Multi-Objective Multi-Label Classification

Chuan Shi* Xiangnan Kong† Philip S. Yu† ‡ Bai Wang*

Abstract

Multi-label classification refers to the task of predicting potentially multiple labels for a given instance. Conventional multi-label classification approaches focus on the single objective setting, where the learning algorithm optimizes over a single performance criterion (e.g. *Ranking Loss*) or a heuristic function. The basic assumption is that the optimization over one single objective can improve the overall performance of multi-label classification and meet the requirements of various applications. However, in many real applications, an optimal multi-label classifier may need to consider the tradeoffs among multiple conflicting objectives, such as minimizing *Hamming Loss* and maximizing *Micro F1*. In this paper, we study the problem of *multi-objective multi-label classification* and propose a novel solution (called MOML) to optimize over multiple objectives simultaneously. Note that optimization objectives may be conflicting, thus one cannot identify a single solution that is optimal on all objectives. Our MOML algorithm finds a set of *non-dominated solutions* which are optimal according to the different tradeoffs of the multiple objectives. So users can flexibly construct various combined predictive models from the solution set, which helps to provide more meaningful classification results in different application scenarios. Empirical studies on real-world tasks demonstrate that the MOML can effectively boost the overall performance of multi-label classification, not limiting to the optimization objectives.

1 Introduction

Traditional supervised learning works on the single label scenario. That is, each instance is associated with one single label within a finite set of labels. However, in many applications, each instance can be associated with more than one label simultaneously. For example, in text categorization, one document can belong to multiple categories [25]; in bioinformatics, one gene sequence may serve multiple functions [9]. This setting is called multi-label classification, which corresponds to the problem of classifying each instance with a set of labels. Multi-label classification has been drawing increasing attention from the machine learning and data mining communities in the past decade [7, 16, 27].

Conventional multi-label classification approaches focus on single objective setting, where the learning algorithm trains one model that optimizes over one single objective. The objective can be a performance evaluation criterion (e.g. *Hamming Loss* [21]) or a heuristic function (e.g. the posteriori principle in ML-KNN [29]). The basic assumption of single objective multi-label classification is that one single objective can evaluate the overall performance of a multi-label classifier. Thus, the optimization over one single objective can comprehensively improve classifier's performance. However, in multi-label classification, many criteria are proposed to evaluate the classification performance from different perspectives (see Section IV.A.2) and some criteria are uncorrelated or even negatively correlated. Dembczyński et al. [8] elaborate the connection among these criteria and point out that some loss functions are essentially conflicting, such as *Hamming Loss* [21] and *Subset 0/1 Loss* [12]. So the optimization over one single objective may not lead to the performance improvement on the other objectives. For example, in a multi-label classification task where the performances on *Hamming Loss* [21] and *Micro F1* [12] are concerned, one may minimize *Hamming Loss*, maximize *Micro F1* (i.e. minimize $1 - Micro\ F1$), or optimize both of them simultaneously. An example of results is shown in Figure 1. Due to the intrinsic conflict between these two objectives, only optimizing over *Hamming Loss* will lead to very bad performance on *Micro F1* (e.g. solution B), or vice versa (e.g. solution A). However, it is obvious that the solution C is better than A and B when we concern the classification performances on both *Hamming Loss* and *Micro F1*. As a consequence, it is necessary to simultaneously optimize over multiple objectives for multi-label classification in such condition where the concerned objectives are conflicting or potential conflicting. This helps to balance the tradeoff among these objectives and comprehensively improve performances of multi-label classification, not limiting to one single criterion. In addition, the simultaneous optimization over

*Beijing University of Posts and Telecommunications, Beijing, China. {shichuan, wangbai}@bupt.edu.cn.

†University of Illinois at Chicago, IL, USA. {xkong4, psyu}@uic.edu.

‡King Abdulaziz University, Jeddah, Saudi Arabia

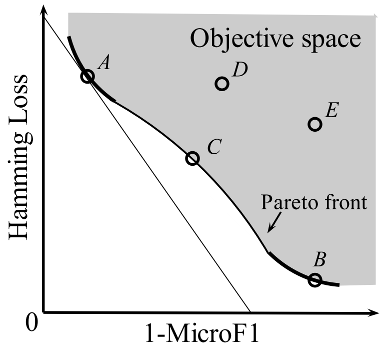
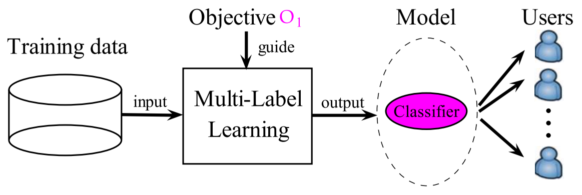
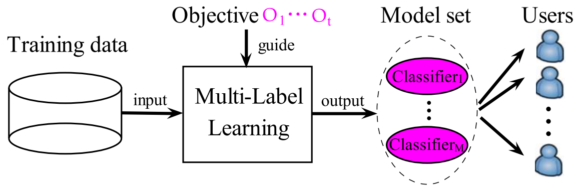


Figure 1: Illustration of optimizing over multiple objectives.



(a) Single objective multi-label classification



(b) Multi-objective multi-label classification

Figure 2: Comparison of single and multiple objective multi-label classification.

multiple objectives is also practically needed in many multi-label classification tasks [21]. For example, in a news filtering application, users must be presented with those interesting articles, but it is also important to only see the most interesting one. So the performances of the multi-label classifier on *One Error* [21] and *Micro F1* [12] both need to be considered.

In conventional multi-label classification (i.e. single objective multi-label classification as shown in Figure 2(a)), one single solution is usually returned to satisfy the requirements of all users. However, it is often the case that users in different application scenarios can have very different expectations on a multi-label classifier [16]. With multiple optimization objectives employed, there is usually no single best solution for this multi-label classification task, but instead, a set of non-dominated solutions that correspond to different tradeoffs among those objectives, so that users can flexibly select appropriate solutions in items of their different applications. For example, in Figure 1, one can select A in a *Hamming Loss*-aware application, or select C in a *Hamming Loss* and *Micro F1* -aware application.

Formally, the multi-objective multi-label classification (as shown in Figure 2(b)) corresponds to simultaneously optimizing over multiple objectives and obtaining a set of multi-label classification models. Despite its value and significance, the multi-objective multi-label classification has not been studied in this context so far, due to the following research challenges. (1) Most evaluation objectives in multi-label classification cannot be directly optimized even in the single objective setting. Among all the performance criteria for multi-label classification, only a small number of them (e.g. *Ranking Loss* [21] and *Hamming Loss* [21]) can be used as the optimization objective directly, since they are differentiable. Most of the other criteria (e.g. *Micro F1* [12] and *Average Precision* [21]) are difficult to be optimized directly. (2) Multi-objective optimization is much more difficult than single objective optimization. It is not easy to effectively tradeoff multiple objectives in multi-label classification. Multi-objective optimization can be converted into single objective optimization with the scalarization method (e.g. weighted sum method [11]) and the tradeoffs among objectives can be exploited by tuning weights. However, it not only is hard to choose the weights in real applications but also cannot discover the solutions in the concave Pareto front [10]. For example, the weighted sum method can find A and B in Figure 1, but it cannot discover C.

