function which models the decision maker's preferences. For example, Deb and Kumar [4] proposed a technique based on Light Beam Search combined with evolutionary algorithm. In this technique, the decision maker is asked to provide local preferential information, which is used to build an outranking relation that is in turn used to reflect the decision makers preference during selection in the evolutionary algorithm.

Sinha *et al.* [15] proposed a technique which, apart from incorporating the preferences to the search also tries to minimize the number of queries to the decision maker. The algorithm queries the decision maker only after "significant progress is made". The progress is measured by the distance between an ideal point and the points in the current population. The decision maker is asked to choose the best point from a set of presented points.

In the next section, we propose a novel framework for the incorporation of user preferences - cwMOEA/D. Although, the cwMOEA/D algorithm is based on MOEA/D and is based on the changing of the weights used in the algorithm, and therefore it cannot be easily used in conjunction with algorithms which do not use decomposition (like the traditional NSGA-II [5]), it provides a way to incorporate most of the above described preference-based approaches into the popular and effective MOEA/D algorithm. Compared to the above mentioned algorithms, the new framework is more general – it expects only binary preference, i.e. each point is either preferred or not. It is trivial to implement most of the reference point based approaches (like the g-dominance) in cwMOEA/D. The interactive approaches, which aim at learning a value function, can also be implemented in the proposed framework in a simple way – after the points are evaluated with a given approach, they can be ranked some

Algorithm 1 cwMOEAD/D

```
Require: $n$: population size
Require: $\delta$: probability of neighborhood operators
 1: $\lambda \leftarrow$ InitUniformWeights()
 2: InitNeighborhood($\lambda$)
 3: $P \leftarrow$ InitPopulation();
 4: $z^* \leftarrow$ InitIdealPoint($P$)
 5: while termination criterion not met do
 6:     Par $\leftarrow \emptyset$                          $\triangleright$ saved parent population
 7:     Off $\leftarrow \emptyset$                          $\triangleright$ saved offspring population
 8:     for all parent $\in P$ do                          $\triangleright$ in random order
 9:         $r \leftarrow$ RandomNumber(0,1)
10:         if $r \leq \delta$ then                        $\triangleright$ as in MOEA/D
11:             child $\leftarrow$ PerformNeighborhoodOperators()
12:         else                                           $\triangleright$ as in MOEA/D
13:             child $\leftarrow$ PerformGlobalOperators()
14:         end if
15:         Evaluate(child)
16:         Par $\leftarrow$ Par $\cup$ {parent}           $\triangleright$ save parent
17:         Off $\leftarrow$ Off $\cup$ {child}            $\triangleright$ save offspring
18:         $z^* \leftarrow$ UpdateReference(child)        $\triangleright$ as in MOEA/D
19:         UpdateProblem(child)                           $\triangleright$ as in MOEA/D
20:     end for
21:     $\lambda \leftarrow$ UpdateWeights($\lambda$, Off $\cup$ Par)
22: end while
```

Algorithm 2 UpdateWeights(λ,P)

```
 1: pref $\leftarrow$ EvaluateUserPreference($P$)   $\triangleright$ provided by user
 2: childWeights $\leftarrow$ MutateWeights($\lambda$, pref)
 3: rPref $\leftarrow$ RefinePreference(pref, $P$)
 4: for $i \leftarrow 1$ to Length($\lambda$) do
 5:     OldIndex $\leftarrow$ AssingIndividual($\lambda[i]$,$P$)
 6:     NewIndex $\leftarrow$ AssingIndividual(childWeights[$i$],$P$)
 7:     if rPref[NewIdx] < rPref[oldIdx] then
 8:         $\lambda[i] \leftarrow$ childWeights[$i$]
 9:     end if
10: end for
11: return $\lambda$
```

and a number of the best may be flagged as preferred points, the rest as not-preferred.

