Soft Comput (2016) 20:3927–3946
DOI 10.1007/s00500-016-2128-8

FOCUS

New mechanism for archive maintenance in PSO-based multi-objective feature selection

Hoai Bach Nguyen[1] · **Bing Xue**[1] · **Ivy Liu**[2] · **Peter Andreae**[1] · **Mengjie Zhang**[1]

Published online: 28 March 2016

Abstract In classification problems, a large number of features are typically used to describe the problem's instances. However, not all of these features are useful for classification. Feature selection is usually an important pre-processing step to overcome the problem of "curse of dimensionality". Feature selection aims to choose a small number of features to achieve similar or better classification performance than using all features. This paper presents a particle swarm Optimization (PSO)-based multi-objective feature selection approach to evolving a set of non-dominated feature subsets which achieve high classification performance. The proposed algorithm uses local search techniques to improve a Pareto front and is compared with a pure multi-objective PSO algorithm, three well-known evolutionary multi-objective algorithms and a current state-of-the-art PSO-based multi-objective feature selection approach. Their performances are examined on 12 benchmark datasets. The experimental results show that in most cases, the proposed multi-objective algorithm generates better Pareto fronts than all other methods.

Keywords Multi-objective · Feature selection · Classification · Particle Swarm Optimization

Communicated by B. Xue and A.G. Chen.

✉ Hoai Bach Nguyen
Hoai.Bach.Nguyen@ecs.vuw.ac.nz

Bing Xue
Bing.Xue@ecs.vuw.ac.nz

Ivy Liu
Ivy.Liu@vuw.ac.nz

Peter Andreae
Peter.Andreae@ecs.vuw.ac.nz

Mengjie Zhang
Mengjie.Zhang@ecs.vuw.ac.nz

[1] School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand

[2] School of Mathematics and Statistics, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

1 Introduction

In machine learning, classification plays an important role, which aims to assign a known class label to an instance based on the instance's properties, called instance features. In many classification algorithms, an instance is described by a large number of features. However, not all features provide useful information. For example, some features do not contain any relevant information about the class label, while other features may be redundant. Those features can conceal the useful information from the relevant features, which results in a low-quality classifier (Zhao et al. 2009). In addition, due to "the curse of dimensionality" (Gheyas and Smith 2010), a large number of features cause a long time to train a classification algorithm. To overcome this problem, feature selection (Guyon and Elisseeff 2003) is proposed to reduce the number of features by removing all redundant and irrelevant features, which will not only speed up the learning/classification process but also improve the classification performance over using all features. However, it is not an easy task to develop an efficient and effective feature selection approach due to the complex interaction between features and its huge search space.

Feature selection aims to extract a good feature subset from a large set of original feature. The extracted feature subset is expected to achieve similar or better performance than the whole feature set. Suppose that there are n original features, then the total number of possible feature subsets

is 2^n. It can be seen that the search space size of feature selection increases exponentially with respect to the number of features. An exhaustive search technique, which considers all feature subsets, can guarantee to produce an optimal feature subset. However, the exhaustive search is clearly too slow especially when there are a large number of original features. Greedy search algorithms, such as sequential forward selection (Whitney 1971) and sequential backward selection (Marill and Green 1963), are more efficient than the exhaustive search algorithm. However, these methods easily get stuck at local optima. To solve the large search space problem, evolutionary computation (EC) techniques are good solutions because of their global search ability. Many EC algorithms, such as genetic programming (GP) (Neshatian and Zhang 2009a), genetic algorithms (GAs) (Yuan et al. 1999), artificial bee colony algorithm (ABC) (Hancer et al. 2015) and particle swarm optimization (PSO) (Unler and Murat 2010; Yang et al. 2008), have been widely applied to feature selection. Compared with other EC algorithms, PSO is preferable because it not only uses fewer parameters but also converges more quickly.

Feature selection has two main goals, which are to minimize the number of selected features and to maximize the classification performance. However, these objectives are usually conflicting with each other. For example, when the number of features is reduced, the classification is likely to be decreased. Therefore, feature selection can be viewed as a multi-objective problem and the searching process needs to consider the trade-off between two objectives. This paper will take the advantages of PSO to develop a multi-objective feature selection approach which can simultaneously achieve both objectives of feature selection problems.

1.1 Goals

The overall goal of this study is to develop a PSO-based multi-objective feature selection approach, which can produce a set of non-dominated solutions that specify a small number of features and achieve better classification performance than using all features. To achieve this goal, we develop a new multi-objective PSO (MOPSO) algorithm, called ISRPSO, which uses local search techniques to improve the Pareto front. In addition, the proposed algorithm is also compared with three well-known evolutionary multi-objective algorithms including Non-dominated sorting genetic algorithm II (NSGAII) (Deb et al. 2000), Strength Pareto evolutionary algorithm 2 (SPEA2) (Zitzler et al. 2001) and Pareto archived evolutionary strategy (PAES) (Knowles and Corne 1999). Finally the proposed algorithm will then be compared with a state-of-the-art multi-objective PSO algorithm, named CMDPSOFS (Xue et al. 2013). Specifically, we will investigate the following:

- Whether the proposed multi-objective PSO algorithm (ISRPSO) evolves a set of non-dominated solutions with a small number of features and better classification performance than using all features.
- Whether applying local search can evolve better Pareto front in comparison with a pure multi-objective PSO.
- Whether ISRPSO can evolve a better Pareto front than NSGAII (Deb et al. 2000), SPEA2 (Zitzler et al. 2001) and PAES (Knowles and Corne 1999) in feature selection problems.
- Whether ISRPSO is able to produce a Pareto front of non-dominated solutions, which can outperform a state-of-the-art multi-objective PSO for feature selection, CMDPSOFS (Xue et al. 2013).

