Multi-Label Classification Based on Multi-Objective Optimization

CHUAN SHI, Beijing University of Posts and Telecommunications
XIANGNAN KONG, University of Illinois at Chicago
DI FU, Beijing University of Posts and Telecommunications
PHILIP S. YU, University of Illinois at Chicago and King Abdulaziz University
BIN WU, Beijing University of Posts and Telecommunications

Multi-label classification refers to the task of predicting potentially multiple labels for a given instance. Conventional multi-label classification approaches focus on 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 trade-offs among multiple inconsistent objectives, such as minimizing *Hamming Loss* while maximizing *Micro F1*. In this article, 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 inconsistent, even 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 different trade-offs among multiple objectives. So users can flexibly construct various predictive models from the solution set, which provides 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 by optimizing over multiple objectives simultaneously.

Categories and Subject Descriptors: I.5.2 [**Pattern Recognition**]: Design Methodology

General Terms: Measurement

Additional Key Words and Phrases: Classification, multi-label classification, multi-objective optimization, model selection, classifier design and evaluation, pattern analysis

ACM Reference Format:
Chuan Shi, Xiangnan Kong, Di Fu, Philip S. Yu, and Bin Wu. 2014. Multi-label classification based on multi-objective optimization. ACM Trans. Intell. Syst. Technol. 5, 2, Article 35 (April 2014), 22 pages.
DOI: http://dx.doi.org/10.1145/2505272

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.

This work is supported by the National Natural Science Foundation of China (No. 60905025, 61375058, 61035003, 71231002). It is also supported by the National Key Basic Research Program (973 Program) of China (2013CB329603). P. Yu is supported in part by NSF through grants CNS-1115234, DBI-0960443, and OISE-1129076.
Authors' addresses: C. Shi (corresponding author), Beijing University of Posts and Telecommunications, Beijing, China; email: shichuan@bupt.edu.cn; X. Kong, University of Illinois at Chicago; email: kongxn@gmail.com; D. Fu, Beijing University of Posts and Telecommunications; email: fudi@bupt.edu.cn; P. S. Yu, University of Illinois at Chicago and King Abdulaziz University; email: psyu@uic.edu; B. Wu, Beijing University of Posts and Telecommunications; email: wubin@bupt.edu.cn.

DOI: http://dx.doi.org/10.1145/2505272

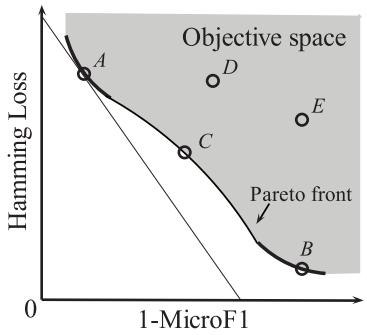
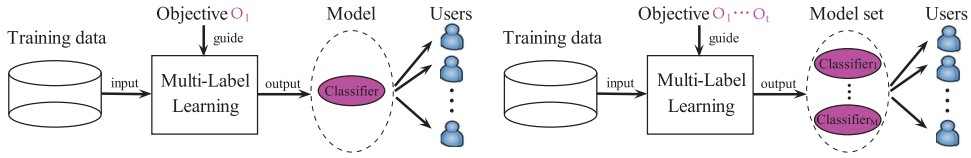


Fig. 1. Illustration of optimizing over multiple objectives.

For example, in text categorization, one document can belong to multiple categories [Yang et al. 2009]; in image classification, an image is usually associated with multiple labels which are characterized by different regions in the image [Zha et al. 2008]; in bioinformatics, one gene sequence may serve multiple functions [Elisseeff and Weston 2002]; in video annotation, an video can be tagged with multiple labels simultaneously [Zha et al. 2009]. 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 [Dembczyński et al. 2010a; Petterson and Caetano 2010; Zhang and Zhang 2010].

Conventional multi-label classification approaches focus on the 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 [Tsoumakas et al. 2010]) or a heuristic function (e.g., the posteriori principle in ML-KNN [Zhang and Zhou 2007]). 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 the classifier's performance. However, in multi-label classification, many criteria are proposed to evaluate the classification performance from different perspectives, and some criteria are inconsistent [Gao and Zhou 2011] or even conflict [Dembczyński et al. 2010b]. Gao and Zhou [2011] prove that no convex surrogate loss is consistent with the ranking loss. Dembczyński et al. [2010b] elaborate the connection among these criteria and point out that some loss functions are essentially conflicting, such as Hamming Loss [Tsoumakas et al. 2010] and Subset 0/1 Loss [Ghamrawi and McCallum 2005]. 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 [Tsoumakas et al. 2010] and Micro F1 [Ghamrawi and McCallum 2005] are concerned, one may minimize Hamming Loss, maximize Micro F1 (i.e., minimize $1 - \textit{Micro } F1$), or optimize both of them simultaneously. An example of results is shown in Figure 1. Due to the inconsistency existing in these two objectives in some conditions, only optimizing over *Hamming Loss* may lead to bad performance on *Micro F1* (e.g., solution B), or vice versa (e.g., solution A). However, it is obvious that 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 conditions where the concerned objectives are inconsistent or potential conflicting. This helps to balance the trade-off among these objectives and comprehensively improve performances of multi-label classification, not limiting to one single



(a) Single-objective multi-label classification. (b) Multi-objective multi-label classification.

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

criterion. In addition, the simultaneous optimization over multiple objectives is also practically needed in many multi-label classification tasks [Tsoumakas et al. 2010]. 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 [Tsoumakas et al. 2010] and Micro F1 [Ghamrawi and McCallum 2005] 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 [Petterson and Caetano 2010]. 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 trade-offs 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 [Gao and Zhou 2011]. The loss functions in multi-label classification are usually difficult to optimize directly because of non-convexity and discontinuity. Many multi-label classification approaches work with surrogate loss functions, such as Ranking Loss [Tsoumakas et al. 2010] and Hamming Loss [Tsoumakas et al. 2010]). (2) Multi-objective optimization is much more difficult than single objective optimization. It is not easy to effectively trade-off 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 [Furnkranz and Flach 2003]) and the trade-offs among objectives can be exploited by tuning weights. However, it is hard to choose the weights in real applications and cannot discover the solutions in the concave Pareto front [Freitas 2006]. For example, the weighted sum method can find A and B in Figure 1, but it cannot discover C.