3. ALGORITHM DESCRIPTION

In this work, we incorporate user preference handling into the original MOEA/D algorithm. In order to do so, we add a co-evolutionary step. In this step, the weights used in the decomposition are adjusted in such a way that the algorithm looks for solutions more preferred by the user. The user needs to specify their preference in a form of function which evaluates the solutions. This function shall return a negative number for preferred solutions and a positive number otherwise. The algorithm does not depend on the particular value, but only on the sign. This makes specifying user preferences easier, and, in fact, the same approach can be used even in the interactive setting, where the user selects preferred solutions as they are offered to them by the algorithm. The new algorithm – cwMOEA/D – is described in more detail in the rest of this section.

The main loop of the algorithm (cf. Algorithm 1) closely matches the main loop of MOEA/D [17]. There are only two additions: during each iteration, the parents and new

Algorithm 3 MutateWeights(λ,pref)

```
Require: $\sigma_p$: std. dev. for preferred individuals
Require: $\sigma_n$: std. dev. for not-preferred individuals
 1: childWeights $\leftarrow \emptyset$
 2: for $i \leftarrow 1$ to Length($\lambda$) do
 3:     if pref[$i$] < 0 then                 $\triangleright$ preferred individual
 4:         childWeights[$i$] $\leftarrow$ GaussianMutation($\lambda[i]$,$\sigma_p$)
 5:     else
 6:         childWeights[$i$] $\leftarrow$ GaussianMutation($\lambda[i]$,$\sigma_n$)
 7:     end if
 8: end for
 9: return childWeights
```

Algorithm 4 RefinePreference(pref, P)

```
Require: $m$: the number of closest individuals in Eq. 1
 1: rPref $\leftarrow \emptyset$
 2: pInds $\leftarrow \{P[i] | \text{pref[i]} < 0\}$        $\triangleright$ preferred individuals
 3: for $i \leftarrow 1$ to Length($P$) do
 4:     if pref[$i$] > 0 then
 5:         rPref[$i$] $\leftarrow \min_{p \in \text{pInds}} |p, P[i]|$
 6:     else             $\triangleright$ average distance to $m$ nearest neighbors
 7:         rPref[$i$] $\leftarrow$ AvgDstMNN(pInds$\setminus\{P[i]\}$, $P[i]$, $m$)
 8:     end if
 9: end for
10: return rPref
```

offspring are saved (lines 6-7 and 16-17), in order to have more evaluated individuals for the selection of weights; and after each MOEA/D iteration, the adaptation of weights is performed (line 21).

3.1 User Preference Specification

We assume, the user provides their preference as a function, $p : D \rightarrow \{-1; 1\}$, where D is the decision space of the problem at hand. This does not in any way limit the preferences to the decision space. The preference function can first evaluate the objective functions and use them to compute the preference later. In fact, the algorithm uses the preference function after all the individuals have already been evaluated, so the objective values are available for all individuals and do not be re-evaluated again. We use the objective space D in the definition of preference function only in order to make the function more general.

Such a general specification of user preference leads to an important advantage – it can be used to express most of the ways, how preferences can be specified. For example, if the user wants to limit the trade-off between the objectives such, that

$$l \leq \frac{f_1(x)}{f_2(x)} \leq h,$$

for some real bounds h and l, they can use the following preference function:

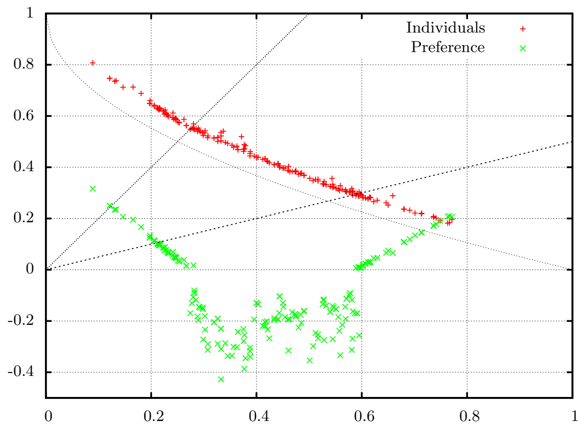
$$p(x) = \begin{cases} -1 & \text{if } l \leq \frac{f_1(x)}{f_2(x)} \leq h; \\ 1 & \text{otherwise.} \end{cases}$$