1.2 Organization

The remainder of this paper is organized as follows. Section provides background information. In Sect. 3, we propose a novel PSO-based multi-objective feature selection algorithm. Section 4 describes the experimental design, and Sect. 5 presents the experimental results. In Sect. 6, we conclude our work with some possible extensions for future work.

2 Background

2.1 Particle swarm optimization (PSO)

In 1995, Kennedy et al. (1995) proposed an evolutionary computation algorithm called particle swarm optimization (PSO), which was developed based on social behaviours. Like other swarm intelligence algorithms, PSO maintained a set of particles, which was also known as a *swarm*. Each particle, which represented a candidate solution, moved around the search space using its own position and velocity. Particularly, the ith particle's position was a D-dimensional vector, $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where D was the dimensionality of the search space. For each iteration, the positions were updated according to the particle's velocity, which was also represented by a vector, $v_i = (v_{i1}, v_{i2}, \ldots, v_{\mathrm{iD}})$. To avoid skipping good positions due to moving too fast, the particle's velocity was limited by a predefined maximum velocity $v_{\max}$, which meant $-v_{\max} \leq v_{\mathrm{id}} \leq v_{\max}$. In PSO, each particle also recorded the best position discovered by itself, called $pbest$. In addition, the best position discovered by a particle's neighbours and itself was also maintained, which was called $gbest$. According to the two $pbest$ and $gbest$, the ith particle moved around the search space by the following updating equations:

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{i1} * \left(p_{id} - x_{id}^{t}\right) + c_2 * r_{i2} * \left(p_{gd} - x_{id}^{t}\right) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

where t represents the tth iteration, d denotes the dth dimension in the search space, w is a predefined constant inertia weight, c_1 and c_2 are acceleration constants, r_{i1} and r_{i2} are two random values uniformly generated in the interval [0,1], and p_{id} and p_{gd} represent the position entry of *pbest* and *gbest* in the dth dimension, respectively.

The above description explains how a continuous PSO works. There is still another kind of PSO, called binary PSO (BPSO). In BPSO, the position entries are binary values. There are some potential limitations of the current binary PSO; for example, the update of particle position is based only on the current velocity while in standard PSO, both current velocity and position are used. Although both versions of PSO have been successfully used to solve feature selection problems (Unler and Murat 2010; Liu et al. 2011; Chuang et al. 2008; Huang and Dun 2008), continuous PSO achieves better performance than binary PSO as shown by Xue (Xue et al. 2012a). Therefore, we will develop a multi-objective feature selection approach using continuous PSO.

2.2 Multi-objective optimization

2.2.1 Basic concepts of multi-objective optimization

In many optimization problems, there is more than one objective that needs to be optimized. Often, the objectives conflict. Multi-objective problems refer to the presence of two or more conflicting objectives. For example, a feature selection problem is a multi-objective problem, in which minimizing both classification error rate and the number of selected features are two conflicting objectives. The conflicting objectives can be expressed in multiple conflicting objective functions, which needs to be maximized or minimized. In mathematical terms, a multi-objective problem can be expressed as in the following formula (as minimization problems):

$$\text{Minimize } (f_1(x), f_2(x), \ldots, f_k(x)) \quad (3)$$

where x is the vector of variables which describe the problem's solutions, $f_i(x)$ is an objective function of x and the integer $k \geq 2$ is the number of objectives.

In multi-objective problems, the trade-off between conflicting objectives is used to measure the quality of a solution. In particular, suppose that y and z are two solutions of a k objective minimization problem. The solution y is better than z or y dominates z if and only if:

$$\forall i : f_i(y) \leq f_i(z) \quad \text{and} \quad \exists j : f_j(y) < f_j(z) \quad (4)$$

where $i, j \in 1, 2, 3, \ldots, k$. When a solution is not dominated by any other solutions, it is called a non-dominated solution. The set of a non-dominated solution forms a trade-off surface in the objective space, namely *Pareto front*. A multi-objective algorithm is designed to evolve a set of non-dominated solutions.

2.2.2 Evolutionary multi-objective algorithms

NAGAII, proposed by Deb et al. (2000), is one of the most popular evolutionary multi-objective algorithms. There are two main ideas introduced in NSGAII: a fast non-dominated sorting technique and a diversity preserving algorithm. The fast non-dominated sorting technique is an efficient approach to sorting all parents and offspring into different level of non-dominated solutions. This sorted population is used to build the parent population for the next generation. The density-preserving algorithm estimates the density of solutions based on the crowding distance and uses the estimates to maintain the diversity of the population.

