IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 11, 2017, accepted September 26, 2017, date of publication October 2, 2017, date of current version November 7, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2758790

RBF-MLMR: A Multi-Label Metamorphic Relation Prediction Approach Using RBF Neural Network

PENGCHENG ZHANG[1], XUEWU ZHOU[1], PATRIZIO PELLICCIONE[2,3], AND HARETON LEUNG[4]

[1]College of Computer and Information, Hohai University, Nanjing 211100, China
[2]Department of Computer Science and Engineering, Chalmers University of Technology, 41296 Göteborg, Sweden
[3]Department of Computer Science and Engineering, University of Gothenburg, 41296 Göteborg, Sweden
[4]Department of Computing, Hong Kong Polytechnic University, HongKong

Corresponding author: Pengcheng Zhang (pchzhang@hhu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572171, Grant 61702159, and Grant 61202136, and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20170893.

ABSTRACT Metamorphic testing has been successfully used in many different fields to solve the test oracle problem. However, how to find a set of appropriate metamorphic relations for metamorphic testing remains a complicated and tedious task. Recently some machine learning approaches have been proposed to predict metamorphic relations. These approaches predicting single label metamorphic relation can alleviate this problem to some extent. However, many applications involve multi-group metamorphic relations, and these approaches are clearly inefficient. To address this problem, in this paper we propose a Multi-Label Metamorphic Relations prediction approach based on an improved radial basis function (RBF) neural network named RBF-MLMR. First, RBF-MLMR uses state-of-the-art soot analysis tool to generate control flow graph and corresponds labels from the source codes of programs. Second, the extracted nodes and the path properties constitute multi-label data sets for the control flow graph. Finally, a multi-label RBF neural network prediction model is established to predict whether the program satisfies multiple metamorphic relations. In order to improve the prediction results, affinity propagation and k-means clustering algorithms are used to optimize the RBF neural network structure of RBF-MLMR. A set of dedicated experiments based on public programs is conducted to validate RBF-MLMR. The experimental results show that RBF-MLMR can achieve accuracy of around 80% for predicting two and three metamorphic relations.

INDEX TERMS Multi-label, metamorphic testing, metamorphic relation, label count vector, RBF neural network.

I. INTRODUCTION

Software testing, a major approach to guarantee software quality, aims at detecting software failures *as early as possible*. However, in practical situations, many programs lack testing oracles [5], [21], and this limits the effective application of software testing. In order to solve the oracle problem, T.Y. Chen et al. have proposed a novel technique called *metamorphic testing* [4], [5], [18], [32]. This technique utilizes *original test cases* (using the random value method and special value method) that are not found in the test process, to construct the new test case (called *follow-up test case*), so as to carry out a more thorough testing of the program. *Metamorphic testing* can detect whether the program has defects by checking whether the output of the *original test case* and the *follow-up test case* meet the expected property. This property is known as *metamorphic relation*.

Currently most state-of-the-art approaches [6], [7], [11] manually construct *metamorphic relations*. When the input satisfies a certain relation, *metamorphic testing* checks whether the output meets the corresponding *metamorphic relations*. If *metamorphic relation* is not satisfied, the program must contain some defects. This technique does not need to verify every testing output, which can effectively alleviate the problem of lacking software oracle. Previous studies have shown that *metamorphic testing* is appropriate for testing program lacking of effective approaches for creating testing oracle. However, the manual process for constructing *metamorphic relations* may miss some important relations that can reveal errors. When the programs are tested with *metamorphic testing*, more than one *metamorphic relation* often exist, and some important relations are easily missed during the construction process. This motivates previous researchers to do further studies on automatic constructing *metamorphic relations*.

In order to solve this limitation, the work in [13] and [14] try to use machine learning techniques for automating prediction functions, which may satisfy some *metamorphic relation*. However, the established machine learning model can

only predict one *metamorphic relation*, which cannot meet the needs of many real programs. It may be very tedious and inefficient for those special programs with many *metamorphic relations*. This problem inspires us to explore novel prediction approach for multi-labels *metamorphic relations*. Artificial Neural Networks (ANNs) are very popular in the prediction of *metamorphic relations*. Artificial Neural Networks are algorithmic mathematical models that mimic the behavioral characteristics of animal neural networks and carry out distributed parallel information processing. This network relies on the complexity of the system, through the adjustment of the relationship between a large number of nodes within the interconnection, so as to achieve the purpose of dealing with information, and self-learning and adaptive ability [1], [2]. In Literature, the research of multi-labels learning approaches has been mainly divided into two categories [9], [28]: 1) *The problem translation*, the classification problems of multi-labels learning are decomposed into a set of independent classification problems, and then the result of multi-labels is summarized by the result of each classification. 2) *Algorithmic adaptability*, the multi-labels sample sets are learned and the predicted model is then established to predict the multi-labels. The second category is a better simulation of a multi-label learning classification problem, and it also meets the cognitive condition.

Consequently, we mainly use the second category of approaches to study the multi-label samples directly in this paper. Among many multi-labels prediction algorithms [9], [10], [22], [27], [28], existing results suggest that ML-RBF (Multi-Labels Radial Basis Function) is one of the best learning algorithms for multi-labels learning [28]. RBF is a well-behaved feed-forward neural network with better approximation capability and global optimal characteristics, and it is widely used in various engineering domains [3], [15], [20]. Thus, this paper proposes a multi-label *metamorphic relation* prediction approach based on RBF neural network. However, in order to improve the efficiency of the approach, the AP algorithm is used to optimize the RBF network structure. The proposed approach can predict multiple metamorphic relations, which can effectively improve construction efficiency of metamorphic relation, as well as establish the foundation for automatic metamorphic testing tools. Specifically, as the complexity of the function program increases, the function contains more and more metamorphic relations. A single label has been unable to meet the reality of the program, so multi-label becomes more and more popular. The multi-label prediction model is established by constructing the multi-label data set and RBF neural networks, which can predict multiple metamorphic relations at one time and can improve the structural efficiency of the metamorphic relations effectively. The contributions of this paper are summarized as follows:

- We propose a novel metamorphic relation prediction approach called RBF-MLMR. To solve the limitations of existing approaches, the RBF-MLMR approach, improved by AP and k-means clustering algorithms is used to automatically construct multiple metamorphic relations that can be satisfied by the programs.
- The RBF-MLMR approach is validated by a set of dedicated experiments. The experiments are based on more than 100 public programs. The results show that RBF-MLMR is effective and efficient compared to other approaches.

The rest of this paper is organized as follows. Section II reviews related works and discusses their limitations. Section III describes the metamorphic testing technique and the basic idea of ML-RBF. Section IV presents and details the RBF-MLMR approach. The experimental validation of RBF-MLMR is performed in Section V. Finally, Section VI offers some conclusions and suggestions for future work.

II. RELATED WORK

A. METAMORPHIC TESTING AND METAMORPHIC RELATIONS

Metamorphic testing has been widely used in different fields such as health care simulations [19], Monte Carlo modeling [7], bioinformatics [6], computer graphics [11], and testing programs with partial differential equations [4]. Some studies have been conducted on how to select ''good'' metamorphic relations. Xie *et al.* [25] compare various metamorphic relations identified for programs of shortest path and critical path, and attempt to distinguish *metamorphic relations* that are effective in detecting software failures. Their study shows that theoretically understanding the application domain is not sufficient for identifying good metamorphic relations. Mayer and Guderlei [17] examine some metamorphic relations for several Java programs of determinant computation. They observe that metamorphic relations that have rich semantic properties normally have high failure-detection capability. All the studies on selecting ''good'' *metamorphic relations* only provide some guidelines that help software testers identify *metamorphic relations* that intuitively have high failure-detection capability. However, all these approaches do not support a systematic methodology on constructing *metamorphic relations*, and testers still need to identify *metamorphic relations* in an ad hoc way. Wang [24] propose a *metamorphic relation* construction method based on the compositional function. The method lacks automatic generation of the composite *metamorphic relations* and tools. Kanewala and Bieman [13] and Kanewala *et al.* [14] present a novel approach for automatically predicting *metamorphic relations* using machine learning techniques and predicting *metamorphic relations* for testing scientific software machine learning approaches using graph kernels. However, these two approaches were designed for single label *metamorphic relation* prediction. They are not suitable and efficient for programs with multiple *metamorphic relations*.

B. MULTI-LABEL LEARNING ALGORITHMS

The key idea of learning multiple *metamorphic relations* is based on multi-label learning algorithms. This

subsection reviews existing multi-label algorithms. Many multi-label learning algorithms, such as BP-MLL, Rank-SVM, Boostexter, Adtboost.MH, ML-KNN and ML-RBF, play different roles on multi-labels learning. BP-MLL [28] adjusts the weight of network constantly to obtain the minimized objective function using BP neural network. Rank-SVM [9] conducts the multi-labels learning with SVM. Boostexter [22] is an expansion of ADABoost that is an ensemble learning algorithm. Adtboost.MH [10] is derived from ADTboost and AdaBoost.HM. ML-KNN [29] lazy learning algorithm transforms multiple-label problems into multiple independent categories of classification problems to solve the multi-label classification problems. With the advantage of easy structure, and global optimum and fast training speed, ML-RBF [27] is excellently adaptive to the multi-labels learning classification. These algorithms are popular, especially ML-RBF has been extensively applied in the multi-labels learning classification.

