Hindawi
Complexity
Volume 2018, Article ID 7837696, 14 pages
https://doi.org/10.1155/2018/7837696

Research Article

MOFSRank: A Multiobjective Evolutionary Algorithm for Feature Selection in Learning to Rank

Fan Cheng,[1,2] **Wei Guo**,[2] **and Xingyi Zhang**[1,2]

[1] *Key Laboratory of Intelligent Computing & Signal Processing of Ministry of Education, Anhui University, Hefei 230039, China*
[2] *School of Computer Science and Technology, Anhui University, Hefei 230601, China*

Correspondence should be addressed to Xingyi Zhang; xyzhanghust@gmail.com

Received 28 May 2018; Revised 23 October 2018; Accepted 10 November 2018; Published 2 December 2018

Academic Editor: Rongqing Zhang

Learning to rank has attracted increasing interest in the past decade, due to its wide applications in the areas like document retrieval and collaborative filtering. Feature selection for learning to rank is to select a small number of features from the original large set of features which can ensure a high ranking accuracy, since in many real ranking applications many features are redundant or even irrelevant. To this end, in this paper, a multiobjective evolutionary algorithm, termed MOFSRank, is proposed for feature selection in learning to rank which consists of three components. First, an instance selection strategy is suggested to choose the informative instances from the ranking training set, by which the redundant data is removed and the training efficiency is enhanced. Then on the selected instance subsets, a multiobjective feature selection algorithm with an adaptive mutation is developed, where good feature subsets are obtained by selecting the features with high ranking accuracy and low redundancy. Finally, an ensemble strategy is also designed in MOFSRank, which utilizes these obtained feature subsets to produce a set of better features. Experimental results on benchmark data sets confirm the advantage of the proposed method in comparison with the state-of-the-arts.

1. Introduction

As a central issue of many applications, such as document retrieval [1], collaborative filtering [2], and expert finding [3], learning to rank has attracted much focus in machine learning area during the last decade. Rank learning, when applied to document retrieval, is a task as follows [1]. In learning, a ranking model is constructed by using the training data that consists of queries, their corresponding retrieved documents, and relevance levels given by human annotators. In ranking, given a new query, the documents are sorted by using the trained ranking model.

Due to the wide usages, a great number of learning to rank algorithms have been proposed, which achieve the ranking models with high accuracies [4–11]. However, in several real ranking applications, such as image retrieval [12, 13] and biomarker finding [14], the number of features in training data is large, which brings great challenges to existing ranking methods, since many features in these applications are redundant or even irrelevant, which reduces the performance of ranking algorithms [15]. To tackle the issue, recently, considerable efforts have been made on designing feature selection algorithms for learning to rank. For example, Geng et al. proposed the first *filter* based work, termed Greedy Search Algorithm (GAS) for feature selection in learning to rank [15]. In GAS, the feature that maximized total importance scores and minimized total similarity scores was iteratively selected to obtain the final feature subset. Experimental results demonstrated the effectiveness of GAS, when compared with traditional ranking algorithms. Since then, many other *filter* based ranking algorithms have been developed [16–19]. Another type of feature selection algorithms for learning to rank belongs to the *wrapper* approach, where a rank learning algorithm is included in the feature selection procedure to create a good feature subset. BRTree [20], RankWrapper [21], BFS-Wrapper [22], and GreedyRankRLS [23] are the representative works of this type. Recently, *embedded* methods have been proposed to solve feature selection for learning to rank, where feature selection is embedded in the ranker construction by introducing a sparse regularization term. For example, RSRank [24], FenchelRank [25], and FSMRank [26] adopted L1 regularization term, whereas in the work of [27], an *embedded* based feature selection algorithm by using a nonconvex regularization was suggested.

The existing feature selection algorithms for learning to rank have shown promising performance in achieving the features with small number and high ranking accuracy. However, all these algorithms solve the problem by only considering the traditional optimization techniques, such as greedy method and gradient descent method. Different from them, in this paper, we tackle the issue by using evolutionary computation as the optimization technique. To be specific, a **M**ulti-**O**bjective evolutionary algorithm for **F**eature **S**election in learning to **R**ank, named MOFSRank is proposed. The main contributions of this paper can be summarized as follows:

(1) A multiobjective feature selection method with an adaptive mutation is suggested, where the features with high ranking accuracy and low redundancy are selected as the feature subsets. Based on the suggested method, a multiobjective evolutionary algorithm, named MOFSRank, is proposed for feature selection in learning to rank.

(2) In MOFSRank, an instance selection strategy is developed to choose the informative instances from the training data, by which the redundant data is removed and the learning process of feature selection is sped up. In addition, an ensemble strategy is also designed in MOFSRank, where the selected feature subsets are further utilized to produce a set of better features.

(3) The effectiveness of the proposed MOFSRank is evaluated on the benchmark data sets, and the experimental results show that compared with the existing work the algorithm we proposed has superior performance in terms of both ranking accuracy and number of selected features.

The remainder of the paper is organized as follows. In Section 2, the preliminaries and related work are presented. Section 3 gives the details of the proposed algorithm and empirical results by comparing our algorithm with several state-of-the-arts on the benchmark data sets are reported in Section 4. Section 5 concludes the paper and discusses the future work.

2. Preliminaries and Related Work

2.1. Learning to Rank. Learning to rank, when applied to document retrieval, can be described as a problem as follows. Assuming that there is a collection of m queries for training, denoted as $\{q_1, q_2, \ldots, q_m\} \in Q$, each query q_k $(1 \leq k \leq m)$ is associated with a list of n_k documents, $d_k = \{d_k^1, d_k^2, \ldots, d_k^{n_k}\}$, whose relevance to q_k is given by a vector $y_k = (y_k^1, y_k^2, \ldots, y_k^{n_k})$, where $y_k^j \in \{r_1, r_2, \ldots, r_l\}$ $(1 \leq j \leq n_k)$ and l is the number of ranks. There exists a total order between the ranks $r_l \succ r_{l-1} \succ \ldots r_1$, where $\succ$ denotes the partial order. With the training data, learning to rank is to construct a ranking model r, which for a given new query q can rank the documents associated with q such that more relevant documents are ranked higher than less relevant ones.

To obtain accurate ranking models, different learning to rank algorithms have been proposed, which can be divided into three categories: *Pointwise* approach, *Pairwise* approach, and *Listwise* approach [1]. The *Pointwise* approach uses each single document as a learning instance, and defines the loss function on individual documents [4, 28]. The *Pairwise* approach regards a pair of documents as a learning instance and transforms the ranking problem into binary classification on document pairs [5–7]. The *Listwise* approach solves the ranking problem in a straightforward fashion, which takes the entire ranked list of documents as a learning instance and defines a *Listwise* loss function for learning [8–11]. Among these three approaches, the *Pairwise* one has attracted much focus, since in the real ranking applications, such as search engine and recommendation system, the training data of this category can be easily obtained from the users' click through [5]. More algorithms for learning to rank can be found in [1].

2.2. Feature Selection Methods for Learning to Rank. The different types of ranking algorithms have shown promising performance in achieving the models with high accuracy. However, in several real ranking applications, the number of training features is large, which brings great challenges to learning to rank algorithms. To tackle the issue, recently, researchers introduced feature selection to the ranking methods and a variety of feature selection algorithms for learning to rank have been suggested, which mainly fall into three categories: *filter* approach, *wrapper* approach, and *embedded* approach [27, 29].

The *filter* approach is independent of the ranking method, and one representative work is GAS proposed by Geng, which is also the first feature selection algorithm for learning to rank [15]. The basic idea of GAS is to select a subset of features with maximum total importance scores and minimum total similarity scores and use selected features to construct a ranking model. Experimental results on LETOR data sets have shown that GAS can achieve good ranking accuracy with a small number of features. Based on this work, several other *filter* based feature selection algorithms have been developed [16–19, 30].

Different from *filter* approach, the *wrapper* approach includes a rank learning algorithm in the feature subset evaluation step, where the ranking algorithm is used as a black box by a wrapper to evaluate the goodness (i.e. the ranking accuracy) of the selected features. Example algorithms include BRTree, which uses boosted regression trees [20], RankWrapper with Ranking SVM [21], BFS-Wrapper utilizing search [22], GreedyRankRLS with Rank RLS algorithm [23], and LMIR using smoothing language model [31].