The disadvantage of such a preference function is that it provides only a binary information. Such information is not very suitable for use with evolutionary algorithms, and therefore we first refine the preference (cf. Algorithm 4). Let $P_p = \{x \in P | p(x) \leq 0\}$ is the set of preferred individuals

Algorithm 5 AssingIndividual(λ,P)

```
1: idx ← $\arg\min_{i \in \{1,\ldots,\text{Length}(P)\}}$ DecompFitness($P[i]$,$\lambda$)
2: return idx
```

Figure 1: Example of refined user preference. The user prefers individuals which lie between the two dashed lines. The red points correspond to individuals, the green points to their refined preference (the distance was multiplied by 5 for negative individuals to make the differences more visible).



from population P and $P_n = \{x \in P | p(x) > 0\}$ is the rest of individuals. Now, the refined preference function

$$p^*(x) = \begin{cases} |x, P_p| & x \in P_n \\ -|x, P_p|_m & x \in P_p \end{cases} \quad (1)$$

where $|x, P_p|$ is the distance of individual x to the closest preferred individual, and $|x, P_p|_m$ is the average distance of x to the m closest preferred individuals (except x itself). Thus, the refined preference p^* is positive, if the original preference is positive and it is negative, if the original is negative. Moreover, not-preferred individuals closer to preferred individuals are preferred over those further from the preferred ones. For the preferred individuals, the refined preference favors those, which are further from other preferred individuals. An example of such a refined preference is presented in Figure 1. The figure was obtained by running the cwMOEA/D algorithm (further described below) on the ZDT1 [18] problem for 20,000 function evaluations. The average distance to the five closest individuals was used to compute the refined preference for the preferred individuals.

3.2 Co-evolution of Weights

The algorithm aims at updating the weights in such a way, that they better match the user preferences specified by the preference function. To this end, a co-evolutionary step is added into the MOEA/D algorithm (cf. Algorithm 2). After each iteration of the algorithm, new weights are generated by means of mutation (line 2). We use simple Gaussian mutation with different standard deviation for preferred and not-preferred individuals (cf. Algorithm 3). After the mutation, the user preference is refined (line 3 of Alg. 2), and the selection of the weights is performed (lines 4-10).

The selection always compares each weight with its offspring (the weight obtained by mutation from the parent weight), this ensures that the weights do not change much and, mostly, that the neighborhoods in MOEA/D change only slowly.

We use the refined user preference function to select the weight. The only problem is, that the function is defined over the individuals in the population and not the weights. Therefore, we need to match the weights with the individuals. To this end, we define the *assigned individual* x to weight w as

$$x = \arg\min_{x \in P} g(x|w),$$

where P is the population and $g(x|w)$ is the fitness of the individual x given the weight w (cf. Algorithm 5). Thus, each weight is assigned to the individual, for which it provides the best (lowest) fitness. Once the individuals have been assigned, both to the parent and to the offspring weight, the preferences of the assigned individuals are compared, and the weight which corresponds to the individual with lower preference is selected. Note that we do not need to compute the assignment in the mutation of the weights (which also depend on the user preference), as MOEA/D keeps the individuals matched to the weights of the subproblem the individuals optimize. However, we need to compute the assignment in the selection step, as the newly generated weight may be far from the original one, and thus may better correspond to a different individual.