In this paper, we study the problem of multi-objective multi-label classification and propose a novel solution, called MOML (Multi-Objective Multi-Label algorithm). Different from conventional multi-label classification approaches, the proposed MOML can simultaneously optimize over multiple objectives based on Evolutionary Multi-objective Optimization (EMO). EMO has unique properties to effectively solve the above challenges. (1) EMO does not require optimization objectives to be differentiable, and thus any evaluation metric in multi-label classification can be used as optimization objectives in our MOML. (2) It can automatically balance the tradeoffs among multiple objectives with population optimization. Due to multiple optimization objectives, MOML returns a set of classification models with different preferences on these objectives, so users can flexibly select different models based on their preferences in different applications. One simple but effective model selection strategy in MOML is to select the top k models over the most preferred objectives on the training data and then make predictions on the testing data. Experiments on seven real-world multi-label classification tasks justify the effectiveness of our MOML with nine popular performance evaluation criteria. Results show that Moml can comprehensively boost the multi-label classification performance on most of the performance criteria. Moreover, Moml can effectively adapt to the user's preferences in different applications by achieving much better performances on the preferred objectives.

The rest of the paper is organized as follows. Section 2 proposes the multi-objective multi-label classification problem, and then we present the Moml solution in Section 3. We validate the effectiveness of Moml through extensive experiments in Section 4. Section 5 briefly compares Moml with those most related works. Finally, Section 6 concludes this paper.

2 Problem Definition

Let $\chi = \mathbb{R}^d$ be the d-dimensional input space and $\mathcal{L} = \{1, 2, \cdots, L\}$ be the finite set of L possible classes. Given a multi-label training set $\mathcal{D} = \{(\mathbf{x}_i, Y_i)|1 \leq i \leq m\}$, where $\mathbf{x}_i \in \chi$ is an instance and $Y_i \subseteq \mathcal{L}$ is the label set associated with $\mathbf{x}_i$. The task of multi-label learning is to learn a multi-label classifier $h : \chi \rightarrow 2^{\mathcal{L}}$ from $\mathcal{D}$ which predicts a set of labels for each unseen instance.

Conventional multi-label classification approaches can be roughly classified into two categories: (1) One type of the approaches train one single model by explicitly or implicitly optimizing a performance criterion. For example, ML-rbf [26] explicitly optimizes the *Hamming Loss*, while *Ranking Loss* is optimized in BP-mll [28] and Rank-svm [9]. (2) The second type of approaches do not explicitly optimize those performance criteria, but they implicitly optimize one single heuristic function which is not directly related to any performance criteria. For example, Ecc [18] and Lead [27] optimize the generalization risk for multi-label predictions by encoding label correlations, and ML-knn [29] maximizes the posteriori principle in multi-label learning. In both types of approaches, the multi-label learning is regarded as a Single objective Optimization Problem (SOP), which can be defined as follows:

Definition 1. ***Single objective multi-label classification***. *It determines a model $\mathcal{M}^*$ for which*

$$O(\mathcal{M}^*) = \min_{\mathcal{M} \in \Omega} O_1(\mathcal{M}) \tag{2.1}$$

Ω is the set of feasible models, $\mathcal{M}$ is a predictive model in Ω. $O_1 : \Omega \rightarrow R$ is an objective function, which can be a performance criterion (e.g. metrics in Section IV.A.2) or any other implicit heuristic function. Without loss of generality, we assume O_1 is to be minimized. Most of conventional algorithms are based on solving this SOP. Different algorithms may vary in the objective function O_1 and optimization techniques.

This paper first formulates multi-label learning as a Multi-objective Optimization Problem (MOP), which can be defined as follows.

Definition 2. ***Multi-objective multi-label classification***. *It determines models $\mathcal{M}^*$ for which*

$$O(\mathcal{M}^*) = \min_{\mathcal{M} \in \Omega} (O_1(\mathcal{M}), O_2(\mathcal{M}), \cdots, O_t(\mathcal{M})) \tag{2.2}$$

t is the number of objectives and O_i represents the i-th objective.

For the MOP, each objective corresponds to an optimal solution. We have to incorporate the different tradeoffs among the multiple objectives. One fundamental difference between SOP and MOP is that, for a MOP, we can find a set of optimal solutions where no single solution can be said to be better than any other. Solving a MOP often implies to search for the set of optimal solutions as opposed to one single solution for a SOP. Here, we define the concept of domination relation to compare the performance of multi-label classification models, similar as in [5].

Definition 3. ***Domination***. *For two models $\mathcal{M}_1, \mathcal{M}_2 \in \Omega$, $\mathcal{M}_1$ dominates $\mathcal{M}_2$ (denoted as $\mathcal{M}_1 \preceq \mathcal{M}_2$) if and only if*

$$\begin{aligned} &\forall\, i \in \{1, \cdots, t\}\ O_i(\mathcal{M}_1) \leq O_i(\mathcal{M}_2) \wedge \\ &\exists\, i \in \{1, \cdots, t\}\ O_i(\mathcal{M}_1) < O_i(\mathcal{M}_2) \end{aligned} \tag{2.3}$$

Similarly, if $\mathcal{M}_1 \npreceq \mathcal{M}_2$ and $\mathcal{M}_2 \npreceq \mathcal{M}_1$, $\mathcal{M}_1$ is non-dominated with $\mathcal{M}_2$. A model $\mathcal{M} \in \Omega$ is said to be Pareto optimal [5] if and only if $\mathcal{M}$ is not dominated by any other model in Ω. The set of all Pareto optimal models is called the Pareto optimal set, or Pareto front. An example is shown in Figure 1. Model C dominates the model D, and C is non-dominated with A and B. A, B, and C are the Pareto optimal set or Pareto front.

3 The MOML Algorithm

In order to solve the multi-objective multi-label classification problem, the traditional approaches on MOP convert multiple objectives into a single objective by using certain schemes and user-specified parameters, such as the weighted sum method [11]. However, these methods cannot be directly applied to multi-label classification problem, since many objectives may not be easily optimized even in SOP setting and the parameter settings are very difficult for these methods. Moreover, these methods cannot effectively solve the concave Pareto front problem [10]. Evolutionary Algorithm (EA) [14] has been proven to be an effective method to solve MOP, which is called the Evolutionary Multi-objective Optimization (EMO) technique [5]. EMO simultaneously optimizes multiple objectives through

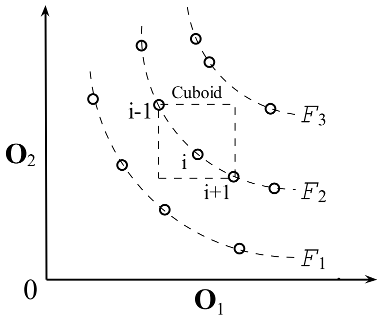


Figure 3: Illustration of non-dominated-sort and diversity-estimate.

population evolution, in which individuals reproduce through evolutionary operation (e.g. crossover and mutation) and obey the Darwinian evolution: survival of the fittest. EMO has many unique advantages: simultaneously generating a set of candidate solutions on one run, easily dealing with the discontinuous and concave Pareto fronts, and no requirement on differentiable optimization objectives [5]. EMO has also effectively solved many data mining problems [3, 10, 20], such as data clustering [15] and marketing predictions [2]. However, it is seldom applied in classification, since the classifier model is difficult to be effectively encoded in EA. Moreover, it is far more difficult to tradeoff the self-learning of classifiers and information exchange among classifiers in EMO.

