An Interactive Decision Tree-Based Evolutionary Multi-Objective Algorithm

Seyed Mahdi Shavarani[1][0000−0002−3316−1252],
Manuel López-Ibáñez[1][0000−0001−9974−1295],
Richard Allmendinger[1][0000−0003−1236−3143], and
Joshua Knowles[1,2][0000−0001−8112−6112]

[1] Alliance Manchester Business School, University of Manchester, UK
{seyedmahdi.shavarani,manuel.lopez-ibanez}@manchester.ac.uk
[2] Schlumberger Cambridge Research, Cambridgeshire, UK

Abstract. Recent research using machine decision makers has revealed that some leading interactive evolutionary multi-objective optimization algorithms do not perform robustly with respect to interactions with preference models (and biases) posited to be representative of human Decision Makers (DMs). In order to model preferences better, we propose an explainable interactive method that uses decision trees to automate (fast) pairwise comparisons based on trade-offs of two given solutions. To cancel out possible biases and errors in estimations, we use the trained tree in holistic comparisons to determine solutions that survive each generation. We test our new method with respect to two different preference models (Tchebychef and Sigmoid) on problems from 2 to 10 objectives, and control both the number of interactions available and various biases. The results suggest the superiority of our method in learning the DM's preferences and in terms of the utility value of the final solution returned by the algorithm compared with some well-known interactive methods.

Keywords: Interactive Evolutionary Multi-Objective Optimization · Decision Tree · Machine Decision Maker · Preference Learning · Bias

1 Introduction

Many real-life optimization problems have several conflicting objectives to be optimized simultaneously [22]. Due to the conflicting nature of objective functions in such Multi-Objective Optimization Problems (MOOPs), it is generally not possible to have a single solution where all the objectives attain their optimal values. Instead, the general goal of solving MOOPs is to reach a small subset of Pareto-optimal solutions with interesting trade-offs or the most preferred one thereof. Evolutionary Multi-Objective Optimization Algorithms (EMOAs) naturally work with a population of solutions and can generate a representation of the Pareto Front (PF) in a single run. Hence they are well aligned to the requirements of MOOPs. However, with an increase in the number of objectives, EMOAs lose their selection pressure, and their performance declines exponentially [7, 16, 17]. Interactive algorithms compensate for this issue by building a

preference model of the Decision Maker (DM) and generating only those parts of the PF that are interesting to the DM [1, 5]. Such *interactive* EMOAs (iEMOAs) alternate between the decision-making and optimization phase in order to reduce computational costs and support the DM in reaching a desirable solution, while incurring minimal cognitive effort.

Despite the potential of iEMOAs as described above, unfortunately scant evidence exists to date to attest to how well such algorithms perform under realistic conditions [1, 19]. The difficulty is partly due to the challenge of testing with human DMs. Rather, recent studies, exploiting a Machine DM (MDM) to enable statistical testing over sufficient interactive runs [19, 25], have indicated that iEMOAs may not perform as expected under realistic conditions. Specifically, their performance declines with a higher number of objectives or when the elicited preference information is affected by human-specific biases, such as inconsistent decisions [25] and fatigue [31]. Thus, even the algorithms that perform well under ideal conditions seem to lack much robustness to biases and other complications that are typical in real-life situations. The primary sources of the problem should be traced back to the core features of an interactive method, which include interaction style, preference model, and the way the preferences are exploited inside the method to direct the search towards the Most Preferred Solution (MPS) [28].

In re-designing these components, we suggest adopting Decision Trees (DTs). DTs have been widely used in learning user preferences [9, 10, 27, 29], but not, to our knowledge, in interactive methods. As non-parametric supervised learning methods, DTs can be seen as a piece-wise constant approximation [6] that delivers classification by recursive partitioning of the solutions space. Utilizing DTs does not require any assumptions about the DM's preference model. DTs are competitive with other learning methods in terms of accuracy and speed, superior in terms of interpretability and comprehensiveness [9], and thus, are very popular among classification techniques [30]. DTs are easy to understand, interpret and visualize (compared with black-box models). This characteristic is particularly beneficial in helping human DMs understand and trust the process and final results [2]. Recently, there have been an emphasis on the explainability of the learning methods in interactive methods and particularly it has been encouraged to employ explainable learning methods, such as DTs for preference learning [20]. DTs do not require any pre-processing or scaling of the data, which can be a tricky task in MOOPs with an unknown PF, and they can work with different types of data, from categorical to continuous [2]. DTs naturally detect relevant features and simplify the learned model [27, 29].

We use binary classification DTs to learn the desirable trade-offs in pairwise comparisons and to predict the preferred solution from the DM's perspective. DTs overcome problems of other learning-to-rank algorithms, which are susceptible to significant errors when confronted with small biases and non-idealities such as inconsistent decisions [9]. The learned preferences are local to the area of the PF that is being explored in that stage of the interaction, which is a desirable property. The experimental results indicate that the proposed algorithm, which

we call DT-based EMOA (DTEMOA), achieves competitive performance when compared with other iEMOAs in terms of the desirability of the final solution returned and robustness to biased or inconsistent preferences.