4. EXPERIMENTS AND DISCUSSION

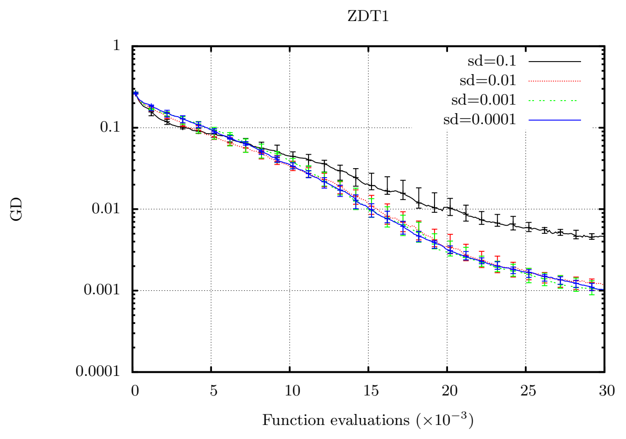
We have implemented the algorithm in the jMetal framework [6] and the source codes are available as supplementary materials. In all the experiments presented in this section, we use the population size of 100 individuals. The offspring are created using a differential crossover operator [13] and polynomial mutation [2]. The parameter δ which controls the percentage of genetic operations performed on the neighborhood and globally was set to $\delta = 0.9$ (i.e. 90 percent of the operations are performed in neighborhoods). The size of the neighborhood was set to 20, and each offspring can replace at most 2 other individuals in its neighborhood. These are the default settings from the jMetal framework and correspond to the settings used in [10]. The same settings were also used for MOEA/D in cases, where cwMOEA/D is compared to MOEA/D. After preliminary experiments, we have set the number of closest individuals used in the preference refinement procedure of preferred individuals to $m = 5$. This value provides good compromise, as the refined value is local, yet not too noisy. The parameters σ_p and σ_n (the standard deviations for the mutations of weights for preferred and not-preferred individuals respectively) were set to $\sigma_p = 0.001$ and $\sigma_n = 0.01$. The reasons for this setting are detailed in the next section.

4.1 The Effect of the Mutation of Weights

Before we show, how the algorithm works for different user preferences, we first investigate the setting of the two standard deviations for the mutation of weights and we provide reasons for using different standard deviation for preferred and not-preferred individuals.

To this end, we run the algorithm on the ZDT1 [18], WFG1, WFG2, and WFG4 [7] problems and report the generational distance (GD) [16] of the population with respect

Figure 2: Comparison of convergence speed for different standard deviation in the mutation of the weights. Median (line) and first and third quartile (error-bars) from 15 independent runs.



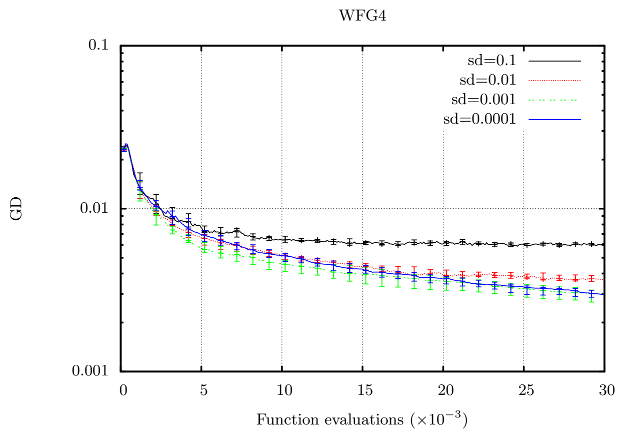
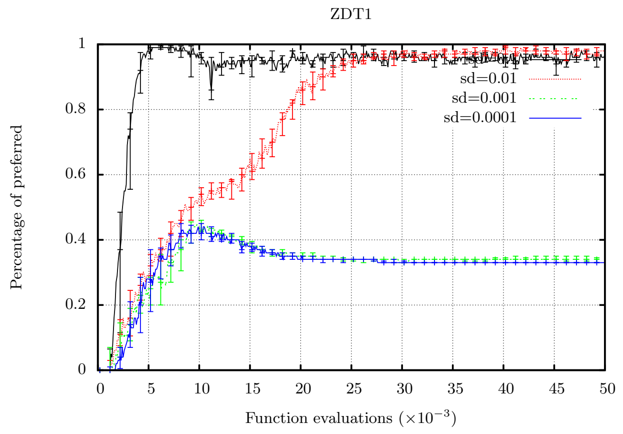
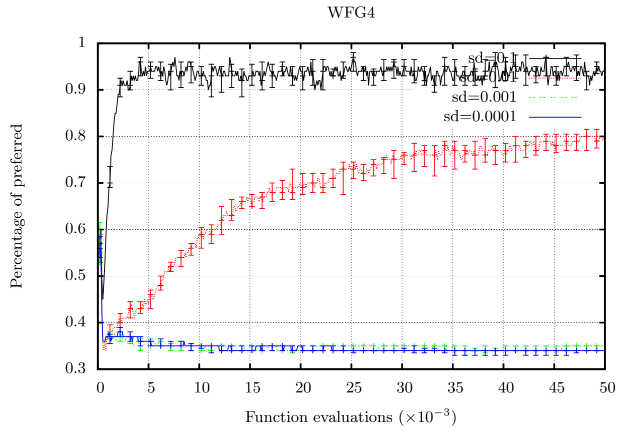