Recently, *embedded* approach (Note that some researchers categorize feature selection algorithms into two groups, where *embedded* approach is included in *wrapper* approach.) has been suggested to solve feature selection for learning to rank, where feature selection and rank learning are integrated into one single process. For example, Sun et al. [24] proposed an *embedded* feature selection ranking algorithm, termed RSRank, where L1 regularization term was introduced into

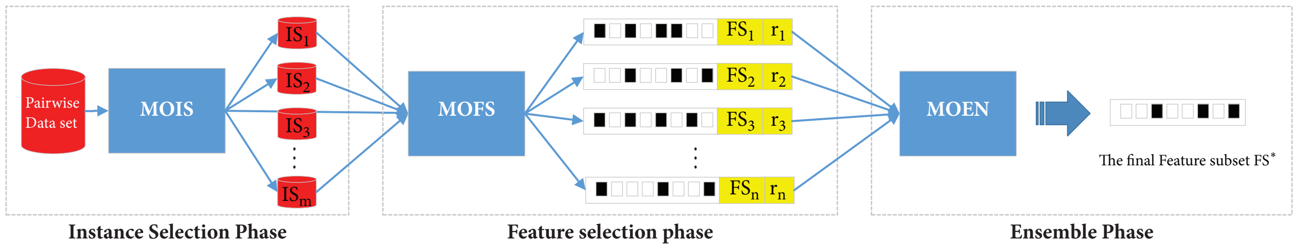


Figure 1: The Main Procedure of MOFSRank.

ranking optimization. The experiment results on OHSUMED and TD2003 data sets have shown that RSRank outperformed several baseline rankers with only selecting thirty percent features. In recognizing the competitiveness of RSRank, many other *embedded* based methods have emerged. Lai et al. suggested another L1 based ranking method, named FenchelRank [25], where Fenchel duality was used to solve the sparse ranking optimization. Empirical evaluations indicated that FenchelRank was not only better than the classical ranking algorithms but also provided better performance than RSRank. Following this work, Lai et al. further developed a new *embedded* feature selection algorithm for learning to rank, termed FSMRank [26]. The algorithm solved a joint convex optimization problem by simultaneously minimizing ranking error and conducting feature selection. Experiments on the LETOR collections demonstrated that FSMRank can obtain better results than the *filter* approach, such as GAS. Different from the algorithms above, which all used convex L1 regularization, Laporte et al. designed a feature selection algorithm for learning to rank with a nonconvex regularization, which resulted in both good ranking accuracy and a small number of selected features [27]. Other embedded feature selection ranking algorithms can also be found in [32, 33].

The algorithms mentioned above have shown the effectiveness of feature selection for learning to rank, and in this paper, we continue this research line by proposing a multiobjective evolutionary algorithm for ranking feature selection. Before giving the details of the proposed algorithm, it should be noted that recently, multiobjective evolutionary algorithms (MOEAs) have been successfully applied to solve different problems in machine learning areas, such as classification [34–36], clustering [37, 38], and pattern mining [39]. In the following, we will propose an MOEA for feature selection in learning to rank.

3. The Proposed Algorithm

The proposed algorithm (MOFSRank) is a feature selection algorithm. To be specific, it is a multiobjective feature selection algorithm for learning to rank, where Pairwise documents are used as the learning instances. To select feature subset from the training set with $O(n^2)$ size (n is the number of training data), we first choose some informative instance subsets from the *Pairwise* training set and then feature selection is performed on those selected instance subsets. Lastly, the outputs of feature selection are combined together to achieve a better feature subset. The main procedure of MOFSRank is shown in Figure 1, which consists of three phases: (1) instance selection phase, (2) feature selection phase, (3) and ensemble phase. In the first phase, a multiobjective evolutionary algorithm, termed MOIS, is suggested to select the informative instances from the original *Pairwise* training set, which has two advantages. First, it removes the possible noisy data in the original set and improves the quality of training set. Second, the instance selection reduces the number of training instances and makes the feature selection more efficient. In the second phase, the final nondominated solutions of MOIS are used for feature selection. To this end, an MOEA for feature selection (MOFS) is proposed, where ranking accuracy and number of the selected features are defined as two optimization objectives. In addition, an adaptive mutation probability is also designed in MOFS, by which the proposed method can choose the features with high ranking accuracy and low redundancy. In the last phase, a mixed coding based multiobjective ensemble algorithm, namely, MOEN, is developed, where the Pareto solutions in the second phase are utilized to produce a better feature subset as the final output. The framework of the proposed MOFSRank is demonstrated in Algorithm 1.

3.1. Instance Selection Phase.

As mentioned before, in this paper, we focus on *Pairwise* ranking, whose training set is of $O(n^2)$ size, where n is the number of training data. Thus, before feature selection, an instance selection operation is carried on the *Pairwise* training set. To be specific, an MOEA named MOIS is proposed for instance selection, where two optimization objectives are the number of selected instances and the value of $1 - RAccuracy$ (in general, the larger value of $RAccuracy$ ($0 \le RAccuracy \le 1$) means better ranking performance; however, since the multiobjective optimization problem is often described as a minimum problem, thus, in this paper, we use $1 - RAccuracy$ as the second objective.), where $RAccuracy$ denotes the accuracy value measured by ranking metrics, such as NDCG or MAP. Thus, the corresponding multiobjective instance selection problem can be described as

$$MOP1: \begin{cases} \min f_1 = n(IS) \\ \min f_2 = 1 - RAccuracy_{r_{IS}} \end{cases} \quad (1)$$

where IS denotes the selected instance subset, $n(IS)$ is the number of the instances in IS, and r_{IS} is a ranker learned on IS set. In this paper, we adopt linear SVM to create the ranker, which has been widely used in many feature selection

```
Input: $maxgen_1$: maximum generations of multi-objective instance selection, $pop_1$: population size of
multi-objective instance selection, $p_{c_1}$: crossover probability of multi-objective instance selection,
$p_{m_1}$: mutation probability of multi-objective instance selection, $maxgen_2$: maximum generations
of multi-objective feature selection, $pop_2$: population size of multi-objective feature selection, $p_{c_2}$: crossover
probability of multi-objective feature selection, $p_{m_2}$: mutation probability of multi-objective
feature selection, $maxgen_3$: maximum generations of multi-objective ensemble, $pop_3$: population size of
multi-objective ensemble, $p_{c_3}$: crossover probability of multi-objective ensemble, $p_{m_3}$: mutation
probability of multi-objective ensemble;
Output: The final selected feature subset $FS^*$;
(1) $OTDS \longleftarrow$ Reading original Pairwise training data set;
    /**Instance Selection Phase**/
(2) $InstanceSubset \longleftarrow$ MOIS($maxgen_1$,$pop_1$,$p_{c_1}$,$p_{m_1}$,$OTDS$);
    /**Feature Selection Phase**/
(3) ($FeatureSubset, rset$) $\longleftarrow$ MOFS($maxgen_2$,$pop_2$,$p_{c_2}$,$p_{m_2}$,$InstanceSubset$);
    /**Ensemble Phase**/
(4) $FS^* \longleftarrow$ MOEN($maxgen_3$,$pop_3$,$p_{c_3}$,$p_{m_3}$,$FeatureSubset$,$rset$);
(5) Return $FS^*$;
```

Algorithm 1: The Framework of MOFSRank.

```
Input: $maxgen_1$: maximum generations of multi-objective instance selection, $pop_1$: population
size of multi-objective instance selection, $p_{c_1}$: crossover probability of multi-objective instance selection,
$p_{m_1}$: mutation probability of multi-objective instance selection, $OTDS$: original training data set;
Output: A set of non-dominated instance subsets $InstanceSubset$;
(1) Initializing the population $P_1 = \{IS_1, \dots, IS_{pop_1}\}$;
(2) for $i = 1$ to $maxgen_1$ do
(3)     $P_i \longleftarrow$ Evaluating $P_i$ by two proposed objectives; // formula (1)
(4)     $M_i \longleftarrow$ Binary Tournament ($P_i$);
(5)     $Q_i \longleftarrow$ Variation ($M_i$, $p_{c_1}$, $p_{m_1}$);
(6)     $P_{i+1} \longleftarrow$ Environmental ($P_i \cup Q_i$);
(7) end for
(8) $InstanceSubset \longleftarrow$ selecting the solutions on the Pareto front;
(9) Return $InstanceSubset$;
```

Algorithm 2: MOIS($maxgen_1$, pop_1, p_{c_1}, p_{m_1}, $OTDS$).

algorithms for learning to rank, such as FenchelRank and FSMRank. $RAccuracy_{r_{IS}}$ is ranking accuracy of the learned ranker r_{IS} on original training set.

For the MOP1, we use binary encoding scheme, which means that the i-th individual (instance subset) can be represented as $IS_i = (is_{i,1}, \dots, is_{i,n})$, where $is_{i,j} \in \{0,1\}$, $j \in \{1, \dots, n\}$, n is the total number of the instances in the original training set. If $is_{i,j} = 1$ denotes that the j-th instance is selected in the i-th individual, otherwise means not. With this encoding scheme, the proposed MOIS adopts a similar framework of NSGA-II [40], and Algorithm 2 presents the procedure of MOIS in detail.

3.2. Feature Selection Phase. We take the non-dominated solutions of MOIS as the training data sets, and the feature selection is carried on them. To this end, a bi-objective evolutionary algorithm with an adaptive mutation for feature selection (MOFS) is suggested, where two conflicting objectives are the number of features and the value of 1 − *RAccuracy*. Thus, the biobjective optimization problem for feature selection is defined as

$$MOP2: \begin{cases} \min f_1 = n(FS) \\ \min f_2 = 1 - RAccuracy_{r_{FS}} \end{cases} \tag{2}$$

where FS denotes the selected feature subset and $n(FS)$ is the number of features in FS set. r_{FS} is the ranker learned with features in FS set. Since each FS is evaluated on the Pareto instance subsets, we choose the largest value as the value of $RAccuracy_{r_{FS}}$.

