Learning to diversify in complex interactive Multiobjective Optimization

Dinara Mukhlisullina[1], Andrea Passerini[2], Roberto Battiti[2]

[1] Bauman Moscow State Technical University
2-nd Baumanskaya Str., 5, 105005, Moscow, Russia
The author was supported by the Erasmus Mundus Action 2 Programme of the European Union
d.mukhlisullina@gmail.com

[2] Università di Trento
Via Sommarive, 14, 38123, Povo (Trento), Italy
{passerini,battiti}@disi.unitn.it

Abstract

Many real-world problems have a natural formulation as Multiobjective Optimization Problems (MOPs), in which multiple conflicting objectives need to be simultaneously optimized. A popular approach to deal with the resulting complexity consists of interacting with the Decision Maker (DM) during optimization, progressively focusing towards her preferred area in the decision space. In BC-EMO, an evolutionary MOP approach based on a "learning while optimizing" strategy, machine learning techniques are used to interactively learn an approximation of the DM utility function, which guides the search for candidate solutions. While extremely effective in early focusing towards the most promising search directions, the algorithm suffers from a lack of diversification in dealing with complex MOP problems: a premature convergence often returns suboptimal solutions. In this paper we address the problem by introducing improved diversification strategies both at the evolutionary level and in DM preference elicitation. Substantial improvements are obtained on challenging benchmark problems with complex Pareto-optimal sets and non-linear DM utility functions.

1 Introduction

A Multiobjective optimization (MOP) problem can be stated as

$$\begin{aligned} &\text{minimize} && \boldsymbol{f}(\mathbf{x}) = \{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\} \\ &\text{subject to} && \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector of n decision variables; $\Omega \subset \mathbb{R}^n$ is the *feasible region* and is typically specified as a set of constraints on the decision variables; $\boldsymbol{f} : \Omega \to \mathbb{R}^m$ is a vector of m objective functions, which need to be optimized simultaneously. The image of a solution $\mathbf{x}$ in the objective space is a point $\mathbf{z} = \{z_1, \ldots, z_m\} = \boldsymbol{f}(\mathbf{x})$, such that $z_i = f_i(\mathbf{x})$, for each $i = 1, \ldots, m$. The main problem when dealing with MOP is that the objectives to minimize are typically conflicting and improving an objective can result in worsening some of the others. We say that an objective vector $\mathbf{z}$ *dominates* $\mathbf{z}'$ ($\mathbf{z} \prec \mathbf{z}'$) if $z_i \leq z'_i$ for all i and there exists at least one j such that $z_j < z'_j$, i.e., it is better for at least one objective and never worse. A point $\hat{x}$ is *P*areto optimal if there is no other feasible point $\mathbf{x} \in \Omega$ such that $f(\mathbf{x})$ dominates $f(\hat{x})$. The set of Pareto optimal points is called *Pareto Set* (PS) and the corresponding set of Pareto optimal objective vector is the *Pareto Front* (PF). These sets represent the full spectrum of possible solutions obtained by varying the relative relevance of conflicting objectives.

Evolutionary algorithms have been successfully applied [5, 15] to approximate the set of Pareto-optimal solutions, relying on the dominance criterion to guide the evolution towards the PF. However, presenting the decision maker (DM) with thousands of candidate solutions can make the decision-making unbearably cumbersome. Furthermore, it is not efficient to spend time on approximating PF areas which are of no interest to the DM. These areas correspond to compromise (non-dominated) solutions she dislikes, these solutions will be further referred to as "suboptimal" solutions. On the other hand, asking to formalize her preferences in a clear mathematical formula is beyond the capability of a typical human

DM. A recent trend of research has thus focused on *interactive* MOP approaches [10], where the search for the most preferred solution is guided by an interaction with the DM providing feedback for selected solutions. A main issue of these approaches is to make effective use of the feedback received, also accounting for the uncertainty and inconsistency which are typical of human decision-making. We recently proposed a machine learning approach called BC-EMO (Brain-Computer Evolutionary Multiobjective Optimization) [1], which interactively learns a model of the DM preferences and uses it to guide the search for improved solutions. The main advantages of BC-EMO in comparison with other known interactive approaches are as follows. First, it learns the utility function based on pairwise comparisons of the solutions provided by the DM. This form of assessment is the most convenient for the DM, as opposed to the methods (e.g., IIMOM [1]), where the DM is forced to give quantitative scores to the solutions. Second, BC-EMO allows to learn an arbitrary utility function without fixing its form in advance and it is robust to incomplete and inconsistent feedback from the DM. Uncertainty in both the form and the parameters of the DM utility function and mistakes in the DM feedback should be considered natural in a real-world scenario, where the DM learns to understand the specific MOP and to perceive a trade-off between the values of the criteria only while interacting with the method. However, the authors of a number of the interactive methods, such as IEM [1] and PI-EMO-VF [7], specify the form of utility functions a priori and suppose that the DM always provides true preference information. Thus they reduce the problem of the DM utility function learning to that of adjusting parameters of a predetermined parametric function, possibly leading to an incorrect modeling of the underlying DM preferences. Conversely, BC-EMO proved quite effective in adapting to the unknown form of the DM utility and robust to uncertainty and inconsistencies in the DM feedback, as evaluated on a wide range of benchmark problems.