In this article, we study the problem of multi-objective multi-label classification and propose a novel solution, called MOML (multi-objective-based 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 these challenges. (1) EMO does not require the 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 trade-offs

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 we propose two model selection strategies to make full use of these models and make predictions on the testing data. And thus, users can flexibly apply these model selection strategies in different applications. 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, in comparison experiments of model selection strategies, MOML can effectively adapt to the user's preferences in different applications by achieving better performances on the preferred objectives.

2. RELATED WORK

Multi-label classification has been well developed in the past decade. There are two basic ways to solve this problem: problem transformation and algorithm adaptation. In problem transformation, a multi-label problem is transformed into multiple single-label problems. For each single-label problem, a single-label classifier is learnt, and then these single-label classifiers are combined for the original multi-label problem. Many base learners have been employed in problem transformation approaches, such as Support Vector Machines [Godbole and Sarawagi 2004], Naive Bayes [Ji et al. 2008], and k-Nearest-Neighbor methods [Zhang and Zhou 2007]. In algorithm adaptation, it modifies specific learning algorithms to solve multi-label data directly. The representative approaches involve decision trees [Vens et al. 2008], AdaBoost [Schapire and Singer 2000], and BP-MLL [Zhang and Zhou 2006]. These algorithms usually optimize only one evaluation metric explicitly or implicitly, whereas our MOML explicitly optimizes multiple objectives at the same time.

Since ensemble learning can effectively improve learners' generalization performances, it has been widely applied in multi-label learning to build a set of base learners [Read et al. 2008, 2009; Shi et al. 2011; Tsoumakas et al. 2008; Tsoumakas and Vlahavas 2007]. For example, RAKEL [Tsoumakas and Vlahavas 2007] trains each single-label base learner for the prediction of each element in the powerset of the label set, and the single-label base learner in EPS [Read et al. 2008] is built for a pruning label subset. Similar to these approaches, MOML also employs the ensemble method in the model selection phase, whereas MOML generates the solution set through evolutionary multi-objective optimization. Recently, some researches began to be aware of conflict existing in measure criteria [Dembczyński et al. 2010b; Petterson and Caetano 2010; Xu and Xu 2010]. Petterson and Caetano [2010] point out the evaluation measures are as diverse as the applications. However, their method still optimizes a single criterion by appropriate surrogate. Different from ML-2OKM [Xu and Xu 2010] which also optimizes two particular objectives with an existing EMO, MOML's optimization objectives can be any evaluation metrics and its base model is a multi-label classifier. Dembczyński et al. [2010b] analyze the connection between loss functions in multi-label classification, which helps to select appropriate optimization objectives in MOML. In addition, there is an increasing attention on the consistency of multi-label learning [Gao and Zhou 2011; Kotlowski et al. 2011; Dembczyński et al. 2012]. Since multi-label loss functions are usually difficult to optimize directly owing to non-convexity and discontinuity, the surrogate loss functions are widely used in multi-label classification. However, Gao and Zhou [2011] find that no convex surrogate loss is consistent with the ranking loss. Then Dembczyński et al. [2012] prove that common convex surrogates used for binary classification are consistent for the minimization of rank loss. These theoretical analysis further disclose the inconsistency among

surrogate loss functions, which implies the importance of multi-objective multi-label classification.

Multi-objective optimization is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints, which is widely existing in many fields (e.g., decision and optimization). Many methods have been proposed to solve this problem (e.g., weighted sum method [Furnkranz and Flach 2003]), among which evolutionary algorithm [Goldberg 1989] has been proven to be an effective solution. This kind of solutions is also called EMO technique [Deb 2001]. EMO simultaneously optimizes multiple objectives through population evolution, in which individuals reproduce through evolutionary operation (e.g., crossover and mutation) and obey the Darwinian evolution: survival of the fittest. Traditional EMO focuses on numerical optimization problems [Deb et al. 2002]. However, EMO begins to be applied in data mining problems in recent years [Freitas 2006], such as data clustering [Handle and Knowles 2007] and click prediction [Agarwal et al. 2011]. Shi et al. [2011] use EMO to generate a set of classifiers, while their work focuses on the ensemble of classifiers. Chen and Yao [2010] employ the multi-objective neural network ensemble to improve classification performances, whereas it focuses on the single-label classification problem.

3. PROBLEM DEFINITION

Let $\chi = \mathbb{R}^d$ be the d-dimensional input space and $\mathcal{L} = \{1, 2, \ldots, L\}$ 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 $h : \chi \to 2^{\mathcal{L}}$ 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 approach trains one single model by explicitly or implicitly optimizing a performance criterion. For example, ML-RBF [Zhang 2009] explicitly optimizes the Hamming Loss, while Ranking Loss is optimized in BP-MLL [Zhang and Zhou 2006] and RANK-SVM [Elisseeff and Weston 2002]. (2) The second type of approach does not explicitly optimize those performance criteria, but implicitly optimizes one single heuristic function which is not directly related to any performance criteria. For example, ECC [Read et al. 2009] and LEAD [Zhang and Zhang 2010] optimize the generalization risk for multi-label predictions by encoding label correlations, and ML-KNN [Zhang and Zhou 2007] 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 $\mathcal{M}^*$ through optimizing one single objective function.

$$\begin{aligned} &\text{minimize } O_1(\mathcal{M}) \\ &\text{s.t. } \mathcal{M} \in \Omega. \end{aligned} \tag{1}$$

Ω is the set of feasible models, $\mathcal{M}$ is a predictive model in Ω. $O_1 : \Omega \to R$ is an objective function, which can be a performance criterion (e.g., metrics in Section 5.1.2) 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 article first formulates multi-label learning as a multi-objective optimization problem (MOP) [Deb 2001], which can be defined as follows.

Definition 2. Multi-objective multi-label classification. It determines models $\mathcal{M}^*$ through simultaneously optimizing multiple objective functions.

$$\begin{aligned} &\text{minimize } \mathbf{O}(\mathcal{M}) = (O_1(\mathcal{M}), O_2(\mathcal{M}), \ldots, O_t(\mathcal{M})) \\ &\text{s.t. } \mathcal{M} \in \Omega. \end{aligned} \tag{2}$$

t is the number of objectives, and O_i represents the ith objective.

For the MOP, each objective corresponds to an optimal solution. We have to incorporate the different trade-offs 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 to Deb [2001].

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

$$\forall\, i \in \{1, \ldots, t\}\ O_i(\mathcal{M}_1) \leq O_i(\mathcal{M}_2) \wedge \exists\, i \in \{1, \ldots, t\}\ O_i(\mathcal{M}_1) < O_i(\mathcal{M}_2). \tag{3}$$

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 [Deb 2001] 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.