We also use the binary encoding scheme for the MOP2. Thus, the i-th individual (feature subset) in population is represented as $FS_i = (fs_{i,1}, \dots, fs_{i,d})$, where $fs_{i,j} \in \{0,1\}$, $j \in \{1, \dots, d\}$, d is the total number of features in $OTDS$ (original training data set). $fs_{i,j} = 1$ denotes that the j-th feature is included in the i-th individual, otherwise means not. With the binary encoding strategy, we solve the MOP2 by adopting a similar framework as NSGA-II. To further improve the

```
Input: $maxgen_2$: maximum generations of multi-objective feature selection, $pop_2$: population size of multi-objective
feature selection, $p_{c_2}$: crossover probability of multi-objective feature selection, $p_{m_2}$: mutation probability
of multi-objective feature selection, $InstanceSubset$: a set of non-dominated instance subsets;
Output: a set of non-dominated feature subsets $FeatureSubset$, and their corresponding rankers set $rset$;
(1) Initializing the population $P_1 = \{FS_1, \ldots, FS_{pop_2}\}$;
(2) for $i = 1$ to $maxgen_2$ do
(3)   /*Evaluating $P_i$ by two proposed objectives with formula (2)*/
(4)   for $j = 1$ to $pop_2$ do
(5)     $f_{j,1} \longleftarrow$ calculating the number of non-zero features in $FS_j$; //the first objective value of $j$-th individual $FS_j$
(6)     $r_j \longleftarrow \mathrm{argmin}_{1\le l\le |InstanceSubset|}(r_{j,l} \mid 1 - RAccuracy_{r_{j,l}})$; //select the ranker with the smallest value of
        $1 - RAccuracy$ on the $InstanceSubset$ as the ranker of individual $FS_j$
(7)     $f_{j,2} \longleftarrow 1 - RAccuracy_{r_j}$; //the second objective value of $j$-th individual $FS_j$
(8)   end for
(9)   $M_i \longleftarrow$ Binary Tournament $(P_i)$;
(10)  $P_m^* \longleftarrow$ calculating with formulas (3), (5) and (6);
(11)  $Q_i \longleftarrow$ Variation $(M_i, p_{c_2}, p_m^*)$;
(12)  $P_{i+1} \longleftarrow$ Environmental $(P_i \cup Q_i)$;
(13) end for
(14) $FeatureSubset \longleftarrow$ selecting the solutions on the Pareto front;
(15) $rset \longleftarrow$ the corresponding ranker set of $FeatureSubset$;
(16) Return $FeatureSubset$ and $rset$;
```

Algorithm 3: MOFS($maxgen_2, pop_2, p_{c_2}, p_{m_2}, InstanceSubset$).

performance of MOFS, an adaptive mutation strategy is also suggested, whose basic idea is from the intuition that during the mutation, the important features should have greater probability of being selected, whereas the redundant features should have greater probability of being removed. Thus, the suggested adaptive mutation probability is defined as

$$p_m^*(j) = \begin{cases} [p_m - \sigma \cdot (Impo(j) - Redu(j))]_{0-1} & \text{if } FS(j) \text{ is } 1 \\ [p_m + \sigma \cdot (Impo(j) - Redu(j))]_{0-1} & \text{if } FS(j) \text{ is } 0 \end{cases} \quad (3)$$

where function

$$[x]_{0-1} = \begin{cases} x, & \textit{if } 0 \le x \le 1 \\ 0, & \textit{if } x < 0 \\ 1, & \textit{else}, \end{cases} \quad (4)$$

$FS(j)$ denotes the value of j-th bit in an individual FS. $p_m^*(j)$ is the adaptive mutation probability of j-th bit in FS, and p_m is the basic mutation probability that used in NSGA-II. σ is a decaying factor and, in this paper, we set $\sigma = e^{-gen}$, where gen is the number of current generation. $Impo(j)$ and $Redu(j)$ represent the important degree and redundant degree of j-th feature in FS, which are formally defined as

$$Impo(j) = \frac{RAccuracy_r(j)}{\sum_{i=1}^{d} RAccuracy_r(j)} \quad (5)$$

$$Redu(j) = \frac{\sum_{(i=1,i\ne j)}^{d} \left(\left|corr_{j,i}\right| \times FS(i)\right)}{\sum_{(i=1,i\ne j)}^{d} \left|corr_{j,i}\right|} \quad (6)$$

where $RAccuracy_r(j)$ denotes the ranking accuracy value of the single j-th feature on original training set. $corr_{j,i}$ is Pearson's correlation coefficient between the j-th feature and the i-th feature ($i \ne j$) in FS. By using the adaptive mutation strategy, we can select the features with high ranking accuracy and low redundancy. The whole procedure of MOFS is presented in Algorithm 3.

3.3. Ensemble Phase.

After the second phase, a set of non-dominated solutions (feature subsets) are obtained. To produce a better final feature subset, a biobjective ensemble algorithm, named MOEN is proposed, where two optimization objectives are the number of selected features and the value of 1-RAccuracy with the selected features. The basic idea of MOEN is that a better feature subset can be achieved by weighted combining these nondominated solutions together. To this end, a mixed coding strategy is developed in MOEN, which consists of two parts. The first part uses the binary encoding, whose length l denotes the number of different features in the nondominated solutions of MOFS, the i-th bit corresponds to the i-th feature, and if this bit is 1, means this feature is selected, 0 indicates otherwise. The second part utilizes real encoding, and its length equals $|rset| \times l$, where $|rset|$ is the number of Pareto solutions in the second phase. Figure 2 provides an example to illustrate the suggested mixed encoding scheme in detail.

In Figure 2, there is an individual ind. The first part of ind has 4 bits, which means there are 4 different features in the non-dominated solutions of MOFS. The second part consists of 3 sub-part, which indicates that the number of feature subsets is 3. Let assume they are FS_1, FS_2 and FS_3, thus the i-th sub-part denotes the ensemble weight for $FS_i (i \in \{1, 2, 3\})$. During the optimization, for the individual ind, we need to

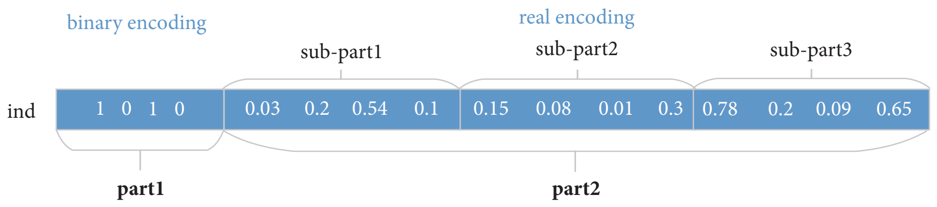


Figure 2: An example to illustrate the suggested mixed encoding scheme.

```
Input: $maxgen_3$: maximum generations of multi-objective ensemble, $pop_3$: population size of multi-objective ensemble, $p_{c_3}$: crossover probability of multi-objective ensemble, $p_{m_3}$: mutation probability of multi-objective ensemble, $FeatureSubset$: a set of non-dominated feature subsets, $rset$: the set of corresponding rankers;
Output: the final feature subset $FS^*$;
(1) Initializing the population $P_1 = \{p_1, \dots, p_{pop_3}\}$;
(2) for $i = 1$ to $maxgen_3$ do
(3)   /*Evaluating $P_i$ by two proposed objectives with formula (2) */
(4)   for $j = 1$ to $pop_3$ do
(5)     $f_{j,1} \longleftarrow$ calculating the number of non-zero features in $ind_j$;// the first objective value of $j$-th individual $ind_j$
(6)     $r_j \longleftarrow$ constructing the ranker with formula (7); // corresponding to the $j$-th individual $ind_j$
(7)     $f_{j,2} \longleftarrow 1 - RAccuracy_{r_j}$;// the second objective value of $j$-th individual $ind_j$
(8)   end for
(9)   $M_i \longleftarrow$ Binary Tournament $(P_i)$;
(10)  $Q_i \longleftarrow$ Variation $(M_i, p_{c_3}, p_{m_3})$;
(11)  $P_{i+1} \longleftarrow$ Environmental $(P_i \cup Q_i)$;
(12) end for
(13) $en\text{-}FS \longleftarrow$ selecting the solutions on the Pareto front;
(14) $FS^* \longleftarrow$ selecting the solution from $en\text{-}FS$ with the minimal value of $1 - RAccuracy$;
(15) Return $FS^*$;
```

Algorithm 4: MOEN($maxgen_3, pop_3, p_{c_3}, p_{m_3}, FeatureSubset, rset$).

calculate its two objectives. The value of the first objective (the number of selected features) can be easily obtained from the *part1* of *ind*. To get the value of second objective (the value of 1-RAccuracy of selected features), first, we should achieve the ranker *en-r* corresponding to *ind*. To this end, we utilize the non-dominated solutions of the second phase and the weights in the *part2* of *ind*. To be specific, let suppose $en\text{-}r = (w_1, w_2, \dots, w_l)$; thus each $w_i \in R$ $(i = 1, \dots, l)$ is obtained by the following formula:

$$w_i = I(part1_i) \times \sum_{j=1}^{|rset|} \left(part2_{i,j} \times r_{i,j}\right) \tag{7}$$

where $I(part1_i)$ is an indicator function which returns 1 if the i-th bit in *part1* is 1 and 0 otherwise. $part2_{i,j} \in R$ denotes the value of i-th bit in $part2_j$, and $part2_j$ is the j-th subpart of *part2*. $r_{j,i} \in R$ represents for the value of i-th bit in the ranker r_j, where r_j is the ranker that corresponds to the output feature subset FS_j in $rset$(Line (15) of Algorithm 3).