A current limitation of the BC-EMO algorithm, which will be highlighted in the experimental evaluation, is a lack of diversification when facing particularly complex MOP problems and nonlinear user utilities. In this paper we propose a modified version of the algorithm providing substantial improvements on these challenging problems. This is achieved by introducing improved diversification strategies both at evolutionary level and in DM preference elicitation. The former relies on recent work on decomposition-based evolutionary algorithms [8], adapting it to our utility-based scenario. This modification is sufficient to successfully address the diversification needs in the artificial setting in which the search is continuously guided by the true (unknown) DM utility. However, in the real setting the search is guided by a learned model, which is only an approximation of the true DM preferences. An excessive trust in this model can lead to suboptimal solutions, especially in the early stages when the approximation can be quite rough. Introducing diversification in the preference elicitation phase contributes to address this problem. The overall algorithm[1] achieves substantially better results than the original one on representative problems characterized by highly complex PS and non-linear DM utility functions.

The remainder of this paper is organized as follows. Section 2 briefly reviews the BC-EMO algorithm and its limitations. The proposed modifications to the original BC-EMO is described in Section 3. The results of the experimental evaluation are reported in Section 4.

2 BC-EMO+NSGA-II

Let us first briefly revise the original BC-EMO formulation [1] and then highlight its main limitations. The algorithm proceeds by alternating search and learning steps. During search, the current approximation of the utility function is used to guide the search for novel preferred solutions, which are presented to the DM for feedback. In this step the original BC-EMO employs the popular multi-objective optimization algorithm NSGA-II (Non-dominated sorting genetic algorithm version II) [1]. During learning, the collected solutions and their assessments by the DM are used to refine the current approximation of the utility function. These two steps are iterated until a stopping criterion is true. The learning algorithm consists of Support Vector Ranking (SVR) [1], an adaptation of SVM to deal with pairwise preferences.

[1] A software package implementing the algorithm is freely available at `http://wwwcdl.bmstu.ru/dinara/bcemo_ld.tar.gz`

The learned utility function is given by:

$$U(\mathbf{z}) = \sum_{(\mathbf{z}_i,\mathbf{z}_j)\in SV} \alpha_{i,j}\left(K(\mathbf{z}_i,\mathbf{z}) - K(\mathbf{z}_j,\mathbf{z})\right)$$

where SV is the set of Support Vectors, i.e., training pairs which contribute to the prediction (all other pairs have α coefficient equal to zero), $K(\mathbf{z}_i,\mathbf{z}_j)$ is a kernel function. Further details on the method can be found in [1].

2.1 Limitations of BC-EMO

BC-EMO is quite effective in early focusing on the relevant region of the decision space for a wide variety of test problems [1], as well as in facing incomplete and inconsistent feedback from the DM [4]. However, as it will be detailed in the experimental evaluation, the algorithm suffers from a lack of diversification for increasing complexity and dimensionality of the PF.

One of the main reasons for this behaviour lays in the characteristics of the underlying evolutionary algorithm, NSGA-II. This algorithm relies on a non-dominated ranking procedure, which alone discourages the diversity of the search [8]. The authors of NSGA-II already observed that the crowding distance operator fails to preserve diversification in optimization problems with more than two objectives [7]. Indeed, an extensive experimental evaluation [8] shows that the algorithm fails to accurately model complex PS. The simulated binary crossover (SBX) [6] operator used in NSGA-II contributes to the loss of population diversity [8]. For a better diversity preservation in the population the authors of NSGA-II in [7] proposed using the modified recombination operator, which combines the SBX operator and some principles inspired by differential evolution (DE) [11]. However, previous work [8] shows that the sole replacement of the SBX operator by the DE operator in the NSGA-II algorithm does not suffice to significantly improve the results.

A second source of suboptimality is the selection of the candidate solutions to present to the DM for feedback. At the first training iteration, the current version of BC-EMO selects the exa non-dominated solutions at random, as the DM preferences are not available yet. Starting with the second iteration, BC-EMO selects the exa highest scoring solutions retaining Pareto dominance. This strategy tends to produce training sets made of highly similar solutions, preventing the population from preserving diversity. Building the utility function based on these solutions possibly leads to premature convergence to suboptimal solutions.