Many multi-label learning algorithms have been proposed by adapting traditional supervised learning techniques to learn from multi-label training instances, such as multi-label decision trees, multi-label Bayesian approaches, multi-label kernel methods, multi-lable RBF, etc. The literature [27] has already demonstrated that ML-RBF is the most efficient algorithm for multi-label learning, and ML-RBF achieves rather competitive performance to other state-of-the-art multi-label learning algorithms.

III. PRELIMINARIES

A. METAMORPHIC TESTING AND METAMORPHIC RELATIONS

The effectiveness of the *metamorphic testing* technique has been validated by research applications in different fields [12], [26], [31], [33]. As we all know, *metamorphic testing* uses unsuccessful test cases to generate follow-up test cases [5]. It is usually carried out according to the following steps [16]:

- An appropriate set of *metamorphic* relations from the specification of the software under testing is identified;
- An initial test case using some traditional test case selection methods such as random values or special values is generated;
- A follow-up test case from the initial test case based on identified *metamorphic relations* is created;
- The initial and follow up test cases are both executed, and their results are compared against the *metamorphic relations*. The violation of the *metamorphic relations* shows that the program has at least one fault.

Metamorphic testing has three outstanding characteristics [8]:

- In order to check the implementation of the program, we need to construct a *metamorphic relation*;
- In order to judge the correctness of the program function from many aspects, the testing is usually based on multiple *metamorphic relations*;
- In order to generate the initial test case, the *metamorphic testing* needs to be combined with other test case generation approaches.

It is quite clear that *metamorphic relation* is the core part of the *metamorphic testing*. It not only contributes to follow-up test cases generation, but also provides a mechanism for the final validation. However, *metamorphic relations* in most previous studies are identified manually by testers in an ad-hoc way.

TABLE 1. MRs and change made to the input.

Metamorphic Relation	Change made to the input
Additive	Add or subtract a count
Permutative	Randomly permute the element
Inclusive	Add a new element
Exclusive	Remove an element
Invertive	Take the inverse of each element
Multiplicative	Multiply by a constant

```
public static double average(int a[]){
        double sum = 0.0;
        double avg = 0.0;
        for(int i = 0;i<a.length;i++){
            sum+=a[i];
        }
        avg = sum/a.length;
        return avg;
}
```

FIGURE 1. Function for calculating the average number of an integer array.

Murphy *et al.* [19] have identified a set of *metamorphic relations* shown in Table 1. These relations are commonly used in mathematical functions, where a *metamorphic relation* represents a function f that may satisfy multiple *metamorphic relations*, and *change made to the input* represents different categories of input modifications. For example, the function in Figure 1 calculates the average of an integer array. Each element in the array is coupled with a number k, and the results are corresponding to an increase of k. This is the *additive* metamorphic relation in Table 1. Each element in the array is multiplied by a non-zero number a, and the result of this output is a times the original. This is the *multiplicative* metamorphic relation shown in Table 1. Randomly permuting the order of the elements in the array, and the corresponding result does not change. This is the *permutative* metamorphic relation in Table 1. From this program, we can see that there are at least three metamorphic relations, which motivate us to do multiple *metamorphic relations* prediction.

B. ML-RBF

Radial basis function is one of the most studied neural network models. Figure 4 illustrates the basic architecture of a typical ML (Multiple layers)-RBF neural network [27]. ML-RBF is derived from the traditional Radial Basis Function (RBF). Briefly, the first layer of ML-RBF neural network

is formed by conducting clustering analysis on instances of each possible class, where the centroid of each clustered group is regarded as the prototype vector of a basis function. After that, the weights of the second layer of the ML-RBF neural network are learned by minimizing a sum-of-squares error function. Specifically, information encoded in the prototype vectors corresponding to all classes are fully exploited to optimize the weights corresponding to each specific class. As shown in Figure 4, in the input layer the input to an ML-RBF neural network corresponds to a d-dimensional feature vector, denoted as $x = (x_1, x_2, ..., x_n)$. L represents the general label category, and $l(l \in L)$ represents a class label. The hidden layer of ML-RBF is composed of L sets of prototype vectors, i.e., $U_{l=1}^{L} C_l$. Here, C_l consists of k_l prototype vectors $\{c_1^l, c_2^l, ..., c_{kl}^l\}$, and c_j^l is used as the center of radial basis function $\phi_j^l(.)$. y represents the output, and W represents the weight of the hidden layer to the output layer.

In addition, the number in each sample is multiplied by the corresponding proportion, and then the results are rounded to obtain the k value. This process is so-called selecting k value with proportion of samples. Each of the output neurons of multi-label neural network is a possible output label. The output of the multi-label RBF neural network is based on formula (1).

$$y_l(x_i) = \sum_{j=0}^{L} w_{jl}\phi_j(x_i) \tag{1}$$

where the weight of $W = [W_{jl}]_{(K+1)\times l}$ dimension between the hidden layer and the output layer is $K + 1$, that is the number of hidden layer nodes. Here, $K = \sum_{i=1}^{L} K_i$ denotes the total number of prototype vectors retained in the hidden layer. In addition, the biases are shown as weights w_{0l} from an extra basis function $\phi_0(.)$ and its output is fixed at 1. Radial basis functions of Gauss function are shown in formula (2).

$$\phi_j(x_i) = exp\left(-\frac{dist(x_i, c_j)^2}{2\sigma_j^2}\right) \tag{2}$$

Here $dist(x_i, c_j)$ represents the Euclidean distance between x_i and c_j. Base function variance smoothing parameter σ is obtained by calculating the average distance between each kind of basis function center, as shown in formula (3).

$$\sigma = \mu\left(\frac{\sum_{p=1}^{k} -1 \sum_{q=p+1}^{k} dist(c_p, c_q)}{\frac{k(k-1)}{2}}\right) \tag{3}$$

where μ is the parameter of scaling factor. Weight w is solved by minimizing the square sum of the error function. The minimum square sum of error function is shown in formula (4).

$$E = \frac{1}{2}\sum_{i=1}^{m}\sum_{l=1}^{L}(y_l(x_i) - t_l^i)^2 \tag{4}$$

where t_l^i is the desired output of x_i on the lth class.

IV. THE RBF-MLMR APPROACH

In this section, this paper first gives an overview of RBF-MLMR in Section IV-A, which describes the steps for building a RBF-MLMR prediction model. Section IV-B shows the construct Function Control Flow Graph (CFG) from source codes. In Section IV-C, the multi-label data set is constructed from the control flow graph. Finally, Section IV-D trains and builds the RBF-MLMR prediction model.

A. OVERVIEW OF RBF-MLMR

Based on the single MR prediction approach presented in [13] and [14] , this paper proposes an approach named RBF-MLMR. This approach models this problem as a multi-label classification problem. RBF neural network uses radial basis function (RBF) as the node activation function of the hidden layer to turn the lower dimensional linear inseparable data of input layer into high-dimensional data of output layer, which makes it linearly separable in the high-dimensional space. Figure 2 gives an overview of this approach. It follows the following steps:

- This approach uses the soot analysis tool to get the control flow graph of the function in the training set, where each function is associated with multiple labels simultaneously.
- From the control flow graph of the function (CFGs), we extract a set of features and corresponding multiple label information to construct a multi-label data set.
- RBF-MLMR model is established from training these data for predictions.

The training phase of RBF-MLMR model also contains three steps.

- The AP (Affinity Propagation) algorithm is used to obtain the k value.
- The k-means algorithm is used to calculate the center of the basis function, and the label counting vector is calculated.
- The label counting vector is overlapped with the center of the base function to obtain the RBF neural network basis function center, and the variance and the weights between the hidden layer and the output layer are set up to predict the model.

During the prediction phase, the characteristic data of the function to be measured are given as input to the established RBF-MLMR model, and then we predict the multiple relations that the measured program may satisfy.

B. CONSTRUCTING FUNCTION CONTROL FLOW GRAPH(CFG)

Definition 1: Control Flow Graph. The CFG $G = (N, E)$ of a function f is a directed graph, where each $n_x \in N$ represents a statement x in f and $E \in N \times N$. An edge $e = (n_x, n_y) \in E$ if x, y are statements in f and y is executed immediately after executing x. Nodes $n_{start} \in N$ and $n_{exit} \in N$ represent the starting and exiting points of f, respectively.