4. THE MOML ALGORITHM

In order to solve the multi-objective multi-label classification problem, a simple approach is to convert multiple objectives into a single objective by using certain schemes and user-specified parameters, such as the weighted sum method [Furnkranz and Flach 2003]. However, this method 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. Here we apply EMO to solve the multi-objective multi-label classification problem. Although EMO has been successfully applied in many numeric optimization problems and some data mining problems, it is seldom applied in classification. The reason lies in these two difficulties: (1) the classifier model is difficult to be effectively encoded in evolutionary algorithm; (2) it is far more difficult to trade off the self-learning of classifiers and information exchange among classifiers in EMO.

This article, proposes a 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 an effective multi-objective optimization mechanism and a novel method of generating new solutions based on a modified ml-RBF base model in the model training phase. In the model selection phase, two model selection strategies are proposed to help users flexibly select their preferred models in terms of their application scenarios.

4.1. Model Training

A good EMO algorithm needs to generate a set of solutions that uniformly distribute along the Pareto front [Veldhuizen and Lamont 2000], 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.

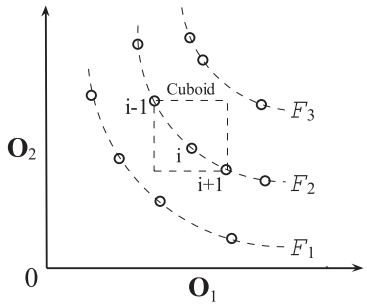


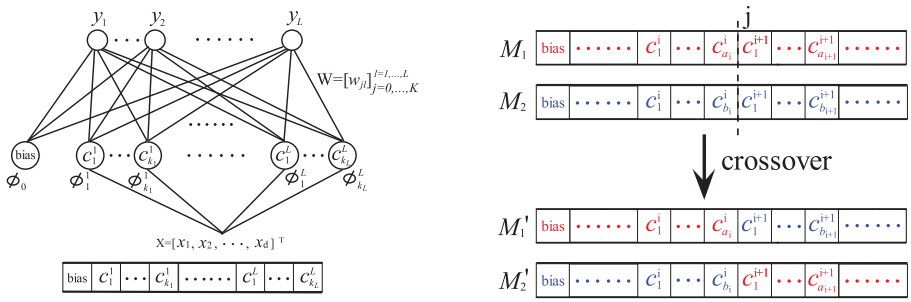
Fig. 3. Illustration of non-dominated-sort and diversity-estimate.

4.1.1. 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 apply 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 article employs the rank-based fitness assignment [Goldberg 1989] 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 $\mathcal{M}_i^{rank} < \mathcal{M}_j^{rank}$. For example, the solutions in F_1 are better than those in F_2. Note that the minimization problem is considered in this article.

Diversity-estimate. Along with convergence to the Pareto front, it is also desired that an evolutionary algorithm 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 Euclidean 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 ith 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.



(a) Genetic representation. (b) Crossover operation.

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

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

$$\mathcal{M}_i^{rank} < \mathcal{M}_j^{rank} \vee \left(\mathcal{M}_i^{rank} = \mathcal{M}_j^{rank} \wedge \mathcal{M}_i^{distance} > \mathcal{M}_j^{distance}\right) \tag{4}$$

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 less 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. In the meantime, those promising solutions are also likely to be contained in the population.

4.1.2. Base Model and Evolutionary Operations. In the framework of MOML, many classification models can be used, such as decision tree [Schietgat et al. 2010], Back Propagation (BP) [Zhang and Zhou 2006], and Radical Basis Function (RBF) [Zhang 2009] 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 ml-RBF neural network in ML-RBF [Zhang 2009] 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 ml-RBF is shown in Figure 4(a). It can be briefly summarized as follows: (1) the input of a ml-RBF corresponds to a d-dimension feature vector. (2) The hidden layer of ml-RBF is composed of L sets of prototype vectors, that is, $\bigcup_{l=1}^{L} C_l$. Here, C_l consists of k_l prototype vectors $<c_1^l, c_2^l, \ldots, c_{k_l}^l>$. 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 jth 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 ml-RBF, the number of clusters k_l is settled to be a fraction α of the number of instances in U_l:

$$k_l = \alpha \times |U_l|. \tag{5}$$

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

Different from the error function in the original ml-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.

$$E = \frac{1}{2}\sum_{i=1}^{m}\sum_{l=1}^{L}\left(y_l(\mathbf{x}_i) - t_l^i\right)^2 + \gamma\sum_{j=0}^{K}\sum_{l=1}^{L} w_{jl}^2, \tag{6}$$

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 Chen and Yao [2010], the optimal output weights W can be computed in closed form by

$$W = (\Phi'\Phi + \gamma I)^{-1}\Phi' T. \tag{7}$$

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 article.

Genetic representation. According to the structure of ml-RBF, we propose a novel genetic representation that is the sequence of prototypes $<bias, c_1^1, c_1^2, \ldots c_{k_L}^L>$. 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 ml-RBF, the initialization operation of MOML generates a set of ml-RBF models with different scale coefficient α. As suggested in Zhang [2009], α is randomly selected from [0.01, 0.02] in the experiments. An advantage of this *Initialization* operation is that it generates a set of ml-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 [Baker 1985] and do crossover and mutation operation to generate new solutions with the ratio of cro_Rat and $1-cro\_Rat$, respectively. In this article, 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 [Baker 1985] to assign each solution with an appropriate selection pressure. That is, the solutions in the lower front have a higher selection probability. It guarantees that the better solution has a high yet appropriate selection probability.

Since different ml-RBFs may have different numbers of prototypes, We adapt the cut and splice crossover [Goldberg et al. 1993] which randomly chooses a crossover point for two ml-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 $<c_1^i, \ldots, c_{a_i}^i>$ and $<c_1^{i+1}, \ldots, c_{a_{i+1}}^{i+1}>$. It guarantees that each prototype vector in the newly generated ml-RBF is unabridged cluster centroid. The width of the centroid of the new ml-RBF is recalculated as in Zhang [2009]. The weights are calculated following Equation (7).

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

ALGORITHM 1: MOML-Training

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

Although the crossover and mutation operations may not generate the optimal combination of prototypes, they provide an effective method to search the prototypes space of ml-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.

4.1.3. 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 ml-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.