In the following, we detail the modifications of BC-EMO proposed in order to address all these limitations, starting with the replacement of NSGA-II with the decomposition-based MOEA/D algorithm [8].

3 BC-EMO+MOEA/D

MOEA/D is a recent genetic algorithm for multi-objective optimization based on a decomposition approach [8]. This algorithm generates s single objective optimization subproblems, where s is the population size. Each of these subproblems is a convolution of all objective functions, which are constructed by using the well-known Tchebycheff approach [9]. The jth single objective optimization subproblem is thus defined as:

$$\begin{array}{ll} \text{minimize} & g^j(\mathbf{x}|\boldsymbol{\lambda}^j,\mathbf{z}^*) = \max_{1\le i\le m}\{\lambda_i^j|f_i(\mathbf{x}) - z_i^*|\} \\ \text{subject to} & \mathbf{x}\in\Omega \end{array} \quad (2)$$

where $\boldsymbol{\lambda}^j = (\lambda_1^j,\ldots,\lambda_m^j)$ is the jth weight vector, $\sum_{i=1}^m \lambda_i^j = 1$, $\lambda_i^j \ge 0$ for all $i = 1,\ldots,m$; $\mathbf{z}^* = (z_1^*,\ldots,z_m^*)$ is the ideal point, i.e. $z_i^* = \min\{f_i(\mathbf{x})|\mathbf{x}\in\Omega\}$ for each $i = 1,\ldots,m$.

Under mild conditions, for each Pareto-optimal point $\mathbf{x}^*$ there exists a weight vector $\boldsymbol{\lambda}^j$ such that $\mathbf{x}^*$ is the optimal solution of (2) and each optimal solution of (2) is also a Pareto-optimal of problem (1) [13]. Thus, the MOP problem (1) can be cast into s single optimization subproblems, with corresponding s weight vectors $\boldsymbol{\lambda}^1,\ldots,\boldsymbol{\lambda}^s$.

Traditional evolutionary algorithms for MOP based on Pareto dominance share a difficulty in generating distributed solutions along complex PFs [8, 12]. In the decomposition approach of MOEA/D, different solutions in the current population are associated with different subproblems. The "diversity" among these subproblems naturally leads to diversity in the population [13]. Thus, the problem of generating solutions reduces to generating the weight vectors of each subproblem. This is done in a space with a typically much smaller dimension than that of the decision space ($m << n$). To preserve diversification during the evolution of the population, MOEA/D uses several techniques [8].

In order to integrate MOEA/D with BC-EMO, it is first necessary to introduce the predicted DM utility function into the evolutionary algorithm. The plain MOEA/D does not use Pareto dominance and crowding distance operator to compare the quality of the solutions. Instead, this algorithm considers the value of the function $g(\mathbf{x})$ (see (2)). Let $\mathbf{x}^i$ be a new generated solution after the reproduction operator. Let $\mathbf{x}^j$ be a randomly picked element from the population, with its own weight vector $\boldsymbol{\lambda}^j$. If $g^j(\mathbf{x}^i|\boldsymbol{\lambda}^j, \mathbf{z}^*) \leq g^j(\mathbf{x}^j|\boldsymbol{\lambda}^j, \mathbf{z}^*)$, then $\mathbf{x}^i$ is accepted as the new individual with vector $\boldsymbol{\lambda}^j$ and $\mathbf{x}^j$ is discarded from the population.

In order to integrate the DM utility into the selection process, we construct a fitness function (FitnessFunction) as follows:

$$\phi^j(\mathbf{x}) = (1-\alpha)\tilde{U}(\mathbf{x}) + \alpha\, \tilde{g}^j(\mathbf{x}) \tag{3}$$

where $\tilde{U}(\mathbf{x})$ is the predicted utility function and $\tilde{g}^j(\mathbf{x}) = g^j(\mathbf{x}|\boldsymbol{\lambda}^j, \mathbf{z}^*)$ is the minimization function of the jth scalarized problem. Both functions are normalized, as $\tilde{U}(\mathbf{x}) = (U(\mathbf{x}) - U_{min}) \,/\, (U_{max} - U_{min})$ and $\tilde{g}^j(\mathbf{x}) = 1/\left(1 + \exp(-g^j(\mathbf{x})\right)$ respectively, in order to be made comparable. The scalar value α is a real weight $0 \leq \alpha \leq 1$ trading-off the two goals: fitting the DM preference versus improving diversification. The comparison of two solutions thus consists of the following steps. 1) Check the standard domination principle of two solutions $\mathbf{x}^1$ and $\mathbf{x}^2$. If $\boldsymbol{f}(\mathbf{x}^2) \prec \boldsymbol{f}(\mathbf{x}^1)$, then $\mathbf{x}^2$ replaces $\mathbf{x}^1$. 2) Otherwise, calculate the value of the fitness function (3). If $\phi^j(\mathbf{x}^2) \leq \phi^j(\mathbf{x}^1)$, then $\mathbf{x}^2$ replaces $\mathbf{x}^1$.