The CFG are created by Soot [30], which utilizes a 3-address intermediate representation of Jimple [23] to

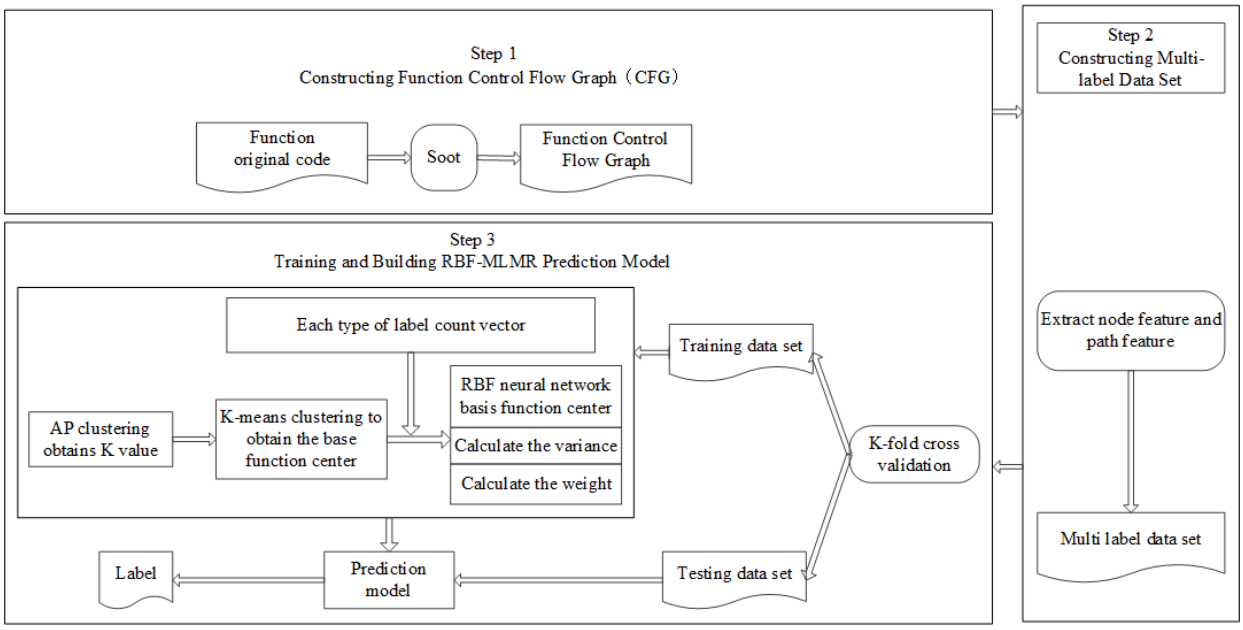


FIGURE 2. Overview of the approach.

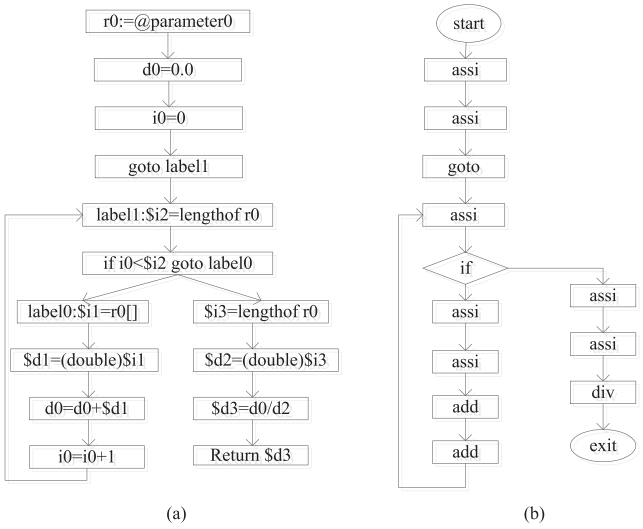


FIGURE 3. Soot analysis tool is used to generate CFG and labels for the function in Figure 1.

complete this process. Using Jimple style, each node is an atomic operation. Figure 3.a presents the Jimple typed control flow graph of Figure 1, called Soot CFG, produced by Soot. During the process of transfer, Java codes are reproduced in Jimple 3-address intermediate representation, then corresponding goto operations and labels are added. These goto operations and labels express the conditional jumps in the original Java codes. As can be seen in Figure 3.b, a labeled CFG will be acquired by adding a label to each node in Figure 3.a to represent their concrete operations.

C. CONSTRUCTING MULTI-LABEL DATA SET

From the control flow graph in Figure 3.b, we extract a set of features, which comprises of five classes, namely *node feature*, *path feature* [13], [14], *node features number*, *path features number*, and *value change path*. The classes are described in the following.

Definition 2: Node Features (NF). The form of node feature is defined as $op_n - d_{in} - d_{out}$, where the operation executed at the given node is denoted by op_n. The outward degree is denoted by d_{out} and the inward degree is denoted by d_{in}.

According to the function of the control flow graph, we calculate the inward degree and outward degree of every node in CFG. Then the number of a specific type of node in CFG is counted and treated as the node feature value. Table 2 shows a set of node features extracted from Figure 3.b. As an example, for *assi-1-1*, where *assi* is the operation performed in the node, 1 is the in-degree of *assi* node, and 1 is the out-degree of *assi* node. Feature value 6 is the number of occurrences of nodes of type *assi-1-1* in Figure 3.b.

Definition 3: Path Features(PF). The path feature means the shortest path from the starting node *Nstart* to a given node and the shortest path from this node to the end node *Nexit*, with the form being $op_1 - op_2 - ... - op_k$, where op_i denotes a node in CFG, namely the specific operation statement in the program. The value of a path feature is the number of occurrences of each shortest path in the CFG.

Table 3 shows a set of path features extracted from Figure 3.b. As an example, *start-assi-assi-goto-assi-if-assi* represents the shortest path node sequence from node *start* to node *assi*, and feature value "2" is the number of occurrences of that shortest path node sequence in Figure 3.b.

Definition 4: Node Features Number (NFN). Calculating the node type number in the function control flow graph as the value of NFN, which can further reflect the complexity of function structure. The number of nodes in a function can reflect the complexity of structure and function to some

TABLE 2. The calculation of the node feature of CFG with label in Figure 3.b.

Node Feature	Feature value
Start-0-1	1
assi-1-1	6
assi-2-1	1
if-1-2	1
div-1-1	1
add-1-1	2
goto-1-1	1
exit-1-0	1
NFN	8

TABLE 3. The calculation of the path feature of CFG with label in Figure 3.b.

Feature	Feature value
start	1
start-assi	1
start-assi-assi	1
start-assi-assi-goto	1
start-assi-assi-goto-assi	1
start-assi-assi-goto-assi-if	1
start-assi-assi-goto-assi-if-assi	2
start-assi-assi-goto-assi-if-assi-assi	2
start-assi-assi-goto-assi-if-assi-assi-div	1
start-assi-assi-goto-assi-if-assi-assi-div-exit	1
start-assi-assi-goto-assi-if-assi-assi-add	1
start-assi-assi-goto-assi-if-assi-assi-add-add	1
assi-assi-goto-assi-if-assi-assi-div-exit	1
assi-goto-assi-if-assi-assi-div-exit	1
goto-assi-if-assi-assi-div-exit	1
assi-if-assi-assi-div-exit	1
if-assi-assi-div-exit	1
assi-assi-div-exit	1
assi-div-exit	1
div-exit	1
exit	1
assi-assi-add-add-assi-if-assi-assi-div-exit	1
assi-add-add-assi-if-assi-assi-div-exit	1
add-add-assi-if-assi-assi-div-exit	1
add-assi-if-assi-assi-div-exit	1
VCP	2
PFN	25

extent, but in some cases there are many nodes with the same node type in a function, that is, performing similar operations. The number of node features further reflects the functional and structural complexity.

Definition 5: Path Features Number (PFN). Calculating the number of path characteristic numbers in the function control flow graph, which is $2n-k-1$, where n is the number of nodes in the control flow graph, and k is the number of path values that are greater than 1.

Definition 6: Value Change Path (VCP). This feature mainly reflects the complexity of data-dependent information, and its value is the number of possible changes in the number of paths.

Then, a multi-label training set is constructed as $D = \{(x_i, Y_i)|1 <= i <= m\}$, where x_i is a function feature data vector and Y_i is the label set associated with x_i.

TABLE 4. MRs and corresponding anticipated changes to the output.

Metamorphic Relation	Output change
Additive	Remains constant or Increase
Permutative	Remains constant
Inclusive	Remains constant or Increase
Compositional	Remains constant or Increase
Exclusive	Remains constant or Increase
Invertive	Decrease or Remains constant
Multiplicative	Multiply by a constant

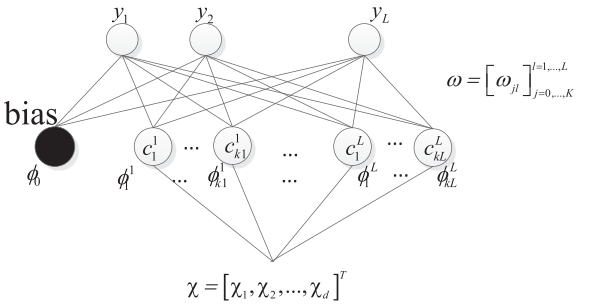


FIGURE 4. Architecture of the ML-RBF neural network.

