Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Available online 1 March 2013 http://www.sciencedirect.com/science/article/pii/S0925231213001938

Multiobjective genetic programming for maximizing ROC performance

Pu Wang [a], Ke Tang [a,*], Thomas Weise [a], E.P.K. Tsang [b], Xin Yao [c]

[a] Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, Anhui 230027, China

[b] Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK

[c] Center of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, The University of Birmingham Edgbaston, Birmingham B15 2TT, UK

ARTICLE INFO

Keywords:
Classification
ROC analysis
AUC
ROCCH
Genetic programming
Evolutionary multiobjective algorithm
Memetic algorithm
Decision tree

ABSTRACT

In binary classification problems, receiver operating characteristic (ROC) graphs are commonly used for visualizing, organizing and selecting classifiers based on their performances. An important issue in the ROC literature is to obtain the ROC convex hull (ROCCH) that covers potentially optima for a given set of classifiers [1]. Maximizing the ROCCH means to maximize the true positive rate (*tpr*) and minimize the false positive rate (*fpr*) for every classifier in ROC space, while *tpr* and *fpr* are conflicting with each other. In this paper, we propose multiobjective genetic programming (MOGP) to obtain a group of nondominated classifiers, with which the maximum ROCCH can be achieved. Four different multiobjective frameworks, including Nondominated Sorting Genetic Algorithm II (NSGA-II), Multiobjective Evolutionary Algorithms Based on Decomposition (MOEA/D), Multiobjective selection based on dominated hypervolume (SMS-EMOA), and Approximation-Guided Evolutionary Multi-Objective (AG-EMOA) are adopted into GP, because all of them are successfully applied into many problems and have their own characters. To improve the performance of each individual in GP, we further propose a memetic approach into GP by defining two local search strategies specifically designed for classification problems. Experimental results based on 27 well-known UCI data sets show that MOGP performs significantly better than single objective algorithms such as FGP, GGP, EGP, and MGP, and other traditional machine learning algorithms such as C4.5, Naive Bayes, and PRIE. The experiments also demonstrate the efficacy of the local search operator in the MOGP framework.

1. Introduction

Classification [2] is one of the most important areas in machine learning. Here, the goal is to find assignments of classes to un-classified and unseen instances (data samples) based on information previously learned. In the most common case, referred to as binary classification, there are two classes or categories and all instances in a data set belong to one of them. Solving classification problems basically means to design good classifier(s) which make right assignments as often as possible.

One open question is how to measure the performance of a classifier. If classifiers are synthesized with optimization algorithms, the choice of the performance measure will have tremendous impact on the results that we will obtain. Simple classification accuracy, though being used as the performance metric for a long time, is actually not a good choice [3]. The receiver operating characteristics, or ROC for short, has been claimed as a generally useful performance visualizing method because its properties are not sensitive to skewed class distributions or unequal misclassification costs, two characteristics which are known to have a negative impact on the utility of the accuracy measure.

The ROC graph is a technique for visualizing, organizing and selecting classifiers based on their performance [1]. It has been widely used in signal detection [4], medical decision making [5], and other fields over the course of the past 40 years. In recent years, because of the ever-increasing use of ROC graphs in the machine learning community, the ROC analysis became a central technique for tackling classification problems. The ROC curve, an important topic in ROC analysis, is obtained by varying discriminative thresholds over the output of a classifier [1]. The area under the ROC curve (AUC) is accepted as a fair indicator to measure the classifier performance for binary classification, since it is invariant to operating conditions such as different misclassification costs and skewed class distributions [6]. ROCCH, another important topic in classification problems, represents the convex hull of a set of points (hard classifiers) obtained from several

* Corresponding author.
E-mail addresses: wuyou308@mail.ustc.edu.cn (P. Wang), ketang@ustc.edu.cn (K. Tang).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.06.054

curves (i.e., soft classifiers) [7]. A classifier is potentially optimal if and only if it touches the ROCCH. Otherwise, we can always find a better classifier. It is possible to get a potentially optimal classifier in ROCCH even if the data sets have skewed class distributions or misclassification costs. Actually, we can consider the ROC curve as a special ROCCH when there is only a single soft classifier. This means that ROCCH could work more powerfully than a plain ROC curve. Consequently, we mainly consider the ROCCH in this paper and we will focus on searching a group of classifiers not only to maximize the ROCCH performance but also try to maximize the AUC of a single soft classifier in binary classification problems.

In this paper, we utilize GP combined with multiobjective techniques to approximate the optimal ROCCH. This work empirically investigates multiobjective genetic programming (MOGP) with four different frameworks on binary classification problems. We show that local search strategies can play a key role in GP for classification problems and that special local search operators can improve the performance.

This paper is organized as follows: Section 2 outlines the related work and in Section 3, we introduce the background and basic algorithms used in our research. Section 4 will describe our framework to classification problems and presents local search operators working in GP. Experiments are studied in Section 5 where four research questions are answered. Section 6 provides the conclusion and a discussions on the important aspects and future perspectives of this work.

2. Related work

2.1. ROCCH in classification

The roots of ROCCH maximization problems can be traced back to [7]. In that work, *iso-performance lines*[1] are translated by operating conditions of classifiers and used to identify a portion of the ROCCH, by which we can choose suitable classifiers for data sets with different skewed class distribution or misclassification costs. In [8], a combination of rule sets to produce instance scores indicating the likelihood that an instance belongs to a given class is described.

Flach et al. [9] investigated a method to detect and repair concavities in ROC curves. The basic idea here is that if a point lies below the line spanned by two other points in ROC, then it can be mirrored to a better point which could perform well beyond the original ROC curve. This can be used to expand the ROCCH. Prati [10] introduced a rule selection algorithm based on ROC analysis to find minimal rule sets with compatible AUC values. Here, selection is based on whether a rule can improve the current ROCCH.

In [11], a method which takes Neyman–Pearson lemma [12] as the theoretical basis for finding the optimal combination of classifiers to maximize the ROCCH is given. Fawcett [13] presents a method for learning rules directly from ROC space. This method utilizes the geometrical properties of the ROC to generate new rules to maximize the ROC performance. Essentially, all above works are searching a rule sets to maximize ROCCH.

2.2. Genetic programming for classification

Genetic programming (GP) [14] is a branch of evolutionary algorithms (EAs). Standard GP has a tree-like representation which can be generated by modular, grammatical, and developmental methods [15]. Tree-based classifiers have a long tradition in machine learning. They are considered to be more explicit, intuitive, and interpretable than, e.g., neural networks. GP therefore has widely been used for solving classification problems [16,17].

An example of using GP to evolve regression rules for a data set with intertwined spirals pattern is already given in Koza's 1992 book [14]. Another early work [18] used in image recognition dates 20 years back. In the area of data mining, GP has been applied most successful in two particular fields: one is classification for data sets with different misclassification costs, as GP is suitable for cost-sensitive learning. [19], e.g., focused on financial forecasting problems by consolidating two types of misclassification errors into a single objective function. GP involving cost-sensitive learning has furthermore been adopted in filtering junk E-mail [20].

The second field is classification of imbalanced data sets, i.e., data sets where one class occurs much more often than the other—one of the areas where the accuracy metric may become useless. Ref. [21] adopts GP to bankruptcy prediction, a prime example for this issue as there are significantly more solvent firms than defaulting ones. Patterson [22] gave a new fitness function for GP applied on highly imbalanced database. Moreover, Bhowan et al. [23] proposed a multiobjective genetic programming approach to evolving accurate and diverse ensembles of genetic program classifiers with good performance on both the minority and majority classes.

Many technologies have been combined with GP to improve the classification performance in these two fields, ranging from ensemble learning over multiobjective methods to local search strategies. As both imbalanced problems and different misclassification costs can be included in the ROCCH [7], this work will focus on GP for maximizing the ROCCH. It should further be mentioned that there is a strong analogy of ROCCH and the Pareto front in multiobjective optimization [24].

In this paper, we use multiobjective GP (MOGP) to approximate the optimal ROCCH. We empirically investigate MOGP with four different frameworks on binary classification problems. Additionally, we show that local search strategies can play a key role in GP for classification problems as special local search operators can carefully be designed to improve the performance.

3. ROCCH, classification, and multiobjective optimization

3.1. Overview of ROCCH in classification problems

3.1.1. ROC Graph and ROCCH

In binary classification problems, each instance I in the data set is marked a certain label from the set $\{\boldsymbol{p}, \boldsymbol{n}\}$ of positive and negative class labels. A classifier is a mapping from instances to predicted classes, and *accuracy* is the most commonly used evaluation measure. However, its disadvantage are known for a long time [25]. Generally, accuracy is not a suitable metric for cost sensitive and skewed class distribution classification problems. To overcome the weakness of accuracy, ROC analysis has been introduced in machine learning. Ref. [26] demonstrated the value of ROC curves in evaluating and comparing algorithms. An important tool of ROC analysis is the ROC graph which is used to visualize the performance of classifiers. The X-axis and Y-axis of ROC graphs display the true positive rate(tpr) and false positive rate (fpr). The performance of a *hard* or *discrete* classifier on a data set can be mapped in a single point in this graph. The upper left point (0,1) represents a perfect classifier which predicts positive (or Yes) to all positive instances and negative (or No) to all negative instances. The points in lower right area are conservative classifiers which produce more negative labels than positive

[1] All classifiers corresponding to the points on one line have the same expected costs.

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054

labels. In contrast, the points in upper right area are liberal classifiers. All the points along the diagonal are totally random classifier and all classifiers below the diagonal have worse-than-random performance. A *soft* classifier which produces a continuous output (e.g. an estimate of an instance's class membership probability) can be mapped in a set of points by varying the threshold. These points then form a ROC curve.

The ROCCH is the convex hull of the set of points in ROC space. These can be obtained from discrete classifiers and soft classifiers alike. The ROC curve of a soft classifier can directly be considered as a ROCCH if there is only one soft classifier. The right side of Fig. 1 shows a convex hull over three different ROC curves. Scott [27] and Fawcett [1] have pointed out that two existing classifiers can be used to create a realizable classifier whose performance (in terms of ROC) lies on the line of connecting the performance of its two endpoints. Hence, any classifiers whose performances are below the ROC convex hull could be defeated by realizable classifiers. A demonstration is shown on the right side of Fig. 1. There are two realizable classifiers whose performance (point *b* and *c*) are better than the performance of point *a*, which is under the ROC convex hull. Point *b* has the same *fpr* with point *a*, but its *tpr* is higher. Point *c* has the same *tpr* with *a*, but its *fpr* is lower. More generally, all the classifiers whose performances are under the convex hull are not optimal. In other words, a classifier is potentially optimal if and only if it lies on the convex hull of the set of points in ROC space [1].

In the following, we will give an example of using ROCCH to search optimal points for different situations such as data sets with different error costs and class distributions. One important target of classification problems is to minimize the total error costs. Suppose $c(\mathbf{Y},\mathbf{n})$ is the cost of a false positive error, $c(\mathbf{N},\mathbf{p})$ is the cost of a false negative error, and *Ntr* is the number of total instances. The class distributions of the data set are noted by the classes' a priori probabilities $p(\mathbf{p})$ and $p(\mathbf{n})=1-p(\mathbf{p})$. The total error costs can be represented as

$$Cost = Ntr \cdot p(\mathbf{n}) \cdot fpr \cdot c(\mathbf{Y},\mathbf{n}) + Ntr \cdot p(\mathbf{p}) \cdot (1-tpr) \cdot c(\mathbf{N},\mathbf{p}) \quad (1)$$

If there are two points $(tpr1,fpr1)$ and $(tpr2,fpr2)$ that have the same total error costs, we can get the following equation:

$$\frac{tpr1-tpr2}{fpr1-fpr2} = \frac{c(\mathbf{Y},\mathbf{n})p(\mathbf{n})}{c(\mathbf{N},\mathbf{p})p(\mathbf{p})} = m \quad (2)$$

In Eq. (2), *m* is defined as a slope of an *iso-performance line* in [7]. In other words, all the classifiers corresponding to the points on an iso-performance line have the same expected cost. Moreover, each set of class and cost distributions (the middle term of above equation) defines a family of iso-performance lines. Moving the iso-performance line until it gets in contact with a point in ROCCH, the joint point with a larger *tpr*-intercept (means the *tpr*-intercept of the line determined by joint point and the slop in ROC graph) represents a classifier which can produce lower expected cost.