In the following, we also take the individual *ind* in Figure 2 as an example, and show how to obtain the ensemble ranker *en-r* of *ind* in detail. Firstly, let us assume that $r_1 = (0, 0.7, 0.3, 0)$, $r_2 = (0.12, 0.6, 0.25, 0.01)$, and $r_3 = (0.08, 0.9, 0.12, 0.1)$ are corresponding rankers of FS_1 to FS_3. Then from the *part2* of *ind*, we have $part2_1 = (0.03, 0.2, 0.54, 0.1)$, $part2_2 = (0.15, 0.08, 0.01, 0.3)$, and $part2_3 = (0.78, 0.2, 0.09, 0.65)$. Thus the ensemble ranker $en\text{-}r = (w_1, w_2, w_3, w_4)$, where

$$\begin{aligned} w_1 &= 1 \times (0 \times 0.03 + 0.12 \times 0.15 + 0.08 \times 0.78), \\ w_2 &= 0 \times (0.7 \times 0.2 + 0.6 \times 0.08 + 0.9 \times 0.2) \\ w_3 &= 1 \times (0.3 \times 0.54 + 0.25 \times 0.01 + 1.2 \times 0.09), \\ w_4 &= 0 \times (0 \times 0.1 + 0.01 \times 0.3 + 0.1 \times 0.65) \end{aligned} \tag{8}$$

With the ranker *en-r*, we can calculate the second objective of *ind* in population and obtain a new set of Pareto solutions by solving the biobjective ensemble algorithm. From the Pareto feature subsets, we choose the one with the minimal value of $1 - RAccuracy$ as the final output FS^*. The whole multiobjective ensemble (MOEN) algorithm is shown in Algorithm 4.

4. Experiments

In this section, we empirically verify the performance of the proposed MOFSRank by comparing it with several state-of-the-arts ranking algorithms. To be specific, we first present the experimental setting (including the data sets, comparison algorithms, and evaluation measures) and then report the comparison results between the proposed algorithm and the

Table 1: Characteristics of the LETOR Data Sets.

Data set	Queries	Features	Rel.levels	Query-document pairs
NP2004	75	64	2	75747
HP2004	75	64	2	80306
TD2004	75	64	2	1079810
MQ2008	784	46	2	80925
OHSUMED	106	45	3	582588

baselines (including the classical ranking algorithms and the representative feature selection algorithms for learning to rank). Lastly, we discuss the effectiveness of the suggested strategies in MOFSRank.

4.1. Experiment Setting

4.1.1. Data Sets. We conduct our experiments on the publicly available LETOR data collections [41], which are considered as the benchmark data sets in learning to rank. We select four data sets (NP2004, HP2004, TD2004, and OHSUMED) from LETOR 3.0 and one data set (MQ2008) from LETOR 4.0. Among them, OHSUMED is a three-level ranking set, while others are all bilevel data sets. The detail characteristics of those data sets are depicted in Table 1.

It should be noted that in LETOR collections, each data set is divided into five-folds and each fold contains a training/validation/test set, respectively. In the following experiments, we adopt the same splits as LETOR provides and report the results by averaging on the five folds.

4.1.2. Comparison Algorithms. The comparison algorithms used in this paper can be divided into two categories. The first group is the classical ranking algorithms provided by the LETOR. In this paper, we select RankSVM-Primal [42], RankSVM-Struct [43], ListNet [8], and AdaRank-NDCG [11] as the comparison algorithms, among which the former two belong to *Pairwise* approach, while the latter two optimize *Listwise* loss functions. The second group of comparison algorithms are the recently suggested feature selection algorithms for learning to rank, which include FenchelRank [25], FSMRank [26], and a nonconvex regularization feature selection method for learning to rank, proposed by Laporte et al. [27]. It is worth noting that, in the work [27], the authors presented three algorithms, and we choose the one, termed $l_{0.5}$, since it has the best mean performance on LETOR data sets.

For fair comparisons, we adopt the recommended parameters values for all comparison algorithms, which were suggested by the authors in their original papers. For the proposed MOFSRank, since it is composed of three sub-MOEAs (MOIS, MOFS, and MOEN), we need to set parameters for each sub-MOEA. The population sizes, cross probabilities and mutation probabilities of three sub-MOEAs are set to $pop_1 = pop_2 = pop_3 = 50$, $p_{c_1} = p_{c_2} = p_{c_3} = 1.0$, $p_{m_1} = 0.01$, $p_{m_2} = p_{m_3} = 1/m$, where m is the length of the individual in the sub-MOEAs. The maximum numbers of generation for MOIS, MOFS, and MOEN are set to $gen_1 = 200$, $gen_2 = 300$, and $gen_3 = 500$, respectively. For *RAccuracy* used in the second objective of each sub-MOEA, we adopt NDCG@10, which is a popular criterion to measure the accuracy of a ranking algorithm and, in the next section, we will discuss this criterion in detail.

4.1.3. Evaluation Measures. On the data sets above (NP 2004, HP2004, TD2004, MQ2008 and OHSUMED), we compare the proposed MOFSRank with several baselines, and the results of different algorithms are reported in terms of NDCG [44] and MAP [45], which are two most widely used metrics in learning to rank. NDCG (Normalized Discounted Cumulative Gain) is often used in the case with multilevel relevance judgments and, for a query, DCG score at position k is formally defined as

$$DCG@k = \sum_{i=1}^{k} \frac{2^{r(i)} - 1}{\log_2 (1 + i)} \tag{9}$$

where $r(i)$ is the relevance label of the i-th document in the sorted list. Then Normalized DCG score at position k in the ranking list of documents can be calculated by the equation as follows:

$$NDCG@k = Z \cdot \sum_{i=1}^{k} \frac{2^{r(i)} - 1}{\log_2 (1 + i)} \tag{10}$$

where Z is the normalization constant so that the value of NDCG ranges from 0 to 1. In the rest of this paper, we use N@k as the abbreviation of NDCG@k.

Another evaluation metric is MAP (Mean Average Precision), which deals with binary relevance judgments: relevant and irrelevant. First, we shall introduce the definition of precision at k, which denotes the proportion of relevant documents at the top k positions:

$$Pre@k = \frac{1}{k} \sum_{i=1}^{k} \zeta_i \tag{11}$$

where ζ_i is an indicator function. If the document at position i is relevant, $\zeta_i = 1$, otherwise $\zeta_i = 0$. Then the average precision of a given query q is defined as the follows:

$$AP(q) = \frac{\sum_{i=1}^{n} Pre@k \times \zeta_k}{n_{rel}} \tag{12}$$

where n and n_{rel} represent the total number of documents and relevant documents associated with query q, respectively.

Table 2: The Comparison Results between MOFSRank and Classical Ranking Algorithms on LETOR Data Sets, Averaged on Five Folds.