In this step, we mainly build a prediction model for predicting whether a function f exhibits multiple metamorphic relations shown in Table 1. When input is changing, the corresponding *change in output* is diverse, and we have considered only the specific output changes given in Table 4 for each metamorphic relation, where *output change* represents a specific output change. We choose six metamorphic relations in Table 1 in the initial experiment. Table 5 describes the multi-label example data sets, where f_i represents a function, $feat_i$ represents a node and path feature. V_{ij} represents the corresponding feature value, and C_{mn} represents the corresponding class label, which denotes whether a function satisfies a specific metamorphic relation or not. "1" expresses satisfaction, "-1" means does not satisfy, and "0" indicates that specific metamorphic relation does not exist. The multi-label data set is input to the RBF-MLMR neural network to create a multi-label predictive model. Then, measured function feature data is input to the prediction model to predict whether multiple metamorphic relations are satisfied or not.

D. TRAINING AND BUILDING RBF-MLMR PREDICTION MODEL

Based on the basic RBF neural network model, we first use the AP (Affinity Propagation) cluster to obtain k value automatically, and then use the k-means clustering algorithm to obtain the base function center. Second, in order to fully consider the relationship between the multi-label, the tag count vector is added. Finally, the label count vector and the k-means clustering algorithm are linearly multiplied to obtain the basis function center of RBF neural network. In the following, this paper introduces the use of AP clustering to obtain k value firstly, followed by the label count vector, and finally the establishment of prediction model.

TABLE 5. Node feature, path feature and corresponding labels constitute multi-label experimental data set.

function	$feat_1$	$feat_2$	$feat_3$	$feat_4$	...	$feat_{n-2}$	$feat_{n-1}$	$feat_n$	$Class_1$	$Class_2$	$Class_3$
f_1	v_{11}	v_{12}	v_{13}	v_{14}	...	v_{1n-2}	v_{1n-1}	v_{1n}	C_{11}	C_{12}	C_{13}
f_2	v_{21}	v_{22}	v_{23}	v_{24}	...	v_{2n-2}	v_{2n-1}	v_{2n}	C_{21}	C_{22}	C_{23}
f_3	v_{31}	v_{32}	v_{33}	v_{34}	...	v_{3n-2}	v_{3n-1}	v_{3n}	C_{31}	C_{32}	C_{33}
...	...	...	...	...	...	...	...	...	...	...	...
f_m	v_{m1}	v_{m2}	v_{m3}	v_{m4}	...	v_{mn-2}	v_{mn-1}	v_{mn}	C_{m1}	C_{m2}	C_{m3}

1) USING AP CLUSTERING TO CALCULATE *k*

The k-means algorithm is an indirect clustering method, which performs classification according to the similarity between the samples. It is a unsupervised learning method. Given parameter k, n objects are divided into k clusters. The algorithm works well in practical application. The scalability is relatively strong, and the processing is efficient. The complexity of this algorithm is $O(Nkt)$. Using the k-means algorithm needs to give a k value, sometimes for a given data set it is difficult to determinate reasonable categories. When the k-means clustering algorithm is used, the AP clustering algorithm is also used to obtain the k value.

The AP (Affinity Propagation) clustering algorithm is different from other clustering algorithms. It is not necessary to specify the number of clustering k in advance or other parameters that describe the number of clusters. The algorithm uses an Euclidean distance to calculate the value between data to obtain a similarity matrix a, and whether the diagonal data can be used as an important reference for the center of the cluster. The iterations are continually updated by the degree of attractiveness and attribution of each point until the mass of m high-quality clusters are generated or the number of set iterations is reached as the termination condition. Then, the rest of the data will be assigned to their own clusters. In this way, this approach first uses the AP clustering algorithm to find the appropriate k, and then uses the k-means algorithm to get the center of the base function of RBF neural networks.

2) MULTI-LABELED DATA SET SPACE

If $X = Rd$ represents the sample data set space, a given multi-label data set is $D = (x_1, Y_1), (x_2, Y_2), ..., (x_n, Y_n)$ $(x_i \in X, Y_i \in Y)$, and d represents the dimension of x_i in the multi-label data set D. $Y = 1, 2, ..., Q$ is a set of labels consisting of tags in the multi-tag data set D. For sample $x \in X$ and label $Y_I \in Y$, y_x is the Q dimension class vector corresponding to x. When $l \in Y$, the value of the L-th dimension of this vector is 1, otherwise the value is -1.

In addition, let $T_l = x_i(x_i, Y_i) \in Y_I$ be the set of samples belonging to label l in the multi-label data set D and k neighboring samples are found in each class T_l as the representative of this class, which is denoted as N_x^l shown in (5).

$$N_x^l = z|z \in T_l \tag{5}$$

where z is among the k nearest neighbor of x in T_l. In order to measure the similarity of sample x_i and sample x_j, we obtain the nearest neighbor sample set N_x^l by using the order of r rank Minkoski distance. We choose $r = 2$. We use the Euclidean distance to calculate the similarity of x_i and x_j .

$$dist(x_i, x_j) = (\sum_{h=1}^{d} \left|\left|x_i^h - x_j^h\right|\right|^2)^{\frac{1}{r}} \tag{6}$$

where x_i^h and x_j^h represent the h-th dimension of samples x_i and x_j, respectively. $||.||$ is the absolute value of the real number returned.

In order to fully exploit the correlation between the various types in the multi-label data set D, the tag count vector $C_x(l)$ is defined. First, the sample data equal to 1 in each label in the multi-label data set D is calculated. Second, we use the sample k neighbors for each class of sample datasets and add each data set after k nearest neighbor into a new set of labels. Finally, the total number of samples of each type of label in the new label data set is set to 1, as shown in (7).

$$C_x(l) = \sum_{q \in Y} \sum_{z \in N_x(q)} [|y_z(l) == 1|], (l \in Y) \tag{7}$$

The base function center calculated by the k-means mean clustering algorithm is overlapped with a new basis function center. The calculation of the center function is shown in (8).

$$C_{(l)} = c_{(l)} \cdot C_x(l), (l \in Y) \tag{8}$$

Among them, $c_{(l)}$ is the value that uses the k-means clustering algorithm to obtain the multi-label data set for each type of k clustering centers, and $C_{(l)}$ is the new clustering center after label vectors are added.

In the following, we use six functions called *max, average*, *meandeviation*, *insertion_sort*, *binarysearch* and *find_euc_dist* as examples of experimental data to introduce the calculation of label count vector.

A set of six rows and six columns is obtained from the six tag information corresponding to the six functions in Table 6,

TABLE 6. Function and the corresponding label.

function	Training phase					
	permutative	**additive**	**inclusive**	**multiplicative**	**invertive**	**exclusive**
max	1	1	1	1	1	1
average	1	1	-1	1	1	-1
meandeviation	1	1	-1	1	1	-1
insertionsort	1	1	-1	1	-1	-1
binarysearch	-1	1	1	1	1	1
findeucdist	-1	1	1	1	1	1

as shown in the following equation.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{9}$$

Each column in the label set represents a category, that is, a transformation relationship. At first, we can count the samples of value ''1'' in first column, resulting in sample 1, sample 2, sample 3 and sample 4. Then the four samples are used to form a new sample set, according to the k nearest neighbor algorithm k samples are retained. For example, if $k = 3$ and sample 2, sample 3, and sample 4 are left, we can get the following set of labels.

$$\begin{pmatrix} 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 \end{pmatrix} \tag{10}$$

We can get another label set using the same process

$$\begin{pmatrix} 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{11}$$

In each category, the k-nearest neighbor algorithm has a value of $k = 3$, and 18 rows are counted. The number of values in each category is 1, and the number of values of 1 in these six categories is 10, 18, 10, 18, 15, and 10, respectively. The label counting vector is obtained by linearly stacking each cluster center with the k-means clustering algorithm to obtain the basis function center of RBF neural network.

3) BUILDING RBF-MLMR MODEL

The RBF-MLMR model is established using the multi-label data set to predict the multiple metamorphic relations that the function may satisfy. The main steps of RBF-MLMR are described in Algorithm 1.

During the training phase, the input training data set is obtained by the K-layer cross-validation of the data set composed of the characteristic data extracted in the above steps, and the output is the set of data constituting the label corresponding to each function. The clustering center number k is calculated by the AP clustering algorithm, and then k clustering centers are obtained using the k-means clustering algorithm. Then, the label count vector $C_x(l)$ of the training data set is calculated according to formula (7). According to formula (8), the tag count vector is clustered with the clustering center of each class obtained by clustering k-means as the center of RBF function.