In what follows, our proposed DTEMOA algorithm, and the way DTs are utilized in it, are explained in Section 2. Our experimental setup is laid out in Section 3, and the results are discussed in Section 4. Finally, conclusions and future research directions are provided in Section 5.

2 Methods

We focus on indirect preference elicitation in our method, where the DM is asked to provide a ranking of a subset of non-dominated solutions at each interaction. We do not attempt to learn a complete ordering of a set of solutions. Instead, similar to [23], we try to learn how the DM performs pairwise comparisons. As in [14], the preferences of the DM are modeled by simple rules. We use DTs to generate such rules to predict the preferred (winner) solution in pairwise comparisons.

DTs consist of nodes that partition data according to a rule. Our method is based on the DT known as CART [6] using a Gini impurity measure to identify good partitions of the data.

2.1 Decision Tree-Based EMOA (DTEMOA)

Preference elicitation. In each interaction, a small sample of solutions is randomly selected from the population ($S \subseteq pop$, $|S| = N^{\text{sol}}$) and their objective vectors $Z = \{\mathbf{z} = \mathbf{f}(\mathbf{x}) | \ \mathbf{x} \in S\}$ are presented to the DM. The DM ranks the options based on their utility. There is no requirement for a complete ordering of the solutions; the algorithm can handle a partial ordering. However, we assume the DM provides a complete ordering for simplicity. Let $a \succ b$ denote that a is preferred over b. We represent the total order provided by the DM with the vector $\mathbf{r}$ such that the rank of $\mathbf{z}_i$ is r_i (lower rank values are better) and $\mathbf{z}_i \succ \mathbf{z}_j \iff r_i < r_j$.

Preference learning. We aim to learn a DM's preferences from the ranking provided as well as predicting (simulating) her preference when evaluating the trade-offs of two objective vectors. DTEMOA infers pairwise orderings from the elicited preference information that can be in the form of complete or partial ranking of a subset of solutions. To construct the training set T, we use the well-known pairwise transformation [15], i.e., for each pair of objective vectors $\mathbf{z}_i, \mathbf{z}_j \in Z$, we create a training example $\mathbf{z}_i - \mathbf{z}_j$ labeled with:

$$\phi(\mathbf{z}_i - \mathbf{z}_j) = \text{sign}(r_j - r_i) = \begin{cases} +1 & \text{if } \ \mathbf{z}_i \succ \mathbf{z}_j \\ -1 & \text{otherwise;} \end{cases} \tag{1}$$

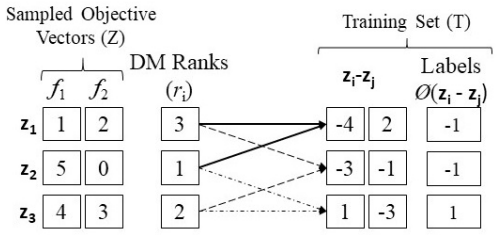

Fig. 1. Construction of the training set from the ranked solutions over a problem with 2 objectives. The number of solutions ranked by the DM is $|Z| = 3$. For each pair of ranked solution $(\mathbf{z}_i, \mathbf{z}_j \in Z)$ an example, labeled by $\phi(\mathbf{z}_i - \mathbf{z}_j)$, is added to the training set. The size of the training set is $\binom{|Z|}{2}$.

i.e., an example is labeled $+1$ if the first solution is preferred over the second one, and -1, otherwise. An instance of such a process is given in Figure 1, where $Z \subset \mathbb{R}^2$, $|Z| = 3$, and each possible pairwise comparison of the ranked set generates an example of the training set T.

In the next step, T is used to train the DT to predict the preferred solution in pairwise comparisons of unseen data as well as the probability of such a decision. Subsequent interactions add more training examples to T, increasing the accuracy of the model. To predict the preferred solution when comparing two objective vectors $\mathbf{z}$ and $\mathbf{z}'$, the trade-off vector $\mathbf{z} - \mathbf{z}'$ is given as the input to the trained DT, which predicts its class (label) $\phi(\mathbf{z} - \mathbf{z}') \in \{+1, -1\}$ indicating whether $\mathbf{z}$ is preferred over $\mathbf{z}'$. The probability of the sample being in class $c \in \{+1, -1\}$ at leaf node o is calculated by $\frac{N_o^c}{N_o}$, where N_o^c is the number of samples of class c in node o, and N_o is the total number of samples in o.