In Fig. 2, there are two iso-performance lines: α with $m=10$ and line β with $m=0.1$. Imagine a scenario where the data set has a class distribution where the negatives outnumber the positives by 10 to 1, but the costs of false positives equal to the false negative costs. A classifier with performance at *C*1 is suitable for this data set and achieves the minimal expected cost in this ROCCH. Consider another scenario in which a data set has a balanced class distribution, but the problem is very cost sensitive. If a false negative is 10 times as expensive as false positive, the suitable classifier is located at *C*2 which is on the iso-performance line β with $m=0.1$.

3.2. ROCCH and multiobjective problems

Essentially, the ROCCH maximization problem aims at finding a set of classifiers to approximate the upmost line and the

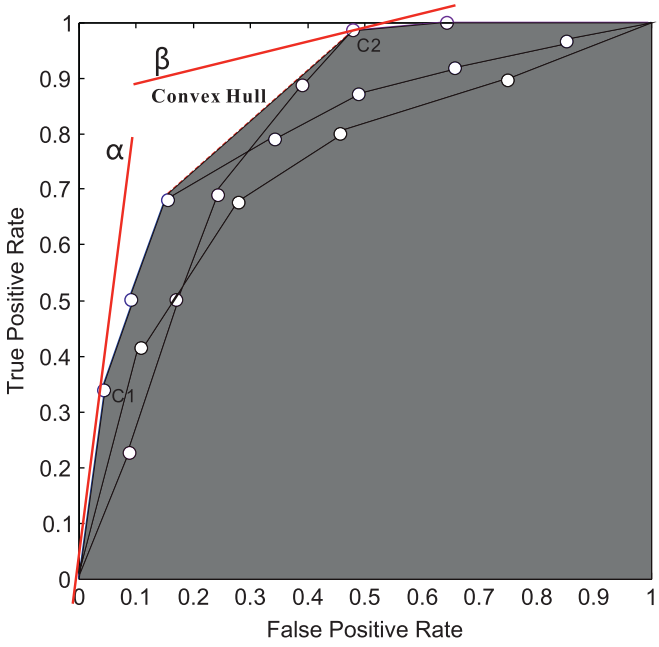

Fig. 2. Find optimal classifiers in ROCCH.

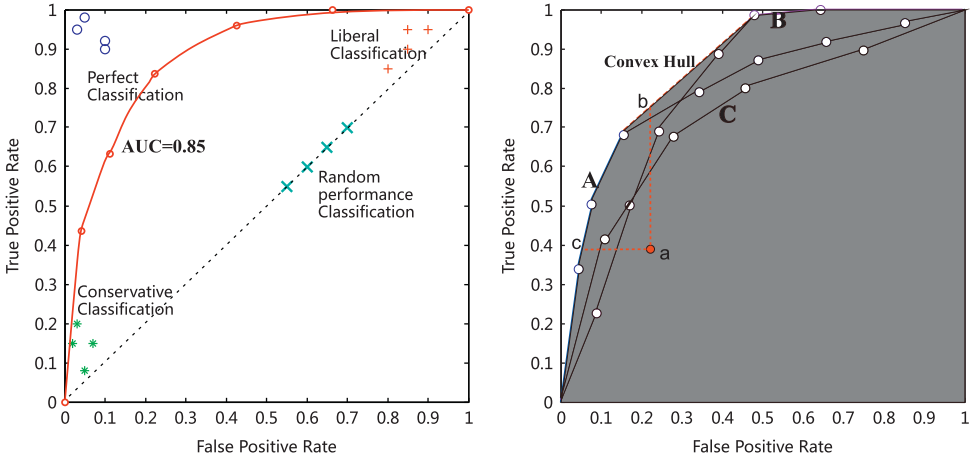

Fig. 1. ROC Graph and ROCCH.

leftmost line in ROC space as closely as possible. Obviously, the ideal performance is point (0,1) which is not easy (and sometimes impossible) to reach. The goal is to find classifiers with a low *fpr* and a high *tpr* at the same time. These two objectives are conflicting because if the classifier labels more instances as positives, it will produce less negatives and vice versa. That means *fpr* and *tpr* are closely related and this relationship is not positive. Hence, a ROCCH maximization problem can be considered as a multiobjective optimization problem. However, minor difference exists between these two concepts. In [24], it is claimed that ROCCH is analogous to the Pareto front (PF) in multiobjective optimization, but no details are given. Here we will describe the relationship between ROCCH and the PF more clearly.

Generally, multiobjective optimization problems (MOP) can be stated as follows:

$$\text{maximize} \quad F(x) = (f_1(x), \ldots, f_m(x))$$
$$\text{subject to} \quad x \in \Omega \tag{3}$$

In Eq. (3), x is the decision variable and $F(x)$ is a vector function representing objective values. In many MOPs, x is continuous and all the objectives are continuous over x. These problems (3) are called *continuous* MOPs. An important term in MOP is *dominance* which can be defined as: let $u = (u_1, \ldots, u_m)$, $v = (v_1, \ldots, v_m)$ be two vectors, u is said to *dominate* v if $u_i \leq v_i$ for all $i = 1 \ldots m$, and $u \neq v$, this is noted as $u \prec v$. If u and v do not dominate each other, we say that u and v are *nondominated*. The nondominated set is a set that each item does not dominate any another one. A point $x^\star$ is called *Pareto optimal* if there is no $x \in \Omega$ such that $F(x)$ dominates $F(x^\star)$ [28,29]. Pareto set (PS) is the collection of all Pareto optimal points. The Pareto front is the set of all the Pareto objective vectors $PF = \{F(x) | x \in PS\}$.

Most of the works on multiobjective optimization methods are searching the PS and PF, which can be realized in many different ways [30–33]. In this work, multiobjective optimization for ROCCH maximization is defined as follows:

$$F(x) = (f_{fpr}(x), f_{tpr}(x))$$
$$\text{minimize } F_1(x) = f_{fpr}(x)$$
$$\text{maximize } F_2(x) = f_{tpr}(x)$$
$$\text{subject to } x \text{ is a classifier} \tag{4}$$

If we take $F_1(x) = -f_{fpr}(x)$, the above formulation is almost the same as the general MOP. The only difference is that the classifier variable x in Eq. (4) is discrete, whereas many multiobjective optimization algorithms have been designed for numerical problems. We will discuss this issue in detail later and give techniques for that situation.

First, let us discuss the relationship between the ROCCH and the Pareto front in multiobjective optimization. In Fig. 3, ROCCH is marked as blue lines and the Pareto front is marked with red broken lines. Any dominated point must obviously be contained under the ROCCH. Hence, the set of dominated solutions contributes nothing to the ROCCH. The points on the ROCCH therefore must be in the nondominated set. Second, not all solutions in nondominated set have contribution to the ROCCH (the points at concavity of the nondominated set are not on the ROCCH). However, when the curve constructed by nondominated set is smooth enough that the difference between ROCCH and Pareto front can be ignored. Hence, the target of using multiobjective optimization to solve ROCCH maximization is to search the nondominated solutions.

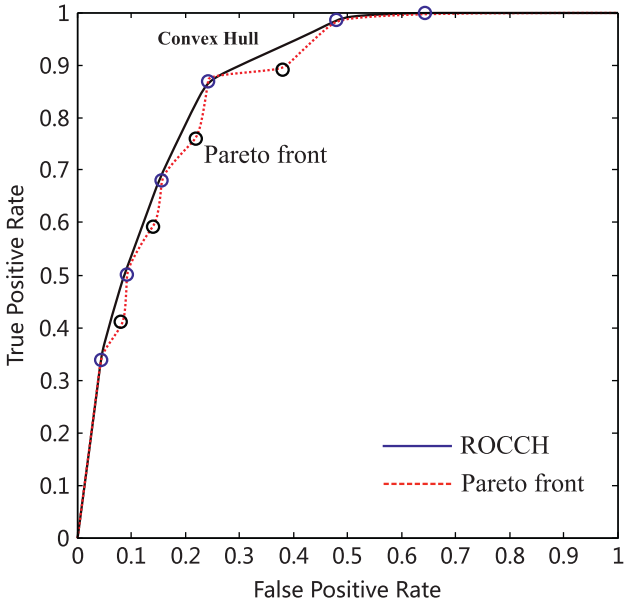

Fig. 3. ROCCH and Pareto front. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. Multiobjective genetic programming for ROCCH maximization

As discussed, we propose to use multiobjective optimization techniques and GP to solve ROCCH maximization problem. Although there exist several evolutionary multiobjective frameworks, most of them are designed for continuous MOPs, and it is unclear which of them suits our problem best. Thus, GP is embedded into four popular MO frameworks to find the suitable evolutionary multiobjective algorithm (EMOA) for ROCCH maximization problems. In this section, we discuss a new framework of GP for classification problem as well as different EMOA strategies in detail. Additionally, to improve the performance of individuals in GP, special local search operators are introduced into multiobjective genetic programming (MOGP) system.

4.1. GP framework for classification problems

GP, like all EAs, maintains a population of candidate solutions. In each iteration of the GP algorithm – referred to as generation – this population undergoes selection and reproduction steps. The three main components of the algorithm are the selection, mutation, and crossover strategies. The latter two, of course, have to be adapted to the tree-based structure of the candidate solutions. Mutation is most often realized by replacing a subtree of a solution with a new, randomly created one and tree-recombination often avails to the exchange of sub-trees of two parent individuals. For the first generation, random trees are created by using Koza's well-known ramped-half-and-half method [14]. The framework of GP for classification problems is described as Algorithm 1.

Algorithm 1. *GP*(M,D).

Require: $M \geq 0 \vee D \neq null$
1: M is the maximum generation
2: D is the data set
Ensure: *GP*
3: Let $gen = 0$
4: Initialize the population using the ramped-half-and-half method

5: **while** $gen \leq M$ **do**
6: Evaluate fitness of each individual
7: Update the best individual
8: Survival Selection + Crossover Operation
9: Mutation Operation
10: gen=gen + 1
11: **end while**

4.1.1. Tree-based individuals for classification

Discriminant functions, classification rules, and decision trees are three common tree-based classifier structures that can be synthesized with GP. If an n-class classifier is to be found, either $n-1$ discriminant functions or classification rules or $n-1$ thresholds for a single discriminant function are needed. Because of the lack of logical conjunctions such as *AND*, *OR*, *NOT*, off-the-shelf decision trees are highly redundant. Fig. 4 shows a genetic decision tree (GDT) [34] that combines a decision trees and classification rules. Besides, a backus normal form (BNF) grammar [35] for GDTs can be defined, which makes it more convenient to generate GDT individuals with GP. In a GDT, *Ai* and *Cj* are the index of features and index of labels in the input data set, respectively, where $0 \leq i \leq |A|$, $|A|$ is number of features and $0 \leq j \leq |C|$, $|C|$ is number of labels (Fig. 5).