The mean distance of the center of RBF function between classes is calculated by formula (12) to obtain the basis function variance - smoothing factor.

$$\sigma = \mu\left(\frac{\sum_{p=1}^{k} -1 \sum_{q=p+1}^{k} dist(c_p, c_q)}{\frac{k(k-1)}{2}}\right) \tag{12}$$

The calculated smoothing factor is substituted into formula (13) to yield the matrix.

$$\phi_j(x_i) = exp(-\frac{dist(x_i, c_j)^2}{2\sigma_j^2}) \tag{13}$$

Then, the weight w is calculated by formulae (14).

$$(\Phi^T \Phi)W = \Phi T \tag{14}$$

During the testing phase, when the test data is input into formula (15), the corresponding label is calculated.

$$y_i(x_i) = \Sigma_{j=0}^{L} w_{jl} \phi_j(x_i) \tag{15}$$

Algorithm 1 RBF-MLMR

Require: Multi-label data set $S = \{(x_i, Y_i), (x_2, Y_2), ..., (x_n, Y_n)\}(x_i \in X, Y_i \in Y)$, and parameter μ

Ensure: The label set which can satisfy the target functions.

Training Phase;
Using AP clustering to get the value of k
Using K-means clustering to yield each type of base function center $c_{(l)}$
Calculating the label count vector $C_x(l)$
for $l \in y$ **do**
  Calculating N_x^l according to formulae (5)
end for
Calculating the tag count vector according to formulae (7)
Calculating the basis function center C_l of RBF neural network according to formula (8)
Calculating the variance according to formulae (3)
Calculating matrix Φ according to formula (12) and (13)
Calculating the weight w according to formulae (4)
Testing Phase;
Inputting the measured data set$P = x_1, x_2, ..., x_n(x_i \in X)$
Calculating the set of labels Z of the measured function according to formulae (1)
for $l \in y$ **do**
  if the output value is bigger than 0 **then**
    $y(l) = 1$
  else
    $y(l) = -1$
  end if
end for

V. EXPERIMENTAL EVALUATION

In this section, we conduct a set of experiments to validate RBF-MLMR based on a public multi-label data set. The experiments are designed to investigate the following basic research questions:

- REQ 1: Is RBF-MLMR effective when compared to other approaches?
- REQ 2: How much time is required for executing RBF-MLMR?
- REQ 3: Are the prediction results influenced by the sizes of data set?

A. EXPERIMENTAL SETUP

The experimental environment is a Windows PC with Acer Intel Pentium G2030 CPU4G RAM. The results are compared with other representative multi-label algorithms, such as, Rank-SVM, ML-KNN, and BP-MLL. A code library including more than 100 functions, written in Java, has been established for the validation of the RBF-MLMR approach. Among them, 75 representative functions are used for the experiments while other programs with similar functions are removed. The process of constructing a multi-label dataset, comprising data extracted from the corresponding control flow graph of specific functions, follows the process described previously. In order to make the experiments repeatable, we construct the multi-label data set, and make it downloadable online.[1] The functions used in the experiment are derived from the following open source projects:

- *The Colt Project* [2]: a set of open source libraries written for high-performance scientific and technical computing in Java;
- *The Apache Mahout* [3]: a machine library written in Java;
- *The Apache Commons Mathematics Library* [4]: a library of mathematical and statistic components written in Java.

There are two stages in the experiment, *training* and *testing*. Furthermore, k-fold cross-validation has been used in the experiment. In this prediction model, the training data set is obtained by k-fold cross-validation. Through the k-fold cross-validation the training set is better; this will ensure that the experimental results are more accurate. The raw data set has been partitioned randomly into k subsets. The first $k-1$ subset is used for *training*, and the remaining is used for *testing*. In other words, we repeat the process k times (the folds) with each of k subsets treated as a test dataset and the remaining data as training data sets. Finally, a single estimation could be accessed by averaging k results of the experiments. Seven evaluation metrics [22], [27], [28] have been used in our study, namely *average precision*, *hamming loss*,*coverage*, *one-error*, *ranking loss*, *accuracy*, and *time consumption*.

- *Average precision* is a classifier that predicts the sample's label as the average proportion of the correct label. Generally speaking, the higher the value, the better the prediction result.

$$f_{AP}(f) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i|}\frac{|L_i|}{f_{rank}(X_i, y)} \tag{16}$$

  where $L_i = \left\{y' | f_{rank}(X_i, y') \leq f_{rank}(X_i, y), y' \notin Y_i\right\}$.
- *Hamming loss* measures the misclassification rate of the whole testing data. Tag marked wrongly or missing over the testing set are considered to evaluate the total probability, and it is given by the following formula:

$$f_{HL}(h) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{Q}|h(x_i)\Delta Y_i| \tag{17}$$

  where Δ stands for the symmetric difference between two sets, Q is the total number of possible class labels, $h(x_i)$ is a multi-label classifier, and n represents the number of experiments.
- *Coverage* refers to the average number of labels that can cover the number of samples corresponding to all the labels. The smaller is the value, the better is the

[1] https://github.com/QXL4515/data-set/
[2] http://acs.lbl.goc/software/colt/
[3] https://mahout.apache.org/
[4] http://commons.apache.org/proper/commons-math/

TABLE 7. The experimental results of four multi-label learning algorithms.

Evaluation standard	ML-RBF	Rank-SVM	ML-KNN	BP-MLL
average precision	0.827	0.736	0.788	0.807
coverage	2.875	4.1	4.5	3.5
hamming loss	0.208	0.458	0.278	0.277
one-error	0.2	0.333	0.4	0.286
ranking loss	0.197	0.375	0.25	0.2

performance. The formula is described as follows:

$$f_C(f) = \frac{1}{n}\sum_{i=1}^{n} maxf_{rank}(X_i, y) - 1, (y \in Y_l) \quad (18)$$

- *one-error* is used to estimate the results of multi-label prediction in the first row of the label that is not the correct label. The smaller is the value, the better is the performance. The formula is described as follows:

$$f_{OE}(f) = \frac{1}{n}\sum_{i=1}^{n} [[(f(X_i, y)) \notin Y_i]], (y \in Y) \quad (19)$$

- *ranking loss* refers to the average error of the order in which the labels are arranged. The smaller is the value, the better is the performance. The formula is described as follows:

$$f_{RL}(h) = \frac{1}{n}\sum_{i=1}^{n} \frac{|D_i|}{|Y_i|\left|\overline{Y}_i\right|} \quad (20)$$

- *accuracy* refers to the proportion of the sample that predicts correctly w.r.t the total number of samples.
- *Time consumption* is the computation time for *training* and *prediction*.

B. EXPERIMENTAL RESULTS

REQ 1: Is RBF-MLMR effective when compared to other approaches?

In order to evaluate the effectiveness of the proposed approach, RBF-MLMR is compared with other multi-label learning algorithms based on the constructed multi-label data set. The experimental dataset contains six labels and the experimental results are shown in Table 7.

As can be seen from Table 7, RBF-MLMR is the best one among five indicators of *average precision*, *coverage*, *hamming loss*, *one-error*, and *ranking loss*, followed by BP-MLL. In order to further evaluate RBF-MLMR, we validate the above-mentioned multi-label data set. In the simulation experiment, RBF-MLMR and BP-MLL were used to predict *single-label*, *double-label*, and *three-label* metamorphic relations. The two approaches are compared by *accuracy*, *loss of Hamming*, and *time*.

Figure 5, Figure 6, and Figure 7 are the results obtained mainly through the 15-layer cross validation. First, the data set is divided into 15 equal parts, 14 sets are used as training data set and the remaining set is used as the test data set, repeated 15 times. Figure 5 shows the experimental results

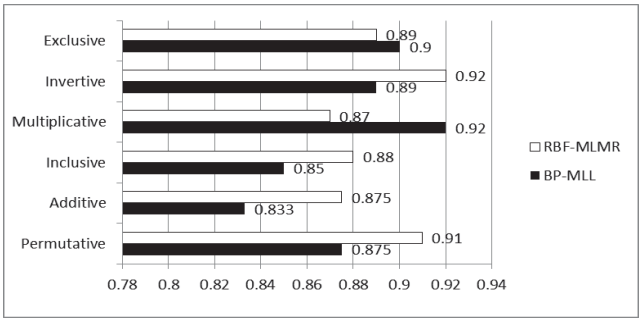


FIGURE 5. RBF-MLMR and BP-MLL predict the accuracy of a single MR.

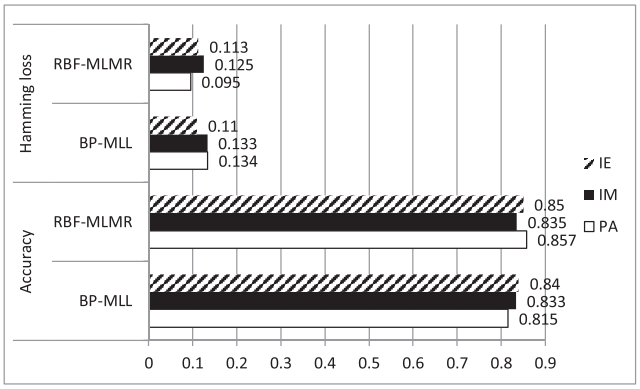


FIGURE 6. RBF-MLMR and BP-MLL predict the accuracy of double-labeled MR and Hamming loss.