Determination of solution score for sorting. Each solution in the population is given a score used to sort solutions. To calculate the score of a solution $\mathbf{x}$, it is compared with all other solutions in the population and the probability that it is preferred over the compared one is calculated using the trained DT. The sum of all such probabilities is the score of the solution $\mathbf{x}$:

$$score(\mathbf{x}) = \sum_{\substack{\mathbf{x}' \in pop \\ \mathbf{x} \neq \mathbf{x}'}} \Pr\{\mathbf{f}(\mathbf{x}) \succ \mathbf{f}(\mathbf{x}')\} = \sum_{\substack{\mathbf{x}' \in pop \\ \mathbf{x} \neq \mathbf{x}'}} \Pr\{\phi(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}')) = 1\} \qquad (2)$$

Integrating DTs into EMOAs. Given the above steps, let us explain how the preferences of the DM are considered in an EMOA to guide the search toward the most preferred parts of the PF. We use NSGA-II [11] as the underlying optimizer. After some generations of the NSGA-II, the algorithm stops to make the first interaction, elicits a ranking from a subset of solutions and consequently builds the training set T as explained above. A DT is trained using T to predict the preferred solution in pairwise comparisons. In the following generations, the

Algorithm 1: DTEMOA

Input: N^{int} : Total number of interactions
N^{sol} : Number of solutions evaluated by the DM per interaction
pop : Population of solutions
gen_1 : Generations before first interaction
gen_i : Generations between two interactions

Output: The most preferred solution

$T \leftarrow \emptyset$
$pop \leftarrow$ run NSGA-II for gen_1 generations
for 1 **to** N^{int} **do**
  $Z \leftarrow$ select N^{sol} solutions
  $\mathbf{r} \leftarrow$ ask the DM to rank the solutions in Z
  $T \leftarrow T \cup \{(\mathbf{z}_i - \mathbf{z}_j, \phi(\mathbf{z}_i - \mathbf{z}_j)) \mid \forall \mathbf{z}_i, \mathbf{z}_j \in Z, \phi(\mathbf{z}_i - \mathbf{z}_j) = \text{sign}(r_j - r_i)\}$
  Train DT using T
  $pop \leftarrow$ run NSGA-II for gen_i generations
    replacing crowding distance with *score* (Eq. 2)
return best $\mathbf{x} \in pop$ ranked first by non-dominated sorting and then *score*

DT is used to calculate the scores of the solutions (Eq. 2) to sort solutions with the same non-dominated sorting rank, i.e., the solution scores replace the crowding distance of NSGA-II to differentiate between non-dominated solutions. The solutions with a higher score have a better chance of survival and participation in mating and the generation of new offspring. The pseudo-code of the algorithm is illustrated in Algorithm 1.

When dealing with DMs, it is crucial to gain the DM's trust and confidence in the process and the results. When using DTs, learned trees can be conveniently visualized, summarizing all the rules elicited from the data set. Such a visualization is illustrated in Fig. 2, showing a DT that was built based on preference information elicited from the DM in an experiment on the DTLZ1 problem.

3 Experimental Design

This section outlines details pertaining to the Machine DM (MDM) we use to simulate a real DM, the benchmark problems, algorithms considered, performance metrics, and algorithm parameters settings as used in the subsequent experimental study.

3.1 Machine Decision Maker (MDM)

To simulate a real DM in the experiments, we have used the MDM introduced in [19] and improved in [25]. The MDM is composed of a utility function (UF) that simulates the true preferences of the DM and simulations of cognitive biases and other non-ideal decision-making behaviors that can happen in interactions with the DM. Shavarani et al. [25] proposed using a version of the Sigmoid