Figure 3: Comparison of the number of preferred individuals for different standard deviation in the mutation of the weights. Median (line) and first and third quartile (error-bars) from 15 independent runs.





to a reference set. The generational distance measures the average distance of each individual in the population to the closest individual in the reference set, as such, the indicator only measures the convergence of the population, but does not in any way reflect the spread. Moreover, we also count the ratio of preferred individuals among the individuals in the population.

The results for the dependence of the generational distance after a given number of evaluations on the standard deviation is presented in Figure 2. We can see, that with higher standard deviation, the convergence is slower. The difference is most pronounced for standard deviation equal to 0.1. This is not surprising, as the larger standard deviation means larger changes in the weights and it is difficult for the algorithm to react to such a dynamic environment.

Figure 3 on the other hand shows the percentage of preferred individuals in the population. We can see that for small standard deviations, the weights are not sufficiently adjusted to react to the user preference. For ZDT1, standard deviation of at least 0.01 is required to ensure most of the individuals are preferred. For WFG3, this number seems to be even larger.

The previous experiments show that for fast convergence we need small standard deviations in the weight mutation, and for large number of preferred individuals we need large values of standard deviation. This is the main reason, we use different standard deviation for preferred and not-preferred individuals. In the rest of the experiments, we use standard deviation of 0.01 for not-preferred individuals and of 0.001 for the preferred ones.