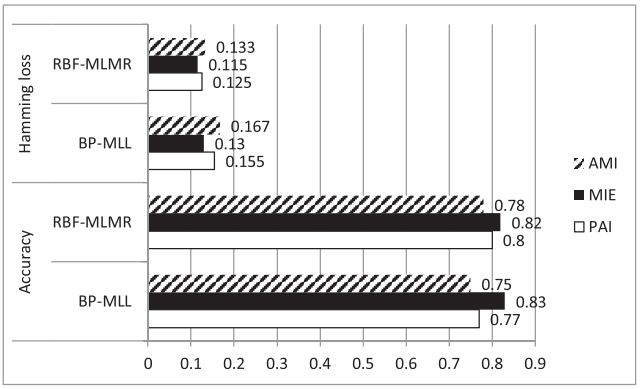


FIGURE 7. RBF-MLMR and BP-MLL predict the accuracy of three labels MR and Hamming loss.

of RBF-MLMR and BP-MLL, respectively. It shows the accuracy of *Permutative*, *Additive*, *Inclusive*, *Multiplicative*, *Invertive* and *Exclusive* metamorphic relation on single-label metamorphic relationship prediction. It can be seen from the

TABLE 8. RBF-MLMR and BP-MLL training and testing time (in second) for single MR prediction.

	Training phase		Testing phase	
Metamorphic relation	**BP-MLL**	**RBF-MLMR**	**BP-MLL**	**RBF-MLMR**
Permutative	2.6208	0.0312	0.1388	0.0173
Additive	2.6801	0.0338	0.1482	0.0234
Inclusive	2.8174	0.0312	0.1498	0.0234
Multiplicative	2.9781	0.0343	0.1498	0.0156
Invertive	2.9515	0.0334	0.1498	0.0223
Exclusive	2.7066	0.0312	0.1466	0.0156

TABLE 9. RBF-MLMR and BP-MLL training and testing time for double-label MR prediction.

	Training phase		Testing phase	
Metamorphic relation	**BP-MLL**	**RBF-MLMR**	**BP-MLL**	**RBF-MLMR**
PA	371.6099	0.0312	0.1466	0.0203
IM	396.5788	0.0296	0.1543	0.0187
IE	410.9675	0.0312	0.1551	0.0203

figure that the accuracy of RBF-MLMR and BP-MLL is better than 0.8 for all six metamorphic relations. The prediction results of RBF-MLMR for *Permutative*, *Additive*, *Inclusive*, and *Invertive* metamorphic relation are significantly higher than that of BP-MLL. The prediction results of BP-MLL for *Exclusive* and *Multiplicative* metamorphic relations are higher than RBF-MLMR. *Accuracy* of RBF-MLMR for *Permutative* and *Invertive* metamorphic relation is more than 0.9.

Figure 6 shows the experimental results of the *accuracy* and *Hamming losses* predicted by RBF-MLMR and BP-MLL for the double-label metamorphic relations. In the figure, PA represents simultaneous prediction of *Permutative* and *Additive* metamorphic relations, IM indicates simultaneous prediction of *Inclusive* and *Multiplicative* metamorphic relations, and IE indicates simultaneous prediction of *Invertive* and *Exclusive* metamorphic relations. From RBF-MLMR and BP-MLL on the double label metamorphic relations prediction results, we can see that accuracy is not higher than that of single label metamorphic relation, but it is still 0.8 or more. It can be seen clearly that the accuracy of RBF-MLMR is higher than that of BP-MLL, and the prediction accuracy of RBF-MLMR is 0.857 and BP-MLL is 0.815. The prediction accuracy of IM metamorphic relation is similar, with the accuracy of RBF-MLMR of 0.835 and BP-MLL of 0.833. The prediction accuracy of RBF-MLMR for metamorphic relation IE is higher than that of BP-MLL, where the accuracy of RBF-MLMR is 0.85 and BP-MLL is 0.84. From the prediction results on double label, we can see *Hamming loss* of RBF-MLMR is about 0.1, a little better than BP-MLL. For the prediction of PA metamorphic relation, the Hamming loss of BPF-MLMR is 0.095 and that of BP-MLL is 0.134. For the IM metamorphic relation the Hamming loss of RBF-MLMR is 0.125, and for BP-MLL is 0.133. For the above two sets of metamorphic relations, the RBF-MLMR is lower than BP-MLL for Hamming loss, indicating that RBF-MLMR is better. For the prediction of IE metamorphic relation, the Hamming loss of RBF-MLMR is 0.113, and that of BP-MLL is 0.11, which is similar. In general, the overall accuracy and Hamming loss are good and RBF-MLMR performs better for double-label metamorphic relation.

Figure 7 shows the results of the RBF-MLMR and BP-MLL predictions on three-label metamorphic relations. In the figure, PAI indicates simultaneous prediction of *Permutative*, *Additive*, and *Inclusive* relations. MIE indicates simultaneous prediction of *Multiplicative*, *Invertive*, and *Exclusive* relations. AMI means simultaneous prediction of *Additive*, *Multiplicative*, and *Invertive* relations. It can be seen from the figure that the accuracy of the overall prediction of the three-label MR is about 0.8, and the prediction accuracy of the RBF-MLMR and BP-MLL for MIE metamorphic relations are 0.82 and 0.83, respectively. The prediction accuracy of PAI metamorphic relations are 0.8 and 0.77, respectively, and the prediction accuracy of AMI metamorphic relations are 0.78 and 0.75, respectively.

Based on the results of *single-label*, *double-label* and *three-label* metamorphic relation prediction, it is concluded that RBF-MLMR is better in terms of *accuracy* and *Hamming loss*.

REQ 2: How much time is required for executing RBF-MLMR?

Table 8, Table 9 and Table 10 give the time overhead for (*training* and *testing*) stages, respectively. It can be seen from the tables that RBF-MLMR has less overhead in *training* and *testing* than BP-MLL. Also the prediction time cost of

TABLE 10. RBF-MLMR and BP-MLL training and testing time for three-label MR prediction.

	Training phase		Testing phase	
Metamorphic relation	**BP-MLL**	**RBF-MLMR**	**BP-MLL**	**RBF-MLMR**
PAI	659.6549	0.0312	0.1681	0.0234
MIE	647.8585	0.0338	0.1658	0.0208
AMI	483.644	0.0312	0.195	0.0218

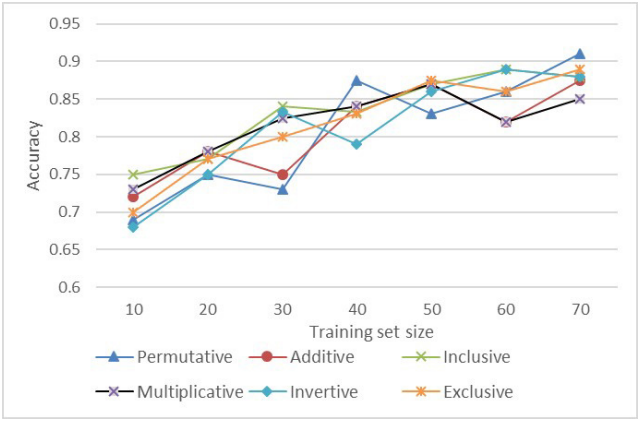


FIGURE 8. Accuracy of various training set sizes for predicting the six MRs.

RBF-MLMR for *single-label*, *double-label*, and *triple-label* metamorphic relations is not significant, but BP-MLL incurs much higher cost. Based on the above results, it can be seen that the time overhead of RBF-MLMR is better for all cases. Consequently, from the time overhead aspect, RBF-MLMR is more suitable for multi-label prediction.

REQ 3: Are the prediction results influenced by the sizes of data set?

In the following we focus on the performance of the RBF-MLMR model as the size of the training set changes during the process of predicting the metamorphic relation. Using a large number of training sets will improve the accuracy of the model, but will increase the cost because of manual work required to identify the metamorphic relation for the training set and extract the characteristic data. In this part, we do the prediction experiment of *single-label*, *double-label*, and *three-label* metamorphic relations for the six different metamorphic relation of *Permutative*, *Additive*, *Inclusive*, *Multiplicative*, *Invertive*, and *Exclusive* under different sized data sets. The six metamorphic relations are used for single label metamorphic relation prediction. In the prediction of two labels metamorphic relations, double labeling PA, PI, AI, IE, and IM are carried out, respectively. The metamorphic relations of PAI, MIE, and AMI are predicted for three labels metamorphic relations.