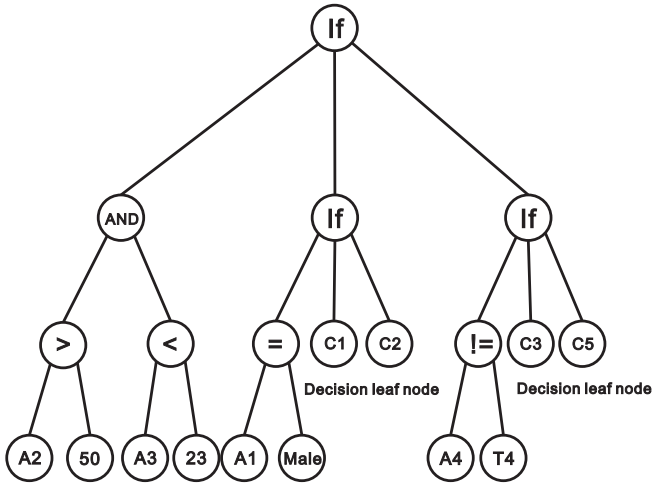

Fig. 4. Genetic decision tree for classification.

4.2. Multiobjective genetic programming

In this section, we will introduce multiobjective genetic programming to maximize the ROC performance in classification problems. There exist many efficient evolutionary multiobjective algorithms invented to solve continuous function optimization problems. It is, however, unclear which one is more suitable for GP for classification problems. Therefore, we conduct an in-depth study to identify the most appropriate framework.

4.2.1. Evolutionary multiobjective algorithms

We take four EMOAs into consideration: NSGA-II [30], MOEA/D [31], SMS-EMOA [36], and AGEMOA [33]. The most important factor in EMOAs is the strategy used to rank the individuals in the population, in other words, the survival mechanism. The reason of why we choose these EMOAs is that all of them adopt different multiobjective frameworks which employ different metrics in trading off conflicting objectives. Since we do not know which is best for ROCCH maximization, we aim for maximizing the diversity in our experiments.

A fast and elitist multiobjective genetic algorithm, which based on nondominated sorting, called NSGA-II, is introduced in [30]. The main contribution of NSGA-II is that it defines a method to rank the individuals by dominance depth and crowding distance. NSGA-II is used here as representative of algorithms that use dominance relationships, i.e., that mainly focus on the dominance count and rank. There are other algorithms such as SPEA [37] and SPEA-II [38] with roughly similar structure.

Multiobjective evolutionary algorithm based on decomposition (MOEA/D) [31] decomposes a multiobjective optimization problem into a number of scalar optimization subproblems and optimizes them simultaneously. The basic idea of MOEA/D is that one solution for a subproblem can use the information provided by its neighbors to improve its performance. Because of that

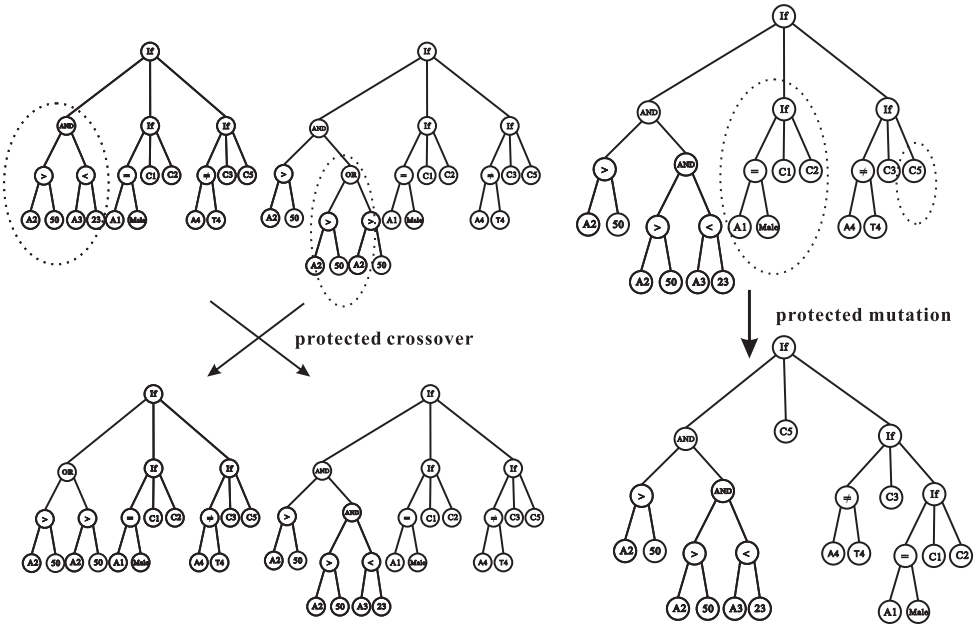

Fig. 5. Tree based crossover is shown on the left side, two subtrees are selected and swapped between different two individuals. The right side shows the mutation operator which swaps two subtrees from the same individual.

character, MOEA/D has a lower computational complexity than NSGA-II.

The hypervolume is a quality metric frequently used in evolutionary approaches to multiobjective optimization problems. SMS-EMOA [36] applies a multiobjective selection mechanism based on the hypervolume measure combined with the concept of nondominated sorting. An individual will survive with a higher probability if it makes higher contribution to the hypervolume covered by the current estimate of the Pareto front.

Bringmann et al. [33] pointed out that the dominance relation and other measures were used to ensure diversity in objective space but are not guided by a formal notion of approximation. In that work, they proposed a measure (approximate distance) to define the additive approximation of one nondominated set to the target set. The selection mechanism in evolutionary optimization is that an individual will survive with higher probability when it has higher contribution to reduce the approximate distance.

4.2.2. Operators used in MOGP

(1) *Tree based crossover and mutation*: First of all, we will outline the search operators used in GP, i.e., the unary mutation operation and the binary crossover operator [15,14,29]. In the crossover operator, one subtree is random selected from each of the two parent solutions. The selected subtrees are then swapped. The most commonly used mutation operation randomly selects a node in a tree and then substitutes the subtree rooted there with a randomly generated one. Another possible mutation operation randomly selects two subtrees from a parent and swaps them. Though tree based mutation operators have been used often in GP works, they may not work very efficiently in classification problems because randomly re-generated subtree-based mutation ignores the information obtained by its parent which is a trained individual. We will introduce tailor-made operators for this domain in the following.

(2) *Decision tree-based local search strategies*: In metaheuristic optimization, it is common to characterize operations as either exploitation (search focused around the currently best known solutions) or exploration (search that visits points in areas of the search space that more distant from the currently investigated points) [39]. Exploration and exploitation are both emphasized in evolutionary algorithms. Crossover and mutation are usually used to explore the search space, they guide the search from one area to another one with a large step. However, to improve the search result, exploitation in a local area around good solution is needed as well.

In GP for classification problems, one classifier divides the instance space into several subspaces that may contain positive and negative instances. The classifier is perfect when all the subspaces contain instances with only one label (positive or negative). We design two types of local search for GP for classification problems, shifting operators and splitting operators are described as follows:

(2.1) *Shifting operator*: The right-hand side of Fig. 6 shows a GDT and the corresponding hyperplane based classification in the left-hand side. The shifting operator improves the performance of the classifier by shifting the hyperplane, which corresponds to threshold adjusting in GDT. In multiobjective optimization problems, it is not easy to improve a classifier to a better classifier which dominates the old one. The dominance relation is a very intensive and rigorous relationship. Therefore, the question of how to define whether an application of the shifting operator was successful or not arises. We choose to use the information gain to measure the improvement. Eqs. (5) and (6) define the weighted sum of the information gain [40]. A successful application of the shifting operator to a classifier x increasing its information gain $E(x)$. In Eq. (5), $P(l)[k]$ and $p(l,k)$ are the number and the probability of instances with label k in the lth decision node

$$p(l,k) = \frac{P(l)[k]}{\sum_{i=1}^{2} P(l)[i]} \tag{5}$$

$$E(x) = \frac{\sum_{\forall leaves\ l \in x}(1 + \sum_{k=1}^{2} p(l,k)\log_2 p(l,k))(\sum_{k=1}^{2} P(l)[k]-1)}{|All\ instances| - |leaves \in x|} \tag{6}$$

(2.2) *Splitting operator*: The splitting operator [40] is another type of local search operator. Different from the shifting operator, it pays more attention to one subspace. In the left side of Fig. 6, the shifting operator cannot make every space "pure" if there are only two hyperplanes. The splitting operator can search a new hyperplane to divide this space into two sub-subspaces, as shown in the left side of Fig. 7. The splitting operator can work well to make two pure subspaces. In this paper, one subspace which is not pure (e.g., information gain < 0.1) will be considered for the splitting operator with a probability equal to the number of instances in this subspace divided by the total number of instances.

As mention in the discussion on the shifting operator, it not necessary to improve the performance of a classifier to a better

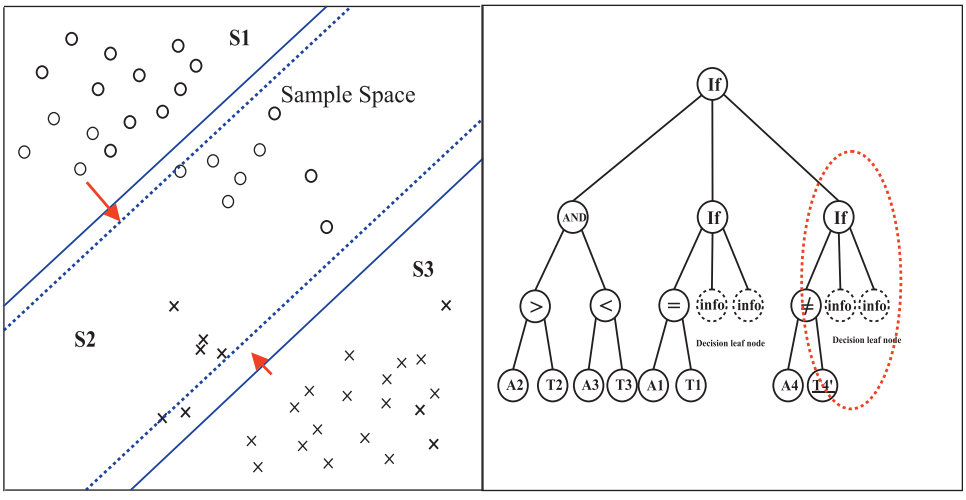

Fig. 6. Shifting operator is done on a genetic decision tree.

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054

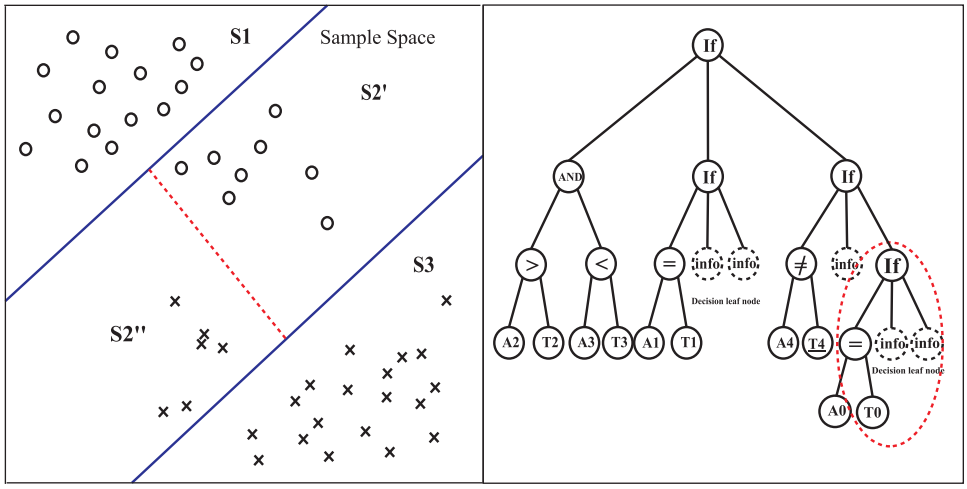

Fig. 7. Splitting operator is done on a genetic decision tree.