This paper first proposes a novel method based on EMO to solve the multi-objective multi-label classification problem. The method is called Multi-Objective Multi-Label algorithm (MOML) which includes two phases: model training and selection. Briefly, MOML designs the effective multi-objective optimization mechanism and the novel method of generating new solutions based on a modified RBF base model in the model training phase. In the model selection phase, users can flexibly select their preferred models in terms of their application scenarios.

3.1 Model Training A good EMO algorithm needs to generate a set of solutions that uniformly distribute along the Pareto front [23], which includes two key issues: (1) solutions prone to converge to the Pareto front and maintain diversity in the evolutionary process; (2) generating promising solutions in each generation. In order to make EMO fit for multi-label learning, we design many novel mechanisms in the following two sections.

3.1.1 Multi-objective Optimization Mechanism Since a good solution is expected to converge to the Pareto front and maintain diversity, the fitness of the solution can be determined by its convergence and diversity. We propose the *non-dominated-sort* and *diversity-estimate* process to effectively evaluate these two measures. Furthermore, the proposed *select-individuals* process selects the best solutions as the next generation population in terms of these measures.

Non-dominated-sort. The *non-dominated-sort* process sorts solutions according to their raw fitness (i.e. objective value O_i). The different value range of objectives (e.g. $Coverage > 1$ and $HammingLoss < 1$) may lead to the situation that some base models reproduce too rapidly. Instead of the raw fitness, this paper employs the rank-based fitness assignment [14] to reassign the fitness (i.e. a rank value) to the solutions, because this method behaves in a more robust manner. In the rank-based fitness assignment, the solution set is divided into different fronts with different ranks. The solutions in the same front are non-dominated to each other and solutions in the higher front are always dominated by some solutions in the lower front. Figure 3 shows an example that 12 solutions are divided into three fronts according to their domination relations. In this way, each solution (i.e. model) $\mathcal{M}_i$ in a front $\mathcal{F}_a$ has a rank value $\mathcal{M}_i^{rank} = a$. It is evident that solution $\mathcal{M}_i$ is better than solution $\mathcal{M}_j$ when $\mathcal{M}_i^{rank} < \mathcal{M}_j^{rank}$.

Diversity-estimate. Along with convergence to the Pareto front, it is also desired that an EA maintains a good spread of solutions. So the solution in the crowded region is more likely to be deleted. To get a diversity estimate of solutions surrounding a particular solution in the population, we design the *diversity-estimate* process that calculates the average distance of two solutions on either side of this solution along each of objectives. It is simple and effective to estimate the diversity of solutions. The diversity estimation of solution $\mathcal{M}_i$, $\mathcal{M}_i^{distance}$, serves as the perimeter of the cuboid formed by using the nearest neighbors as the vertices. As shown in Figure 3, the diversity of this i-th solution in its front is the average side length of the cuboid. The small $\mathcal{M}_i^{distance}$ means solution $\mathcal{M}_i$ is in a more crowded region, which implies a bad diversity.

Select-individuals. Every solution $\mathcal{M}_i$ in the population has two feature values: (1) non-domination rank $\mathcal{M}_i^{rank}$; (2) diversity estimation $\mathcal{M}_i^{distance}$. We define a partial order $\prec$ to compare two solutions, which comprehensively considers both of features.

DEFINITION 4. ***partial order*** $\prec$. *For two solutions $\mathcal{M}_i$ and $\mathcal{M}_j$, $\mathcal{M}_i \prec \mathcal{M}_j$, if and only if*

$$\begin{aligned} &\mathcal{M}_i^{rank} < \mathcal{M}_j^{rank} \vee \\ &(\mathcal{M}_i^{rank} = \mathcal{M}_j^{rank} \wedge \mathcal{M}_i^{distance} > \mathcal{M}_j^{distance}) \end{aligned} \tag{3.4}$$

That is, between two solutions with different non-domination ranks, we prefer the solution with the lower rank. Otherwise, if both solutions belong to the same front, then we prefer the solution that is located in a lesser crowded region. After sorting the population with $\prec$, the *select-individuals* process selects top solutions, which guarantees that good solutions (with low rank and high diversity) will be kept. At the meantime, those promising solutions are also likely to be contained in the population.

The multi-objective optimization mechanism realizes the essence of EA: survival of the fittest. In the optimization process, those solutions performing good on multiple objectives are more likely to survive, while those solutions performing bad or only performing good on some of the optimization objects are more likely to eliminate. As a consequence, solutions in the population automatically tradeoff multi-objectives and make a good balance among them.

3.1.2 Base Model and Evolutionary Operations

In the framework of MOML, many classification models can be used, such as decision tree [19], BP [28] and RBF [26] neural network. Different base models will lead to different genetic representation and operation. Because the structure can be effectively encoded and the weights can be efficiently calculated in close form, the RBF neural network in ML-RBF [26] is selected as the base model in MOML, however with an additional regularization term added to reduce overfitting risks as explained later.

The architecture of RBF is shown in Figure 4(a). It can be briefly summarized as follows: (1) The input of a RBF corresponds to a d-dimension feature vector. (2) The hidden layer of RBF is composed of L sets of prototype vectors, i.e. $\bigcup_{l=1}^{L} C_l$. Here, C_l consists of k_l prototype vectors $< c_1^l, c_2^l, \cdots, c_{k_l}^l >$. For each class $l \in L$, the popular k-means clustering is performed on the set of instances U_l with label l. Thereafter, k_l clustered groups are formed for class l and the j-th centroid $(1 \le j \le k_l)$ is regarded as a prototype vector c_j^l of basis function $\phi_j^l(\cdot)$. (3) Each output neuron is related to a possible class. In the hidden layer of RBF, the number of clusters k_l is settled to be a fraction α of the number of instances in U_l:

$$k_l = \alpha \times |U_l| \tag{3.5}$$

The scale coefficient α controls the structure and complexity of RBF model.

Different from the error function in the original RBF, we add a regularization term into the error function. The regularization term greatly reduces the overfitting risk and improves the stability of solutions as observed in the experiments.

$$E = \frac{1}{2}\sum_{i=1}^{m}\sum_{l=1}^{L}(y_l(\mathbf{x}_i) - t_l^i)^2 + \gamma \sum_{j=0}^{K}\sum_{l=1}^{L} w_{jl}^2 \tag{3.6}$$

where $y_l(\mathbf{x}_i)$ represents the predicted value of instance $\mathbf{x}_i$ on label l, t_l^i is the real value of instance i on label l, $K = \sum_{l=1}^{L} k_l$, and γ is the regularization coefficient. Similar to the derivation of minimizing the error function by scaled-conjugate-gradient descent in [4], the optimal output weights W can be computed in closed form by

$$W = (\Phi'\Phi + \gamma I)^{-1}\Phi' T \tag{3.7}$$

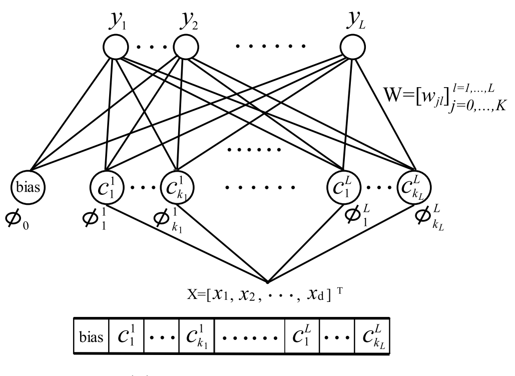
Here $\Phi = [\phi_{ij}]_{m\times(K+1)}$ with elements $\phi_{ij} = \phi_j(\mathbf{x}_i)$, $W = [w_{jl}]_{(K+1)\times L}$ with elements w_{jl}, and $T = [t_{il}]_{m\times L}$ with elements $t_{il} = t_l^i$. Through extensive experiments, the regularization coefficient γ is fixed at 0.1 in this paper.