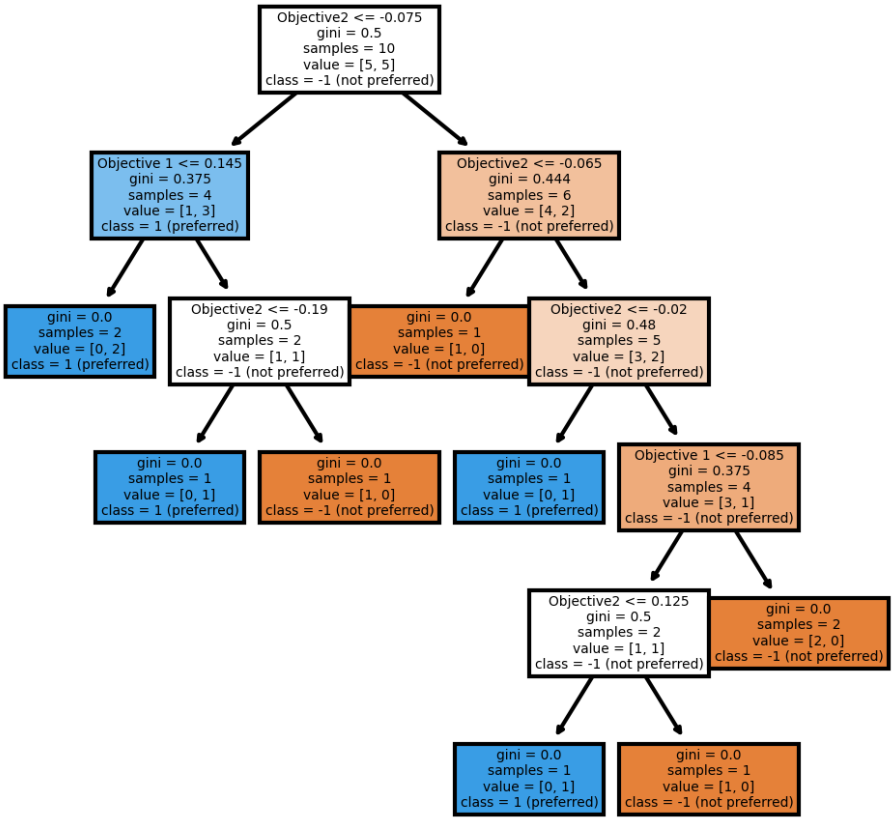

Fig. 2. A decision tree example built on the information elicited in one interaction on problem DTLZ1 with $m = 2$ objectives. Here objective l indicates the trade-off value for l^{th} objective, i.e., $z_{il} - z_{jl}$ when comparing solution i with solution j. The *samples* field indicates the number of samples that fall on that node. The c^{th} element of *value* in each node o indicates N_o^c, the number of samples that belong to the c^{th} class. Gini impurity is a measure (from 0 to 0.5) of the quality of the split.

UF introduced by Stewart [26] for experimenting on iEMOAs and modified it for minimization problems without violating any of its underlying assumptions. The modified UF used here (and hereafter called Stewart UF) is formulated as follows:

$$U'(\mathbf{z}) = \sum_{i=1}^{m} w_i u_i(z_i) \tag{3}$$

$$u_i(z_i) = \begin{cases} \lambda_i + \dfrac{(1-\lambda_i)(1-e^{-\beta_i(\tau_i - z_i)})}{1-e^{-\beta_i \tau_i}} & \text{if } 0 \le z_i \le \tau_i \\ \dfrac{\lambda_i \cdot (e^{\alpha_i(1-z_i)} - 1)}{e^{\alpha_i(1-\tau_i)} - 1} & \text{if } \tau_i < z_i \le 1 \end{cases}$$

Table 1. Description of different DM behaviors simulated by combinations of τ_i and λ_i for minimization problems (based on its counterpart for maximization [26]).

Type	τ_i	λ_i	**Description**
1	$[0.6, 0.9]$	$[0.1, 0.4]$	Mainly compensatory preferences.
2	$[0.6, 0.9]$	$[0.6, 0.9]$	Mainly compensatory preferences, but with sharp preference threshold.
3	$[0.1, 0.4]$	$[0.1, 0.4]$	Limited range of compensation, all **higher** values nearly equally undesirable.
4	$[0.1, 0.4]$	$[0.6, 0.9]$	Limited range of compensation, plus sharp preference threshold.

There are four parameters in Equation 3 that control the shape of this UF:

τ_i : Reference level, i.e., the point where losses are separated from gains with a steep inflation rate (loss can be interpreted as those values of the objective that are not satisfying to the DM).
λ_i : The utility value at the reference level.
α_i : The non-linearity of the function over losses.
β_i : The non-linearity of the function over the gains. Having $\alpha_i > \beta_i > 0$ satisfies various assumptions about the behavior of human DMs [25, 26].

One of the benefits of Stewart UF is the ability to simulate different decision-making behaviors, as depicted in Table 1. It has been shown that Stewart UF imposes more difficulties on the algorithms [25].

We also make use of a Tchebychef UF, which is formulated as follows:

$$U(\mathbf{z} = \mathbf{f}(\mathbf{x})) = \max_{i=1,\ldots,m} w_i |z_i - z_i^*|, \tag{4}$$

where w_i is the weight of each objective function, and $\mathbf{z}^*$ is the ideal or Utopian point. Since the selected benchmark problems are to be minimized and preserve consistency, it is assumed that the DM prefers solutions with lower utility values.

Aside from the tests under ideal conditions, we use the MDM capability of simulation of inconsistencies in the decisions of the DM. The MDM adds a normally distributed random noise with mean 0 and variance σ. The variance σ controls the amount of noise. To investigate the robustness of the algorithms, we test with $\sigma \in \{0.005, 0.01, 0.1, 0.2\}$.

3.2 Benchmark Problems

We follow [3, 18] and perform our experiments on well-known DTLZ benchmark problems [12]. DTLZ1, DTLZ2, DTLZ7 are selected from the DTLZ test suite. Each of these problems exposes different difficulties to the algorithm and allows one to investigate its performance from various aspects. DTLZ1 contains $11^k - 1$ local PFs, and each of them can attract an EMOA before reaching the global PF. DTLZ2 investigates the performance of an EMOA in getting close to true PF.

Finally, DTLZ7 has 2^{m-1} disconnected Pareto-optimal regions in the objective space. It is used to check the diversity of the solutions. The problem dimension n and the dimension of the objective space m are selected in a way that n is $m+4$ for DTLZ1, $m+9$ for DTLZ2 and $m+19$ for DTLZ7, as suggested in [12]; we consider $m \in \{2, 4, 10\}$. Also to make the problems more challenging, for DTLZ1 we follow [3] and limit x_i to $[0.25, 0.75]$, and for DTLZ2 we map x_i to $x_i/2 + 0.25$ as suggested by [8]. We consider all possible combinations of different problem sets, number of objective functions, different value functions and different decision-making behaviors. Full details about the set of 45 test configurations can be found in the supplementary materials [24].