Table 1
27 UCI data sets.

Data set	No. of features	Class distribution	Data set	No. of features	Class distribution	Data set	No. of features	Class distribution
australian	14	383:307	*house-votes*	16	168:267	*pima*	8	268:500
bands	36	228:312	*hypothyroid*	25	151:3012	*wpbc*	33	151:47
bcw	9	458:241	*ionosphere*	34	225:126	*sonar*	60	97:111
crx	15	307:383	*kr-vs-kp*	36	1669:1527	*spambase*	57	1813:2788
euthyroid	25	293:2870	*mammographic*	5	445:516	*spect*	22	212:55
german	24	700:300	*monks-1*	6	216:216	*spectf*	44	212:55
haberman	3	225:81	*monks-2*	6	290:142	*tic-tac-toe*	9	626:332
hill-valley	100	600:612	*monks-3*	6	228:204	*transfusion*	4	178:570
parkinsons	22	147:48	*mushroom*	22	3916:4208	*wdbc*	30	212:357

one which dominates the old one. The success of the splitting operator can be defined as follows: suppose one space S contains p positive instances and n negative instances. If the splitting operator is applied on this space, there are two subspaces. The first subspace $S1$ contains $p1$ positive instances and $p2$ negative instances. The second subspace $S2$ holds $p-p1$ positive instances and $n-n1$ negative instances. The information gain improvement is $InfoGain(S)$ is defined as Eq. (8). $InfoGain(S)$ is larger than 0, the operation is a success

$$Info(S) = -\left(\frac{p}{n+p}\log_2\frac{p}{n+p} + \frac{n}{n+p}\log_2\frac{n}{n+p}\right) \tag{7}$$

$$InfoGain(S) = \frac{p1+n1}{n+p}Info(S1) + \frac{n+p-n1-p1}{n+p}Info(S2) \tag{8}$$

4.2.3. MOGP for classification problems

In our first set of experiments, four simple MOGP algorithms with tree-based crossover and mutation (but without local search) are used. We will therefore refer to them as S-NSGP-II, S-MOGP/D, S-SMS-EMOA, and S-AG-EMOA. Additionally, we extend these four simple MOGP methods with the local search algorithms. These algorithms are defined in Algorithms 2– 9 and we name them NSGP-II, MOGP/D, SMS-MOGP, and AG-MOGP. All algorithms work similar to the frameworks of their original version, but differ from them in two key points: first, in order to generate the initial population, we adopt the ramped-half-and-half method [14]. Second, simple-version MOGP methods use tree-based crossover and mutation into their evolutionary process. MOGP approaches with local search utilize local shifting and splitting operators to improve the performance of individuals. These differences will be underlined in the algorithm pseudo-codes. The inputs for these algorithms are classification data sets and the results are approximations of nondominated set, which in turn, are approximated ROCCHs.

5. Experimental studies

In the following, we will describe the data sets and give the configurations for different algorithms. With our experiments, we consider four important questions. First, we want to verify whether local search operators can work well to improve the performance of different MOGP methods for maximizing the ROCCH. Second, we want to see whether multiobjective optimization frameworks actually work better than single-objective algorithms. Third, we want to know whether MOGP approaches with local search have advantages on ROCCH maximization compared with traditional machine learning algorithms such as Naive Bayes (NB), C4.5 and PRIE which is good at constructing classifiers to maximize ROCCH. The last question is to find which multiobjective optimization framework is better for classification problems and why.

5.1. Data sets

Twenty-seven data sets are selected from the UCI repository [41] and described in Table 1. In this paper, we focus on binary classification problems, so all the data sets are two-class problems. Balanced and imbalanced benchmark data sets are

carefully selected. The scale in terms of the number of instances of these data sets ranges from hundreds to thousands.

5.2. Algorithms involved

Besides the eight MOGP approaches already mentioned, we also apply four single-objective genetic programming methods: *FGP* [34], *GGP* [40], and *EGP* [40] that use the framework of GP for classification in Algorithm 1. The fitness function is the only difference amongst them. *MGP* [40] also employs the two local search operators as described above. Table 2 lists these algorithms and their characteristics.

5.3. Validation and configuration

5.3.1. Cross-validation

To verify the generalization performances of different classifiers produced by different algorithms, cross-validation is employed. We apply each algorithm on each 27 data sets with five-fold cross-validation for 20 times. The different GP approaches as well as the MOGP algorithms have different convergence speed. In this work, the generation limit M for each algorithm *Alg* on the different data sets *Data* is defined as follows. The train part *Train* of the five-fold cross-validation on *Data* is taken and *Alg* is applied to *Train* by five-fold cross-validation for one time. We then set M to the generation index at which *Alg* had the best performance.

5.3.2. Configurations

In Table 3, we provide the parameter configurations for all algorithms in Table 2. In this work, we take the representation from [34] called GDT as the individual in the above eight new multiobjective genetic programming algorithms. For binary classification problems, 0 and 1 (standing for negative and positive) are selected as the terminals of GP. Every classifier (individual) is constructed as *if–then–else* tree which involves *and*, *or*, *not*, $>$, $<$ and $=$ as operator symbols. Most offspring individuals are obtained by the crossover operator with probability 0.9. The mutation, shifting, and splitting operators are applied with probability 0.1. Tournament selection is adopted as the selection strategy and the tournament size is set to 4. To avoid overfitting, the maximum depth of each individual tree is limited to 17.

5.4. Results

Our experiments are composed of three parts. First, to verify the effectiveness of local search in MOGP methods, we list the results (area under the ROCCH) of MOGP approaches without and with local search. In the second part of this discussion, we show that a multiobjective genetic programming framework with local search

Table 2
15 Algorithms.

Algorithm name	Character
S-NSGP-II	$p_{sf}=p_{sp}=0$ (without local search)
S-MOGP/D	$p_{sf}=p_{sp}=0$ (without local search)
S-SMS-MOGP	$p_{sf}=p_{sp}=0$ (without local search)
S-AG-MOGP	$p_{sf}=p_{sp}=0$ (without local search)
NSGP-II	$p_{sf}\neq 0, p_{sp}\neq 0$ (with local search)
MOGP/D	$p_{sf}\neq 0, p_{sp}\neq 0$ (with local search)
SMS-MOGP	$p_{sf}\neq 0, p_{sp}\neq 0$ (with local search)
AG-MOGP	$p_{sf}\neq 0, p_{sp}\neq 0$ (with local search)
FGP	Cost sensitive based fitness function
GGP	G-mean based fitness function
EGP	Entropy based fitness function
MGP	EGP + local search operators
C4.5	As described in [42]
Naive Bayes	As described in [43]
Prie [13]	The state of the art of machine learning algorithm to construct classifiers for ROCCH maximization

p_{sf} and p_{sp} are the probabilities of shifting operator and splitting operator.

Table 3
Parameters for 15 algorithms.

	Objective	Maximize ROCCH	
Terminals of GP	{0,1} with 1 representing "Positive"; 0 representing "Negative"	Function set of GP	If–then–else, and, or, not, $>$, $<$,$=$.
Data sets	27 UCI data sets	Algorithms	15 algorithms in Table 2
Crossover rate	0.9	Mutation rate	0.1
Shifting rate	0.1	Splitting rate	0.1
Parameters for GP	P (population size)=100; G (maximum evaluation times)=M Number of runs: 5-fold cross-validation 20 times	Termination criterion	Maximum of G of evaluation time has been reached
Selection strategy	Tournament selection, Size=4	Max depth of initial/inprocess individual program	3/17

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054

can work better than single-objective genetic programming without or with local search. Finally, traditional machine learning algorithms are compared to MOGP with local search to certify the efficiency of our algorithms in maximizing the ROC performance.

5.4.1. Results of MOGP without and with local search

Table 4 describes the performance of MOGP methods with and without local search on 27 UCI data sets. In the second column of Table 4, the first sub-column shows the results of NSGP-II without local search (named S-NSGP-II) and the results of NSGP-II with local search (named NSGP-II) are shown in the second sub-column. A number is printed in bold face if the results are statistically significantly better than the other variant of the same algorithm according to the Wilcoxon sum-rank tests [44] with a confidence level of 0.95. The results of MOGP/D, SMS-MOGP and AG-MOGP with and without local search are described in the third, fourth, and fifth columns. The last row of Table 4 gives the number of wins, the number of draws, and the number of losses for all data sets.

From this table we can see that S-NSGP-II wins never, loses 22 times to NSGP-II, and does not perform different in five of the 27 UCI data sets. This means that the local search operator works well and improves the performance of S-NSGP-II. Additionally, MOGP/D wins against S-MOGP/D on all data sets, SMS-MOGP and AG-MOGP lose only one time against their version without local search operators. Table 4 therefore testifies the effectiveness of the local search in MOGP approaches for maximizing the ROC performance.

5.4.2. Results of single-objective GP and traditional machine learning algorithms

In this subsection, we present all results of the four single-objective genetic programming algorithms [40] on 27 data sets. In Table 5, the first column gives the names of the data sets involved and the second to the fifth column the results (mean and standard deviation are multiplied by 100). The last four rows list the outcomes of Wilcoxon rank-sum tests of all seven tested approaches versus the four MOGP methods with local search. In Table 6, we list the outcomes of the Wilcoxon rank-sum test applied to the results of single- and multi-objective genetic programming algorithms. As FGP and GGP have the same representation (GDT) with the MOGP methods without local search, the main different factor is the multi-objective strategy. At the same time, the main difference between the MGP and MOGP methods with local search is the multiobjective strategy. From this table, it becomes obvious that multiobjective strategies in genetic programming are able to improve the ROC performance in classification problems.

As described in Table 5, the second to fifth columns report the results of the four single-objective genetic programming algorithms on 27 data sets. From the sixth to the last column, we list three traditional machine learning algorithms NB, C4.5, and PRIE which is the state of art in ROCCH maximization. In the last four rows, the Wilcoxon rank-sum test is used to compare their results with those of the MOGP methods with local search. Taking 21−6−0 as example, AG-MOGP wins 21 times, losses 0 time, and six times scores equal against EGP. Obviously, AG-MOGP is far better than EGP in ROCCH maximization on the data sets we used. From the last four Wilcoxon rank-sum test results, it is clear that multiobjective GP strategies with local search outperform significantly the four single-objective genetic programming algorithms and the three standard machine learning algorithms.

5.4.3. Analysis on MOGP for maximizing the ROC performance

Table 7 shows the Wilcoxon rank-sum test results among the MOGP methods with and without local search. Here, we use Fig. 8 to illustrate the relationship of these four MOGP approaches with local search. The algorithm at the head of arrow is better than the one at the end of arrow, and the results show that NSGP-II is the best algorithm among the tested ones. At the same time, MOGP/D is slightly better than SMS-MOGP and AG-MOGP which have a roughly similar performance on all 27 data sets. Fig. 9 describes

Table 4
Performance of MOGP methods without and with local search on UCI data sets, mean and standard deviation, multiplied by 100, are given in this table.