In this experiment, the training set size is set as 10, 20, ..., and 70. In each experiment K-layer cross-test is used. In single labels we mainly use *accuracy rate* as the evaluation index. Figure 8 shows the results when using RBF-MLMR to predict *Permutative*, *Additive*, *Inclusive*, *Multiplicative*, *Invertive*, and *Exclusive* metamorphic relation, respectively. As the size of the training set changes the accuracy of the corresponding prediction results also change. We can know that using the RBF-MLMR prediction model with a small training set is better than a random classifier. With the increase of the size of the training set, the accuracy curve tends to rise as a whole, and the change is more obvious when the training data size reaches 50 with the accuracy above 0.85 after 50. For the metamorphic relation *Permutative*, when the training set size is 30-40, the accuracy rate exceeds 0.85. For the metamorphic relation *Additive*, the accuracy is close to 0.85 when the training set size is 40. When the *Inclusive* and *Invertive* metamorphic relations are predicted, the accuracy is greater than 0.85 when the training set size is 50. For *Multiplicative* and *Exclusive* metamorphic relations, as the data set size approaches 45, the accuracy is close to 0.85. These results show that RBF-MLMR is also effective for small training sets for single label metamorphic relation.

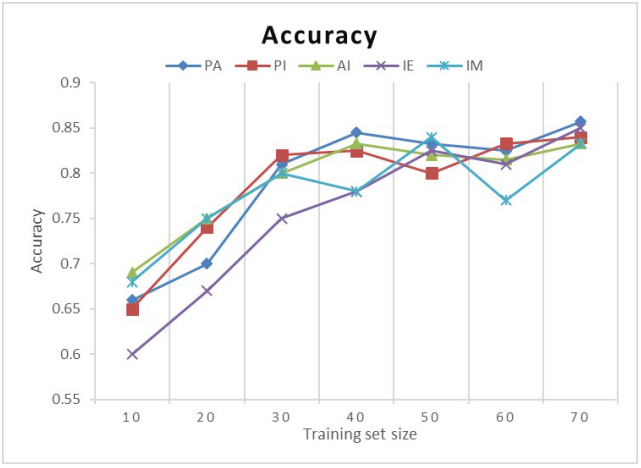


FIGURE 9. Accuracy of various training set sizes for predicting MRs of double-label.

For multi-label metamorphic relations prediction, *accuracy* and *Hamming loss* are used as an evaluation index. Figure 9 and Figure 10 show the use of RBF-MLMR for the double-label metamorphic relations prediction. It can be seen from the experimental results that RBF-MLMR is also effective for small training data sets for double label metamorphic relations. In Figure 9 and Figure 10, it is shown that *accuracy* and *Hamming Loss* of PA (Permutative and

FIGURE 10. Hamming Loss of various training set sizes for predicting MRs of double-label.

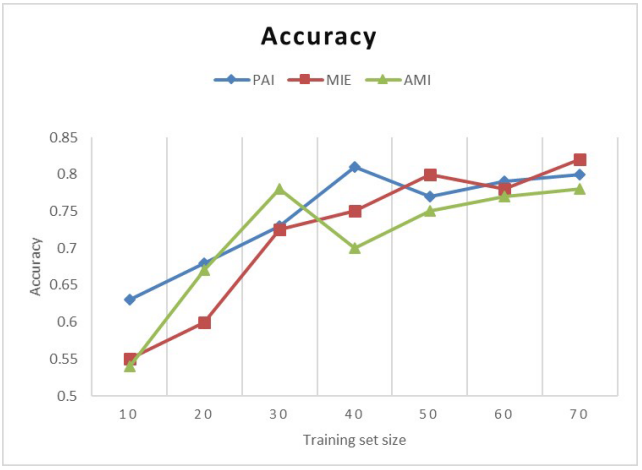


FIGURE 11. Accuracy of various training set sizes for predicting MRs of three-label.

Additive) metamorphic relations vary with the size of the experimental data set. When the training set size is 40, the accuracy rate is more than 0.8, and the Hamming loss is about 0.1. Figure 9 and Figure 10 show the change of *accuracy* and *Hamming loss* for simultaneous prediction of PI (Permutative and Inclusive) metamorphic relations. We can see that when the training set size is more than 30 the accuracy rate is more than 0.8. When the training set size reaches 60, the Hamming loss is about 0.1. Figure 9 and Figure 10 show the *accuracy* and *Hamming loss* for prediction of AI (Additive and Inclusive) metamorphic relations, we can see from the figure when the training set size is 30, the accuracy rate is 0.8. When the training set size is 40, the Hamming loss is close to 0.1. Figure 9 and Figure 10 show changes of *accuracy* and *Hamming loss* simultaneous prediction of IE (Invertive and Exclusive) metamorphic relations. When the data set size is 50 the accuracy rate is 0.8, and Hamming loss also close to 0.1. Figure 9 and Figure 10 show the change of *accuracy* and *Hamming loss* for simultaneous prediction of IM (Inclusive and Multiplicative) metamorphic relations. When the data set size is 50, the accuracy rate is close to 0.85, and Hamming loss is close to 0.1. Based on the results of the double-labelled metamorphic relations, it can be seen that when the data set size exceeds 45, the accuracy rate exceeds 0.8, and Hamming loss tends to be 0.1. This result shows that RBF-MLMR is effective in predicting double-labelled metamorphic relations under small training set.

Figure 11 and Figure 12 show the curves of the *accuracy* and *Hamming loss* with the changing training set for the three-label metamorphic relations. In Figure 11 and Figure 12, the accuracy of the PAI (*Permutative*, *Additive*, and *Inclusive*) metamorphic relation and the changes in Hamming losses are shown. It can be seen from the figure when the training set size exceeds 40, the *accuracy* is close to 0.8. When the data set size exceeds 60, the *Hamming loss* tends to 0.1. Figure 11 and Figure 12 show the *accuracy* and *Hamming loss* of the MIE (*Mutiplicative*, *Invertive*, and

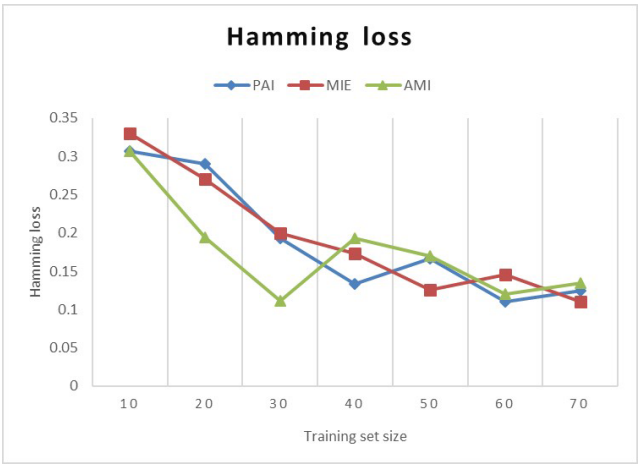


FIGURE 12. Hamming Loss of various training set sizes for predicting MRs of three-label.

Exclusive) metamorphic relations. When the training set size exceeds 50, the *accuracy* is about 0.8. When the training set increases, Hamming loss is close to 0.1. Figure 11 and Figure 12 show the *accuracy* and *Hamming loss* of the AMI (*Additive*, *Multiplicative* and *Invertive*) metamorphic relations. When the training set size exceeds 50, the accuracy rate is close to 0.8. Hamming loss tends to 0.1 when the training set size increases. These results also show that RBF-MLMR is better than other approaches for the prediction of triple label metamorphic relations.

According to the prediction results of single label, double label and triple label metamorphic relations, it can be seen that in general RBF-MLMR is effective for small and big training sets.

C. THREAT TO VALIDITY

The experimental results show that RBF-MLMR has the highest precise among all the approaches; however, there are some threats which may affect experimental results. Firstly,

the analysis of function control flow graph needs the characteristic extracted from the function control flow graph. Sometime it cannot fully reflect the relationship between the metamorphic relation and the program structure. Consequently, some important metamorphic relations may be missing due to lacking information. The authors plan to further study the representation of the function, fully exploit the relationship between the structure of the program and metamorphic relation, and improve the performance of RBF-MLMR. Secondly, the experimental results show that *accuracy* is sensitive to different sizes of data. It increases with the size of the experimental data set when the single label metamorphic relation is predicted. For data set size of 50 or less, the increase is relatively large. The increase is not large when the size is greater than 50. In multi-label metamorphic relations prediction when the data set size is 60 or less, the *accuracy* rate increases greatly and the *Hamming loss* reduction is obvious. When the size of the data set is more than 60, the overall *accuracy* rate is still increasing, and the *Hamming loss* is decreasing. Further experiments should be performed to give a reasonable experimental data set size range that can improve the accuracy of the prediction. Thirdly, currently RBF-MLMR can only predict at most three metamorphic relations at a time. When four or more are predicted, RBF-MLMR cannot generate the results or the *accuracy* is very low. This also affects its wide applications.

VI. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

This paper proposed a novel approach called RBF-MLMR, which is an automatic multi-label metamorphic relation prediction approach based on the RBF neural network. This paper uses a set of programs without test oracle to evaluate the performance and accuracy of RBF-MLMR. Experimental results show that RBF-MLMR has high accuracy, improves the efficiency of the metamorphic test, and it also lays the foundation for the metamorphic test tool development.