Genetic representation. According to the structure of RBF, we propose a novel genetic representation that is the sequence of prototypes $< bias, c_1^1, c_1^2, \cdots c_{k_L}^L >$. An example is shown in Figure 4(a). The genetic representation has the following advantages. (1) When the prototypes (c) are determined, the basis functions (ϕ) and the weights (W) can be efficiently computed, which means the performance of RBF mostly depends on the selection of the prototypes. (2) It is easy to design the crossover and mutation operators by tuning these prototypes.

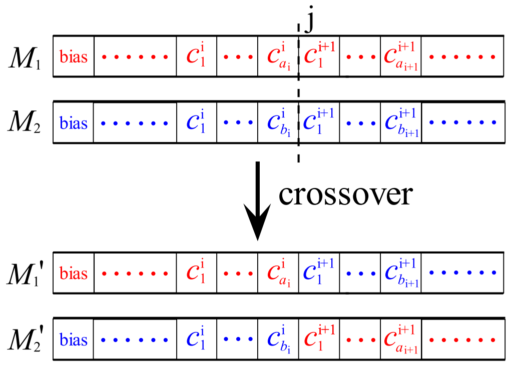
Initialization. When the base model is RBF, the initialization operation of MOML generates a set of RBF models with different scale coefficient α (see Equation 5). As suggested in [26], α is randomly selected from [0.01, 0.02] in the experiments. An advantage of this *Initialization* is that it generates a set of RBF models with different structures, which contributes to the population diversity.

Generate-individuals. Generating new solutions is realized by the *generate-individuals* process. The basic idea is to randomly select parent solutions from the current population based on the roulette wheel selection [1] and do crossover and mutation operation to generate new solutions with the ratio of *cro_Rat* and $1 - $ *cro_Rat*, respectively. Following the general rule in EA, *cro_Rat* is fixed at 0.8, which helps to converge to the Pareto front and maintain the appropriate diversity of the population. MOML applies the roulette wheel selection [1] to assign each solution with an appropriate selection pressure. It guarantees that the better solution has a high yet appropriate selection probability.

Since different RBFs may have different numbers of prototypes, this paper adapts the cut and splice crossover [13] which randomly chooses a crossover point


y_1
y_2
y_L
$W=[w_{jl}]_{j=0,\ldots,K}^{l=1,\ldots,L}$
bias
ϕ_0
X=[$x_1, x_2, \cdots, x_d$]^T

(a) genetic representation



(b) crossover operation

Figure 4: (a) Architecture of RBF and its genetic representation. (b) The crossover operation. The crossover point j is selected between two prototype vectors.

for two RBFs and swaps their prototypes beyond this point. Different from the traditional cut and splice crossover, the crossover point in MoML is randomly selected between two prototype vectors, rather than in an arbitrary position. Figure 4(b) shows such an example, in which the crossover point j is selected between the prototype vector $< c_1^i, \cdots, c_{a_i}^i >$ and $< c_1^{i+1}, \cdots, c_{a_{i+1}}^{i+1} >$. It guarantees that each prototype vector in the newly generated RBF is unabridged cluster centroid. The width of the centroid of the new RBF is recalculated as in [26]. The weights are calculated following Equation 7.

Algorithm 1 MoML-Training

```
Input:
  $\mathcal{D}$: training data      $\mathcal{M}$: base model
  $N$: # base models      $G$: # generations
procedure TRAINING
    Generate $P = \{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_N\}$ randomly
    for $t = 1 : G$ do
        $Q$=generate-individuals($P$)
        $R = P \bigcup Q$
        $F = (\mathcal{F}_1, \mathcal{F}_2, \cdots)$=non-dominated-sort($R$)
        diversity-estimate($F$)
        $P$=select-individuals($F$)
    end for
    return $P$
end procedure
```

According to the structure of RBF, two mutation operations are designed. The mutation operator randomly selects some prototype vectors in a RBF and does the following two structural mutation operations with the same probability. (1) Deleting one prototype. Randomly select one prototype and delete it. (2) Adding one prototype. The center of the new prototype is determined by a random combination of all centroids in this prototype vector.

Although the crossover and mutation operations may not generate the optimal combination of prototypes, they provide an effective method to search the space of prototypes of RBF. The crossover operator reassembles the prototypes of parent solutions, which not only maintains the good genes but also generates new combinations. The mutation operator deletes and adds new prototypes, which helps to extend the search space and maintain diversity. Once a good solution is found in the space of prototypes, it will be kept in population until it becomes a bad one.

3.1.3 Algorithm Framework The training phase of MoML is described in Algorithm 1. MoML transforms the t optimization objectives to a fitness measure by the creation of a number of fronts, sorted according to *non-dominated-sort*. After the fronts have been created, *diversity-estimate* assigns its members density value later to be used for diversity maintenance. In each generation, N new solutions are generated with *generate-individuals*. Of the $2N$ solutions, *select-individuals* selects the N best solutions for the next generation. In this way, a huge elite can be kept from generation to generation.

In MoML, the multi-objective optimization mechanism guides the solutions to converge to Pareto front and maintain the diversity. The genetic operations effectively search the prototypes space of RBF and generate promising solutions. A particular advantage of MoML is that any function can be used as the optimized objective, only if the function can be calculated, without the requirement of being differentiable.

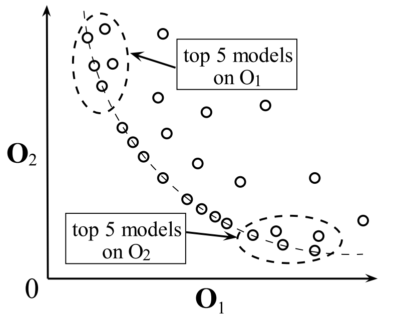


Figure 5: Illustration of model selection.

3.2 Model Selection The model training phase of MOML returns a solution set, which is a unique feature of the multi-objective multi-label classification. The user can make full use of these solutions in terms of their applications. For example, a "best" model can be selected with Gap statistic [15], or the ensemble method can be applied on all models [18]. We can utilize all found models or only the models in the Pareto front. In order to validate the benefits of multiple objectives and the effect of objectives on performances, this paper proposes a dynamic model selection method. That is, we select the top k models on optimization objectives and then make predictions with a majority vote. Assume that instances are independent identical distribution, these selected models will also perform well on the corresponding objective on the testing data. This dynamic model selection method has two advantages: (1) users can flexibly select the preferred models in terms of their applications; (2) the ensemble of the top k models can improve the generalization performances. As shown in Figure 5, the top 5 models are selected from the obtained solution set based on the user preference. The model selection algorithm of MOML is shown in Algorithm 2, in which $\mathcal{M}_i(\mathbf{x}, l)$ means the output of model $\mathcal{M}_i$ on label l for instance $\mathbf{x}$.