Data set	S-NSGP-II	NSGP-II	S-MOGP/D	MOGP/D	S-SMS-MOGP	SMS-MOGP	S-AG-MOGP	AG-MOGP
australian	90.93 ± 2.52	**92.00 ± 2.46**	88.09 ± 5.37	**91.68 ± 2.44**	89.13 ± 4.21	**92.02 ± 2.33**	90.39 ± 3.03	**91.05 ± 9.49**
bands	71.71 ± 5.43	**77.70 ± 3.49**	69.05 ± 4.47	**76.47 ± 4.05**	68.51 ± 4.28	**75.52 ± 3.71**	73.12 ± 4.91	**77.02 ± 3.70**
bcw	98.12 ± 0.80	98.19 ± 0.99	97.71 ± 1.33	**98.07 ± 1.13**	97.04 ± 1.83	97.95 ± 1.14	97.92 ± 1.21	97.25 ± 9.87
crx	90.18 ± 3.12	**91.79 ± 2.47**	89.53 ± 4.88	**91.58 ± 2.32**	89.34 ± 4.56	**91.65 ± 2.21**	90.71 ± 3.09	**90.88 ± 9.47**
euthyroid	79.27 ± 9.20	**96.78 ± 1.37**	72.46 ± 10.39	**94.47 ± 6.91**	75.05 ± 10.51	**96.49 ± 1.32**	80.19 ± 8.83	**96.34 ± 7.45**
german	73.00 ± 3.94	**74.03 ± 2.81**	68.08 ± 5.35	**73.52 ± 2.97**	71.75 ± 4.23	**73.68 ± 2.56**	72.64 ± 3.51	**73.11 ± 7.82**
haberman	65.55 ± 6.60	**67.08 ± 6.19**	63.68 ± 7.17	**66.60 ± 6.58**	65.03 ± 7.48	**65.50 ± 7.12**	66.88 ± 6.56	66.27 ± 9.44
hill-valley	50.30 ± 1.47	**53.19 ± 2.61**	50.07 ± 1.47	**53.02 ± 2.59**	50.58 ± 1.37	**52.65 ± 2.61**	50.30 ± 1.32	**52.42 ± 5.95**
house-votes	97.01 ± 3.82	98.10 ± 1.39	96.50 ± 2.87	**97.84 ± 1.46**	96.95 ± 2.42	**98.13 ± 1.23**	97.75 ± 1.58	96.94 ± 9.89
hypothyroid	79.63 ± 11.60	**97.99 ± 1.52**	77.41 ± 14.60	**97.11 ± 2.06**	81.54 ± 12.99	**97.77 ± 1.54**	90.06 ± 11.45	**98.13 ± 10.28**
ionosphere	86.81 ± 6.76	**91.83 ± 3.98**	84.61 ± 6.73	**91.42 ± 3.56**	84.89 ± 5.74	**90.37 ± 5.02**	87.46 ± 4.86	**90.67 ± 9.97**
kr-vs-kp	88.67 ± 7.32	**98.01 ± 0.85**	80.41 ± 8.04	**98.12 ± 0.99**	83.80 ± 6.48	**98.26 ± 0.93**	87.20 ± 7.35	**97.16 ± 9.84**
mammographic	89.08 ± 2.05	89.79 ± 1.80	87.71 ± 2.52	**89.45 ± 2.00**	88.17 ± 2.18	**89.23 ± 1.84**	89.14 ± 1.95	88.55 ± 9.11
monks-1	94.80 ± 3.43	**99.93 ± 0.53**	88.75 ± 11.39	**99.45 ± 1.97**	83.75 ± 12.99	**97.95 ± 3.88**	94.20 ± 5.97	**97.48 ± 10.18**
monks-2	77.65 ± 9.50	**93.60 ± 5.25**	68.18 ± 10.74	**89.82 ± 16.76**	69.57 ± 10.18	**85.44 ± 6.85**	76.01 ± 9.69	**82.80 ± 10.11**
monks-3	98.22 ± 4.26	**100.00 ± 0.00**	94.51 ± 9.50	**99.84 ± 0.45**	92.52 ± 9.99	**98.60 ± 5.08**	99.52 ± 0.49	98.76 ± 9.99
mushroom	98.70 ± 1.61	**99.95 ± 0.10**	96.93 ± 3.15	**99.77 ± 0.30**	97.09 ± 2.71	**99.79 ± 0.36**	98.59 ± 2.45	**99.35 ± 3.07**
parkinsons	85.09 ± 6.58	86.17 ± 5.96	80.87 ± 8.00	**86.96 ± 5.02**	81.76 ± 2.32	**85.82 ± 5.81**	83.08 ± 6.22	**84.78 ± 10.20**
pima	77.22 ± 3.52	**80.61 ± 3.21**	72.54 ± 5.07	**80.35 ± 2.86**	75.20 ± 4.12	**80.11 ± 3.07**	77.97 ± 3.62	**79.16 ± 8.54**
sonar	70.42 ± 6.01	**80.09 ± 5.55**	67.51 ± 7.43	**79.68 ± 6.05**	69.34 ± 7.97	**79.38 ± 5.93**	71.74 ± 6.48	**78.04 ± 9.39**
spambase	70.97 ± 8.55	**96.36 ± 0.57**	64.17 ± 7.67	**95.80 ± 0.60**	64.85 ± 8.94	**96.04 ± 0.63**	93.38 ± 0.92	**95.02 ± 9.61**
spect	75.47 ± 5.05	76.52 ± 6.91	73.90 ± 8.34	**76.97 ± 7.85**	73.19 ± 7.39	**76.00 ± 7.11**	75.21 ± 7.27	75.20 ± 10.19
spectf	68.30 ± 5.95	**73.38 ± 5.55**	66.43 ± 8.58	**73.58 ± 5.65**	68.38 ± 7.47	**75.29 ± 6.31**	71.52 ± 6.52	**74.90 ± 9.69**
tic-tac-toe	73.39 ± 8.99	**86.19 ± 11.46**	67.52 ± 11.04	**84.18 ± 9.06**	69.12 ± 11.85	**77.06 ± 4.28**	71.31 ± 10.70	**75.90 ± 12.95**
transfusion	68.97 ± 4.89	**72.12 ± 4.44**	64.94 ± 4.75	**71.88 ± 4.63**	67.04 ± 5.45	**71.94 ± 2.58**	68.40 ± 5.17	**70.98 ± 8.43**
wdbc	93.52 ± 4.95	**97.28 ± 1.49**	92.42 ± 4.73	**97.02 ± 1.63**	92.91 ± 3.95	**97.08 ± 1.83**	94.14 ± 3.08	**96.23 ± 9.86**
wpbc	59.52 ± 8.15	**67.41 ± 8.33**	59.42 ± 7.76	**66.61 ± 7.41**	61.00 ± 8.36	**66.92 ± 8.88**	60.94 ± 8.36	**66.04 ± 9.85**
Win–draw–loss	0–5–22	22–5–0	0–0–27	27–0–0	0–1–26	26–1–0	0–6–21	21–6–0

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054

Table 5
Performance of single-objective genetic programming and traditional machine learning algorithms on UCI data sets.

Data set	EGP	FGP	GGP	MGP	NB	Prie	C4.5
australian	90.05 ± 3.06	85.56 ± 4.87	85.54 ± 3.83	90.66 ± 2.68	89.47 ± 2.78	91.75 ± 2.36	85.52 ± 4.05
bands	70.04 ± 5.05	53.99 ± 5.56	64.88 ± 4.89	76.18 ± 4.97	73.91 ± 4.68	76.07 ± 4.81	74.65 ± 0.00
bcw	97.35 ± 1.37	93.73 ± 2.11	93.85 ± 2.45	97.23 ± 1.52	98.92 ± 0.62	98.16 ± 1.09	95.05 ± 2.55
crx	90.68 ± 2.49	85.91 ± 3.57	86.36 ± 3.32	90.75 ± 2.53	87.88 ± 3.16	90.65 ± 2.77	85.51 ± 0.00
euthyroid	93.37 ± 5.81	50.01 ± 0.11	79.41 ± 13.12	97.47 ± 1.41	91.91 ± 2.03	96.24 ± 1.31	92.97 ± 2.49
german	70.81 ± 3.42	51.75 ± 3.51	67.14 ± 5.36	71.69 ± 3.30	78.42 ± 2.94	75.95 ± 3.25	65.36 ± 0.00
haberman	62.97 ± 7.63	50.66 ± 4.25	63.98 ± 6.68	64.14 ± 7.69	65.00 ± 7.19	69.58 ± 7.26	63.96 ± 0.00
hill-valley	50.18 ± 2.15	50.09 ± 1.39	49.90 ± 3.25	53.34 ± 3.00	50.64 ± 3.65	51.82 ± 3.93	50.00 ± 0.00
house-votes	97.75 ± 1.63	94.63 ± 4.00	95.23 ± 3.85	97.74 ± 1.62	98.05 ± 1.04	97.80 ± 1.49	96.35 ± 2.04
hypothyroid	96.55 ± 2.55	52.35 ± 3.27	93.45 ± 5.91	98.19 ± 1.77	98.02 ± 1.52	96.51 ± 2.45	95.56 ± 3.23
ionosphere	87.22 ± 5.84	80.87 ± 7.60	79.86 ± 7.01	90.09 ± 4.76	93.57 ± 3.18	93.68 ± 4.23	88.20 ± 5.65
kr-vs-kp	85.71 ± 6.65	62.16 ± 8.52	71.89 ± 6.02	98.44 ± 1.06	93.21 ± 1.00	98.26 ± 0.44	99.71 ± 0.23
mammographic	88.96 ± 1.97	82.76 ± 3.60	84.73 ± 3.46	88.68 ± 2.24	89.77 ± 1.96	89.70 ± 2.02	87.66 ± 0.00
monks-1	85.96 ± 11.96	51.21 ± 9.96	75.03 ± 5.25	99.64 ± 1.66	73.18 ± 4.58	70.93 ± 5.59	75.22 ± 0.00
monks-2	80.48 ± 12.05	50.01 ± 6.29	53.28 ± 6.92	94.76 ± 4.88	52.38 ± 7.04	51.25 ± 6.16	94.17 ± 5.93
monks-3	99.59 ± 0.49	87.48 ± 10.72	86.75 ± 9.04	99.90 ± 0.29	95.94 ± 2.17	99.60 ± 0.27	100.00 ± 0.00
mushroom	98.68 ± 1.88	84.67 ± 8.24	89.44 ± 4.47	99.95 ± 0.13	92.60 ± 0.71	99.49 ± 0.14	100.00 ± 0.00
parkinsons	81.92 ± 7.80	76.62 ± 8.22	75.97 ± 7.19	85.87 ± 7.20	85.91 ± 6.11	88.24 ± 5.83	78.91 ± 9.76
pima	76.27 ± 4.94	50.88 ± 1.29	70.73 ± 3.44	78.77 ± 3.71	81.40 ± 3.01	79.58 ± 2.92	75.23 ± 4.93
sonar	73.33 ± 7.01	54.17 ± 6.81	68.22 ± 7.38	77.59 ± 7.57	80.12 ± 7.03	69.92 ± 8.64	73.85 ± 7.84
spambase	85.28 ± 5.53	76.14 ± 7.16	76.58 ± 4.30	94.79 ± 1.04	93.98 ± 0.69	96.72 ± 0.56	93.72 ± 0.00
spect	74.36 ± 7.01	68.21 ± 10.68	71.99 ± 7.18	75.33 ± 8.59	84.09 ± 6.03	83.51 ± 7.01	76.88 ± 8.91
spectf	71.76 ± 7.04	58.69 ± 9.06	69.16 ± 7.16	73.10 ± 8.45	84.94 ± 5.19	78.90 ± 6.37	63.36 ± 9.07
tic-tac-toe	71.89 ± 12.11	63.35 ± 9.73	63.35 ± 10.15	90.04 ± 10.24	61.52 ± 14.76	70.41 ± 12.51	84.91 ± 13.91
transfusion	71.31 ± 5.21	50.48 ± 0.89	67.46 ± 4.37	71.31 ± 4.88	70.93 ± 4.94	70.87 ± 5.39	71.08 ± 5.08
wdbc	95.12 ± 2.92	87.25 ± 4.54	90.39 ± 2.83	96.05 ± 1.92	98.14 ± 1.33	96.58 ± 1.94	92.74 ± 3.16
wpbc	66.83 ± 9.90	56.47 ± 7.41	60.15 ± 8.92	64.21 ± 10.66	66.42 ± 8.85	68.22 ± 9.25	58.19 ± 10.77
NSGP-II	23–4–0	27–0–0	27–0–0	15–9–3	13–8–6	13–7–7	20–5–2
MOGP/D	22–5–0	27–0–0	27–0–0	10–11–6	12–8–7	9–11–7	20–2–5
SMS-MOGP	21–6–0	27–0–0	27–0–0	9–10–8	12–8–7	9–11–7	20–2–5
AG-MOGP	21–6–0	27–0–0	26–1–0	9–10–8	14–6–7	11–9–7	20–2–5