Several directions for future work are possible:

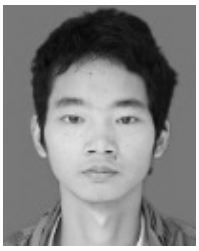
- *The representation of the function*. The experimental data set of this paper consists of *node feature*, *path feature*, *value change path*, *node feature number*, *path feature number*, and *function label information*. The authors will further study the function representation method and find the approach to fully reflect the function characteristics so as to extract the characteristics of the function, and fully indentify the relationship between the program structure and the metamorphic relation.
- *Optimizing RBF-MLMR*. The maximum number of labels to be predicted is three in this proposed approach. The accuracy of prediction for more labels is not high enough and further research is needed to improve the accuracy rate. In addition, the RBF-MLMR model can predict more metamorphic relations at a time, and improve the efficiency of metamorphic testing and reduce the cost of testing, support a wider application of metamorphic testing.
- *Large scale programs*. Currently, RBF-MLMR is mainly used for single function test or unit test. As future work, the RBF-MLMR should be applied in large scale programs to validate its usability.

REFERENCES

[1] D. Bibicu and L. Moraru, "Cardiac cycle phase estimation in 2-D echocardiographic images using an artificial neural network," *IEEE Trans. BioMed. Eng.*, vol. 60, no. 5, pp. 1273–1279, May 2012.
[2] D. Bibicu and L. Moraru, "Run-length textural descriptors and artificial neural networks for steatosis liver disease detection," *Adv. Sci.*, vol. 5, no. 11, pp. 1137–1143, 2013.
[3] S. Chatterjee, S. Sarkar, S. Hore, N. Dey, A. S. Ashour, and V. E. Balas, "Particle swarm optimization trained neural network for structural failure prediction of multistoried RC buildings," *Neural Comput. Appl.*, vol. 28, no. 8, pp. 1–12, 2016.
[4] T. Y. Chen, J. Feng, and T. H. Tse, "Metamorphic testing of programs on partial differential equations: A case study," in *Proc. COMPSAC*, 2002, pp. 327–333.
[5] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Dept. Comput. Sci., Hong Kong Univ. Sci. Technol., Hong Kong, Tech. Rep. HKUST-CS98-01, 1998.
[6] T. Y. Chen, J. W. Ho, H. Liu, and X. Xie, "An innovative approach for testing bioinformatics programs using metamorphic testing," *BMC Bioinformatics*, vol. 10, no. 1, p. 24, 2009.
[7] J. Ding, T. Wu, D. Wu, J. Q. Lu, and X. H. Hu, "Metamorphic testing of a Monte Carlo modeling program," in *Proc. 6th Int. Workshop Autom. Softw. Test*, 2011, pp. 1–7.
[8] G. Dong, B. Xu, L. Chen, C. Nie, and L. Wang, "Survey of metamorphic testing," *J. Frontiers Comput. Sci. Technol.*, vol. 3, no. 2, pp. 130–143, 2009.
[9] B. A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification. Neural information processing systems," in *Proc. NIPS*, 2012, pp. 681–687.
[10] D. C. Francesco and M. Tommasi, *Learning Multi-label Alternating Decision Trees from Texts and Data*. Berlin, Germany: Springer-Verlag, 2003.
[11] R. Guderlei and J. Mayer, "Statistical metamorphic testing testing programs with random output by means of statistical hypothesis tests and metamorphic testing," in *Proc. QSIC*, Oct. 2007, pp. 404–409.
[12] M. Jiang, T. Y. Chen, F. C. Kuo, and Z. Ding, "Testing central processing unit scheduling algorithms using metamorphic testing," in *Proc. IEEE Int. Conf. Softw. Eng. Service Sci.*, May 2013, pp. 530–536.
[13] U. Kanewala and J. M. Bieman, "Using machine learning techniques to detect metamorphic relations for programs without test oracles," in *Proc. IEEE Int. Symp. Softw. Rel. Eng.*, Nov. 2013, pp. 1–10.
[14] U. Kanewala, J. M. Bieman, and A. Ben-Hur, "Predicting metamorphic relations for testing scientific software: A machine learning approach using graph kernels," *Softw. Test. Verification Rel.*, vol. 26, no. 3, pp. 245–269, 2015.
[15] Z. Li *et al.*, "Rule-based back propagation neural networks for various precision rough set presented kansei knowledge prediction: A case study on shoe product form features extraction," *Neural Comput. Appl.*, vol. 28, no. 3, pp. 1–18, 2016.
[16] H. Liu, X. Liu, and T. Y. Chen, "A new method for constructing metamorphic relations," in *Proc. 12th Int. Conf. Quality Softw. (QSIC)*, 2012, vol. 430. no. 4, pp. 59–68.
[17] J. Mayer and R. Guderlei, "An empirical study on the selection of good metamorphic relations," in *Proc. COMPSAC*, 2006, pp. 475–484.
[18] C. Murphy, *Metamorphic Testing Techniques to Detect Defects in Applications Without Test Oracles*. Columbia, SC, USA: Columbia Univ., 2010.
[19] C. Murphy, G. E. Kaiser, L. Hu, and L. Wu, "Properties of machine learning applications for use in metamorphic testing," in *Proc. 12th Int. Conf. Softw. Eng. Knowl. Eng.*, 2008, pp. 867–872.
[20] L. Saba *et al.*, "Automated stratification of liver disease in ultrasound: An online accurate feature classification paradigm," *Comput. Methods Programs Biomed.*, vol. 130, p. 118, Jul. 2016.
[21] R. Sanders and D. Kelly, "The challenge of testing scientific software," in *Proc. Conf. Assoc. Softw. Test.*, vol. 233. 2008, pp. 30–36.
[22] R. E. Schapire and Y. Singer, "BoosTexter: A boosting-based system for text categorization," *Mach. Learn.*, vol. 39, nos. 2–3, pp. 135–168, May 2000.

[23] R. Vallee-Rai and L. J. Hendren, "Jimple: Simplifying java bytecode for analyses and transformations," *Citeseer*, pp. 1–15, 1998.
[24] R. Wang, "Researches on basic criterion and strategy of constructing metamorphic relations," *Comput. Sci.*, vol. 39, no. 1, pp. 115–119, 2012.
[25] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *J. Syst. Softw.*, vol. 84, no. 4, pp. 544–558, 2011.
[26] S. Yan, X. Yang, M. Li, H. Liu, and Z. Liu, *Research of Testing for Scientific Computing Software in the Area of Nuclear Power Based on Metamorphic Testing*. Singapore: Springer-Verlag, 2016.
[27] M. L. Zhang, "ML-RBF: Rbf neural networks for multi-label learning," *Neural Process. Lett.*, vol. 29, no. 2, pp. 61–74, 2009.
[28] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1338–1351, Oct. 2006.
[29] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.
[30] Y. Zhang, Y. Y. Liu, Y. Zhang, and Y. Y. Liu, "Java bytecode optimization framework," *Comput. Eng.*, vol. 34, no. 2, pp. 69–71, 2008.
[31] Z. Y. Zhang, W. Chan, T. Tse, and P. F. Hu, "Experimental study to compare the use of metamorphic testing and assertion checking," *J. Softw.*, vol. 20, no. 10, pp. 2637–2654, 2009.
[32] Z. Q. Zhou, D. Huang, T. Tse, Z. Yang, H. Huang, and T. Chen, "Metamorphic testing and its applications," in *Proc. 8th Int. Symp. Future Softw. Technol. (ISFST)*, 2004, pp. 346–351.
[33] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *IEEE Trans. Softw. Eng.*, vol. 42, no. 3, pp. 264–284, Mar. 2016.

PENGCHENG ZHANG received the Ph.D. degree in computer science from Southeast University in 2010. He is currently an Associate Professor with the College of Computer and Information, Hohai University, Nanjing, China. His research interests include modeling, analysis, testing and verification of component-based systems, software architectures, real-time and probabilistic systems, service-oriented systems.



XUEWU ZHOU received the B.S. degree in computer science from the Wuhan Institute of Technology, Wuhan, China, in 2016. He is currently working toward the M.S. degree with the College of Computer and Information, Hohai University, Nanjing, China. His current research interests include data mining and software engineering.

PATRIZIO PELLICCIONE is currently an Associate Professor (Docent in Software Engineering) with the Department of Computer Science and Engineering, Chalmers University of Technology and the University of Gothenburg. He is an Assistant Professor with the Computer Science Department, University of L'Aquila (on leave). The fingerprint of his research activity can be summarized in one sentence: moving formal methods from theory to practice in software architecture modeling and verification.

HARETON LEUNG joined Hong Kong Polytechnic University in 1994, where he is currently the Director of the Laboratory for Software Development and Management. His research interests include testing, metrics, project management, risk management, and quality and process improvement.