The preference elicitation phase consists of ordering the current population according to the learned fitness function (3) and selecting the exa highest scoring individuals. The value of α in (3) has to provide a trade-off between fitting the DM preferences and preventing the algorithm from a premature convergence to a suboptimal solution: if $\alpha = 0$, the search focuses on looking for a single preferred solution and may end up in a premature convergence; if $\alpha = 1$, the search process resembles plain search towards the whole PF, without considering the DM utility. In this work, we consider the following three cases: $\alpha = 0$; $\alpha = 0.5$; linearly decreasing α. Our preliminary experiment shows that the third case provides the best approximation accuracy (Section 4.2).

3.1 Diversifying candidates for DM feedback

Algorithm 1 Fitness+Clustering Based Preference Ordering

1: **procedure** PREFORDER(P_i, U_i, exa, r)
Input:
P_i input population; U_i current version of the utility function; exa number of individuals for DM feedback; r ratio of the number of clusters to size of population
Output:
P_o ordered individuals for DM feedback
2: $s_o \leftarrow r * len(P_i)$
3: identify Pareto non-dominated individuals P_i^* in P_i
4: sort P_i^* by FITNESSFUNCTION based on U_i
5: $P_o^* \leftarrow$ collect first s_o elements of P_i^*
6: cluster P_o^* by using k-$medoids$ algorithm ($k = exa$)
7: $P_o \leftarrow$ collect the exa *medoids* of P_o^*
8: **return** P_o
9: **end procedure**

While tackling the problem of insufficient exploration of the decision space, this version of the algorithm does not fully address the problem of diversity of the candidates for the DM feedback. By selecting the best exa examples according to the fitness function (3), it still suffers from a lack of diversification, which can lead to premature convergence to a suboptimal solution, even if guided by the true utility function (Section 4.2). We addressed this problem by *clustering* the population to select training examples for the DM feedback. To this purpose, we choose $r * s$ solutions with the best values of the fitness function, where s is the population size and $0 < r \leq 1$ a parameter. We then cluster this set of solutions into exa clusters and present cluster representatives to the DM. In this work we employ a simple *k-medoids* algorithm, where the medoids represent the candidates for the DM feedback. Note that we cannot straightforwardly employ a standard *k-means*, as we are clustering in the objective space and we are not guaranteed that a novel point corresponds to a feasible one in the decision space. The resulting preference elicitation procedure is shown in Algorithm 1. The value r is a parameter of the algorithm, trading-off the importance of the utility function with respect to the diversity of the training set. For $r = 1$ the method resembles plain search towards the whole PF, without considering the DM preferences. Algorithm 2 describes the updated training procedure including the novel preference function. The overall BC-EMO plus MOEA/D is presented in Algorithm 3.

Algorithm 2 Training procedure of BC-EMO+MOEA/D

1: **procedure** TRAIN(P_i, U_i, s, exa, r)
Input:
P_i current population; U_i current version of the utility function; s size of output population; exa number of training individuals for iteration; r ratio of the number of clusters to size of population
Output:
P_o current population with the updated fitness values; U_o refined version of the utility function; res_o estimated performance of refined utility function
2: $dist \leftarrow 0$, $res_o \leftarrow 0$
3: $P_{tr} \leftarrow$ PREFORDER(P_i, U_i, exa, r)
4: obtain pairwise preferences for P_{tr} from the DM
5: add P_{tr} to the current list of training instances
6: choose best kernel K by k-fold cross validation
7: $U_o \leftarrow$ function trained on full training set with K
8: $res_o \leftarrow$ k-fold cv estimate of function performance
9: $P_o \leftarrow$ update P_i by FITNESSFUNCTION based on U_o
10: **return** P_o, U_o, res_o
11: **end procedure**

4 Experimental Results

This experimental section highlights the shortcomings of the previous version of the BC-EMO algorithm, and shows how the proposed modifications succeed in addressing them. We will thus analyze in turn the problems discussed in Section 2.1. Our aim is not to boost the algorithm performance by fine tuning its parameters, but rather to show the improvements achieved by our novel version. Therefore, we chose the following default values for the non-critical parameters (see Algorithm 3), to be kept fixed in all experiments: $s \approx 600$ - population size[2]; $maxgen = 1,000$ - number of generations; $gen_1 = 200$ - number of generations before the first training iteration; $gen_i = 10$ - number of generations between two training iterations; $r = 0.5$ - ratio of solutions for the clustering algorithm. Preliminary experiments

[2]Note that in MOEA/D the population size, that is equaled to the number of weight vectors, is controlled by an integer H, such that each component of s weight vectors $\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^s$ takes a value from $\{0/H, 1/H, \ldots, H/H\}$. Thus, $s = C^{m-1}_{H+m-1}$, where m is the number of objective functions (more details can be found in [8]). For three-objective test problems $H = 33$, so $s = 595$. In NSGA-II the population size must be divisible by four because of the tournament selection operator.