4.2. 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, users can select one good model according to some criteria, such as AUPRC [Vens et al. 2008]. Here we design two strategies to select a set of prediction models according to users' preferences.

The dynamic model selection strategy (called DYN) selects the top-k models on the preference objective and then makes predictions with a majority vote. Assume that instances are independently and identically distributed, these selected models will also perform well on the corresponding objective on the testing data. This dynamic model selection strategy not only can flexibly select the preferred models in terms of users' applications but also can improve the generalization performances with ensemble learning. Note that the preference objective may be or not be the optimization objects. As we know, the model training process is expensive and is not often done. The optimization objectives are usually fixed ahead of time. In different applications, users have diverse preference on performances, so they can flexibly determine their

ALGORITHM 2: MOML-Testing-DYN

Input: $\mathcal{U}$: testing data; O: preference objective; P: model set; k: # top models
Output: label set Y
procedure TESTING
  Sort P in an ascending order by O
  Select top-k models $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ 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 procedure

ALGORITHM 3: MOML-Testing-EN

Input: $\mathcal{U}$: testing data; P: model set; N: # models
Output: label set Y
procedure TESTING
  for $\mathbf{x} \in \mathcal{U}$ **do**
    $Y(\mathbf{x}) = \{l | \frac{1}{N} \sum_{i=1}^{N} \mathcal{M}_i(\mathbf{x}, l) > 0, \mathcal{M}_i \in P, l \in \mathcal{L}\}$
  end for
end procedure

preference objectives. Since the prediction process is fast, it can be done online according to different user preferences. The DYN strategy 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 $\mathbf{x}$.

The ensemble model selection strategy (called EN) combines all models and then makes predictions with a majority vote, which can be seen in Algorithm 3. On the one hand, this strategy can be used for users without obvious preferences. The EN strategy ensembles all models, so it may have no preference on a certain objective. On the other hand, it is promising to uniformly promote the performances on all criteria, since the ensemble learning is employed.

4.3. 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 ml-RBF first. Two main time-consuming components of ml-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 ml-RBF has the following complexity:

$$O(\alpha m^2/L + \alpha d m^2). \tag{8}$$

For MOML, it needs to generate N ml-RBFs and evaluate NG new ml-RBFs. The complexity of MOML in ml-RBF is $O(\alpha N m^2/L + \alpha NG d m^2)$. 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

$$O(\alpha N m^2/L + \alpha NG d m^2). \tag{9}$$

Table I. Summary of the Experimental Datasets

	Dataset						
Property	Yeast	Image	RCV1-1	RCV1-2	RCV1-3	RCV1-4	RCV1-5
# instances	2,417	2,000	3,000	3,000	3,000	3,000	3,000
# features	103	294	472	472	472	472	472
# labels	14	5	101	101	101	101	101
Domain	biology	media	text	text	text	text	text

There are k models to make predictions on the testing data (k is N for the EN strategy), so the time complexity of the testing phase is

$$O(\alpha k d n^2). \tag{10}$$

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

5. EXPERIMENTS

5.1. Experimental Setup

5.1.1. Data Collection. We tested our algorithm on seven real-world multi-label datasets from three different domain, as summarized in Table I. The first dataset is Yeast [Read et al. 2009; Zhang 2009; Zhang and Zhang 2010; Zhang and Zhou 2006] in biology, where the task is to predict the gene functional classes of the Yeast Saccharomyces cerevisiae. The second dataset Image [Read et al. 2009; Zhang 2009; Zhang and Zhang 2010; Zhang and Zhou 2006] involves the task of automatic image annotation for scene images. The other five datasets RCV1-1–RCV1-5 are the subsets of RCV1 [Yang et al. 2009; Zhang and Zhang 2010], where the task is to predict topic categories of each text document. These five datasets have different multi-label distributions, such as label cardinality and density [Zhang and Zhang 2010].

5.1.2. 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 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 in Table II, where "↓" indicates the smaller the value, the better the performance; "↑" indicates the larger the value, the better the performance. Assume we have a multi-label dataset $\mathcal{U}$ containing n multi-label instances (x_i, y_i), where $y_i \in \{0, 1\}^L (i = 1, \ldots, n)$. Let $h(x_i)$ denote a multi-label classifier's predicted label set for x_i.

5.1.3. Compared Methods. We compare our method with four baseline methods which optimize over different single objectives. In MOML, any subset of metrics listed here can be used as the optimization objectives. Here, we employ two pairs representative subsets of evaluation metrics, that is, $\{HL, RL\}$ and $\{MicF1, AP\}$. The $\{HL, RL\}$ objective subset includes two popular objectives that have already been directly optimized in previous single objective approaches [Elisseeff and Weston 2002; Zhang 2009; Zhang and Zhou 2006]. The $\{MicF1, AP\}$ objective subset includes two most useful performance criteria which are not often been directly optimized before. In addition, these two pairs of objectives are potentially conflicting. Here the DYN model selection strategy is employed. 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 the DYN model selection based on the top k models in terms of HL and RL, respectively. The