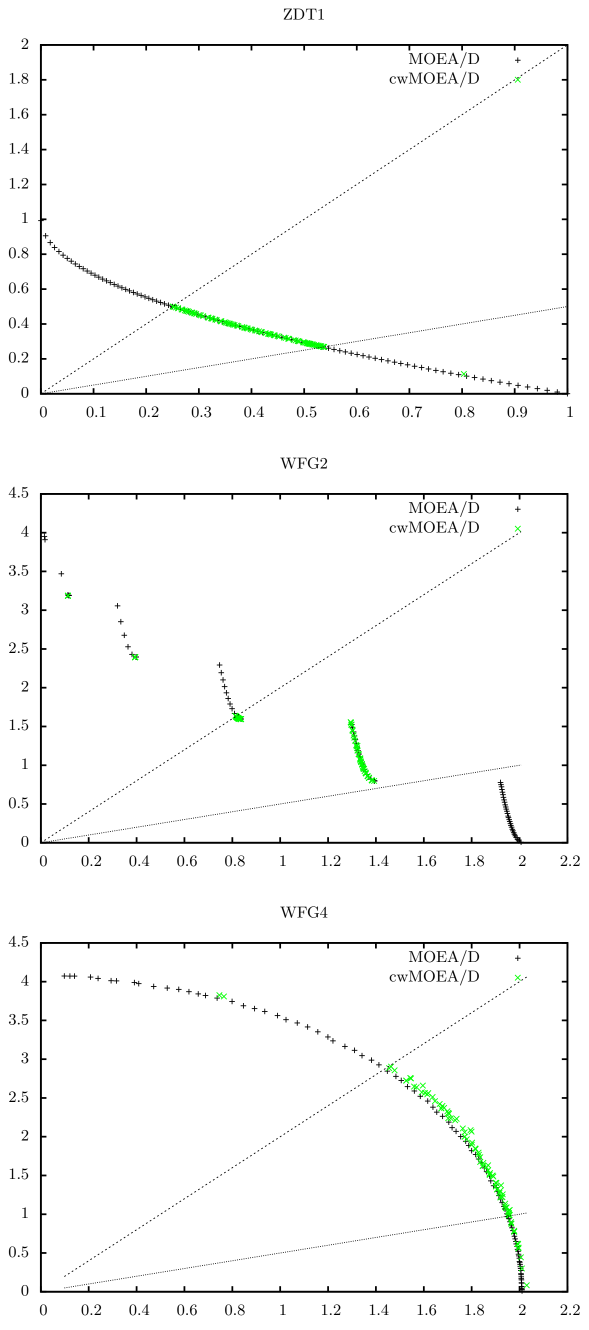
4.2 Limited trade-off

In this section, we provide the results of experiments with limited trade-off between two objectives. The preference is specified as

$$0.5 \leq \frac{f_1(x)}{f_2 x} \leq 2.$$

In this case, the preferred area is a continuous part of the Pareto front (in case the Pareto front itself is continuous). We let the algorithm run for 150,000 function evaluations, with population size of 100 individuals, and the standard deviation for weight mutation set as described in the previous section (i.e. 0.01 for not-preferred individuals and 0.001 for the preferred ones). The initial weights were generated uniformly, and the Tchebycheff decomposition was used to define the subproblems. We made 15 independent runs of the algorithms and show the population in the last genera-

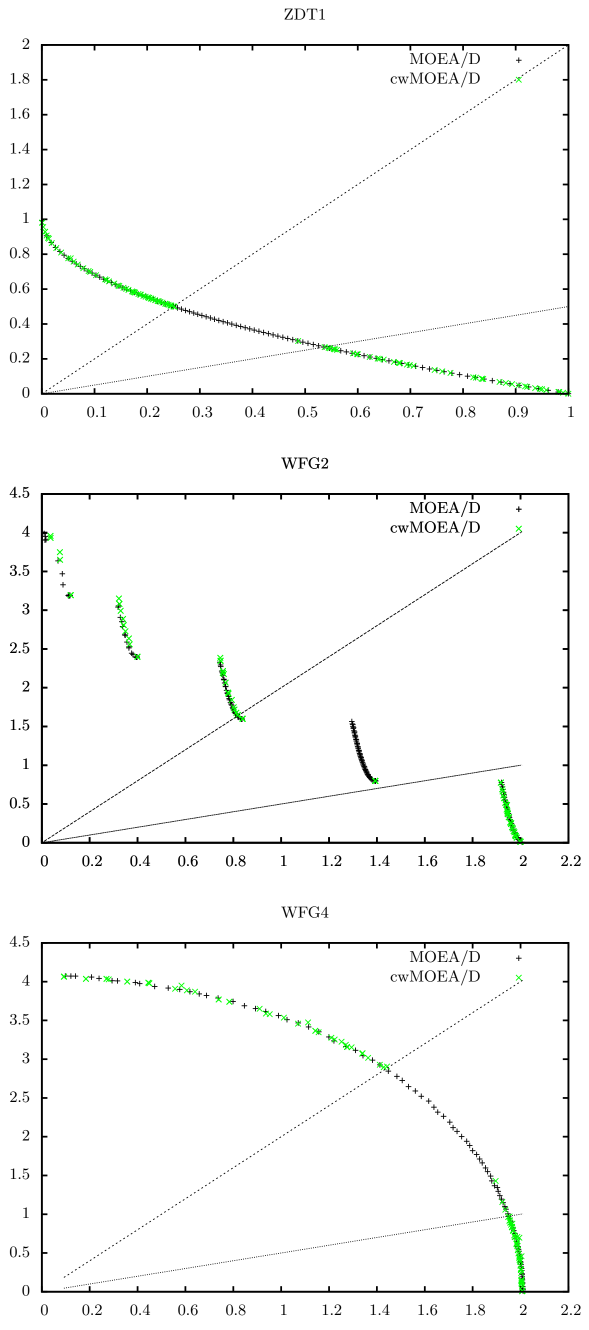
Figure 4: The resulting populations after 150,000 evaluations for the ZDT1, WFG2, and WFG4 problems. Population with median hyper-volume from 15 independent runs is plotted.



tion with median hyper-volume (of the preferred individuals) in the plots in Figure 4. For comparison, we also show the set of solutions found by MOEA/D without any user preferences.