3.3 Evaluating Performance and Competing Algorithms

Interactive methods are supposed to develop the best solution or a small subset of non-dominated solutions that maximize the DM's satisfaction. Thus, it makes sense to evaluate the performance of the interactive methods based on the utility value of the returned solution [1]. We compare the performance of DTEMOA against two state-of-the-art algorithms: BCEMOA [3] and iTDEA [18]. Preference learning in BCEMOA is similar to the way it is done by DTEMOA in that both learn to rank solutions, however, BCEMOA uses support vector machines (SVM) for preference learning. iTDEA uses a completely different preference learning scheme, where the preferences of the DM are reflected in the search process by prioritizing solutions in the proximity of the DM's selected solution. Both BCEMOA and DTEMOA use NSGA-II as their underlying optimizer. iTDEA still uses the same non-domination and mating methods, but in each generation only one solution is created and whether or not it is accepted to the population depends on the position of that solution in the objective space and its distance to its closest solution in the population. The iTDEA is sensitive to the threshold parameter that controls the acceptable distance for the new solution. A large threshold would prevent solutions to enter the population, while a small one would make the population grow uncontrolled and increase computational costs, specially in many-objective problems. These similarities and differences were the main motivation behind selecting these two algorithms.

3.4 Algorithm Parameter Settings

The parameters of the selected algorithms are illustrated in Table 2. iTDEA generates and evaluates only one solution per generation. Thus, the number of generations is set to $N^{\text{gen}} = 80\,000$ to have an equal number of objective evaluations in all compared algorithms. As suggested in [18], the first interaction of iTDEA happens after $\frac{N^{\text{gen}}}{3}$ generations, and the number of generations between each subsequent interaction is given by $gen_i = \frac{N^{\text{gen}}}{2(N^{\text{int}}-1)}$. Thus, with a larger number of interactions, gen_1 becomes smaller. The experimental study will investigate the impact of different numbers of interactions, N^{int}. The initial and final territory parameters of iTDEA are respectively set to 0.1 and 0.00001

Table 2. Parameter settings of the iEMOAs. In addition, total solution evaluations is 80 000, the population size $|pop| = 200$, the number of iterations is $N^{\text{int}} \in \{2, 3, 4\}$, and the DM ranks $N^{\text{sol}} = 5$ solutions per interaction. After N^{int} interactions, the algorithms continue running without further interaction until reaching N^{gen} generations.

Parameter	iTDEA	BCEMOA	DTEMOA
Total generations (N^{gen})	80 000	400	400
Generations before 1st interaction (gen_1)	$N^{\text{gen}}/3$	200	200
Generations between further interactions (gen_i)	$\frac{N^{\text{gen}}}{2(N^{\text{int}}-1)}$	20	20

in problems with two objectives, which was one of the alternatives suggested in [18]. For problems with $m = 4$ and 10 objectives, these values change to 0.5 and 0.25, respectively. Any smaller values for these parameters would make the size of the archive population large and the computational costs unaffordable. Other parameters for BCEMOA and iTDEA are set as suggested in [3, 18]. The hyper-parameters of the DT in DTEMOA and SVM in BCEMOA are tuned by grid search cross-validation at each interaction [24].

We run experiments with a different number of interactions $N^{\text{int}} \in \{2, 3, 4\}$ to test the effect of interactions on the results. Our initial experiments indicate that a higher number of interactions does not seem to increase the quality of the solutions substantially. Each experiment is repeated 40 times with different random seeds.

Implementations. BCEMOA, DTEMOA, iTDEA, MDM and utility functions are implemented in Python version 3.7.6. The NSGA-II and DTLZ benchmark implementations were acquired from the Pygmo library 2.16.0 [4], the SVM ranking model from the Preference Learning Toolbox [13] powered by scikit-learn 0.23.1 [21]. Scikit-learn is also used to implement the DT models.

4 Results and Discussion

This section is divided into two parts. In the first part, we focus on our proposed preference learning technique and provide some insight into its performance. In the second part, the performance of the proposed DTEMOA is compared with iTDEA and BCEMOA.

4.1 Assessing Ranking Performance

One of the main differentiating characteristics of iEMOAs is the way they adapt to the DM's preferences and the way they learn and model the preferences [28]. The methods compared in our study differ significantly in how they interact and adapt to the DM. Both BCEMOA and DTEMOA try to learn a model to rank solutions but use different learning techniques. We can directly compare their accuracy independently of other algorithmic aspects. We evaluate the accuracy of the preference learning models of BCEMOA and DTEMOA on a

Table 3. Mean accuracy (and standard deviation) of preference learning in DTEMOA and BCEMOA on the DTLZ7 problem for different UFs. The different number of interactions indicate how the algorithms exploit the accumulated preference data over interactions. The number of solutions presented to the DM at each interaction is $N^{\text{sol}}= 5$. The best accuracy for each UF, m and N^{int} is highlighted in bold face.