Table II. Summary of Metrics in Multi-Label Classification

Criteria	Description	Formula
Hamming Loss (*HL*)↓ [Tsoumakas et al. 2010]	Evaluate the average error rate over all the binary labels.	$HammingLoss(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{L}\|h(x_i)\bigoplus y_i\|_1$ $\bigoplus$ stands for the symmetric difference of two sets (XOR operation), and $\|.\|_1$ denotes the l_1-norm.
Micro F1 (*MicF1*)↑ [Ghamrawi and McCallum 2005]	Evaluate a classifier's label set prediction performance, which considers both micro-average of *precision* and *recall* on all binary labels with equal importance.	$Micro-F1(h,\mathcal{U}) = \frac{2\times\sum_{i=1}^{n}\|h(x_i)\bigcap y_i\|_1}{\sum_{i=1}^{n}\|h(x_i)\|_1+\sum_{i=1}^{n}\|y_i\|_1}$
Macro F1(*MacF1*)↑ [Ghamrawi and McCallum 2005]	Evaluate a classifier's label set prediction performance, which considers both macro-average of *precision* and *recall* with equal importance.	$Macro-F1(h,\mathcal{U}) = \frac{1}{L}\sum_{k=1}^{L}\frac{2\times\sum_{i=1}^{n}h^k(x_i)y_i^k}{\sum_{i=1}^{n}h^k(x_i)+\sum_{i=1}^{n}y_i^k}$ y_i^k is the k-th entry of y_i and $h^k(x_i)$ is the k-th entry of $h(x_i)$.
Subset 0/1 Loss (*SL*)↓ [Ghamrawi and McCallum 2005]	Evaluate the average percentage when a classifier's label set prediction is exactly correct.	$SubsetLoss(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n}I(h(x_i)\neq y_i)$ $I(\cdot)$ denotes the indicator function, i.e. $I(\pi)=1$ iff π holds, otherwise $I(\pi)=0$.
Accuracy (*Acc*)↑ [Tsoumakas et al. 2010]	Evaluate the average fraction of correct labels across all examples.	$Accuracy(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\|h(x_i)\bigcap y_i\|}{\|h(x_i)\bigcup y_i\|}$
Ranking Loss(*RL*)↓ [Tsoumakas et al. 2010]	Evaluate the average fraction of label pairs that are disordered for an example.	$RankingLoss(h,\mathcal{U}) = \frac{1}{L}\sum_{i=1}^{L}\frac{1}{\|Y_i\|\|\overline{Y_i}\|}\|\mathrm{R}_i\|$ $R_i = \{(y_1,y_2)\|h(x_i,y_1)\leq h(x,y_2),(y_1,y_2)\in Y_i\times\overline{Y_i}\}$. $\overline{Y_i}$ denotes the complementary set of Y_i in Y.
One Error (*OE*)↓ [Elisseeff and Weston 2002]	Evaluate how many times the top-ranked label by a classifier is not in the true label set of an example.	$OE(h,\mathcal{U}) = \frac{1}{L}\sum_{i=1}^{L}\|argmax_{y\in Y}h(x_i,y)\notin Y_i\|$ $\|\pi\|$ equals 1 if π holds and 0 otherwise.
Coverage (*Cov*)↓ [Elisseeff and Weston 2002]	Evaluate how many steps are needed, on average, to move down the label list in order to cover all the true labels of an example.	$coverage(h,\mathcal{U}) = \frac{1}{L}\sum_{i=1}^{L}max_{y\in Y_i}rank^h(x_i,y)-1$
Average Precision (*AP*)↑ [Elisseeff and Weston 2002]	Evaluate the average fraction of true labels ranked above a particular label.	$avgprec(h,\mathcal{U}) = \frac{1}{L}\sum_{i=1}^{L}\frac{1}{\|Y_i\|}\times\frac{\|P_i\|}{rank^h(x_i,y)}$ $P_i = \{y'\|rank^h(x_i,y')\leq rank^h(x_i,y),y'\in Y_i\}$

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 optimization objectives.

—MOML$_{\{MicF1,AP\}}$. The proposed MOML approach with the second objective subset $\{MicF1, AP\}$. Similarly, we report two versions of the DYN 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 [Zhang 2009]. Based on ml-RBF neural network, the method explicitly optimizes the HL criterion.

—BP-MLL [Zhang and Zhou 2006]. This method is based on BP neural network, which explicitly optimizes the RL criterion.

—ML-KNN [Zhang and Zhou 2007]. The KNN based lazy multi-label learning method optimizes a posterior principle which is not directly related to any single performance criterion.

—ECC [Read et al. 2009]. 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 [Zhang 2009]. For BP-MLL, as indicated in the literature [Zhang and Zhou 2006], 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 [Zhang and Zhou 2007]. For ECC, the ensemble size is set to 10 and sampling ratio is set to 67% [Read et al. 2009].

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

Due to the limited space, we only show the results of the average values of nine metrics on Yeast, Image and RCV1-1 in Tables III–V, where '*' indicates the best result on each criterion and '_' indicates the performance of MOML on its preference objective. The other four datasets on RCV1 have similar results with RCV1-1. From these tables, we can observe that the four versions of the MOML method rank among the first four on most metrics and they always have the best average ranks on each dataset. Furthermore, Table VI summarizes the mean and standard deviation of the rank values for each method over nine metrics on all seven datasets. To statistically measure the significance of performance improvement, pairwise t-test at 5% significance level are conducted between MOML and other compared algorithms for each dataset. Here the MOML's performances are the average performances of four versions of MOML. Table VII illustrates the number of win/tie/loss of MOML against other compared algorithms on all seven datasets. The results indicate that, although MOML only optimizes two objectives, the performances of MOML are significantly better than the baselines on most metrics. Moreover, Table VI 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

Table III. Results of Different Algorithms on the Yeast Dataset

Criteria	Methods							
	MOML$_{\{HL,RL\}}$	MOML$_{\{HL,RL\}}$	MOML$_{\{MicF1,AP\}}$	MOML$_{\{MicF1,AP\}}$	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)

Note: The results are reported as "average performance + (rank)", where "↓" indicates that the smaller the value, the better the performance; "↑" indicates the larger the better.

Table IV. Results of Different Algorithms on the Image Dataset

Criteria	Methods							
	MOML$_{\{HL,RL\}}$	MOML$_{\{HL,RL\}}$	MOML$_{\{MicF1,AP\}}$	MOML$_{\{MicF1,AP\}}$	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)

Note: The results are reported as "average performance + (rank)", where "↓" indicates that the smaller the value, the better the performance; "↑" indicates the larger the better.

Table V. Results of Different Algorithms on the RCV1-1 Dataset

Criteria	Methods							
	MOML$_{\{HL,RL\}}$	MOML$_{\{HL,RL\}}$	MOML$_{\{MicF1,AP\}}$	MOML$_{\{MicF1,AP\}}$	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)

Note: The results are reported as "average performance + (rank)", where "↓" indicates that the smaller the value, the better the performance; "↑" indicates the larger the better.

tradeoff among multiple objectives and avoid the local optimal to improve the overall performance almost on all metrics.

Table VIII 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.