	N@1	N@2	N@3	N@4	N@5	N@6	N@7	N@8	N@9	N@10	MAP
					NP2004						
RankSVM-Primal	**0.5600**	0.7000	0.7236	0.7662	0.7719	0.7816	0.7864	0.7908	0.7950	0.7950	0.6755
RankSVM-Struct	**0.5600**	0.7000	0.7321	0.7746	0.7746	0.7843	0.7890	0.7935	0.7977	0.7977	0.6771
ListNet	0.5333	0.7267	0.7587	0.7879	0.7965	0.8037	0.8084	0.8128	0.8128	0.8128	0.6720
AdaRank-NDCG	0.5067	0.6133	0.6722	0.7122	0.7122	0.7225	0.7273	0.7384	0.7384	0.7384	0.6269
MOFSRank	0.5547	**0.8000**	**0.8218**	**0.8411**	**0.8466**	**0.8505**	**0.8514**	**0.8518**	**0.8543**	**0.8543**	**0.7064**
					HP2004						
RankSVM-Primal	0.5733	0.6867	0.7129	0.7413	0.7528	0.7683	0.7706	0.7706	0.7720	0.7720	0.6715
RankSVM-Struct	0.5867	0.6733	0.7248	0.7465	0.7523	0.7652	0.7652	0.7666	0.7666	0.7666	0.6784
ListNet	0.6000	0.6867	0.7213	0.7618	0.7694	0.7797	0.7845	0.7845	0.7845	0.7845	0.6899
AdaRank-NDCG	0.5867	0.7333	0.7512	0.7862	0.7920	0.7971	0.7971	0.8016	0.8037	0.8057	0.6914
MOFSRank	**0.6720**	**0.8080**	**0.8343**	**0.8406**	**0.8477**	**0.8508**	**0.8541**	**0.8585**	**0.8594**	**0.8622**	**0.7614**
					TD2004						
RankSVM-Primal	0.3067	0.3067	0.3131	0.3026	0.3062	0.3003	0.2951	0.2921	0.2942	0.2913	0.2061
RankSVM-Struct	0.3467	0.3467	0.3371	0.3287	0.3192	0.3129	0.3102	0.3146	0.3084	0.309	0.2196
ListNet	0.3600	0.3467	0.3573	0.3469	0.3325	0.327	0.3251	0.3206	0.3188	0.3175	0.2231
AdaRank-NDCG	0.4267	0.3800	0.3688	0.3524	0.3514	0.3376	0.3297	0.3284	0.3212	0.3163	0.1936
MOFSRank	**0.4533**	**0.4467**	**0.4105**	**0.3901**	**0.3795**	**0.3711**	**0.3616**	**0.3579**	**0.3551**	**0.3560**	**0.2427**
					MQ2008						
RankSVM-Primal	0.3725	0.4065	0.4333	0.4566	0.4765	0.4893	0.4940	0.4604	0.2263	0.2309	0.4744
RankSVM-Struct	0.3626	0.3984	0.4285	0.4508	0.4695	0.4851	0.4905	0.4564	0.2239	0.2279	0.4785
ListNet	0.3754	0.4112	0.4324	0.4568	0.4747	0.4894	0.4978	0.4630	0.2265	0.2303	0.4833
AdaRank-NDCG	0.3826	0.4211	0.4420	0.4653	0.4821	0.4948	0.4993	0.4636	0.2270	0.2307	0.4763
MOFSRank	**0.3957**	**0.4300**	**0.4521**	**0.4718**	**0.4888**	**0.5021**	**0.5064**	**0.4700**	**0.2360**	**0.2406**	**0.4867**
					OHSUMED						
RankSVM-Primal	0.5460	0.5010	0.4855	0.4766	0.4689	0.4552	0.4534	0.4500	0.4490	0.4504	0.4447
RankSVM-Struct	0.5515	0.5000	0.4850	0.4820	0.4729	0.4584	0.4570	0.4587	0.4568	0.4523	0.4478
ListNet	0.5326	0.4810	0.4732	0.4561	0.4432	0.4400	0.4409	0.4460	0.4459	0.4410	0.4457
AdaRank-NDCG	0.5330	0.4922	0.4790	0.4688	0.4673	0.4597	0.4596	0.4575	0.4541	0.4496	**0.4498**
MOFSRank	**0.5707**	**0.5353**	**0.5144**	**0.4970**	**0.4872**	**0.4774**	**0.4719**	**0.4676**	**0.4646**	**0.4641**	0.4489

Based on (11) and (12), MAP can be formally defined as

$$MAP = \frac{1}{|Q|}\sum_{q\in Q} AP(q) \tag{13}$$

where Q is the set of all queries.

4.2. Experimental Results and Analysis

4.2.1. Comparison Results between MOFSRank and Classical Ranking Algorithms. In the first part of experiments, we compare our method with several classical ranking algorithms, which are all the algorithms without using feature selection. Specifically, we evaluate MOFSRank with RankSVM-Primal, RankSVM-Struct, ListNet, and AdaRank-NDCG on five LETOR data sets. Table 2 presents the performances of different algorithms, averaged on five-folds.

From Table 2, we can find that on all data sets, the proposed algorithm performs significantly better than the existing classical ranking methods. The comparison results have shown that MOFSRank can achieve the best ranking accuracy on 53 of 55 statistical points, which demonstrates the superiority of MOFSRank on LETOR data set and indicates the effectiveness of feature selection for learning to rank.

4.2.2. Comparison Results between MOFSRank and Feature Selection Algorithms for Learning to Rank. In the second part of experiments, we are interested in how our MOFSRank performs, when compared with other feature selection baselines for learning to rank. To this end, we report the comparison results between the proposed algorithm and FenchelRank, FSMRank, and $l_{0.5}$, which are all recently suggested ranking feature selection methods with good performances. Tables 3 and 4 depict the ranking accuracy and the number of the selected features with different algorithms on the LETOR data sets, averaged on five-folds.

It can be observed from Table 3 that the proposed MOFSRank achieves the highest ranking values on most statistical points, which is much better than the existing feature selection baselines for learning to rank. Here, we

Table 3: The Comparison Results between MOFSRank and Feature Selection Baselines for Ranking on LETOR Data Sets, Averaged on Five Folds.

	N@1	N@2	N@3	N@4	N@5	N@6	N@7	N@8	N@9	N@10	MAP
						NP2004					
FenchelRank	0.5600	0.7400	0.7636	0.7728	0.7808	0.7962	0.801	0.8054	0.8096	0.8157	0.6830
FSMRank	0.5467	0.7800	0.7784	0.7943	0.8000	0.8071	0.8190	0.8279	0.8279	0.8279	0.6837
$l_{0.5}$	**0.5867**	0.7533	0.7686	0.7711	0.7848	0.7899	0.7899	0.8055	0.8097	0.8137	0.6963
MOFSRank	0.5547	**0.8000**	**0.8218**	**0.8411**	**0.8466**	**0.8505**	**0.8514**	**0.8518**	**0.8543**	**0.8543**	**0.7064**
						HP2004					
FenchelRank	0.6667	0.7667	0.7961	0.8095	0.8181	0.8232	0.8232	0.8232	0.8274	0.8274	0.7447
FSMRank	0.6133	0.7933	0.8070	0.8187	0.8187	0.8255	0.8302	0.8383	0.8383	0.8383	0.7205
$l_{0.5}$	0.6267	0.7867	0.8035	0.8135	0.8135	0.8135	0.8182	0.8182	0.8224	0.8265	0.7242
MOFSRank	**0.6720**	**0.8080**	**0.8343**	**0.8406**	**0.8477**	**0.8508**	**0.8541**	**0.8585**	**0.8594**	**0.8622**	**0.7614**
						TD2004					
FenchelRank	0.3600	0.3933	0.3725	0.3606	0.3462	0.3363	0.3329	0.3263	0.3220	0.3202	0.2368
FSMRank	0.3600	0.3400	0.3384	0.3260	0.3151	0.3080	0.3092	0.3128	0.3138	0.3133	0.2267
$l_{0.5}$	0.3733	0.3933	0.3630	0.3462	0.3324	0.3279	0.3262	0.3261	0.3213	0.3205	0.2314
MOFSRank	**0.4533**	**0.4467**	**0.4105**	**0.3901**	**0.3795**	**0.3711**	**0.3616**	**0.3579**	**0.3551**	**0.3560**	**0.2427**
						MQ2008					
FenchelRank	0.3762	0.4164	0.4402	0.4598	0.4790	0.4933	0.4989	0.4619	0.2277	0.2317	0.4785
FSMRank	0.3750	0.4211	0.4404	0.4611	0.4809	0.4938	0.4986	0.4624	0.2283	0.2321	0.4760
$l_{0.5}$	0.3720	0.4167	0.4390	0.4614	0.4805	0.4934	0.4995	0.4625	0.2268	0.2307	0.4800
MOFSRank	**0.3957**	**0.4300**	**0.4521**	**0.4718**	**0.4888**	**0.5021**	**0.5064**	**0.4700**	**0.2360**	**0.2406**	**0.4867**
						OHSUMED					
FenchelRank	0.5456	**0.5390**	**0.5166**	**0.4989**	0.4826	0.4742	**0.4721**	**0.4680**	**0.4652**	0.4637	0.4486
FSMRank	0.5397	0.5124	0.5070	0.4938	0.4808	0.4725	0.4692	0.4637	0.4567	0.4534	0.4455
$l_{0.5}$	0.5489	0.5337	0.5032	0.4812	0.4765	0.4688	0.4631	0.4591	0.4594	0.4610	0.4477
MOFSRank	**0.5707**	0.5353	0.5144	0.4970	**0.4872**	**0.4774**	0.4719	0.4676	0.4646	**0.4641**	**0.4489**

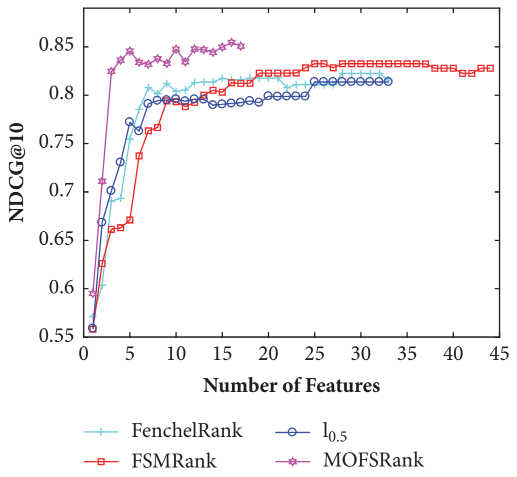
Table 4: The number of selected features between MOFSRank and feature selection baselines for ranking on LETOR data sets, averaged on five-folds.