Algorithm 3 The BC-EMO+MOEA/D algorithm

procedure BC-EMO($maxit$, exa, gen_1, gen_i, $thres$, s,r)
Input:
$maxit$ maximum number of allowed training iterations; exa number of training individuals for iteration; gen_1 generations before first training iteration; gen_i generations between two training iterations; $thres$ performance requirement to stop training; s size of population; r ratio of the number of clusters to size of population
Output:
P final ordered population
    $res \leftarrow 0$, $it \leftarrow 0$, $U \leftarrow$ RAND
    run plain MOEA/D for gen_1 generations
    collect last population P
    while $it \leq maxit \wedge res < thres$ **do**
        $P, U, res \leftarrow$ TRAIN(P, U, s, exa,r)
        run MOEA/D for gen_i generations guided by FITNESSFUNCTION with U
        collect last population P
    end while
    run MOEA/D for remaining number of generations guided by FITNESSFUNCTION with U
    return the final population P
end procedure

showed that the algorithm is rather robust to modifications of these parameters. We fixed the number of training examples per iteration (exa) to 5 in order to maintain a reasonable per-iteration burden on the DM. We fixed the C regularization parameter used in SVR to 10^6. A preliminary analysis showed it to be a reasonable value considering the cost of the resulting optimization problem and an assumption of no noise in the DM evaluation. The kernel to be employed in the model is not chosen *a priori* but selected and continuously adapted for each problem and each training iteration by an internal cross-validation procedure (Algorithm 2). A similar procedure can be implemented for the C regularization parameter in a noisy scenario, a more realistic context when emulating a human DM. Preliminary studies on the original version of the BC-EMO algorithm proved its robustness in this type of scenario [4].

4.1 Test Problems

We already proved that the original BC-EMO succeeds in early converging to the optimal solution on a number of diverse test problems [1]. Here we challenge the algorithm by focusing on the benchmark suite developed for the CEC-2009 competition [14], which consists of test problems with complex Pareto sets and fronts.

When testing any interactive methods for multi-objective optimization problems, one faces the problem of simulating the DM. A general approach that is commonly employed is to use a particular value function that acts as a representative of the DM. We refer to this function as the *true utility function* or the *value function*. The simplest possible model is a linear function, in which positive weights encode the relative importance of the different objectives. However, its appropriateness is rather questionable [1]. To simulate the complexity of a realistic utility function, we focus here on non-linear functions achieving their optimum in the central area of the decision space, rather than at some extreme point. The rationale behind this choice is twofold: incorporating non-linear correlations, as commonly found between objectives; accounting for the tendency of human DM to prefer compromise solutions trading-off the various optimality criteria.

We selected the UF8 and UF9 test problems from the benchmark suite (see [14] for their mathematical formulation). These problems were chosen for the complexity and diversity of their PFs, and because we did manage to build utility functions satisfying the mentioned complexity requirements.

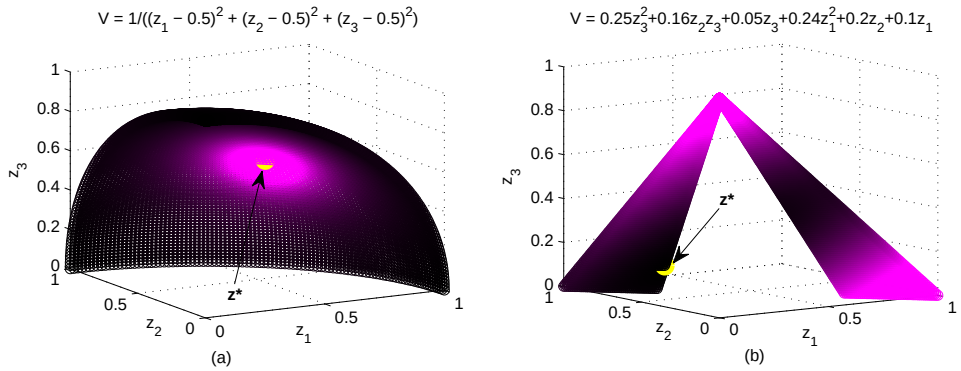

Figure 1: Pareto front, gold solution: (a) for the UF8 test problem, (b) for the UF9 test problem.