To demonstrate the results, we have chosen three benchmark problems with different properties. ZDT1 is a simple problem with convex Pareto front, WFG2 is a more complicated problem with non-continuous Pareto front and WFG4 has concave Pareto front.

Figure 5: The resulting populations after 150,000 evaluations for the ZDT1, WFG2, and WFG4 problems. Population with median hyper-volume from 15 independent runs is plotted.



The results show, that the algorithm is able to adjust the weights in such a way that the user preference is met. Most of the solutions correspond to the preferred ones. However, in all cases there are a few solutions outside the preferred zone.

For ZDT1, the algorithm found a good spread of preferred solutions converged close to the Pareto front. There is also only one solution outside of the preferred zone.

The WFG2 problem provides and interesting challenge as its Pareto front is not continuous. We can see that the algorithm found the correct parts of the Pareto front and that the solutions are well spread and converged. However, there are a few solutions on the extreme ends of some of the other non-continuous parts of the fronts. The algorithm was not able to change some of the weights enough to ignore these parts. The number of such solutions is low and it does not harm the overall performance much. Also, there are a lot of solutions on the small preferred part on the boundary of the preferred area.

For the WFG4 problem, we can see that the algorithm again moved most of the solutions to the preferred area. In comparison to the other test problems, there are more not-preferred individuals in this set of solutions. Also, the solutions are slightly worse converged compared to MOEA/D without user preferences.

Overall, the algorithm found a large number of well-spread and converged solutions, however, the distribution is not as uniform as the one found by MOEA/D.

4.3 Non-continuous preference

Apart from testing a continuous preference, it may also be interesting to test some non-continuous preference, i.e. such that there are at least two preferred areas in the objective space. To this end, we invert the preference we used in the previous section. Now, the preferred solutions are those, where

$$\frac{f_1(x)}{f_2(x)} < 0.5 \text{ or } \frac{f_1(x)}{f_2(x)} > 2.$$

Such a preference poses new challenge, the algorithm now should divide the weights into two groups instead of grouping them together. We again show the results on the ZDT1, WFG2, and WFG4 benchmark functions in Figure 5. The parameters of the algorithm are the same as in the previous section, the only part that changed is the user preference.

We can again see that the algorithm is able to respect the user preference and most of the solutions are in the preferred areas. In all cases, the solutions are well-converged and spread, however, we can observe that each of the two parts of the preferred space contains different number of solutions.

For the ZDT1 benchmark, we can see that the algorithm found more solutions in the upper part of the search space than in the lower part, however, both parts contain more solutions than what MOEA/D was able to find.

The cwMOEA/D algorithm was able to find all the preferred part of the non-continuous Pareto front of the WFG2 problem, this time, the density of solutions is higher in the lower part, and the upper part is slightly under-represented.

Finally, for WFG4, the algorithm again correctly found the two preferred parts of the Pareto front, but again, the upper part contains less solutions than the lower part.

The differences between the number of solutions in the upper and lower part of the preferred area can be explained by the way the algorithm converges toward the Pareto front. If there are more solutions in one part of the preferred area in the beginning, it is difficult for the algorithm to move them to the other part. Such a move would require large changes in the weights and the algorithm is not able to make such changes in its presents form. It could be beneficial to make more aggressive selection of the weight which would allow the algorithm to move the weight further. For example, the weights can be compared to other weights than just its parents. This would allow the algorithm to remove the weight from the over-represented area and move it to the under-represented one. However, such an aggressive move might require re-computation of the neighborhoods for the underlying MOEA/D algorithm. Such an extension is left as a future work.