		$m = 2$		$m = 4$		$m = 10$	
UF	N^{int}	BCEMOA	DTEMOA	BCEMOA	DTEMOA	BCEMOA	DTEMOA
Stewart	2	79.4(11.3)	84.8(12.1)	84.2(7.1)	83(7.2)	68.1(7.2)	62.2(6.5)
	3	82.9(9)	84.9(11.4)	82.2(7)	82.9(5.7)	69.5(5.8)	67(6)
	4	85.7(7.7)	86.5(5.5)	81.6(7.1)	83.5(2.9)	71.1(6.1)	69.3(6.3)
	5	**87.5(8.1)**	84(11.3)	81(6.7)	81.8(6.5)	70.3(4.5)	**71.3(7.1)**
Tchebychef	2	**100(0)**	99.9(0.2)	75.5(11.4)	95(10.1)	67.3(5.8)	96.4(8.5)
	3	**100(0)**	99.7(1.1)	74(7.8)	96.8(8.8)	67.1(6.5)	98.8(3.4)
	4	**100(0)**	**100(0)**	75.6(5.8)	97(9.6)	67.7(5.8)	97.2(9.9)
	5	**100(0)**	**100(0)**	76.7(5.1)	**98.2(9.9)**	68.3(6)	**100(0)**

randomly-generated population of 400 solutions for DTLZ7 problem with different number of objectives $m \in \{2, 4, 10\}$. It is also interesting to see how each algorithm exploits accumulated information elicited in different interactions. Thus, we simulate 5 interactions. Before each interaction, NSGA-II is used to evolve the population for 20 generations, and 5 solutions are selected randomly from the non-dominated front and ranked by a UF. The preference learning of both algorithms is applied to the elicited data. Then the accuracy of each model in ranking the non-dominated solutions in the population is measured by counting the proportion of correct pairwise rankings to all possible pairwise rankings:

$$\text{Acc} = 100 \cdot \frac{\sum_{i=1}^{|pop|-1} \sum_{j=i+1}^{|pop|} \text{I}\{r_i < r_j \wedge \hat{r}_i < \hat{r}_j\}}{\sum_{i=1}^{|pop|-1} \sum_{j=i+1}^{|pop|} \text{I}\{r_i < r_j\}}, \tag{5}$$

where I is the indicator function which is equal to 1 if the given condition is true, otherwise 0; $r_i = U(\mathbf{f}(\mathbf{x}_i))$ is the true rank of the solution $\mathbf{x}_i \in pop$ according to the UF, and $\hat{r}_i$ is the rank predicted by the learning models for the same solution. The experiments are repeated 40 times for each problem.

Table 3 summarizes the results. In particular, the accuracy of both methods is similar with 2 objectives. However, the preference learning technique in DTEMOA performs better when the number of objectives increases.

4.2 Comparison of the Performance with other iEMOAs

The overall performance of the algorithms, measured in terms of the true utility of the returned solution (lower values are better), over all tests is illustrated in Tables 4 and 5. Table 4 summarizes the performance of the algorithms over different number of interactions. Table 5 show the results over different number

Table 4. Performance of the algorithms over different number of interactions under ideal conditions. The utility values are averaged over all tests with all possible combinations of different problems, number of objective functions, different UFs and different decision-making behaviors. The results are rounded to 3 decimal places. The p-values obtained from the Wilcoxon test using Holm's method to adjust for multiple comparisons are also reported.

	Mean utility value			Wilcoxon test's p-values		
N^{int}	BCEMOA	DTEMOA	iTDEA	DTEMOA vs. BCEMOA	DTEMOA vs. iTDEA	BCEMOA vs. iTDEA
2	0.206	0.184	0.214	0.000	0.032	0.192
3	0.206	0.179	0.259	0.000	0.000	0.000
4	0.203	0.177	0.278	0.000	0.000	0.000

of objective functions categorized into experiments under ideal conditions and those with simulation of inconsistent decisions. The results of this table are over experiments with 3 interactions, which is reasonable number of interactions based on the observations in Table 4. The results are aggregated over all tests with different UFs, problems and decision making behaviors. DTEMOA generally returns a solution with a better true utility than BCEMOA and iTDEA. The Wilcoxon test indicates that the differences in utility are significant (p-value ≈ 0). The second important observation, as illustrated in Table 5, is the robustness of the DTEMOA towards inconsistencies in the DM's decisions. Considering all the results over all tests, the performance of DTEMOA has declined by 1.6% when inconsistent decisions are simulated; this change is insignificant compared to the deterioration in the performance of BCEMOA (17%) and iTDEA (3.5%). Furthermore, the superiority of DTEMOA becomes more significant when the number of objectives increases.

Due to space limitations, we have not presented results for each test but reported only the aggregated results. The interested reader is referred to the supplementary materials [24] for the full set of results.