For the UF8 test problem, we used the value function from [7][3]:

$$V(z_1, z_2, z_3) = 1/((z_1 - 0.5)^2 + (z_2 - 0.5)^2 + (z_3 - 0.5)^2). \quad (4)$$

The optimal minimum of this utility function, which we refer to as the *gold solution*, is in $\mathbf{z}^* = (0.5773, 0.5773, 0.5775)$ of the central area of the PF. Figure 1(a) shows the PF for the UF8 test problem and the gold solution. The brightness of the points corresponds to the value of the true utility function (4): the darker the point, the lower the corresponding utility function value. In this case to minimize each objective function one needs to maximize the value function (4). Thus the gold solution $\mathbf{z}^*$ corresponds to the brightest point. For the UF9 test problem, we employed a polynomial true utility function having the following form:

$$V(z_1, z_2, z_3) = 0.25 \cdot z_3^2 + 0.16 \cdot z_2 \cdot z_3 + 0.05 \cdot z_3 + 0.24 \cdot z_1^2 + 0.2 \cdot z_2 + 0.1 \cdot z_1. \quad (5)$$

The global minimum of this utility function (the gold solution) is found[4] in $\mathbf{z}^* = (0.2198, 0.6595, 0.1207)$. Figure 1(b) shows the PF for the UF9 test problem and the discovered gold solution, again with brightness indicating utility. In this case, minimization of the objective functions leads to minimization of the utility function (5). Thus the gold solution $\mathbf{z}^*$ corresponds to the darkest (rather than the brightest) point.

In all the experiments, the evaluation measure is the approximation error, which is calculated as the Euclidean distance between the gold solution and the final solution obtained by the algorithm. We repeated each experiment 100 times varying the random seed of the EMOA search, and report box-and-whisker plots of the results. All the results are displayed with logarithmic scaling of Y axis.

4.2 True Utility Function

The first set of experiments aims at testing the ability of BC-EMO with NSGA-II and MOEA/D to find the gold solutions under the true utility function guidance, i.e. (4) and (5) for UF8 and UF9 respectively. In these experiments, we switched off the training phase. It is obvious that if the method has problems in finding the gold solution under the true utility guidance, then it will surely not be able to do it when guided by its learned approximation.

Figure 2 shows the results on the UF8 (Figures (a) and (b)) and UF9 (Figures (c) and (d)) test problems with the true utility function. First, we tested BC-EMO+NSGA-II and BC-EMO+MOEA/D with the same number of generations $maxgen = 1,000$. For BC-EMO+MOEA/D we used three various cases

[3]Note that this utility function does not retain Pareto dominance. Let us consider two points in the objective space $\mathbf{z}^1 = (0.4, 0.4, 0.4)$ and $\mathbf{z}^2 = (0.6, 0.6, 0.6)$, these two points have the same value of the true utility function (4), but it is evident that $\mathbf{z}^1$ dominates $\mathbf{z}^2$ (in the case of minimization of each objective functions). We could not find an alternative polynomial function providing the most preferred solution in non-extreme points.

[4]The solution is found using the *quadprog* function for quadratic optimization which is part of the MatLab^{TM} package.

for α (see Section 3). Concerning the UF8 test problem, the median value of the final solutions obtained with BC-EMO+NSGA-II is equal to 0.1991 (Figure 2(a)), whereas the median error value obtained with BC-EMO+MOEA/D is two orders of magnitude smaller, regardless of the choice for α (Figure 2(b)). To find out whether the number of generations affects the final results we carried out additional experiments with increased number of generations with BC-EMO+NSGA-II. Increasing generations up to $5,000$ does not provide improvements. For $10,000$ generations, BC-EMO+NSGA-II achieves a median error of 0.0109, still an order of magnitude larger than that found by BC-EMO+MOEA/D with one tenth of the generations: 0.0024 for $\alpha = 0$; 0.0004 for $\alpha = 0.5$ and 0.0008 for linearly decreasing α.