	NP2004	HP2004	TD2004	MQ2008	OHSUMED
FenchelRank	18.60	12.00	32.40	15.80	13.00
FSMRANK	32.00	13.80	28.20	13.20	18.00
$l_{0.5}$	14.60	7.00	17.20	**2.40**	9.60
MOFSRank	**5.72**	**5.12**	**5.88**	4.80	**5.40**

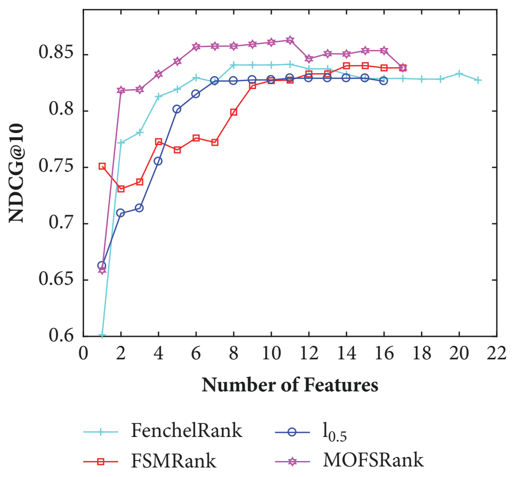
present a few statistics on different data sets in terms of N@10. On NP2004, HP2004 and MQ2008 data sets, MOFSRank obtains the NDCG values of 0.8543, 0.8622 and 0.2406. Compared to the second best algorithms (FSMRank), its performances increase 3.2%, 2.9% and 3.7%, respectively. On TD2004 data set, the value of N@10 for MOFSRank is 0.3560, which shows 11.1% improvement than the second best algorithms ($l_{0.5}$). Similarly, the increase of MOFSRank on OHSUMED set is 0.1%, in comparison with the second best algorithm, FenchelRank.

Table 4 presents the number of selected features of different algorithms on LETOR data sets, averaged on five folds. From the table, we can find that, on NP2004, HP2004, TD2004, and OHSUMED data sets, the features selected by MOFSRank are much fewer than those of other baselines. On MQ2008 data set, the proposed algorithm achieves the second best performance, whose number of selected features is slightly larger than the nonconvex feature selection algorithm $l_{0.5}$. The statistics in Tables 3 and 4 have demonstrated the competitiveness of MOFSRank, when compared with other feature selection algorithms for learning to rank.

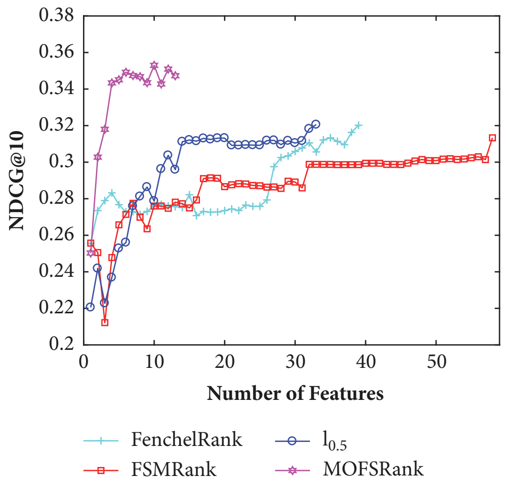
To further investigate the performance of different feature selection algorithms on the LETOR data sets, in the following, we detailed report the value of N@10 (y-axis) with respect to different number of selected features (x-axis), and the results are plotted in Figure 3. Note that since three feature selection baselines cannot directly select a given number of features, we adopt the strategy used in [26], which can choose top k ($k \geq 1$) best features from the whole features. From the figures, we can find that although the NDCG accuracy of different algorithms varies with the number of selected features, our MOFSRank can always achieve the best trade-off between the



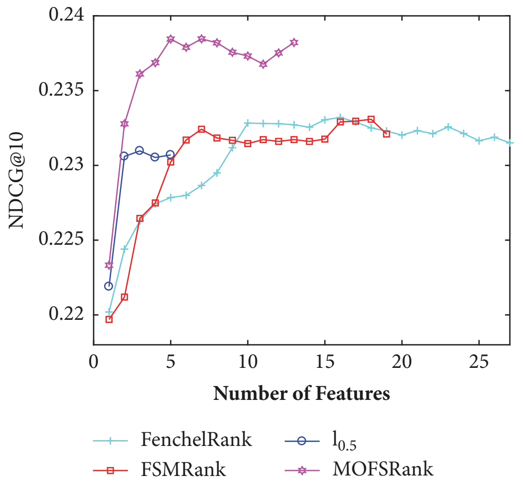
(a) NP2004



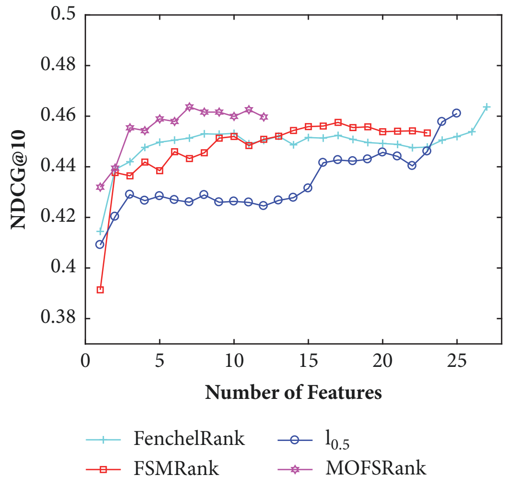
(b) HP2004



(c) TD2004



(d) MQ2008



(e) OHSUMED

Figure 3: The NDCG accuracy of four feature selection algorithms on LETOR sets with different feature numbers.

Table 5: The training instances of MOFSRank and MOFSRank-NonIS on LETOR data sets.

	NP2004	HP2004	TD2004	MQ2008	OHSUMED
#Ins of MOFSRank-NonIS	45445	48183	647886	48555	349552
#Ins of MOFSRank	14493	8563	20407	2611	10399
Instances ratio	0.32	0.19	0.04	0.06	0.03

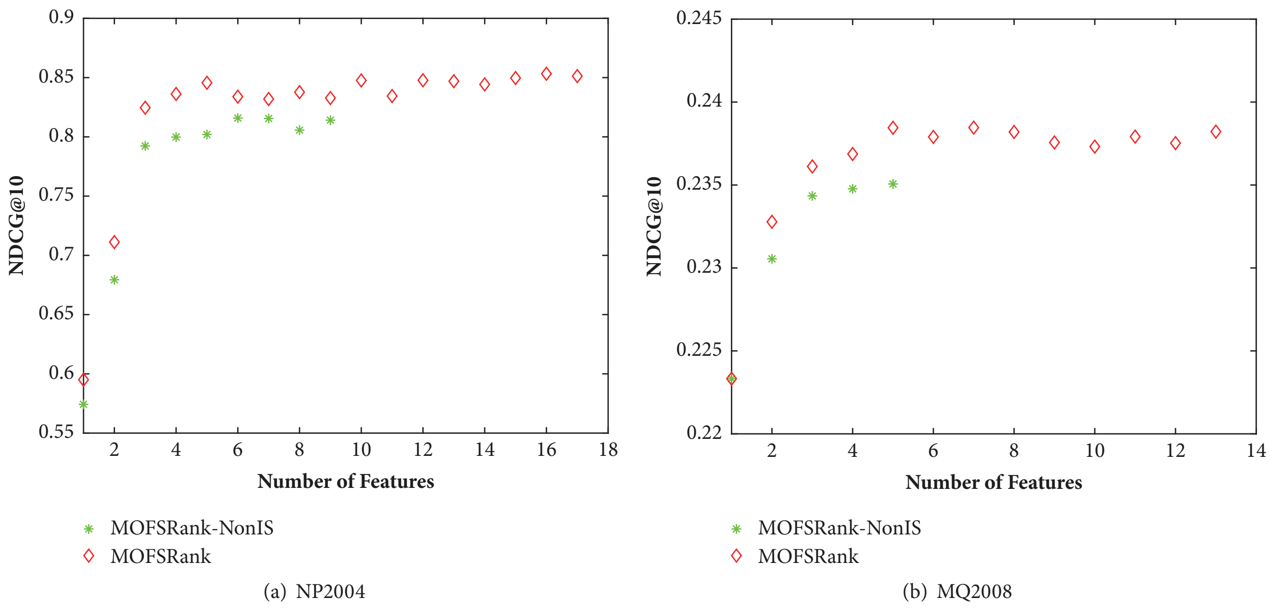


Figure 4: The final nondominated Solutions Obtained by MOFSRank and MOFSRank-NonIS on LETOR Data Sets in Objective Space.

accuracy and the number of selected features, which indicates the superior performance of the proposed method.

4.3. Effectiveness of the Suggested Strategies in MOFSRank. As mentioned before, in the proposed MOFSRank, three strategies (instance selection, adaptive mutation, and Pareto based ensemble) are suggested and, in the following, we will empirically investigate the influence of these strategies on the performance of MOFSRank for LETOR data sets, respectively.