Algorithm 2 MOML-Testing

```
Input:
  $\mathcal{U}$: testing data      $O$: optimization objective set
  $P$: model set                   $k$: # top models
procedure TESTING
    for optimization objective $O_i \in O$ do
        Sort $P$ in an ascending order by $O_i$
        Select top-$k$ models $\{\mathcal{M}_1, \cdots, \mathcal{M}_k\}$ from $P$
        for $\mathbf{x} \in \mathcal{U}$ do
            $Y(\mathbf{x}) = \{l | \frac{1}{k} \sum_{i=1}^{k} \mathcal{M}_i(\mathbf{x}, l) > 0, l \in \mathcal{L}\}$
        end for
    end for
end procedure
```

3.3 Complexity Analysis Let d be the number of features of instances, m and n be the number of training and testing instances respectively, L be the number of labels. We consider the time complexity of RBF first. Two main time-consuming components of RBF are the k-means clustering and calculating $\Phi = [\phi_{ij}]_{m \times (K+1)}$ for all training instances. For simplicity, suppose each label has the same number of instances $\frac{m}{L}$, and thus the number of centroid is $\alpha \frac{m}{L}$. The complexity of a k-means clustering is $O(\alpha (\frac{m}{L})^2)$ (the iteration number in k-means is fixed, so it is omitted here). L k-means clustering are needed, so the total complexity is $O(\alpha m^2/L)$. ϕ_{ij} needs to calculate the distance to each prototype vector c_j for each instance $\mathbf{x}_i$, and thus its complexity is $O(\alpha d m^2)$. In all, the RBF has the following complexity:

$$O(\alpha m^2/L + \alpha d m^2) \tag{3.8}$$

For MOML, it needs to generate N RBFs and evaluate NG new RBFs. The complexity of MOML in RBF is $O(\alpha N m^2/L + \alpha N G d m^2)$. The complexity of the genetic operation in MOML is $O(GN^2)$. Since $N \ll m$, the total time complexity of MOML in the training phase is

$$O(\alpha N m^2/L + \alpha N G d m^2) \tag{3.9}$$

There are k models to make predictions on the testing data, so the time complexity of the testing phase is

$$O(\alpha k d n^2) \tag{3.10}$$

Since $k \ll NG$, the testing phase is much faster than the training phase.

4 Experiments

4.1 Experimental Setup

4.1.1 Data Collection We tested our algorithm on seven real-world multi-label datasets from three different domains as summarized in Table 1. The first dataset is Yeast [18, 26, 27, 28] in biology, where the task is to predict the gene functional classes of the Yeast Saccharomyces cerevisiae. The second dataset Image [18, 26, 27, 28] involves the task of automatic image annotation for scene images. The other five dataset RCV1-1–RCV1-5 are from RCV1-v2 [25, 27], where the task is to predict topic categories of each text document.

4.1.2 Evaluation Metrics The performance evaluation for multi-label learning is much more complicated than single-label problems. Here, we adopt nine state-of-the-art multi-label evaluation metrics which are most

Table 1: Summary of the experimental datasets.

	Dataset		
Property	Yeast	Image	RCV1(1-5)
# instance	2417	2000	3000
# feature	103	944	101
# label	14	294	5
Domain	biology	media	text

popular in the literature. To the best of our knowledge, few works on multi-label learning have conducted experimental evaluation on such comprehensive comparisons over the nine metrics. These metrics are briefly summarized here, where "↓" indicates the smaller the value the better the performance; "↑" indicates the larger the value the better the performance.

• *Hamming Loss* (*HL*)↓ [21]: evaluates the average error rate over all the binary labels.

• *Micro F1* (*MicF1*)↑ [12]: evaluates a classifier's label set prediction performance, which considers both micro-average of *precision* and *recall* on all binary labels with equal importance.

• *Macro F1* (*MacF1*)↑ [12]: evaluates a classifier's label set prediction performance, which considers both macro-average of *precision* and *recall* with equal importance.

• *Subset 0/1 Loss* (*SL*)↓ [7, 12]: evaluates the average percentage when a classifier's label set prediction is exactly correct.

• *Accuracy* (*Acc*)↑ [21]: evaluates the average fraction of correct labels across all examples.

• *Ranking Loss* (*RL*)↓ [21]: evaluates the average fraction of label pairs that are disordered for an example.

• *One Error* (*OE*)↓ [9, 21]: evaluates how many times the top-ranked label by a classifier is not in the true label set of an example.

• *Coverage* (*Cov*)↓ [9, 21]: evaluates how many steps are needed, on average, to move down the label list in order to cover all the true labels of an example.

• *Average Precision* (*AP*)↑ [9, 21]: evaluates the average fraction of true labels ranked above a particular label.

4.1.3 Compared Methods We compare our method with four baseline methods which optimize over different single objectives. In Moml, any subset of metrics listed above can be used as the optimization objectives. Here, we employ two pairs representative subsets of evaluation metrics, i.e. $\{HL, RL\}$ and $\{MicF1, AP\}$. The $\{HL, RL\}$ objective subset includes two popular objectives that have already been directly optimized in previous single objective approaches [9, 26, 28]. The $\{MicF1, AP\}$ objective subset includes two most useful performance criteria which have not yet been directly optimized before. In addition, these two pairs of objectives are potentially conflicting. These compared methods are summarized as follows.

• Moml$_{\{HL,RL\}}$: The proposed Moml approach with the first objective subset ($\{HL, RL\}$), which outputs a set of models with different preferences on each objective. In order to verify the quality of the outputted solution set, we report two versions of model selection based on the top k models in terms of HL and RL, respectively. The corresponding algorithms are called Moml$_{\{\underline{HL},RL\}}$ and Moml$_{\{HL,\underline{RL}\}}$. These two combined models correspond to the two application preferences over the two objectives.

• Moml$_{\{MicF1,AP\}}$: The proposed Moml approach with the second objective subset $\{MicF1, AP\}$. Similarly, we report two versions of model selection in terms of $MicF1$ and AP and the corresponding algorithms are called Moml$_{\{\underline{MicF1},AP\}}$ and Moml$_{\{MicF1,\underline{AP}\}}$, respectively. Note that, in order to be fit for the minimization problem, $1 - MicF1$ and $1 - AP$ are used in Moml.

• ML-rbf [26]: Based on RBF neural network, the method explicitly optimizes the HL criterion.

• BP-mll [28]: This method is based on BP neural network, which explicitly optimizes the RL criterion.

• ML-knn [29]: The KNN based lazy multi-label learning method optimizes a posterior principle which is not directly related to any single performance criterion.

• Ecc [18]: It is an ensemble of classifier chains which encode the multi-label correlations in the multi-label classification process.

The population size and running generation of Moml are set as 30 and 10. k is 9 (i.e. 30% of the population size) in the top k model selection. ML-rbf is implemented with fixed parameters of $\alpha = 0.01$ and $\mu = 1.0$, as suggested in the literature [26]. For BP-mll, as indicated in the literature [28], the number of hidden neurons is set to be 20% of the number of input neurons, and the number of training epochs is fixed at 100 with learning rate of 0.05. For ML-knn, the number of nearest neighbors considered is set to 10 and Euclidean distance is used as the distance measure [29]. For Ecc, the ensemble size is set to 10 and sampling ratio is set to 67% [18].