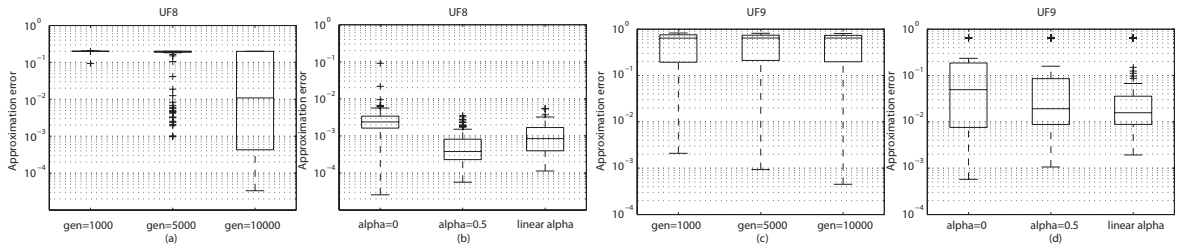

Figure 2: Approximation error obtained with guidance of the true utility function in search process: (a) UF8: BC-EMO+NSGA-II; (b) UF8: BC-EMO+MOEA/D; (c) UF9: BC-EMO+NSGA-II; (d) UF9: BC-EMO+MOEA/D

Results on the UF9 test problem show a similar trend. Figure 2(c) reports results for BC-EMO+NSGA-II, achieving a median approximation error of 0.6410 for $maxgen = 1,000$; 0.6425 for $maxgen = 5,000$ and 0.6427 for $maxgen = 10,000$. Figure 2(d) reports results for BC-EMO+MOEA/D with different choices of α. All three choices have a median error an order of magnitude smaller than that of BC-EMO+NSGA-II: 0.0494 for $\alpha = 0$; 0.0192 for $\alpha = 0.5$ and 0.0157 for linearly decreasing α.

These experiments clearly show that BC-EMO+MOEA/D substantially outperforms BC-EMO in combination with NSGA-II. The high approximation error of the latter indicates that NSGA-II itself has a problem of lack of diversification and premature convergence to a suboptimal solution, as previously discussed (Section 2.1). Concerning the choice of α, there are no significant differences in performance in this artificial setting with true utility function. We focused on the linearly decreasing strategy as it provides slightly better overall results (see test problem UF9).

4.3 Predicted Utility Function

In the next set of experiments, we aim at evaluating the number of DM queries needed in order to achieve a certain level of approximation. In real situations it is the DM who decides when the interaction process should be stopped. In experimental stage we simulate this behaviour by introducing a stopping criterion. The search process is stopped if one of the two conditions is met: 1) the Euclidean distance between two consecutive best solutions is equal or smaller than a predefined threshold $\epsilon = 10^{-3}$; 2) the maximum number of allowed training iterations is reached. Thus, in all experiments we vary the maximum number of iterations in order to determine the minimum number of them required to satisfy the first condition of the stopping criterion. Therefore, the X axis range of all graphs below is limited by this value.

Figures 3 (a) and (b) report experimental results for the BC-EMO+MOEA/D using predicted utility functions for problems UF8 and UF9 respectively. The algorithm achieves a median approximation error of 0.1325 and 0.5236 for the two problems. These results are comparable with the ones achieved by BC-EMO+NSGA-II guided by the true utility function. Indeed, guiding BC-EMO+NSGA-II with predicted utility function also produces similar results. By looking at the gap with the results when guided with the true utility, it is clear that the learning stage is badly affecting the performance of BC-EMO+MOEA/D. Improving this stage could help to reduce this gap, whereas the original BC-EMO+NSGA-II could hardly be improved without substantial modifications to its evolutionary strategy (i.e. regardless of DM adaptation), given the bad results it achieves in the optimal case (true utility).