4.3.1. Effectiveness of the Instance Selection Strategy. In the first phase of MOFSRank, an instance selection strategy is suggested, which can reduce the number of training instances, and improve the performance of MOFSRank. To verify this fact, we compare the proposed algorithm with MOFSRank-NonIS, which is the same one as our MOFS-Rank, except that it excludes the instance selection strategy, and uses the original *Pairwise* instances in the training set. The comparison results on LETOR data sets are shown from two aspects. Firstly, we present the real training instances of two algorithms in Table 5, where #Ins of MOFSRank and #Ins of MOFSRank-NonIS denote the numbers of real training instances of MOFSRank and MOFSRank-NonIS. It can be easily observed from Table 5, that on all the LETOR data sets the suggested instance selection strategy does reduces the training instances greatly, especially on the data sets with hundreds of thousands of training instances (TD2004 and OHSUMED), and the ratios of the selected instances are only 0.04 and 0.03.

Secondly, we take N@10 as the ranking measure, and plot the final non-dominated solutions obtained by MOFSRank and MOFSRank-NonIS in objective space in Figure 4. Note that due to space limitation, in the following experiments, we only list the results on one LETOR 3.0 data set (NP2004) and one LETOR 4.0 data set (MQ2008), and the results on other LETOR data sets are similar. As can be seen from Figure 4, on both data sets, the MOFSRank can obtain better nondominated solutions than the MOFSRank-NonIS, which demonstrates the effectiveness of the suggested instance selection strategy in MOFSRank.

4.3.2. Effectiveness of the Adaptive Mutation Strategy. In the second phase of MOFSRank, an adaptive mutation strategy is developed, which can enhance the performance of MOFSRank. To confirm the fact, we compare the proposed method with MOFSRank-NonAM, where the adaptive mutation strategy is removed from the original MOFSRank. The final nondominated solutions obtained by MOFSRank and MOFSRank-NonAM in objective space for LETOR data sets are plotted in Figure 5, from which, we can find that compared with MOFSRank-NonAM, the MOFSRank achieves better nondominated solutions on the experimental sets, which indicates the effectiveness of the adaptive mutation strategy.

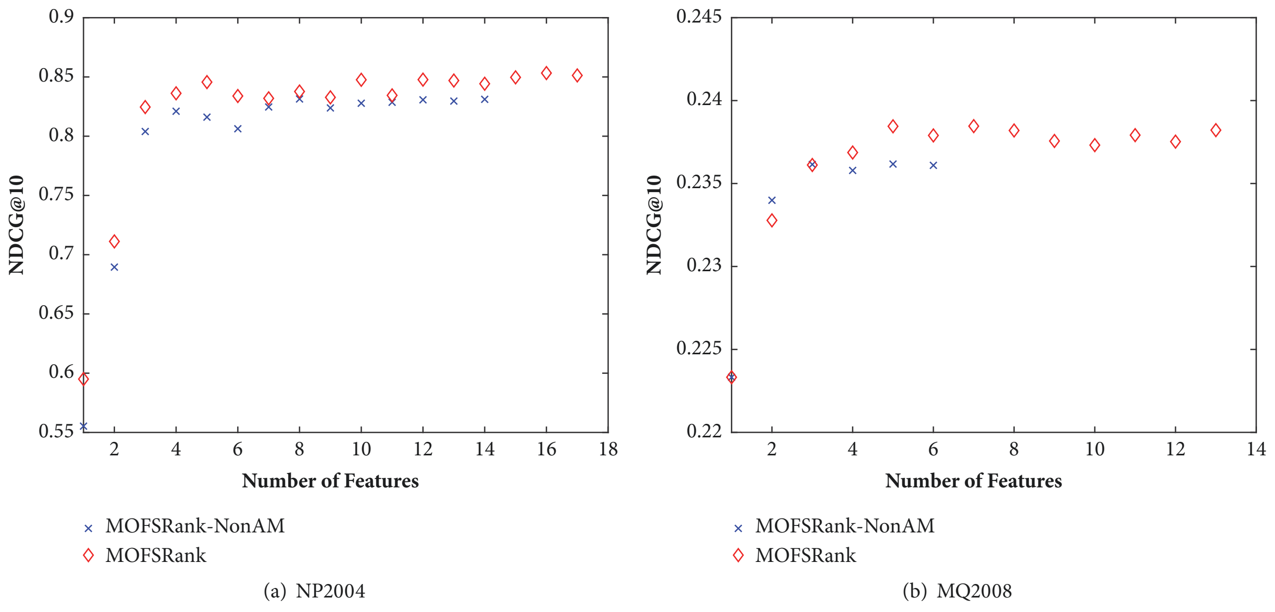


Figure 5: The final nondominated solutions obtained by MOFSRank and MOFSRank-NonAM on LETOR data sets in objective space.

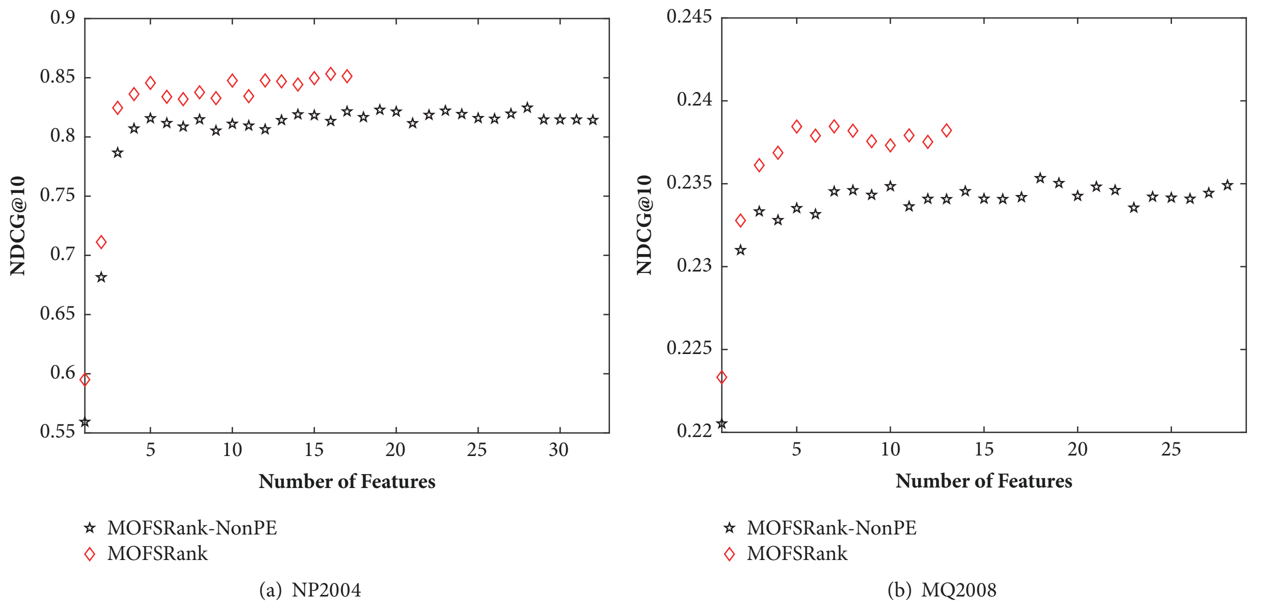


Figure 6: The final nondominated solutions obtained by MOFSRank and MOFSRank-NonPE on LETOR data sets in objective space.

4.3.3. Effectiveness of Pareto Based Ensemble Strategy. In the third phase of MOFSRank, to obtain a better feature subset, a Pareto based ensemble strategy is suggested, where the Pareto solutions of the second phase are combined together. In order to verify the effectiveness of this ensemble strategy, we compare the MOFSRank with MOFSRank-NonPE. The only difference between them lies in the fact that MOFSRank-NonPE does not include the Pareto based ensemble operation. The experimental results of two algorithms on LETOR data sets are plotted in Figure 6, from which we can clearly find that with the suggested ensemble strategy, the proposed algorithm achieves better nondominated solutions than MOFSRank-NonPE. This fact demonstrates the effectiveness of the suggested Pareto based ensemble strategy.

5. Conclusion

In this paper, we have proposed a multiobjective evolutionary algorithm, termed MOFSRank, for feature selection in ranking. In MOFSRank, an MOEA for instance selection (MOIS) has been suggested, where the informative instances were chosen from the original training set and made the following feature selection more effective and efficient. Then a multiobjective feature selection (MOFS) algorithm with an adaptive mutation has been performed on these chosen instances subsets, which can obtain the features with high ranking accuracy and low redundancy. Finally, a multiobjective ensemble (MOEN) algorithm has been developed to integrate the Pareto solutions of MOFS, by which the performance of MOFSRank can be further improved. Experimental results on LETOR data sets have demonstrated the competitiveness of the proposed algorithm.