Table VI. The Average Ranks (mean±std) for Each Method over 7 Datasets

	Methods							
Criteria	MOML$_{\{HL,RL\}}$	MOML$_{\{HL,RL\}}$	MOML$_{\{MicF1,AP\}}$	MOML$_{\{MicF1,AP\}}$	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 VII. The Win/Tie/Loss Results for MOML against the Compared Algorithms Based on Pairwise *t*-test at 5% Significance Level on Seven Datasets in Terms of Different Evaluation Metrics

MOML against	Criteria									
compared methods	*HL*	*RL*	*SL*	*OE*	*Cov*	*MicF1*	*AP*	*Acc*	*MacF1*	In Total
ML-RBF	7/0/0	7/0/0	7/0/0	6/1/0	7/0/0	7/0/0	7/0/0	7/0/0	7/0/0	62/1/0
BP-MLL	7/0/0	7/0/0	7/0/0	7/0/0	7/0/0	7/0/0	7/0/0	7/0/0	7/0/0	63/0/0
ML-KNN	7/0/0	7/0/0	7/0/0	7/0/0	7/0/0	7/0/0	7/0/0	6/1/0	7/0/0	62/1/0
ECC	7/0/0	7/0/0	5/0/2	6/0/1	7/0/0	5/0/2	7/0/0	5/0/2	6/0/1	55/0/8

Table VIII. Average Running Time (Second)

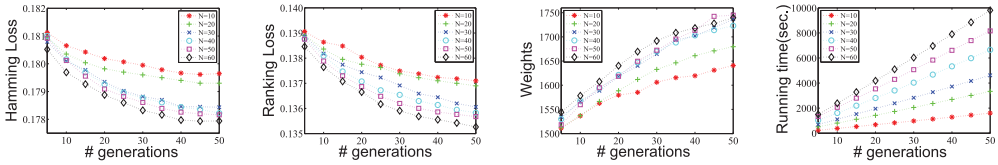
	Methods									
Data	MOML$_{\{HL,RL\}}$		ML-RBF		BP-MLL		ML-KNN		ECC	
Set	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Yeast	757	3.9	15.6	0.6	12,100	17.5	2.5	1.2	39.7	5.9
Image	343	1.6	2.8	0.2	12,500	5.3	9.1	1.2	39.3	4.2
RCV1-1	34,400	89	816	10.5	62,300	164	149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

5.3. Parameter Settings

There are two genetic operation related parameters governing the MOML, that is, the population size N and the running generations G. Figure 5 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 5 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. In the meantime, the weights of ml-RBF and running time also increase. Figure 5 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 5(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 ml-RBF (see Equation (6)), the weights will increase sharply, which means these models are overfitting. Figure 5(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 (9).

In addition, the number of top models k also affects the performance of MOML. In order to observe its effect on performance, we do experiments on Image data with MOML$_{\{HL,RL\}}$ method. Figure 6 show the performance of MOML$_{\{HL,RL\}}$ on different k. Note that MOML$_{\{HL,RL\}}$ has the same parameter setting with that in Section 5.2 and



(a) Hamming loss ↓. (b) Ranking loss ↓. (c) Weights. (d) Running time.

Fig. 5. The impact of the running generations and population size on performance. "↓" indicates the smaller the better; "↑" indicates the larger the better.

(a) Dynamic model selection on HL. (b) Dynamic model selection on RL.

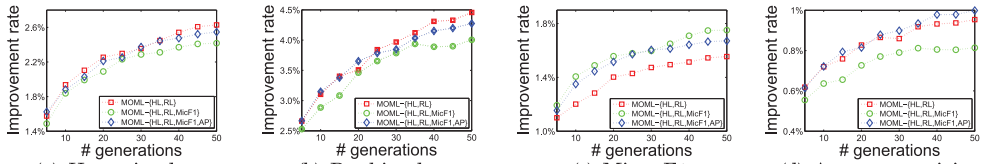
Fig. 6. The effect of the parameter k on the performance of MOML$_{\{HL,RL\}}$.

k is the ratio of selected models. Figure 6(a) clearly illustrates that MOML$_{\{HL,RL\}}$ has the best performance on criteria Hamming Loss and Ranking Loss when k are 0.3 and 0.4. The same phenomenon is also shown in Figure 6(b). When k is small, there are only few classifiers to make a prediction. When k becomes larger, more classifiers will be ensembled, which usually leads to better performances. However, ensembling more classifiers makes the prediction have less preference to the optimization objective, so the performance of MOML$_{\{HL,RL\}}$ on criteria Hamming Loss and Ranking Loss will degrade for large k. This is the reason why the performances of MOML$_{\{HL,RL\}}$ increase first and then decrease as k increases. The experiments also imply that MOML will achieve better performance when k is 0.3 and 0.4. In this setting, the model selection process not only has the benefit of ensembling classifiers but also keeps the preference on optimization objectives.

5.4. Influence of the Number of Objectives

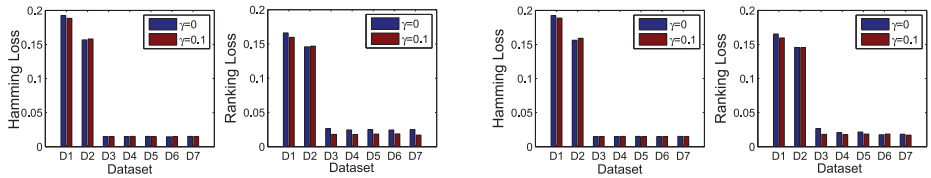
Our previous experiments only show the cases with a pair of objectives. However, more objective functions also can be included in MOML. In order to study the performances of MOML with different numbers of objectives, here we consider four objective functions (i.e., HL, RL, MicF1, and AP) and three versions of MOML which optimize the first 2, 3, and 4 objectives, respectively. The corresponding algorithms are called MOML-$\{HL, RL\}$, MOML-$\{HL, RL, MicF1\}$, and MOML-$\{HL, RL, MicF1, AP\}$. We also consider a special case of MOML, called MOML-$\{Ens\}$, where the running generation of MOML is 0. That is, MOML-$\{Ens\}$ does not do any genetic operation and multi-objective optimization. So it is just the ensemble of multiple ml-RBFs, and its performances are constant along the evolutionary process. Considering MOML-$\{Ens\}$ as baseline, we observe the improvement rate of performances of other algorithms against MOML-$\{Ens\}$. Ten-fold cross-validation are reported on the Yeast data.

The results are shown in Figure 7. It is obvious that the three versions of MOML achieve the consistent and steady performance promotion against MOML-$\{Ens\}$ on all four objectives. It illustrates that the genetic operation and multi-objective optimization in MOML is really helpful to train better models. We can also find that MOML has better performances on the optimization objectives than on non-optimization objectives.



(a) Hamming loss. (b) Ranking loss. (c) Micro F1. (d) Average precision.

Fig. 7. Influence of the number of objectives on MOML. It shows the improvement rate of performances of MOML against the ensemble of multiple base models.



(a) Performance on MOML$_{\{\underline{HL},RL\}}$. (b) Performance on MOML$_{\{HL,\underline{RL}\}}$.