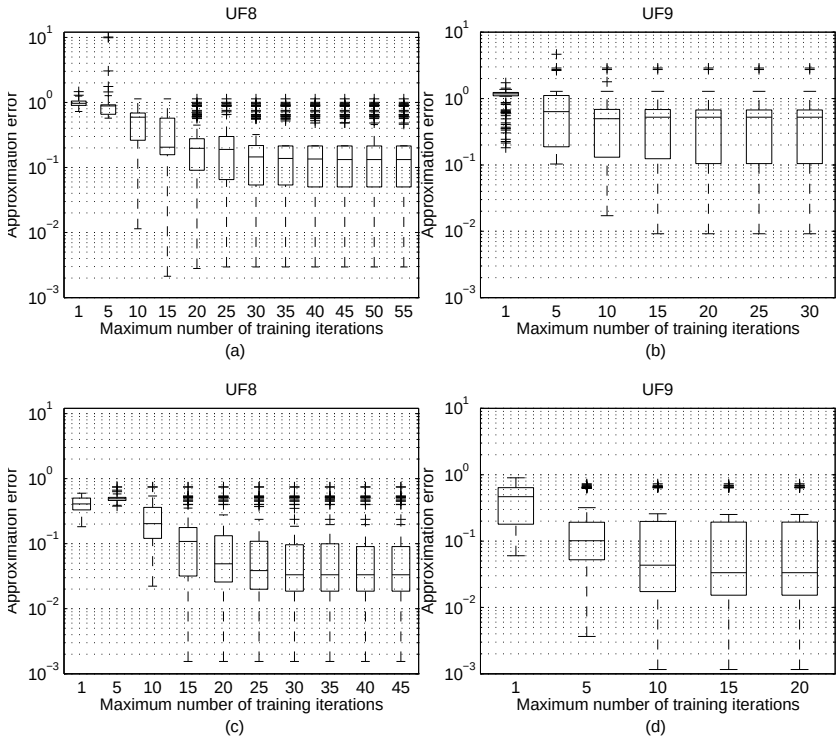

Figure 3: Approximation error obtained on the UF8 and UF9 test problems with guidance of the predicted utility function in search process: (a) UF8: BC-EMO+MOEA/D; (b) UF9: BC-EMO+MOEA/D; (c) UF8: BC-EMO+MOEA/D with clustering; (d) UF9: BC-EMO+MOEA/D with clustering.

4.4 Clustering

The aim of the next set of experiments is to evaluate the performance of the clustering strategy in improving diversification of the training set for the DM feedback (Section 3.1). The clustering stage was performed by using *k-medoids* algorithm implemented in the KMlocal package[5].

Figure 3 (c) and (d) show the results of the modified BC-EMO+MOEA/D algorithm on the UF8 and UF9 test problems respectively. Introducing clustering significantly improves the median results in both cases: on the UF8 test problem, the median goes from 0.1325 to 0.0332, and on the UF9 test problem from 0.5236 to 0.0334. These improvements are statistically significant, as measure by a two-tailed paired t-test ($p < 0.0001$). Clustering also tends to improve robustness of the approach, as shown by the reduced variance of the results, especially for the UF9 problem. Finally, this modification helps in speeding up convergence, thus reducing training iterations and DM effort: from 55 to 45 on the UF8 test problem, from 29 to 18 on the UF9 test problem.

5 Conclusion

We described an improved interactive MOP approach capable of effectively dealing with complex PS and non-linear (unknown) DM utility functions. Key to the result are better diversification strategies in both the search and the preference elicitation phases. A number of research directions can be identified in order to further decrease the DM's cognitive workload and improve the usability of the proposed system. We have some preliminary results of interacting with the real DM while solving the applied problem of four-crank sheet-metal press designing (with eight decision variables and three objectives). According to the DM the method is easy to use and it allows the DM to get the preferred solution just for seven iteration. However a proper evaluation would require extensive study with DM. The effort of the DM in providing

[5]Available at `http://www.cs.umd.edu/~mount/Projects/KMeans/`.

feedback is the main bottleneck of interactive decision making. While we minimize the complexity of the required feedback by focusing on pairwise comparisons, advanced *query learning* strategies can be explored to further reduce the number of questions asked to the DM. Bayesian preference elicitation strategies [2, 3] are a promising direction to pursue.

References

[1] R. Battiti and A. Passerini. Brain-computer evolutionary multiobjective optimization: A genetic algorithm adapting to the decision maker. *IEEE Transactions on Evolutionary Computation*, 14(5):671–687, oct. 2010.

[2] A. Birlutiu, P. Groot, and T. Heskes. Efficiently learning the preferences of people. *Machine Learning*, pages 1–28, May 2012.

[3] E. Bonilla, S. Guo, and S. Sanner. Gaussian Process Preference Elicitation. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems*, pages 262–270. 2010.

[4] P. Campigotto, A. Passerini, and R. Battiti. Handling concept drift in preference learning for interactive decision making. In *Online proceedings of the First International Workshop on Handling Concept Drift in Adaptive Information Systems (HaCDAIS 2010)*, Barcelona, Spain, Sep. 2010.

[5] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001.

[6] K. Deb and R.B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[7] K. Deb, A. Sinha, P.J. Korhonen, and J. Wallenius. An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *IEEE Transactions on Evolutionary Computation*, 14(5):723 –739, oct. 2010.

[8] H. Li and Q. Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evolutionary Computation*, 13(2):284 –302, april 2009.

[9] K. Miettinen and M.M. Makela. On scalarizing functions in multiobjective optimization. *OR Spectrum*, 24:193–213.

[10] K. Miettinen, F. Ruiz, and A.P. Wierzbicki. Introduction to Multiobjective Optimization: Interactive Approaches. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 27–57. Springer-Verlag Berlin, Heidelberg, 2008.

[11] R. Storn and K. Price. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359.

[12] R. Wang, R. Purshouse, and P. Fleming. Preference-inspired co-evolutionary algorithms for many-objective optimisation. *IEEE Transactions on Evolutionary Computation*, PP(99):1, 2012.

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

[14] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. Technical Report CES-487, The Shool of Computer Science and Electronic Engineering, University of Essex, 2008.

[15] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov 1999.