5 Conclusions

There have been many improvements in the field of interactive methods, and yet it seems there is room for more. Recent research on benchmarking of these methods has revealed that they may not perform as expected when confronted with non-ideal conditions that were not anticipated in their development [1, 25]. At present, the field has little idea of how to explain or predict which components of iEMOA methods are most vulnerable to non-idealities or what conditions affect their performance the most. However, we believe the core of any interactive method is the preference model and the accuracy of these models in

Table 5. Comparing the performance of algorithms over various number of objective functions (m). The utility values are averaged over all tests with all possible combinations of different problem sets, different UFs and different decision-making behaviors ($N^{\text{int}}= 3$). The results are rounded to 3 decimal places. Algorithms are compared pairwise, reporting the p-value for a Wilcoxon test using Holm's method to adjust for multiple comparisons.

	Under ideal conditions						With simulation of DM inconsistencies					
	mean utility value			p-value			mean utility value			p-value		
m	BCEMOA	DTEMOA	iTDEA	DTEMOA vs. BCEMOA	DTEMOA vs. iTDEA	BCEMOA vs. iTDEA	BCEMOA	DTEMOA	iTDEA	DTEMOA vs. BCEMOA	DTEMOA vs. iTDEA	BCEMOA vs. iTDEA
2	0.270	0.270	0.256	1.000	1.000	1.000	0.292	0.266	0.268	0.000	0.012	0.055
4	0.215	0.193	0.319	0.002	0.000	0.000	0.244	0.200	0.328	0.000	0.000	0.000
10	0.122	0.077	0.097	0.000	0.652	0.000	0.116	0.081	0.103	0.000	0.158	0.000

reflecting the DM's preferences is crucial in the optimization process. With these observations, this study proposed an innovative preference model and learning approach using decision trees to predict the result of pairwise comparisons from the DM's perspective to manage the convergence direction of the population. The method's performance is found to be stable in many-objective problems because more objectives translate to more attributes that can participate in classification. Further, when the number of objectives increases, DTs can naturally identify the most important ones. Our method is also found to be more robust to biases as we make holistic comparisons to cancel out any estimation errors in single predictions. In contrast, other learning methods such as SVM rely on kernel performance, and complications such as higher dimensions, nonlinearity, and biases make kernel selection an intensive task and generally less accurate.

Another critical advantage of DTs in this context is their intuitive interpretation and ease of visualization, both being essential elements in gaining the DM's trust in the process and the results. We used the suggested experimental design of [25] to evaluate the performance of the algorithm and examine the efficiency of the method. Further research may extend this study by investigating the application of weighted decision trees to emphasize the most recently elicited information. Another interesting research direction is to explore the use of decision tree regressors instead of classifiers in the preference model.

Reproducibility. We make source code and data for reproducing our results publicly available as supplementary materials [24] to motivate further research in this direction.

References

1. Afsar, B., Miettinen, K., Ruiz, F.: Assessing the performance of interactive multiobjective optimization methods: A survey. ACM Computing Surveys **54**(4) (2021), doi: 10.1145/3448301
2. Allen, J., Moussa, A., Liu, X.: Human-in-the-loop learning of qualitative preference models. In: Barták, R., Brawner, K.W. (eds.) Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, pp. 108–111, AAAI Press (2019)
3. Battiti, R., Passerini, A.: Brain-computer evolutionary multiobjective optimization: A genetic algorithm adapting to the decision maker. IEEE Transactions on Evolutionary Computation **14**(5), 671–687 (2010), doi: 10.1109/TEVC.2010.2058118
4. Biscani, F., Izzo, D., Yam, C.H.: A global optimisation toolbox for massively parallel engineering optimisation. Arxiv preprint arXiv:1004.3824 (2010), URL http://arxiv.org/abs/1004.3824
5. Branke, J., Deb, K., Miettinen, K., Słowiński, R. (eds.): Multiobjective Optimization: Interactive and Evolutionary Approaches, Lecture Notes in Computer Science, vol. 5252. Springer, Heidelberg (2008)
6. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and regression trees. CRC Press (1984)
7. Brockhoff, D., Zitzler, E.: Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In: Runarsson, T.P., Beyer, H.G., Burke, E.K., Merelo, J.J., Whitley, D., Yao, X. (eds.) Parallel Problem Solving from Nature – PPSN IX, Lecture Notes in Computer Science, vol. 4193, pp. 533–542, Springer, Heidelberg (2006)
8. Brockhoff, D., Zitzler, E.: Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In: Proceedings of the 2007 Congress on Evolutionary Computation (CEC 2007), pp. 2086–2093, IEEE Press, Piscataway, NJ (2007), doi: 10.1109/CEC.2007.4424730
9. Cheng, W., Hühn, J., Hüllermeier, E.: Decision tree and instance-based learning for label ranking. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) Proceedings of the 26th International Conference on Machine Learning, ICML 2009, pp. 161–168, ACM Press, New York, NY (2009), doi: 10.1145/1553374.1553395
10. Dalip, D.H., Gonçalves, M.A., Cristo, M., Calado, P.: Exploiting user feedback to learn to rank answers in q&a forums: A case study with stack overflow. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, p. 543–552, SIGIR '13, Association for Computing Machinery, New York, NY, USA (2013), ISBN 9781450320344, doi: 10.1145/2484028.2484072, URL https://doi.org/10.1145/2484028.2484072
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002), doi: 10.1109/4235.996017
12. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization, pp. 105–145, Advanced Information and Knowledge Processing, Springer, London, UK (Jan 2005), doi: 10.1007/1-84628-137-7_6
13. Farrugia, V.E., Martínez, H.P., Yannakakis, G.N.: The preference learning toolbox (2015), URL arXiv:1506.01709