4.2 Performance Comparison Ten-fold cross-validation is performed on each experimental dataset. On each dataset, we report the average values of each algorithm with the ranks based on its results. All

Table 2: Results on the Yeast dataset. The results are reported as "average performance + (rank)", where "↓" indicates that the smaller the value, the better the performance; "↑" indicates the larger the better.

	Methods							
Criteria	Moml$_{\{\underline{HL},RL\}}$	Moml$_{\{HL,\underline{RL}\}}$	Moml$_{\{\underline{MicF1},AP\}}$	Moml$_{\{MicF1,\underline{AP}\}}$	ML-rbf	BP-mll	ML-knn	Ecc
HL ↓	0.1883 (1)*	0.1887 (3)	0.1885 (2)	0.1889 (4)	0.1935 (5)	0.2120 (8)	0.1949 (6)	0.2056 (7)
RL ↓	0.1596 (2)	0.1595 (1)*	0.1600 (3)	0.1603 (4)	0.1621 (5)	0.1723 (7)	0.1669 (6)	0.2776 (8)
SL ↓	0.8051 (5)	0.8039 (3)	0.7997 (2)	0.8047 (4)	0.8163 (6)	0.8519 (8)	0.8167 (7)	0.7968 (1)*
OE ↓	0.2197 (6)	0.2172 (2)	0.2193 (5)	0.2180 (3)	0.2189 (4)	0.2308 (8)	0.2304 (7)	0.1742 (1)*
Cov ↓	6.2027 (3)	6.2122 (4)	6.1868 (2)	6.1861 (1)*	6.2465 (5)	6.3562 (7)	6.2647 (6)	7.1431 (8)
MicF1 ↑	0.6572 (3)	0.6562 (5)	0.6576 (1)*	0.6569 (4)	0.6486 (6)	0.6468 (7)	0.6398 (8)	0.6574 (2)
AP ↑	0.7752 (4)	0.7753 (3)	0.7756 (2)	0.7759 (1)*	0.7720 (5)	0.7534 (7)	0.7650 (6)	0.7313 (8)
Acc ↑	0.5267 (2)	0.5248 (5)	0.5261 (3)	0.5257 (4)	0.5170 (7)	0.5185 (6)	0.5087 (8)	0.5404 (1)*
MacF1 ↑	0.3888 (3)	0.3871 (4)	0.3889 (2)	0.3897 (1)*	0.3668 (6)	0.3457 (8)	0.3737 (5)	0.3647 (7)
AveRank↓	**(3.22)**	**(3.33)**	**(2.44)**	**(2.89)**	(5.44)	(7.33)	(6.56)	(4.78)

Table 3: Results on the Image dataset. The results are reported as "average performance + (rank)", where "↓" indicates that the smaller the value, the better the performance; "↑" indicates the larger the better.

	Methods							
Criteria	Moml$_{\{\underline{HL},RL\}}$	Moml$_{\{HL,\underline{RL}\}}$	Moml$_{\{\underline{MicF1},AP\}}$	Moml$_{\{MicF1,\underline{AP}\}}$	ML-rbf	BP-mll	ML-knn	Ecc
HL ↓	0.1581 (1)*	0.1591 (4)	0.1589 (3)	0.1583 (2)	0.1653 (5)	0.2559 (8)	0.1703 (6)	0.1786 (7)
RL ↓	0.1468 (2)	0.1454 (1)*	0.1476 (3)	0.1479 (4)	0.1558 (5)	0.3532 (8)	0.1708 (6)	0.2411 (7)
SL ↓	0.5695 (2)	0.5750 (4)	0.5765 (6)	0.5745 (3)	0.6020 (7)	0.7890 (8)	0.5755 (5)	0.5385 (1)*
OE ↓	0.2695 (4)	0.2655 (2)	0.2680 (3)	0.2650 (1)*	0.2860 (5)	0.5700 (8)	0.3150 (7)	0.2935 (6)
Cov ↓	0.8615 (3)	0.8610 (2)	0.8650 (4)	0.8570 (1)*	0.8955 (5)	1.6790 (8)	0.9500 (6)	0.9715 (7)
MicF1 ↑	0.6062 (3)	0.6038 (5)	0.6067 (2)	0.6052 (4)	0.5798 (7)	0.3524 (8)	0.5925 (6)	0.6380 (1)*
AP ↑	0.8223 (3)	0.8232 (2)	0.8219 (4)	0.8241 (1)*	0.8118 (5)	0.6139 (8)	0.7967 (7)	0.7977 (6)
Acc ↑	0.5126 (2)	0.5083 (6)	0.5084 (5)	0.5096 (4)	0.4778 (7)	0.2769 (8)	0.5097 (3)	0.5985 (1)*
MacF1 ↑	0.6065 (2)	0.6033 (5)	0.6048 (4)	0.6054 (3)	0.5773 (7)	0.2687 (8)	0.5936 (6)	0.6441 (1)*
AveRank ↓	**(2.44)**	**(3.44)**	**(3.78)**	**(2.56)**	(5.89)	(8.00)	(5.56)	(4.33)

Table 4: Results on the RCV1-1 dataset. The results are reported as "average performance + (rank)", where "↓" indicates that the smaller the value, the better the performance; "↑" indicates the larger the better.

	Methods							
Criteria	Moml$_{\{\underline{HL},RL\}}$	Moml$_{\{HL,\underline{RL}\}}$	Moml$_{\{\underline{MicF1},AP\}}$	Moml$_{\{MicF1,\underline{AP}\}}$	ML-rbf	BP-mll	ML-knn	Ecc
HL ↓	0.0147 (1)*	0.0149 (3)	0.0148 (2)	0.0150 (4)	0.0165 (5)	0.0320 (8)	0.0222 (7)	0.0214 (6)
RL ↓	0.0180 (1)*	0.0181 (2)	0.0183 (4)	0.0182 (3)	0.0196 (5)	0.0826 (7)	0.0684 (6)	0.2506 (8)
SL ↓	0.6423 (4)	0.6410 (2)	0.6373 (1)*	0.6411 (3)	0.6873 (6)	1.0000 (8)	0.7770 (7)	0.6673 (5)
OE ↓	0.0647 (3)	0.0650 (4)	0.0640 (2)	0.0637 (1)*	0.0743 (5)	0.5340 (8)	0.2850 (7)	0.1033 (6)
Cov ↓	6.7567(1)*	6.7630 (2)	6.7893 (3)	6.7993 (4)	6.9390 (5)	20.597 (7)	17.523 (6)	35.973 (8)
MicF1 ↑	0.7097 (2)	0.7082 (3)	0.7098 (1)*	0.7081 (4)	0.6774 (5)	0.4177 (8)	0.5421 (7)	0.6483 (6)
AP ↑	0.8620 (4)	0.8629 (1)*	0.8624 (3)	0.8628 (2)	0.8443 (5)	0.4717 (8)	0.6666 (7)	0.6990 (6)
Acc ↑	0.6070 (2)	0.6063 (3)	0.6079 (1)*	0.6058 (4)	0.5689 (6)	0.2655 (8)	0.4113 (7)	0.5820 (5)
MacF1 ↑	0.2546 (2)	0.2537 (4)	0.2553 (1)*	0.2543 (3)	0.2203 (5)	0.0539 (8)	0.1960 (7)	0.2177 (6)
AveRank ↓	**(2.22)**	**(2.67)**	**(2.00)**	**(3.11)**	(5.22)	(7.78)	(6.78)	(6.22)

Table 5: The average ranks (mean±std) for each method over 7 datasets, including Yeast, Image and RCV1 (subset 1-5).