4.4 Discussion

In the experimental section, we presented the results of the algorithm only for the problems with two objectives, however, the approach is general and does not depend on the number of objectives. We left the experiments with more than two objectives as a future work. We will also investigate, how the number of objectives affects the required magnitude of the weights mutation.

The same approach can also be used to handle constrained problems – the solutions which do not satisfy the constraints can be flagged as not-preferred and the algorithm will automatically lead the search to the areas which do satisfy the constraints. The testing and evaluation of cwMOEA/D for constrained multi-objective optimization is also left as a future work.

5. CONCLUSION AND FUTURE WORK

We have developed a co-evolution-based algorithm which incorporates user preferences into MOEA/D – cwMOEA/D. The experiments have shown that the algorithm is able to find well-spread and converged solutions in the area of the search space specified by the user. This leads to more effective use of computational resources as uninteresting areas of the space are not examined. The larger number of interesting solutions found by the algorithm allow the decision maker to choose from a wider set of possibilities.

We have also found several directions for future work. The solutions are not as regularly spread as those found by MOEA/D, it may be beneficial to look for ways, which would make the solution set more regularly spread. A better mutation of the weights may be useful to reach this goal. We have also observed that in case of non-continuous preference, some parts of the preferred space may contain more individuals than other. It may be interesting to improve the selection of the weights in such a way that a weight from the over-represented can "move" to the under-represented part. We will also compare cwMOEA/D to different preference-based methods and experiment with the possibilities of constrained handling based on the ideas from cwMOEA/D.

Acknowledgments

This work has been supported by Czech Science Foundation project no. P103-15-19877S.

6. REFERENCES

[1] J. Branke. Consideration of partial user preferences in evolutionary multiobjective optimization. In J. Branke, K. Deb, K. Miettinen, and R. Slowinski, editors, *Multiobjective Optimization*, volume 5252 of *Lecture Notes in Computer Science*, pages 157–178. Springer Berlin Heidelberg, 2008.

[2] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[3] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *Evolutionary Computation, IEEE Transactions on*, 18(4):577–601, Aug 2014.

[4] K. Deb and A. Kumar. Light beam search based multi-objective optimization using evolutionary algorithms. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 2125–2132. IEEE, 2007.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[6] J. J. Durillo and A. J. Nebro. jMetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

[7] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *Evolutionary Computation, IEEE Transactions on*, 10(5):477–506, 2006.

[8] H. Jain and K. Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *Evolutionary Computation, IEEE Transactions on*, 18(4):602–622, Aug 2014.

[9] S. Jiang, Z. Cai, J. Zhang, and Y.-S. Ong. Multiobjective optimization by decomposition with pareto-adaptive weight vectors. In *Natural Computation (ICNC), 2011 Seventh International Conference on*, volume 3, pages 1260–1264, July 2011.

[10] H. Li and Q. Zhang. Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 13(2):284–302, April 2009.

[11] A. Mohammadi, M. Omidvar, X. Li, and K. Deb. Integrating user preferences and decomposition methods for many-objective optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 421–428, July 2014.

[12] J. Molina, L. V. Santana, A. G. Hernández-Díaz, C. A. C. Coello, and R. Caballero. g-dominance: Reference point based dominance for multiobjective metaheuristics. *European Journal of Operational Research*, 197(2):685–692, 2009.

[13] K. Price, R. M. Storn, and J. A. Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[14] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu. MOEA/D with adaptive weight adjustment. *Evolutionary Computation*, 22(2):231–264, June 2014.

[15] A. Sinha, P. Korhonen, J. Wallenius, and K. Deb. An interactive evolutionary multi-objective optimization algorithm with a limited number of decision maker calls. *European Journal of Operational Research*, 233(3):674–688, 2014.

[16] D. A. Van Veldhuizen and G. B. Lamont. Evolutionary computation and convergence to a pareto front. In J. R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1998 Conference*, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Stanford University Bookstore.

[17] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, Dec 2007.

[18] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195, June 2000.