Fig. 8. The effect of the regularization term on the performance of MOML$_{\{HL,RL\}}$. When γ is 0, there is no regularization term. The datasets D1–D7 denote the Yeast, Image, and RCV1(1-5), respectively.

However, when more objectives are included in MOML, the performance of MOML on the optimization objectives will degrade. For example, compared to MOML-$\{HL, RL\}$, MOML-$\{HL, RL, MicF1, AP\}$ can achieve better performances on *MicF1* and *AP* by including them in its objective set, while its performances are slightly worse than MOML-$\{HL, RL\}$ on *HL* and *RL*. As the number of optimization objectives increases, the space of Pareto frontier is greatly extended, which results in the exponential increase of the number of non-dominated solutions [Saxena et al. 2013]. So the domination-based selection operators in EMO do not work well in this case [Saxena et al. 2013].

5.5. Influence of Regularization Term

As we noted, a regularization term is added in the error function of MOML (see Equation (6)), which is different from the error function in the original RRF [Zhang 2009]. This section will validate the effect of the regularization term on MOML. We run MOML$_{\{HL,RL\}}$ with and without the regularization term on all seven datasets. The same parameters are set with that in Section 5.2. Note that we only need to set γ with 0 (see Equation (6)) for MOML$_{\{HL,RL\}}$ without the regularization term.

The experiment results are shown in Figure 8. It is clear that MOML with the regularization term is better than that without the regularization term for almost all the datasets. Particularly, the superiority is more obvious for the text data D3–D7 (i.e., RCV1(1–5)). The experiments illustrate the importance of the regularization term for MOML. Compared to ML-RBF [Zhang 2009], MOML has more overfitting risk, since MOML trains models more times due to its evolution process. The regularization term effectively reduces the overfitting risk, which helps MOML achieve good performances.

5.6. Comparison of MOML and Weighted Sum Method

We know that a direct approach for the multi-objective multi-label classification problem is the weighted-sum method [Furnkranz and Flach 2003]. This section will compare the MOML$_{\{HL,RL\}}$ with the weight-sum method. The weight-sum method optimizes the

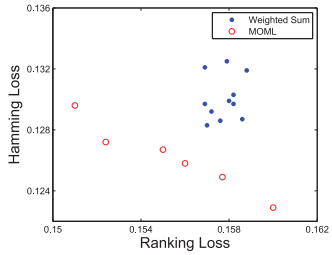


Fig. 9. The comparison of MOML and weighted-sum method.

objective:

$$\begin{aligned} & w_1 * HL + w_2 * RL \\ & s.t. : w_1 + w_2 = 1. \end{aligned} \tag{11}$$

The optimization technique employs the same evolutionary algorithm framework with MOML$_{\{HL,RL\}}$. The experiments are done on Image data and the two methods have the same parameters setting with that in Section 5.2. The weighted-sum method varies w_1 from 0 to 1 with the interval 0.1. For each weight setting, it obtains a best solution and employs the solution to make a prediction. So the weighted-sum method generates 11 results for varying weights. After one run, MOML$_{\{HL,RL\}}$ returns a set of solutions, from which we select the Pareto optimal solutions to make predictions. Figure 9 shows the prediction performance on Hamming Loss and Ranking Loss. It clearly shows that the solutions generated by MOML overwhelmingly dominate those of the weighted-sum method. Moreover, the MOML's solutions are widely spread, which implies that users can flexibly select prediction models in terms of their preferences. The experiment not only illustrates MOML's potential to generate better solutions compared to the weighted-sum method but also shows MOML's advantages in computational efficiency. MOML only needs to run once to generate these solutions. However, the weighted-sum method needs to be run many times through varying weights. As a consequence, MOML is much more convenient and efficient than the weighted-sum method.

5.7. Comparison of Model Selection Strategies

This section will compare different model selection strategies and clarify their characteristics and application scenarios. In experiments, we use the same model training phase with the optimization objectives HL and RL (see Algorithm 1) and different model selection strategies (see Algorithm 2–3). These compared strategies and methods are summarized as follows.

—DYN(HL) and DYN(RL). These two dynamic model selection strategies select the optimization objective as the preference objective. The DYN(HL) denotes the strategy selects the top models according to the optimization objective HL. Similarly, the DYN(RL) selects models according to RL. In fact, DYN(HL) and DYN(RL) are MOML$_{\{\underline{HL},RL\}}$ and MOML$_{\{HL,\underline{RL}\}}$ in Section 5.2, respectively.
—DYN(MicF1) and DYN(AP). The preference objective in these two dynamic model selection strategies is not the optimization objective. The DYN(MicF1) and DYN(AP) denotes the strategy selects the top models according to the preference objective $MicF1$ and AP, respectively.
—EN. This is the ensembling model selection strategy.
—ML-RBF [Zhang 2009]. This is the base model in our algorithm, which is used as baseline.

Table IX. The Average Ranks (mean±std) for Each Model Selection Strategy over 7 Datasets

Ave. ranks	Methods					
for criteria	DYN(HL)	DYN(RL)	DYN(MicF1)	DYN(AP)	EN	ML-RBF
HL	1.71±1.11*	3.43±1.71	4.00±1.52	2.43±0.78	2.71±1.38	6.00±0.00
RL	3.71±1.25	2.29±1.70*	2.29±1.11*	2.57±1.27	3.86±1.46	6.00±0.00
SL	2.71±1.11	3.00±1.41	2.00±1.73*	3.00±1.15	4.14±1.21	6.00±0.00
OE	2.29±1.97*	2.71±1.38	3.43±1.13	4.14±1.46	2.57±1.27	5.57±0.78
Cov	3.86±0.69	3.86±0.89	1.86±0.69	1.29±0.48*	4.29±1.38	5.86±0.37
MicF1	2.29±0.75	3.57±0.97	1.43±1.13*	3.29±1.25	4.43±0.97	6.00±0.00
AP	2.57±1.13	2.57±1.27	4.00±1.73	1.17±0.95*	4.00±0.57	6.00±0.00
Acc	2.43±0.97	4.29±0.48	2.57±1.13	1.29±0.48*	4.43±0.78	6.00±0.00
MacF1	2.00±1.15*	3.14±1.46	3.86±1.34	3.86±1.34	2.14±0.89	6.00±0.00

Note: The "*" indicates the best rank for each criterion.