	Methods							
Criteria	Moml$_{\{\underline{HL},RL\}}$	Moml$_{\{HL,\underline{RL}\}}$	Moml$_{\{\underline{MicF1},AP\}}$	Moml$_{\{MicF1,\underline{AP}\}}$	ML-rbf	BP-mll	ML-knn	Ecc
HL	1.14±0.38*	3.14±0.38	2.00±0.58	3.71±0.76	5.00±0.00	8.00±0.00	6.71±0.49	6.29±0.49
RL	1.71±0.49	1.29±0.49*	3.57±0.53	3.43±0.53	5.00±0.00	7.14±0.38	6.00±0.00	7.86±0.38
SL	3.43±1.40	2.43±0.98	2.14±1.77*	3.29±0.49	6.14±0.38	8.00±0.00	6.71±0.76	3.86±1.95
OE	3.57±1.13	3.43±0.98	2.57±1.13	1.29±0.76*	4.86±0.38	8.00±0.00	6.86±0.38	5.43±1.99
Cov	2.00±1.29*	2.29±0.76	3.00±0.58	2.71±1.60	5.00±0.00	7.14±0.38	6.00±0.00	7.86±0.38
MicF1	2.14±0.38	3.57±0.98	1.29±0.49*	4.00±0.67	5.43±0.79	7.86±0.38	7.00±0.58	4.71±2.21
AP	3.86±0.38	1.86±0.69	2.86±0.69	1.43±0.79*	5.00±0.00	7.57±0.53	6.43±0.53	7.00±1.00
Acc	2.00±1.32	3.71±1.25	1.86±1.57*	4.00±0.58	6.29±0.49	7.71±0.76	6.57±1.62	3.86±1.95
MacF1	2.14±0.69	4.14±0.38	1.57±1.13*	2.71±0.76	5.71±0.76	8.00±0.00	6.57±0.79	5.14±1.95

Table 6: Average running time (second).

Data	Methods									
	MOML$_{\{\underline{HL},RL\}}$		ML-RBF		BP-MLL		ML-KNN		ECC	
Set	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Yeast	757	3.9	15.6	0.6	12,100	17.5	2.5	1.2	39.7	5.9
Image	343	1.6	2.8	0.2	12,500	5.3	9.1	1.2	39.3	4.2
RCV1-1	34,400	89	816	10.5	62,300	164	149	4.2	273	137

experiments are conducted on machines with Intel Xeon Quad-Core CPUs of 2.26 GHz and 24 GB RAM.

Due to the limited space, we only show the results of the average values of 9 metrics on Yeast, Image and RCV1-1 in Tables 2-4, where '*' indicates the best result on each criterion and '_' indicates the performance of MOML on its optimization objective. The other four datasets on RCV1 have very similar results with RCV1-1. From these tables, we can observe that the four versions of the MOML method rank first four on most metrics and they always have the best average ranks on each dataset. Furthermore, Table 5 summarizes the mean and standard deviation of the rank values for each method over 9 metrics on all seven datasets. Although each MOML only optimizes two objectives, it always performs better than the baselines on all metrics. Moreover, Table 5 shows that each variant of the four MOML algorithms does provide the best average rank on its primary objective, such as MOML$_{\{\underline{HL},RL\}}$ on HL, MOML$_{\{HL,\underline{RL}\}}$ on RL, etc. Other methods may occasionally outperform our approach on some of the metrics in a few of the datasets, but not consistently. These results validate our intuition that the multi-objective optimization in our MOML can effectively tradeoff among multiple objectives and avoid the local optimal to improve the overall performance almost on all metrics. It is worth noticing that each version of MOML can achieve the best average performance on its primary objective, which supports the quality of the outputted solution set and the flexibility of our approach to different application scenarios with different preferences.

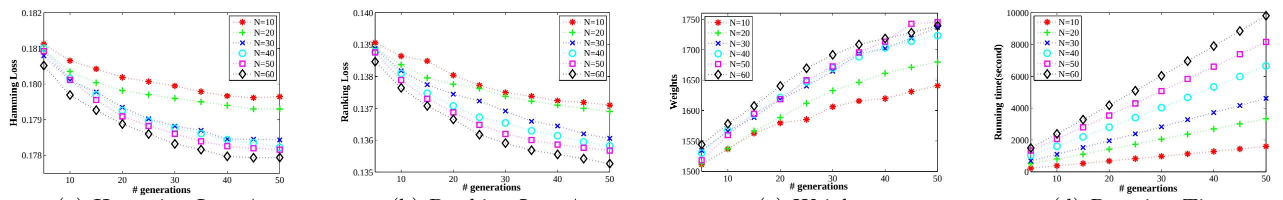
Table 6 shows the average running time. We only show one result of four versions of MOML, since the four versions have the same time complexity. Although MOML is slower than ML-RBF, ML-KNN and ECC, it is still faster than BP-MLL in the training phase. In the testing phase, MOML is faster than BP-MLL and ECC.

4.3 Parameter Settings There are two genetic operation related parameters governing the MOML, i.e. the population size N and the running generations G. Figure 6 illustrates the evolutionary characteristics of MOML$_{\{HL,RL\}}$ on the Yeast data with ten-fold cross-validation, under different parameter configurations. Specifically, when the population size N increases from 10 to 60 with an interval of 10, we report the average of performances, running time and weights (the sum of absolute value of W) by combining all the models in the population.

It is evident from Figure 6 that, when the population size N is fixed, the performance (i.e. *Hamming Loss* and *Ranking Loss*) of MOML consistently improves as the running generation increases. At the meantime, the weights of RBF and running time also increase. Figure 6 also clearly shows that the large population size usually leads to better performances accompanying with the increase of weights and running time. Observing the trend of weight curves in Figure 6(c), we can find that, although the weights consistently increase, the rate of increase becomes small. If we do not add the regularization term in the error function of RBF (see Equation 6), the weights will increase sharply, which means these models are overfitting. Figure 6(d) illustrates that the running time of MOML increases linearly with the population size N and running generation G, which validates the time complexity of MOML in Equation 11.

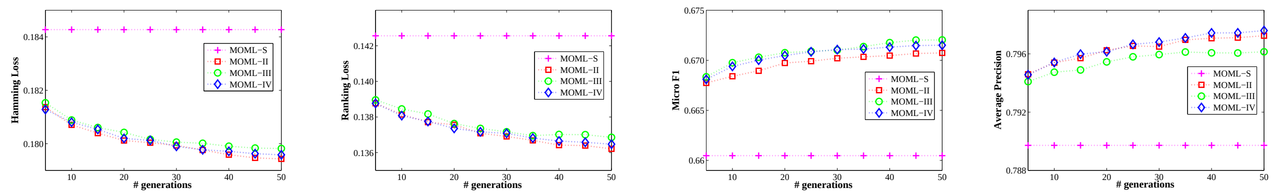
In addition, the number of top models k also affects the performance of MOML. The larger k means ensembling more classifiers, which usually leads to better performances and longer running time. However, it has less preference to the optimization objective. In the experiments, we settled the appropriate, not optimal, parameters for our MOML. It achieves good performances with an acceptable running time. In real applications, users can settle these parameters in terms of the tradeoff of the effectiveness and efficiency.