There still remains some interesting work related to MOFSRank that deserves to be further investigated. The proposed MOFSRank has shown that MOEA is a promising method to solve feature selection for learning to rank and, in this paper, we mainly focus on the *Pairwise* ranking approach. In the future, we plan to further design feature selection algorithm for other type of learning to rank approach, such as *Listwise* approach. In addition, in our MOFSRank, we adopt NSGA-II as the framework, it is also interesting to combine the proposed method with other frameworks of MOEA, such as MOEA/D [46], SPEA2 [47], and AR-MOEA [48].

Data Availability

The data used to support the findings of our study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61672033, 61502004, 61502001, and 61502012), the Natural Science Foundation of Anhui Province (1708085MF166), Humanities and Social Sciences Project of Chinese Ministry of Education (Grant no. 18YJC870004), and the Key Program of Natural Science Project of Educational Commission of Anhui Province (KJ2017A013).

References

[1] T.-Y. Liu, "Learning to rank for Information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–231, 2009.

[2] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys*, vol. 47, no. 1, 2014.

[3] C. Moreira, P. Calado, and B. Martins, "Learning to rank academic experts in the DBLP dataset," *Expert Systems with Applications*, vol. 32, no. 4, pp. 477–493, 2015.

[4] D. Cossock and T. Zhang, "Subset ranking using regression," in *Proceedings of the Conference on Learning Theory*, pp. 605–619, 2006.

[5] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 133–142, July 2002.

[6] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 933–969, 2003.

[7] C. Burges, T. Shaked, E. Renshaw et al., "Learning to rank using gradient descent," in *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*, pp. 89–96, ACM, August 2005.

[8] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pp. 129–136, ACM, Corvallis, Ore, USA, June 2007.

[9] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank - Theory and algorithm," in *Proceedings of the International Conference on Machine Learning*, pp. 1192–1199, 2008.

[10] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 271–278, 2007.

[11] J. Xu and H. Li, "AdaRank: a boosting algorithm for information retrieval," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 391–398, 2007.

[12] J. Weston, S. Bengio, and N. Usunier, "Large scale image annotation: learning to rank with joint word-image embeddings," *Machine Learning*, vol. 81, no. 1, pp. 21–35, 2010.

[13] J. Yu, D. Tao, M. Wang, and Y. Rui, "Learning to Rank Using User Clicks and Visual Features for Image Retrieval," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 767–779, 2015.

[14] R. Leaman, R. I. Doğan, and Z. Lu, "DNorm: disease name normalization with pairwise learning to rank," *Bioinformatics*, vol. 29, no. 22, pp. 2909–2917, 2013.

[15] X. Geng, T.-Y. Liu, T. Qin, and H. Li, "Feature selection for ranking," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 407–414, 2007.

[16] G. Hua, M. Zhang, Y. Liu, S. Ma, and L. Ru, "Hierarchical feature selection for ranking," in *Proceedings of the International Conference on World Wide Web, WWW 2010*, pp. 1113-1114, Raleigh, North Carolina, USA, 2010.

[17] K. D. Naini and I. S. Altingovde, "Exploiting Result Diversification Methods for Feature Selection in Learning to Rank," in *Proceedings of the European Conference on Information Retrieval*, pp. 455–461, 2014.

[18] M. B. Shirzad and M. R. Keyvanpour, "A feature selection method based on minimum redundancy maximum relevance for learning to rank," in *Proceedings of the Ai & Robotics*, pp. 1–5, 2015.

[19] A. Gigli, C. Lucchese, F. M. Nardini, and R. Perego, "Fast feature selection for learning to rank," in *Proceedings of the International Conference on the Theory of Information Retrieval*, pp. 167–170, 2016.

[20] F. Pan, T. Converse, D. Ahn, F. Salvetti, and G. Donato, "Feature selection for ranking using boosted trees," in *Proceedings of the ACM Conference on Information and Knowledge Management*, pp. 2025–2028, 2009.

[21] H. Yu, J. Oh, and W. Han, "Efficient feature weighting methods for ranking," in *Proceedings of the ACM Conference on Information and Knowledge Management*, pp. 1157–1166, 2009.

[22] V. Dang and B. Croft, “Feature selection for document ranking using best first search and coordinate ascent,” in *Proceedings of the SIGIR Workshop on Feature Generation and Selection for Information Retrieval*, pp. 1–5, 2010.

[23] T. Pahikkala, A. Airola, P. Naula, and T. Salakoski, “Greedy rankrls: a linear time algorithm for learning sparse ranking models,” in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 11–18, 2010.

[24] Z. Sun, T. Qin, Q. Tao, and J. Wang, “Robust sparse rank learning for non-smooth ranking measures,” in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 259–266, 2009.

[25] H. Lai, Y. Pan, C. Liu, L. Lin, and J. Wu, “Sparse learning-to-rank via an efficient primal-dual algorithm,” *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 62, no. 6, pp. 1221–1233, 2013.

[26] H.-J. Lai, Y. Pan, Y. Tang, and R. Yu, “FSMRank: Feature selection algorithm for learning to rank,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 940–952, 2013.

[27] L. Laporte, R. Flamary, S. Canu, S. Dejean, and J. Mothe, “Nonconvex regularizations for feature selection in ranking with sparse SVM,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1118–1130, 2014.

[28] P. Li, C. J. C. Burges, and Q. Wu, “Mcrank: learning to rank using multiple classification and gradient boosting,” in *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 897–904, 2007.

[29] M. B. Shirzad and M. R. Keyvanpour, “A Systematic Study of Feature Selection Methods for Learning to Rank Algorithms,” *International Journal of Information Retrieval Research*, vol. 8, no. 3, pp. 46–67, 2018.

[30] X. Han and S. Lei, “Feature selection and model comparison on microsoft learning-to-rank data sets,” https://arxiv.org/abs/1803.05127, 2018.

[31] Y. Lin, H. Lin, K. Xu, and X. Sun, “Learning to rank using smoothing methods for language modeling,” *Journal of the Association for Information Science and Technology*, vol. 64, no. 4, pp. 818–828, 2013.

[32] D. X. Sousa, S. D. Canuto, T. C. Rosa, W. S. Martins, and M. A. Gonçalves, “Incorporating Risk-Sensitiveness into Feature Selection for Learning to Rank,” in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, pp. 257–266, ACM, 2016.

[33] L. Du, Y. Pan, J. Ding, H. Lai, and C. Huang, “EGRank: an exponentiated gradient algorithm for sparse learning-to-rank,” *Information Sciences*, vol. 467, pp. 342–356, 2018.

[34] Z. Wang, M. Li, and J. Li, “A multi-objective evolutionary algorithm for feature selection based on mutual information with a new redundancy measure,” *Information Sciences*, vol. 307, pp. 73–88, 2015.

[35] J. Lee, W. Seo, and D. W. Kim, “Effective Evolutionary Multilabel Feature Selection under a Budget Constraint,” *Complexity*, vol. 2018, Article ID 3241489, 14 pages, 2018.

[36] G. Acampora, F. Herrera, G. Tortora, and A. Vitiello, “A multi-objective evolutionary approach to training set selection for support vector machine,” *Knowledge-Based Systems*, vol. 147, pp. 94–108, 2018.

[37] W. Ying, Y. Xie, Y. Wu, B. Wu, S. Chen, and W. He, “Universal partially evolved parallelization of MOEA/D for multi-objective optimization on message-passing clusters,” *Soft Computing*, vol. 21, no. 18, pp. 5399–5412, 2017.

[38] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, “A Decision Variable Clustering-Based Evolutionary Algorithm for Large-Scale Many-Objective Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 97–112, 2018.

[39] X. Zhang, F. Duan, L. Zhang, F. Cheng, Y. Jin, and K. Tang, “Pattern Recommendation in Task-oriented Applications: A Multi-Objective Perspective,” *IEEE Computational Intelligence Magazine*, vol. 12, no. 3, pp. 43–53, 2017.

[40] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[41] T. Qin, T.-Y. Liu, J. Xu, and H. Li, “LETOR: A benchmark collection for research on learning to rank for information retrieval,” *Information Retrieval*, vol. 13, no. 4, pp. 346–374, 2010.

[42] O. Chapelle and S. S. Keerthi, “Efficient algorithms for ranking with SVMs,” *Information Retrieval*, vol. 13, no. 3, pp. 201–215, 2010.

[43] T. Joachims, “Training linear SVMs in linear time,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 217–226, 2006.

[44] K. Rvelin, Kek, and J. Inen, “Cumulated gain-based evaluation of ir techniques,” *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002.

[45] B. Y. Ricardo, R. N. Berthier et al., “Modern information retrieval,” *ACM*, vol. 43, no. 1, pp. 26–28, 1999.

[46] H. B. Nguyen, B. Xue, H. Ishibuchi, P. Andreae, and M. Zhang, “Multiple reference points MOEA/D for feature selection,” in *Proceedings of theGenetic and Evolutionary Computation Conference Companion*, pp. 157-158, Berlin, Germany, 2017.

[47] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization,” in *Evolutionary Methods for Design, Optimization, and Control*, pp. 95–100, 2002.

[48] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, “An Indicator Based Multi-Objective Evolutionary Algorithm with Reference Point Adaptation for Better Versatility,” *IEEE Transactions on Evolutionary Computation*, 2017.