14. Greco, S., Matarazzo, B., Słowiński, R.: Interactive evolutionary multiobjective optimization using dominance-based rough set approach. In: Ishibuchi, H., et al. (eds.) Proceedings of the 2010 Congress on Evolutionary Computation (CEC 2010), pp. 1–8, IEEE Press, Piscataway, NJ (2010)
15. Herbrich, R., Graepel, T., Obermayer, K.: Support vector learning for ordinal regression. In: ICANN'99: Proceedings of the 9th International Conference on Artificial Neural Networks, pp. 97–102 (1999), doi: 10.1049/cp:19991091
16. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: Proceedings of the 2008 Congress on Evolutionary Computation (CEC 2008), pp. 2419–2426, IEEE Press, Piscataway, NJ (2008), doi: 10.1109/CEC.2008.4631121
17. Khare, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) Evolutionary Multi-criterion Optimization, EMO 2003, Lecture Notes in Computer Science, vol. 2632, pp. 376–390, Springer, Heidelberg (2003)
18. Köksalan, M., Karahan, İ.: An interactive territory defining evolutionary algorithm: iTDEA. IEEE Transactions on Evolutionary Computation **14**(5), 702–722 (Oct 2010), doi: 10.1109/TEVC.2010.2070070
19. López-Ibáñez, M., Knowles, J.D.: Machine decision makers as a laboratory for interactive EMO. In: Gaspar-Cunha, A., Antunes, C.H., Coello Coello, C.A. (eds.) Evolutionary Multi-criterion Optimization, EMO 2015 Part II, Lecture Notes in Computer Science, vol. 9019, pp. 295–309, Springer, Heidelberg (2015), doi: 10.1007/978-3-319-15892-1_20
20. Misitano, G., Afsar, B., Larraga, G., Miettinen, K.: Towards explainable interactive multiobjective optimization: R-XIMO. Autonomous Agents and Multi-Agent Systems **36**(42) (2022), doi: 10.1007/s10458-022-09577-3
21. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
22. Purshouse, R.C., Deb, K., Mansor, M.M., Mostaghim, S., Wang, R.: A review of hybrid evolutionary multiple criteria decision making methods. COIN Report 2014005, Computational Optimization and Innovation (COIN) Laboratory, University of Michigan, USA (Jan 2014)
23. Quan, G., Greenwood, G.W., Liu, D., Hu, S.: Searching for multiobjective preventive maintenance schedules: Combining preferences with evolutionary algorithms. European Journal of Operational Research **177**(3), 1969–1984 (2007), doi: 10.1016/j.ejor.2005.12.015
24. Shavarani, S.M., López-Ibáñez, M., Allmendinger, R., Knowles, J.D.: An interactive decision tree-based evolutionary multi-objective algorithm: Supplementary material. https://doi.org/10.5281/zenodo.7429806 (2022)
25. Shavarani, S.M., López-Ibáñez, M., Knowles, J.D.: Realistic utility functions prove difficult for state-of-the-art interactive multiobjective optimization algorithms. In: Chicano, F., Krawiec, K. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 457–465, ACM Press, New York, NY (2021), doi: 10.1145/3449639.3459373
26. Stewart, T.J.: Robustness of additive value function methods in MCDM. Journal of Multi-Criteria Decision Analysis **5**(4), 301–309 (1996)
27. Xia, F., Zhang, W., Li, F., Yang, Y.: Ranking with decision tree. Knowledge and information systems **17**(3), 381–395 (2008), doi: 10.1007/s10115-007-0118-y

28. Xin, B., Chen, L., Chen, J., Ishibuchi, H., Hirota, K., Liu, B.: Interactive multiobjective optimization: A review of the state-of-the-art. IEEE Access **6**, 41256–41279 (2018), doi: 10.1109/ACCESS.2018.2856832
29. Yu, P.L.H., Wan, W.M., Lee, P.H.: Decision tree modeling for ranking data. In: Fürnkranz, J., Hüllermeier, E. (eds.) Preference Learning, pp. 83–106, Springer, Heidelberg (2011), ISBN 978-3-642-14125-6, doi: 10.1007/978-3-642-14125-6_5
30. Zhao, H., Ram, S.: Constrained cascade generalization of decision trees. IEEE Transactions on Knowledge and Data Engineering **16**(6), 727–739 (2004), doi: 10.1109/TKDE.2004.3
31. Zujevs, A., Eiduks, J.: Adaptive multi–objective optimization procedure model. In: Scientific Proceeding of Riga Technical University, vol. 5, pp. 46–54 (2008)