4.4 Influence of The Number of Objectives Our previous experiments only show the cases with a pair of objectives. However, more objective functions also can be included in MOML. In order to study the performances of MOML with different number of objectives, here we consider four objective functions (i.e. HL, RL, $MicF1$, and AP) and three versions of MOML which optimize the first 2, 3 and 4 objectives respectively. The corresponding algorithms are called MOML-II, MOML-III, and MOML-IV. We also consider a special case of



(a) Hamming Loss ↓ (b) Ranking Loss ↓ (c) Weights (d) Running Time

Figure 6: The impact of the running generations and population size on performance. "↓" indicates the smaller the better; "↑" indicates the larger the better.



(a) Hamming Loss ↓ (b) Ranking Loss ↓ (c) Micro F1 ↑ (d) Average Precision ↑

Figure 7: Influence of the number of objectives. MOML-II, MOML-III and MOML-IV represent the MOML with 2, 3, and 4 optimization objectives, respectively. MOML-S does not do any multi-objective optimization. "↓" indicates the smaller the better; "↑" indicates the larger the better.

MOML, called MOML-S, where the running generation of MOML is 0. That is, MOML-S does not do any genetic operation and multi-objective optimization. So it is just the ensemble of multiple RBFs, and its performances are constant along the evolutionary process. Ten-fold cross-validation are reported on the Yeast data.

The results are shown in Figure 7. It is obvious that MOML-(II,III,IV) perform better than MOML-S on all four objectives, which supports the effectiveness of the genetic operation and multi-objective optimization in MOML. We can also find that MOML has better performance on the optimization objectives than on non-optimization objectives. However, when more objectives are included in MOML, the problem becomes even more difficult, and the model space extends greatly. Thus, the performances of MOML-IV are not as good as MOML-II on *HL* and *RL*. However, MOML-IV can achieve the better performance on *MicF1* by including *MicF1* in its objective set.

5 Related Work

Our work is related to multi-label learning and evolutionary multi-objective optimization. Here we briefly discuss the most related work in these two domains. MOML has the same base classifier with ML-RBF [26]: RBF. However, MOML adds a regularization term in the error function of RBF, which greatly reduces the over-fitting risk of RBF. Similar to ECC [18], EPS [17], and RAKEL [22], MOML also employs the ensemble method in the model selection phase, whereas MOML generates a solution set through evolutionary multi-objective optimization. Petterson and Caetano [16] have been aware that the evaluation measures are as diverse as the applications. However, their method still optimizes a single criterion by appropriate surrogate. Different from ML-2OKM [24] which also optimizes two particular objectives with an existing EMO, MOML's optimization objectives can be any evaluation metrics and its base model is the multi-label classifier. Dembczyński et al. [8] analyze the connection between loss functions in multi-label classification, which helps to select appropriate optimization objectives in MOML.

Traditional evolutionary multi-objective optimization focuses on numerical optimization problems [6]. It is also a promising method for data mining [3, 5]. Handle and Knowles [15] apply EMO to boost clustering performance. Shi et al. [20] use EMO to generate a set of classifiers, while their work focuses on the ensemble of classifiers. Chen and Yao [4] employ the multi-objective neural network ensemble to improve classification performances, whereas it focuses on the single-label classification problem.

6 Conclusion

In this paper, we first studied the multi-objective multi-label classification problem and proposed a novel Multi-Objective Multi-Label algorithm (MOML). MOML can simultaneously optimize over multiple objectives and return a set of solutions. In applications, users can select the top k models in terms of the preferred objective and make predictions on the testing data. Experiments show that MOML not only achieves best performances on the optimization objectives, but also improves the performances on most of the other state-of-the-art criteria for multi-label classification.

Acknowledgments. It is supported by the National Natural Science Foundation of China (No. 60905025, 61074128, 61035003).

References

[1] J. Baker. Adaptive selection methods for genetic algorithms. In *ICGA*, pages 100–111, 1985.

[2] S. Bhattacharyya. Evolutionary algorithms in data mining: Multi-objective performance modeling for direct marketing. In *KDD*, pages 465–473, 2000.

[3] J. B. Bi. Multi-objective programming in SVMs. In *ICML*, pages 35–42, 2003.

[4] H. Chen and X. Yao. Multiobjective neural network ensembles based on regularized negative correlation learning. *Transactions on Knowledge and Data Engineering*, 22(12):1738–1751, 2010.

[5] K. Deb. *Multiobjective Optimization using Evolutionary Algorithms*. Wiley, UK, 2001.

[6] K. Deb, A. Pratab, S. Agarwal, and T. MeyArivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation*, 6(2):182–197, 2002.

[7] K. Dembczyński, W. Cheng, and E. Hullermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286, 2010.

[8] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hullermeier. Regret analysis for performance metrics in multi-label classification: the case of hamming and subset zero-one loss. In *ECML/PKDD*, pages 280–295, 2010.

[9] A. Elisseeff and J. Weston. A kernel method for multilabelled classification. In *NIPS*, pages 681–687, 2002.

[10] A. A. Freitas. A critical review of multi-objective optimization in data mining: a position paper. *SIGKDD Exploration*, 6(2):77–86, 2006.

[11] J. Furnkranz and P. A. Flach. An analysis of rule evaluation metrics. In *ICML*, pages 202–209, 2003.

[12] N. Ghamrawi and A. McCallum. Collective multilabel classification. In *CIKM*, pages 195–200, 2005.

[13] D. Goldberg, K. Deb, H. Kargupta, and G. Harik. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. In *ICGA*, pages 56–64, 1993.

[14] D. E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Wesley, USA: Boston, 1989.

[15] J. Handle and J. Knowles. An evolutionary approach to multiobjective clustering. *Transaction on Evolutionary Computation*, 11(1):56–76, 2007.

[16] J. Petterson and T. Caetano. Reverse multi-label learning. In *NIPS*, pages 421–426, 2010.

[17] J. Read, B. Pfahringer, and G. Holmes. Multi-label classification using ensembles of pruned sets. In *ICDM*, pages 995–1000, 2008.

[18] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In *ECML*, pages 254–269, 2009.

[19] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, and S. Dzeroski. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11(2):38–47, 2010.

[20] C. Shi, X. Kong, P. S. Yu, and B. Wang. Multi-label ensemble learning. In *ECML/PKDD*, 2011.

[21] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685, 2010.

[22] G. Tsoumakas and I. P. Vlahavas. Random k-labelsets: an ensemble method for multilabel classification. In *ECML*, pages 406–417, 2007.

[23] D. A. V. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 18(2):125–147, 2000.

[24] H. Xu and J. Xu. Designing a multi-label kernel machine with two-objective optimization. In *AICI*, pages 282–291, 2010.

[25] B. S. Yang, J. T. Sun, T. J. Wang, and Z. Chen. Effective multi-label active learning for text classification. In *KDD*, pages 917–925, 2009.

[26] M.-L. Zhang. ML-RBF: RBF neural networks for multi-label learning. *Neural Process Letters*, 29(2):61–74, 2009.

[27] M.-L. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *KDD*, pages 999–1007, 2010.

[28] M.-L. Zhang and Z.-H. Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[29] M.-L. Zhang and Z.-H. Zhou. ML-kNN: a lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.