Table 6
Wilcoxon rank-sum test results for MOGP methods without local search and single-objective genetic programming. FGP, EGP and four MOGP methods without local search have the same representation. MGP and four local search-based MOGP approaches take shifting and splitting operators into their search strategies.

Algorithms	FGP	GGP		MGP
S-NSGP-II	26–0–1	20–5–1	NSGP-II	15–9–3
S-MOGP/D	25–1–1	16–6–5	MOGP/D	10–11–6
S-SMS-MOGP	26–0–1	18–6–3	SMS-MOGP	9–10–8
S-AG-MOGP	26–1–0	22–5–0	AG-MOGP	9–10–8

Table 7
Wilcoxon rank-sum test results for MOGP methods with and without local search.

Algorithms	NSGP-II	MOGP/D	SMS-MOGP	AG-MOGP
NSGP-II	–	9–18–0	9–16–2	6–20–1
MOGP/D	0–18–9	–	3–20–4	4–20–3
SMS-MOGP	2–16–9	3–20–4	–	2–23–2
AG-MOGP	1–20–6	3–20–4	2–23–2	–
Algorithms	S-AGE-MOGP	S-MOGP/D	S-SMS-MOGP	S-NSGP-II
S-AG-MOGP	–	22–5–0	19–8–0	5–20–2
S-MOGP/D	0–5–22	–	3–18–6	0–8–19
S-SMS-MOGP	0–8–19	6–18–3	–	0–9–18
S-NSGP-II	2–20–5	19–8–0	18–9–0	–

the relationship of these MOGP approaches without local search. Here, S-AG-MOGP is the best algorithms among four methods and it is slightly better than S-NSGP-II which is quite a lot better than S-SMS-MOGP. S-MOGP/D is the worst algorithm. First of all, it should be emphasized that NSGP-II is the best algorithm of all approaches, with and without local search.

There are two factors affecting the performance of the tested algorithms, one is the different ranking mechanisms used in multiobjective optimization algorithms, the other is that the local search operators have different efficacy when they are introduced into different EMOAs. Comparing Fig. 9 with Fig. 8, we can find that NSGP-II and MOGP/D are improved to the first and the second place. This means local search operators work well in multiobjective optimization frameworks of NSGA-II and MOEA/D, and not as good in SMS-EMOA and AG-EMOA.

More evidence can be found in Table 10, which shows the total time cost in seconds of all MOGP methods and the difference of the approaches with and without local search. Local search

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054

operators seem to consume more time in MOGP/D framework and the second longest time in NSGP-II framework. There are no huge differences of time cost in the SMS-MOGP and AG-MOGP frameworks. The reason for this is that the SMS-MOGP and AG-MOGP frameworks are very greedy strategies. As outline in Section 4.2.1, the contribution of each individual to their target metrics (hypervolume or approximate distance) is directly used into select mechanism. Additionally, the hypervolume is similar to the area under the ROCCH and minimizing the approximated distance is also very similar to maximizing the hypervolume. As SMS-MOGP and AG-MOGP are greedy at searching the maximum ROCCH, it becomes harder to escape a local optimum since the shifting and splitting operators only perform exploitation. Hence, these two operators cannot contribute much to SMS-MOGP and AG-MOGP.

NSGP-II, on the other hand, ranks the individuals by dominance level and crowding-distance and MOGP/D just compares individuals with others in their neighborhood. Both algorithms are not very greedy at selecting individuals to survive and the survivors can potentially be improved by the local search operators. From another perspective, SMS-MOGP and AG-MOGP have larger selection pressure than NSGP-II and MOGP/D in searching genetic decision trees to maximize the ROC performance.

The reason for why NSGP-II is better than MOGP/D is the discordancy of the genotype and phenotype of genetic programming: two genetic programming individuals can be very similar in decision space, but may have a very long distance in objective space. In MOGP/D, an offspring is produced by two parents in a neighborhood which is defined in the objective space, but it will not be in this neighborhood. This causes the framework of MOEA/D to not work well because it supposes that subproblems can be optimized by individuals in their neighborhood.

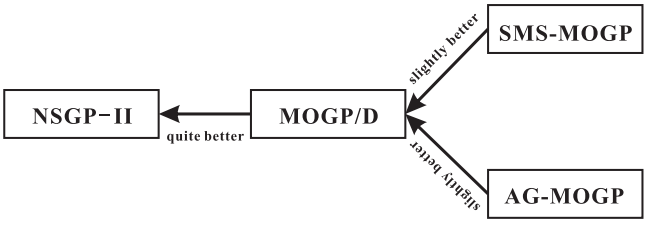

Fig. 8. Comparisons among MOGP methods with local search.

5.4.4. Time cost of all algorithms

Tables 8 and 9 report the time cost of all algorithms. The experiment environment is an 8 core CPU with 2.13 GHz and 24 GB RAM. Obviously, GP-based algorithms need much more time than traditional machine learning algorithms. Because of the metaheuristic character of EAs, GP needs to evaluate many classifiers until it converges. The Naive Bayes method, on the other hand, calculates an a posteriori probability and the C4.5 adopts uses a greedy method to increase information gain. PRIE employs a greedy strategy to construct classifiers (more than one, usually dozens of classifiers) to maximize the ROCCH, so it cost a little more time than NB and C4.5, but still much less than GP-based algorithms.

Among the single-objective GP algorithms, MGP costs much more time than the others (EGP, FGP, GGP). The reason is that local search operators exploit each individual. For the same reason, MOGP methods with local search will consume more time than their counterparts without local search.

Fig. 9. Comparisons among MOGP methods without local search.

Table 8
Time cost of MOGP algorithms in seconds.

Time (s)	S-NSGP-II	S-MOGP/D	S-SMS-MOGP	S-AG-MOGP	NSGP-II	MOGP/D	SMS-MOGP	AG-MOGP
australian	30.34	5.31	42.71	28.12	51.65	31.18	22.88	48.74
bands	81.84	122.68	24.55	109.3	70.65	117.45	24.25	122.61
bcw	4.96	5.51	9.82	11.69	3.06	7.24	5.03	15.66
crx	83.69	131.33	197.94	300.69	82.38	87.43	82.81	155.47
euthyroid	154.83	22.46	238.75	313.61	806.02	529.17	159.92	214
german	64.98	46	62.84	58.48	258.99	274.96	76.43	67.62
haberman	5.22	5.25	26.20	9.98	3.82	3.88	5.04	4.13
hill-valley	274.38	170.05	339.33	378.59	971.96	1204.87	606.45	543.85
house-votes	6.86	1.79	91.90	189.56	3.85	18.23	4.29	35.59
hypothyroid	607.35	1376.31	799.97	65.32	52.5	484.17	487.80	1569.87
ionosphere	12.07	3.98	16.32	20.74	28.5	19.28	8.96	82.59
kr-vs-kp	293.09	127.52	301.78	312.55	159.18	882.05	304.78	334.64
mammographic	15.22	11.92	24.44	33.58	34.95	98.57	17.74	104.11
monks-1	29.23	27.64	37.54	45.6	21.86	19.64	15.80	28.09
monks-2	134.2	121.59	138.32	142.11	318.01	489.36	162.39	208.77
monks-3	1.1	3.81	159.33	421.66	27.57	43.82	3.98	7.98
mushroom	857.63	698.38	682.44	491.46	607.59	718.55	662.13	805.04
parkinsons	1.55	1.09	1.96	2.44	2.2	2.22	2.71	1.44
pima	25.75	4.77	22.46	18.16	51.64	74.73	25.50	54.3
sonar	2.76	2.34	5.94	9.06	7.92	24.69	6.61	6.36
spambase	134.62	103.62	1189.80	2856.4	2910.43	3509.06	1711.28	1520.9
spect	7.71	2.53	13.40	14.79	32.66	21.06	6.29	6.99
spectf	1.1	1.08	3.33	5.81	2.12	1.97	6.79	10.33
tic-tac-toe	84.52	81.51	96.32	98.32	699.82	399.07	156.73	109.5
transfusion	4.71	4.36	4.52	4.69	36.69	41.49	7.05	4.67
wdbc	4.79	1.89	4.88	6.45	7.14	5.78	7.04	9.81
wpbc	0.7	3.84	1.20	1.68	2.5	4.33	1.54	1.88
Sum time	2925.20	3088.56	4538.00	5950.84	7255.66	9114.25	4582.20	6074.94

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054

Table 9
Time cost of single-objective genetic programming and machine learning algorithms.

Time(s)	EGP	FGP	GGP	MGP	NB	Prie	C4.5	Time (s)	EGP	FGP	GGP	MGP	NB	Prie	C4.5
australian	5.50	2.75	2.88	2.90	0.05	4.18	0.03	*bands*	5.50	2.50	2.70	121.26	0.09	15.85	0.08
bcw	10.93	13.55	13.14	6.77	0.03	0.53	0.01	*crx*	11.95	4.21	3.18	6.10	0.05	2.92	0.03
euthyroid	212.36	8.35	14.05	198.61	0.48	2.74	0.34	*german*	24.13	4.25	5.67	16.95	0.12	4.79	0.13
haberman	10.70	3.20	2.78	1.54	0.01	0.43	0.00	*hill-valley*	29.59	9.41	7.11	1507.34	0.42	380.82	0.08
house-votes	2.83	3.04	5.51	1.37	0.04	0.48	0.01	*hypothyroid*	79.04	47.37	48.36	45.38	0.50	3.20	0.20
ionosphere	20.76	142.17	13.91	15.67	0.05	5.77	0.04	*kr-vs-kp*	419.03	30.57	23.84	554.33	0.65	1.58	0.31
mammographic	20.74	7.64	5.00	46.07	0.03	0.87	0.02	*monks-1*	5.93	1.51	2.16	16.27	0.02	0.29	0.00
monks-2	200.47	20.76	14.98	59.35	0.01	0.30	0.01	*monks-3*	8.49	7.10	6.11	5.40	0.01	0.31	0.01
mushroom	1107.35	664.66	531.60	903.85	1.01	2.31	0.57	*parkinsons*	4.52	1.52	1.77	5.42	0.02	1.62	0.01
pima	30.50	20.73	12.10	20.81	0.03	16.46	0.02	*sonar*	5.90	5.77	5.56	10.12	0.05	31.45	0.03
spambase	90.32	65.99	16.67	2313.91	1.14	374.98	1.80	*spect*	4.92	2.91	1.04	9.35	0.03	0.39	0.02
spectf	9.71	8.76	2.88	8.62	0.05	3.81	0.03	*tic-tac-toe*	218.47	164.91	9.86	102.80	0.04	0.48	0.03
transfusion	33.79	93.27	3.48	6.71	0.02	4.34	0.01	*wdbc*	15.23	9.57	5.47	8.74	0.08	20.86	0.04
wpbc	7.77	1.96	1.36	3.04	0.02	7.89	0.04								

Table 10
Total time cost of local search operator in different MOGP methods.