We do experiments on all seven datasets with the same parameters in Section 5.1.3. Similarly, we test the average values of all strategies on nine metrics and summarize their rank values. We only show the average rank results in Table IX due to the space limitation. Generally, these strategies achieve their best performances on different criteria and they consistently and significantly outperform the baseline ML-RBF. In addition, we can find that MOML usually achieves best performances on its preference objective no matter whether the preference objective is or not the optimization objective. For example, DYN(HL) and DYN(RL) perform best on their preference objectives HL and RL, DYN(MicF1) and DYN(AP) have the performance improvement on the preference objectives (i.e., $MicF1$ and AP) as well as other objectives (e.g., SL, Cov, Acc). Compared to the baseline ML-RBF, the ensembling model selection strategy (i.e., EN) overall improves performances on almost all objectives, but it cannot achieve best performances on any special objectives. As we have noted, the time-consuming model training phase can be done offline, whereas the model selection phase is fast, which can be done online according to user's preference. If a user has apparent preference, he can employ the preference objective in the DYN strategy to make better classification on his preference. If users have no obvious preferences, the EN strategy can be adopted.

6. CONCLUSION

In this article, we first studied the multi-objective multi-label classification problem and proposed a novel algorithm MOML. MOML can simultaneously optimize over multiple objectives and return a set of solutions. In applications, users can flexibly select models in terms of their preferences. Experiments show that MOML not only achieves the better performances on the optimization objectives, but also improves the performances on most of the other state-of-the-art criteria for multi-label classification.

REFERENCES

Deepak Agarwal, Bee Chung Chen, Pradheep Elango, and Xuanhui Wang. 2011. Click shaping to optimize multiple objectives. In *Proceedings of KDD*. 132–140.

J. Baker. 1985. Adaptive selection methods for genetic algorithms. In *Proceedings of ICGA*. 100–111.

H. Chen and X. Yao. 2010. Multiobjective neural network ensembles based on regularized negative correlation learning. *Trans. Knowl. Data Eng.* 22, 12, 1738–1751.

K. Deb. 2001. *Multiobjective Optimization Using Evolutionary Algorithms*. Wiley, U.K.

K. Deb, A. Pratab, S. Agarwal, and T. MeyArivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolution. Comput.* 6, 2, 182–197.

K. Dembczyński, W. Cheng, and E. Hullermeier. 2010a. Bayes optimal multilabel classification via Probabilistic classifier chains. In *Proceedings of the ICML*. 279–286.

K. Dembczyński, W. Kotlowski, and E. Hullermeier. 2012. Consistent multilabel ranking through univariate loss minization. In *Proceedings of ICML*. 1319–1326.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hullermeier. 2010b. Regret analysis for performance metrics in multi-label classification: The case of Hamming and subset zero-one loss. In *Proceedings of ECML/PKDD*. 280–295.

A. Elisseeff and J. Weston. 2002. A kernel method for multilabelled classification. In *Proceedings of NIPS*. 681–687.

A. A. Freitas. 2006. A critical review of multi-objective optimization in data mining: A position paper. *SIGKDD Explor.* 6, 2, 77–86.

J. Furnkranz and P. A. Flach. 2003. An analysis of rule evaluation metrics. In *Proceedings of ICML*. 202–209.

W. Gao and Z. H. Zhou. 2011. On the consistency of multi-label learning. In *Proceedings of COLT*. 341–358.

N. Ghamrawi and A. McCallum. 2005. Collective multilabel classification. In *Proceedings of CIKM*. 195–200.

Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative methods for multi-labeled classification. In *Proceedings of PAKDD*. 22–30.

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

D. E. Goldberg. 1989. *Genetic Algorithms in Search Optimization and Machine Learning*. Wesley, Boston, MA.

J. Handle and J. Knowles. 2007. An evolutionary approach to multiobjective clustering. *Trans. Evolution. Comput.* 11, 1, 56–76.

Shuiwang Ji, Lei Tang, Shipeng Yu, and Jieping Ye. 2008. Extracting shared subspace for multi-label classification. In *Proceedings of KDD*. 381–389.

W. Kotlowski, K. Dembczynski, and E. Hullermeier. 2011. Bipartite ranking through minimization of univariate loss. In *Proceedings of ICML*. 1113–1120.

J. Petterson and T. Caetano. 2010. Reverse multi-label learning. In *Proceedings of NIPS*. 1912–1920.

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

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

D. Saxena, J. A. Duro, A. Tiwari, K. Deb, and Q. Zhang. 2013. Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Trans. Evolution. Comput.* 17, 1, 77–99.

Robert E. Schapire and Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learn.* 2, 39, 135–168.

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

Chuan Shi, Xiangnan Kong, Philip S. Yu, and Bai Wang. 2011. Multi-label ensemble learning. In *Proceedings of ECML/PKDD*. 223–239.

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

G. Tsoumakas, I. Katakis, and I. P. Vlahavas. 2008. Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of ECML/PKDD Workshop*.

G. Tsoumakas and I. P. Vlahavas. 2007. Random K-labelsets: An ensemble method for multilabel classification. In *Proceedings of ECML*. 406–417.

D. A. V. Veldhuizen and G. B. Lamont. 2000. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolution. Comput.* 18, 2, 125–147.

Celine Vens, Jan Struyf, Leander Schietgat, Saso Dzeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine Learn.* 3, 73, 185–214.

H. Xu and J. Xu. 2010. Designing a multi-label kernel machine with two-objective optimization. In *Proceedings of AICI*. 282–291.

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

Z. J. Zha, T. Mei, J. D. Wang, Z. F. Wang, and X. S. Hua. 2009. Graph-based semi-supervised learning with multiple labels. *J. Visual Commun. Image Represent.* 20, 2, 97–103.

Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, and Zengfu Wang. 2008. Joint multi-label multi-instance learning for image classification. In *Proceedings of CVPR*. 1–8.

M.-L. Zhang. 2009. ML-RBF: RBF Neural networks for multi-label learning. *Neural Process Lett.* 29, 2, 61–74.

M.-L. Zhang and K. Zhang. 2010. Multi-label learning by exploiting label dependency. In *Proceedings of KDD*. 999–1007.

M.-L. Zhang and Z.-H. Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *Trans. Knowl. Data Eng.* 18, 10, 1338–1351.

M.-L. Zhang and Z.-H. Zhou. 2007. ML-kNN: A Lazy learning approach to multi-label learning. *Pattern Recognition* 40, 7, 2038–2048.

Received January 2013; revised April 2013, July 2013; accepted July 2013