Algorithms	Time cost		Time cost	Difference
S-AG-MOGP	5950.84	AG-MOGP	6074.94	124.10
S-MOGP/D	3088.56	MOGP/D	9114.25	6025.69
S-SMS-MOGP	4538.00	SMS-MOGP/D	4582.20	44.20
S-NSGP-II	2925.20	NSGP-II	7255.66	4330.46

6. Conclusion and future work

In this work, we first pointed out that ROCCH is very suitable to measure the ROC performance in binary classification, especially for especially if the class distribution is unknown or the misclassification cost are skew. Then, we discussed the relationship between ROCCH and the Pareto front in multiobjective optimization: both can be considered as analogous to each other.

Maximizing the ROCCH can be archived by using evolutionary multiobjective algorithms to search a group of nondominated solutions. Four different MO frameworks for synthesizing genetic decision trees are proposed: S-NSGP-II, S-MOGP/D, S-SMS-MOGP, and S-AG-MOGP, each employing a different fitness measure. We then proposed to use local search in genetic programming for classification problems. Two different local search operators called shifting and splitting are defined. They are introduced into the MOGP methods to improve the performance in searching Pareto front.

We found that these operators contribute differently in the different MOGP methods. NSGP-II with local search outperforms the other the MOGP algorithms both with and without local search. We furthermore compare the new approaches to single-objective genetic programming algorithms and traditionally machine learning algorithms and found that they perform very favorable. In conclusion, NSGP-II with local search is the best overall algorithms for ROCCH maximization.

As pointed out in Section 3.2, ROCCH is not the same as the Pareto front in multiobjective optimization. In this work, MOGP approaches are adopted to search a group of nondominated genetic decision trees to approximate the ROCCH and to maximize the area under the curve constructed by these nondominated solutions. Our plan for future work is to combine the concepts of ROCCH and Pareto front in a better way in order to derive new multiobjective evolutionary algorithms for maximizing the ROC performance.

In Section 5.4.1, we found that the contribution of the local search methods to improve the final results strongly depends on the MO framework. Therefore, further research should pay attention on how local search operators work in multiobjective scenarios. Generally, the concepts of memetic computing should be investigated more thoroughly in this domain, as they seemingly can lead to significant improvements in ROCCH maximization problems.

The last issue we want to tackle is the dissatisfying runtime of our methods. This is a disadvantage for EAs in general, but may be mitigated by using parallelization and new hardware such as GPUs.

Acknowledgment

This work was partially supported by the 973 Program of China (Grant no. 2011CB707006), National Natural Science Foundation of China (Grant nos. U0835002, 61028009 and 61175065), the National Natural Science Foundation of Anhui Province (No. 1108085J16), and the European Union Seventh Framework Programme under Grant agreements no. 247619.

Appendix A. MOGP algorithms

Algorithm 2. *fast-nondominated-sort*(P) [30].

Require: $P \neq null$
1: P is a solution set
Ensure: fast-nondominated-sort
2: **for** each $p \in P$ **do**
3: $S_p = \emptyset$
4: $n_p = 0$
5: **for** each $q \in P$ **do**
6: **if** $p \prec q$ **then**
7: $S_p = S_p \cup \{q\}$

```
    else {q ≺ p}
      n_p = n_p + 1
    end if
    end for
    if n_p = 0 then
      P_rank = 1
      F_1 = F_1 ∪ {p}
    end if
  end for
  while F_i ≠ ∅ do
    Q = ∅
    for each p ∈ F_i do
      for each q ∈ S_p do
        n_q = n_q − 1
        if n_q = 0 then
          q_rank = i + 1
          Q = Q ∪ {q}
        end if
      end for
    end for
    i = i + 1
    F_i = Q
  end while
```

Algorithm 3. *crowding-distance-assignment* (T) [30].

```
Require: T ≠ null
    T is a nondominated solution set
Ensure crowding-distance-assignment
    l = |T|
    for each i ∈ [1,l] do
      T[i]_distance = 0
    end for
    for each objective m do
      T = sort(T,m)
      T[1]_distance = T[l]_distance = ∞
      for i = 2 to l − 1 do
      T[i]_distance = T[i]_distance + (T[i+1].m − T[i−1].m)/(f_m^max − f_m^min)
      end for
    end for
```

Algorithm 4. *NSGP-II*(P,*Max*,N).

```
Require: Max ≥ 0 ∨ P = null ∨ N > 0
    Max is the maximum|evaluations
    P is the population
    N is the population size
Ensure: NSGA-II
    Let m = 0, t = 0
    Initialize the population P_t by ramped-half-and-half method
    Evaluate each individual in P_t and m = m + N
    while m ≤ Max do
      Generate offspring Q_t from P_t by tree-based crossover
      Shifting operator with probability p_sf
      Splitting operator with probability p_sp
      Evaluate each changed offspring in Q_t
      m = m + | changed-offspring |
      R_t = P_t ∪ Q_t
      F = fast- nondominated-sort (R_t)
      P_{t+1} = ∅ and i = 0
      while |P_{t+1}| + |F_i| ≤ N do
        crowding-distance-assignment(F_i)
        P_{t+1} = P_{t+1} ∪ F_i
        i = i + 1
```

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054

end while
Sort($\mathbf{F}_i$, $\prec_n$)
$P_{t+1} = P_{t+1} \cup \mathbf{F}_i[1 : (N - |P_{t+1}|)]$
$P = P_{t+1}$
$t = t+1$
end while
Return P

Algorithm 5. *MOGP/D*(*EP*,*P*,*N*,*M*,*T*).

Require: $EP = null \vee N > 0 \vee M \geq 2 \vee T > 0$
EP(External Population) is an archive to collect pareto-optimal solutions
P is the population contains *N* solutions $(x^1, \ldots, x^N)$ where x^i is the current solution for the *i*th subproblem
N is the population size and is also the number of subproblems in MOEA/D
M is the number of objectives
A uniform spread of N weight vectors: $\lambda^1, \ldots, \lambda^N, \lambda^i = (\lambda_1^i, \ldots, \lambda_M^i)$ *for* $1 \leq i \leq N$
T is the number of weight vectors in the neighborhood of each weight vector
Reference point $\mathbf{z}^*$
$FV^{(k)}$ is the F-value of x^i $1 \leq i \leq N$
Ensure: MOGP/D
Step 1) Initialization
Step 1.1) Set $EP = \emptyset$
Step 1.2) Compute the Euclidean distances between any two weight vectors and then work out the *T* closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, Set $B(i) = i_1, \ldots, i_T$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the *T* closest weight vectors to λ^i
Step 1.3) Generate an initial population $(x^1, \ldots x^N)$by ramped-half-and-half method
Step 2) Update
for $i = 1, \ldots, N$ **do**
Step 2.1) Reproduction: Randomly select two indexes *k*,*l* for *B*(*i*), and then generate a new solution *y* by x^k and x^l by using tree-based crossover operators
Step 2.2) Improvement: Apply shifting operator probability p_{sf} and splitting operator with probability p_{sp} on *y* to produce y'
Step 2.3) Update of neighboring solutions: For each index $j \in B(i)$, if $g^{te}(y' | \lambda^j, \mathbf{z}^*) \leq g^{te}(x^j | \lambda^j, \mathbf{z}^*)$, then set $x^j = y'$ and $FV^j = F(y')$
Step 2.4) Update of *EP*
Remove from *EP* all the vectors dominated by $F(y^i)$
Add FV' to *EP* if no vectors in *EP* dominate $F(y^i)$
end for
Step 3) Stopping Criteria: If stopping criteria is satisfied then stop and output *EP*. Otherwise, go to **Step 2**

Algorithm 6. *Reduce* (*Q*) [36].

Require: $Q \neq \emptyset$
Q is a solution set
Ensure: Reduce
$\Re_1, \ldots, \Re_v \leftarrow$ *fast-nondominated-sort*(*Q*)
if $v > 1$ **then**
$r \leftarrow argmax_{s \in \Re_v}[d(s,Q)]$
else
$r \leftarrow argmin_{s \in \Re_v}[\Delta_\varsigma(s, \Re_v)]$
end if
Return (Q $\{r\}$)

Algorithm 7. *SMS-MOGP* (*Max*,*N*).

Require: $Max > 0, N > 0$
Max is the maximum of evaluations
N is the population size
Ensure: SMS-MOGP
$P_0 = init()$ by ramped-half-and-half method
$t = 0$
$m = 0$
while $m < Max$ **do**
$q_{t+1} \leftarrow$ Shifting operator with probability p_{sf}
splitting operator with probability p_{sp} *are done on* P_t
$P_{t+1} \leftarrow Reduce(P_t \cup q_{t+1})$

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054

$t \leftarrow t+1$
$m \leftarrow m+1$
end while

Algorithm 8. *MeasureQuality*(A,P) [33].

Require: $A \neq \emptyset \vee P \neq \emptyset$
A Archive
P Population
Ensure: MeasureQuality
$S \leftarrow \emptyset$
for each $a \in A$ **do**
  $\delta \leftarrow \infty$
  for each $p \in P$ **do**
    $\rho \leftarrow -\infty$
    for $i \leftarrow d$ **do**
      $\rho \leftarrow max\{\rho, a_i - p_i\}$
    end for
    $\delta \leftarrow min\{\delta, \rho\}$
  end for
  $S \leftarrow S \cup \delta$
end for
sort S decreasingly
Return S

Algorithm 9. *AG-MOGP*(Max,A,P, μ, λ).

Require: $A = \emptyset \vee P = \emptyset \vee Max > 0 \vee \mu > 0 \vee \lambda > 0$
A Archive
P Population
Max is the maximum of evaluations
μ and λ are the size of parent and offspring population
Ensure AGEMOA
Initialize population P with μ by ramped-half-and-half method
Set archive $A \leftarrow P$, $m \leftarrow 0$
while $m < Max$ **do**
  Initialize offspring population $O \leftarrow \emptyset$
  for $j \leftarrow 1$ to λ **do**
    Select two random individuals from P
    Apply crossover operator
    Shifting operator with probability p_{sf}
    Splitting operator with probability p_{sp}
    Add new individual into O
  end for
  for each $p \in O$ **do**
    Insert offspring p in archive A
  end for
  Add offsprings to population, i.e., $P \leftarrow P \cup O$
  while $|P| > \mu$ **do**
    for each $p \in P$ **do**
      MeasureQuality(A, $P \backslash \{p\}$)
    end for
    Remove p from P for which $S_\alpha(A, P \backslash \{p\})$ is lexicographically smallest
  end while
end while

References

[1] T. Fawcett, An introduction to roc analysis, Pattern Recogn. Lett. 27 (8) (2006) 861–874.
[2] C.M. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag GmbH, Berlin, Germany, 2006.
[3] F. Provost, T. Fawcett, Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions, in: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Amer Assn for Artificial, 1997, pp. 43–48.
[4] J. Egan, Signal Detection Theory and ROC Analysis, Series in Cognition and Perception, Academic Press, 1975.
[5] H. Sox, M. Higgins, Medical Decision Making, Amer College of Physicians, 1988.
[6] A. Bradley, The use of the area under the roc curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (7) (1997) 1145–1159.
[7] F. Provost, T. Fawcett, Robust classification for imprecise environments, Mach. Learn. 42 (3) (2001) 203–231.

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054

[8] T. Fawcett, Using rule sets to maximize roc performance, in: Proceedings IEEE International Conference on Data Mining, 2001. ICDM 2001, IEEE, pp. 131–138.
[9] P. Flach, S. Wu, Repairing concavities in roc curves, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI05), 2005, pp. 702–707.
[10] R. Prati, P. Flach, Roccer: an algorithm for rule learning based on roc analysis, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc., 2005, pp. 823–828.
[11] M. Barreno, A.A. Cárdenas, J.D. Tygar, Optimal roc curve for a combination of classifiers, Adv. Neural Inf. Process. Syst. 20 (X) (2008) 57–64.
[12] L. Cam, Neyman–Pearson Lemma, Encyclopedia of Biostatistics.
[13] T. Fawcett, Prie: a system for generating rule lists to maximize roc performance, Data Min. Knowl. Discov. 17 (2) (2008) 207–224.
[14] J. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press, 1992.
[15] R. Poli, W. Langdon, N. McPhee, A Field Guide to Genetic Programming, Lulu Enterprises UK Ltd, 2008.
[16] P. Espejo, S. Ventura, F. Herrera, A survey on the application of genetic programming to classification, IEEE Trans. Syst. Man Cybernet. Part C: Appl. Rev. 40 (2) (2010) 121–144.
[17] P. Wáng, T. Weise, R. Chiong, Novel evolutionary algorithms for supervised classification problems: an experimental study, Evolut. Intell. 4 (1) (2011) 3–16, http://dx.doi.org/10.1007/s12065-010-0047-7.
[18] W. Tackett, Genetic programming for feature discovery and image discrimination, in: Proceedings of the Fifth International Conference on Genetic Algorithms, ICGA-93, Citeseer, 1993, pp. 303–309.
[19] J. Li, X. Li, X. Yao, Cost-sensitive classification with genetic programming, in: 2005 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2005, pp. 2114–2121.
[20] Y. Zhang, H. Li, M. Niranjan, P. Rockett, Applying cost-sensitive multiobjective genetic programming to feature extraction for spam e-mail filtering, Genetic Program. (2008) 325–336.
[21] E. Alfaro-Cid, K. Sharman, A. Esparcia-Alcázar, A genetic programming approach for bankruptcy prediction using a highly unbalanced database, AI 2007: Appl. Evolut. Comput. (2007) 169–178.
[22] G. Patterson, M. Zhang, Fitness functions in genetic programming for classification with unbalanced data, AI 2007: Adv. Artif. Intell. (2007) 769–775.
[23] U. Bhowan, M. Johnston, M. Zhang, X. Yao, Evolving diverse ensembles using genetic programming for classification with unbalanced data, IEEE Trans. Evolut. Comput, http://dx.doi.org/10.1109/TEVC.2012.2199119, in press.
[24] P. Flach, Roc Analysis, Encyclopedia of Machine Learning, Springer, Berlin, Heidelberg, 2010, pp. 869–874.
[25] G. Hughes, On the mean accuracy of statistical pattern recognizers, IEEE Trans. Inf. Theory 14 (1) (1968) 55–63.
[26] K. Spackman, Signal detection theory: valuable tools for evaluating inductive learning, in: Proceedings of the Sixth International Workshop on Machine Learning, Morgan Kaufmann Publishers Inc., 1989, pp. 160–163.
[27] M. Scott, M. Niranjan, R. Prager, Realisable classifiers: improving operating performance on variable cost problems, in: Proceedings of the Ninth British Machine Vision Conference, vol. 1, Citeseer, 1998, pp. 304–315.
[28] H. Li, Q. Zhang, Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii, IEEE Trans. Evolut. Comput. 13 (2) (2009) 284–302.
[29] T. Weise, Global Optimization Theory and Application, it-weise.de (Self-Published), Germany, 2009. URL 〈http://www.it-weise.de/projects/book.pdf〉.
[30] K. Deb, A. Pratab, S. Agrawal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evolut. Comput. (IEEE-EC) 6 (2) (2002) 182–197.
[31] Q. Zhang, H. Li, Moea/d: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evolut. Comput. 11 (6) (2007) 712–731.
[32] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Trans. Evolut. Comput. 3 (4) (1999) 257–271.
[33] K. Bringmann, T. Friedrich, F. Neumann, M. Wagner, Approximation-guided evolutionary multi-objective optimization, in: Proceeding of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Spain, 2011, pp. 1198–1203.
[34] L. Jin, FGP: A Genetic Programming Based Financial Forecasting Tool, Ph.D. Thesis, Citeseer, 2000.
[35] J. Backus, The syntax and semantics of the proposed international algebraic language of the Zurich acm-gamm conference, in: Proceedings of the International Conference on Information Processing, 1959.
[36] N. Beume, B. Naujoks, M. Emmerich, Sms-emoa: multiobjective selection based on dominated hypervolume, Eur. J. Operat. Res. 181 (3) (2007) 1653–1669.
[37] E. Zitzler, L. Thiele, An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach, TIK-Report 43, Eidgenössische Technische Hochschule (ETH) Zürich, Department of Electrical Engineering, Computer Engineering and Networks Laboratory (TIK), Zürich, Switzerland, May 1998.
[38] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, TIK-Report 101, Eidgenössische Technische Hochschule (ETH) Zürich, Department of Electrical Engineering, Computer Engineering and Networks Laboratory (TIK), Zürich, Switzerland, Errata Added 2001-9-27, May 2001.
[39] T. Weise, R. Chiong, K. Táng, Evolutionary optimization: pitfalls and booby traps, J. Comput. Sci. Technol. 27 (5) (2012) 907–936.
[40] P. Wáng, K. Táng, E.P.K. Tsang, X. Yáo, A memetic genetic programming with decision tree-based local search for classification problems, in: Proceedings of the 12th IEEE Congress on Evolutionary Computation (CEC'11), 2011, pp. 917–924. http://dx.doi.org/10.1109/CEC.2011.5949716.
[41] UCI, Uc Irvine Machine Learning Repository, 2009. URL 〈http://archive.ics.uci.edu.c/ml〉.
[42] J. Quinlan, C4.5: Programs for Machine Learning, Morgan, Kaufmann, 1993.
[43] I. Rish, An empirical study of the Naive Bayes classifier, in: IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, vol. 3, 2001, pp. 41–46.
[44] F. Wilcoxon, Individual comparisons by ranking methods, Biomet. Bull. 1 (6) (1945) 80–83.

Pu Wang received the bachelors degree in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2008. He is currently pursuing the PhD degree from the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, USTC.

His current research interests include evolutionary computation, memetic algorithms, and multiobjective optimization for classification problems in machine learning, and genetic programming for model checking in software engineering.

Ke Tang received the BEng degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002, and the PhD degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007. In 2007, he joined the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China, Hefei, China, as an Associate Professor, and was promoted to Professor in 2011. He is also an Honorary Senior Research Fellow in the School of Computer Science, University of Birmingham, UK. He has authored/co-authored more than 50 refereed publications. His major research interests include evolutionary computation, machine learning, meta-heuristic algorithms, and real-world applications. He is an Associate Editor of IEEE Computational Intelligence Magazine and served as a program co-chair of CEC2010, held in Barcelona.

Thomas Weise received the Diplom Informatiker (equivalent to MS) degree from the Department of Computer Science, Chemnitz University of Technology, Chemnitz, Germany, in 2005, and the PhD degree from the Distributed Systems Group of the Fachbereich Elektrotechnik and Informatik, University of Kassel, Kassel, Germany, in 2009.

Since 2009, he has been a Researcher with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China. He has been an Associate Professor in USTC since 2011. His experience ranges from applying genetic programming to distributed systems and multiagent systems, efficient web service composition for service oriented architectures, to solving large-scale real-world vehicle routing problems for multimodal logistics and transportation. Besides being the author and co-author of over 40 refereed publications, he also authors the electronic book Global Optimization Algorithms—Theory and Application which is freely available at his website http://www.itweise.de. His major research interests include evolutionary computation, genetic programming, and real-world applications of optimization algorithms.

Edward Tsang received the bachelor degree in business administration from the Chinese University of Hong Kong, Shatin, Hong Kong, in 1977, and the MSc and PhD degrees in computer science from the University of Essex, Colchester, UK, in 1983 and 1987, respectively. He is currently a Professor in Computer Science at the School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK, where he is also the Co-Founder and the Director of the Center for Computational Finance and Economic Agents, an interdisciplinary research center. He was with companies including British Telecom, London, UK, Honda Europe, Ghent, Belgium, and OANDA, NY. Main techniques used in his research include heuristic search, optimization, and evolutionary computation. With background and experience in industry, he has

broad interest in business applications of artificial intelligence, including computational finance, computational economics, and constraint satisfaction.

He was the Founding Chair of the IEEE Computational Finance and Economics Technical Committee in which he is an Active Member.

Xin Yao received the BSc degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the MSc degree from the North China Institute of Computing Technology, Beijing, China, in 1985, and the PhD degree from USTC in 1990.

From 1985 to 1990, he was an Associate Lecturer and Lecturer with USTC, while working toward the PhD degree in simulated annealing and evolutionary algorithms. In 1990, he was a Postdoctoral Fellow with the Computer Sciences Laboratory, Australian National University, Canberra, Australia, where he continued his work on simulated annealing and evolutionary algorithms. In 1991, he was with the Knowledge-Based Systems Group, Commonwealth Scientific and Industrial Research Organization, Division of Building, Construction and Engineering, Melbourne, Australia, where he worked primarily on an industrial project on automatic inspection of sewage pipes. In 1992, he returned to Canberra to take up a lectureship in the School of Computer Science, University College, University of New South Wales, Australian Defense Force Academy, Sydney, Australia, where he was later promoted to a Senior Lecturer and Associate Professor. Since April 1999, he has been a Professor (Chair) of computer science in the University of Birmingham, Birmingham, UK. He is currently the Director of the Center of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, Birmingham, UK and also a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, USTC. He has given more than 50 invited keynote and plenary speeches at conferences and workshops worldwide. He has more than 300 referenced publications. His major research interests include evolutionary artificial neural networks, automatic modularization of machine learning systems, evolutionary optimization, constraint-handling techniques, computational time complexity of evolutionary algorithms, coevolution, iterated prisoner's dilemma, data mining, and real-world applications.

Yao was the Editor-in-Chief of the IEEE Transactions on Evolutionary Computation from 2003 to 2008, an Associate Editor or editorial board member of 12 other journals, and the Editor of the World Scientific Book Series on Advances in Natural Computation. He was the recipient of the President's Award for Outstanding Thesis by the Chinese Academy of Sciences for his PhD work on simulated annealing and evolutionary algorithms in 1989. He was the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.

Please cite this article as: P. Wang, et al., (2013), http://dx.doi.org/10.1016/j.neucom.2012.06.054