SPEA2 is another popular evolutionary multi-objective algorithm proposed by Zitzler et al. (2001). The most important principle introduced in SPEA2 is the incorporation between the fine-grained fitness assignment strategy, a density estimation technique and an enhanced archive truncation method. In particular, each individual fitness is the sum of its strength raw fitness and a density estimation. A new population is constructed by selecting non-dominated solutions from both the original solution and the archive. If the number of non-dominated solutions exceeds the population size, the archive truncation method is applied to determine which solutions will be selected. The truncation method bases on the distance between a solution and its kth nearest neighbour.

PAES is an evolutionary multi-objective algorithm proposed by Knowles and Corne (1999). The author argued that PAES is the simplest possible non-trivial algorithm, which is able to generate diverse solutions in the Pareto optimal set. The main idea of PAES is the combination of local search and the usage of an archive of previously found non-dominated solutions to identify the approximate dominance rank of the current and candidate solutions.

The success of the PSO algorithm for a single-objective problem has encouraged researchers to extend PSO to solve multi-objective problems (MOPSO). As explained in the previous section, in PSO for a single-objective problem, each particle has exactly one leader *gbest* to update its position. However, in most of current MOPSO algorithms, instead of recording *gbest* for each particle, an *archive set* is used to maintain a set of non-dominated solutions being discovered by the population. Each particle will select a solution from the *archive set* as its *gbest* to update its position. When the final iteration is reached, the archive set represents the final result of the MOPSO algorithm, which forms the *Paretofront*. This work will apply a local search on the archive set to help MOPSO better explore *Pareto front*. Since these algorithms

were successfully applied to solve feature selection problems, all these above algorithms will be used as benchmark techniques for the proposed algorithm, which is specifically designed for feature selection.

2.3 Related work on feature selection

2.3.1 Traditional feature selection methods

A basic version of feature selection is feature ranking (Dash and Liu 1997), where a score is assigned to each feature according to an evaluation criterion. Feature selection can be performed by selecting features with the highest scores. However, this type of algorithm ignores the interaction between features. Additionally, the features with the highest scores are usually similar. Therefore, these algorithms tend to select redundant features.

Sequential search techniques were also applied to solve feature selection problems. In particular, sequential forward selection (SFS) (Whitney 1971) and sequential backward selection (SBS) (Marill and Green 1963) were proposed. SFS started with an empty feature set. For each iteration, a single feature from the unselected features, which resulted in the best accuracy when using with selected features, was added into the selected feature set. On the other hand, SBS started from a full set of features and iteratively removed a single feature at each step. These algorithms stopped when adding or removing any single feature did not result in any improvement of the classification accuracy. Although these local search techniques achieved better performance than the feature ranking method, they might suffer the "nesting" problem, in which a feature was permanently added or removed from the selected feature set. To avoid the nesting effect, Stearns (1976) proposed a "plus-l-takeaway-r" method in which l features were added before r features were removed from the feature subset. However, it was not an easy task to determine the best values of (l,r). Instead of using a fixed pair of values (l,r), Pudil et al. (1994) proposed floating sequential selection algorithms, in which the values (l,r) were dynamically determined.

2.3.2 EC approaches (non-PSO) for feature selection

EC techniques are well known because of their global search ability. EC algorithms have been applied to feature selection problems, such as GAs (Zhu et al. 2007), GP (Neshatian and Zhang 2009b). Zhu et al. (2007) proposed a hybrid feature selection approach, which combined both local search and GAs. In this algorithm, a filter method was used to rank features individually. Basing on the ranking information, GAs deleted or added a feature to achieve better fitness value, which was the classification accuracy. The experiments showed that this algorithm outperformed the GAs alone and other algorithms.

Oreski and Oreski (2014) proposed a hybrid genetic algorithm with neural-networks (HGA-NN) to evolve an optimum feature subset. In the initialization step, the feature set was narrowed by different fast filter techniques. So important features, which were selected by the filter approaches, were used to initialise the major part of population. The rest of population was filled randomly. In HGA-NN, an incremental stage was applied to enhance the creation of the initial population, which increased the diversity of the genetic material. The proposed algorithm was evaluated on two real-world credit datasets. The experimental results showed that HGA-NN achieved better classification performance than GA-NN technique proposed by Wang (2005).

Lin et al. (2014) proposed a novel GA-based feature selection approach, in which the prior knowledge about financial distress prediction was used to group similar features. After that a filter approach was used to rank all features in the same group and only top-rank features from each group were chosen to participate in the selection process by GAs algorithm. Although the two-step selection approach was efficient, it skipped the interaction between features. Other GA-based feature selection approaches were developed recently to solve real-world problems, such as Chaaraoui and Flórez-Revuelta (2013); Seo et al. (2014); Liang et al. (2015). Neshatian and Zhang (2009b) proposed a wrapper GP-based approach, which evaluated and ranked feature subset in binary classification tasks. Experiments showed that the proposed methods detected subset of relevant features in different situations, where other methods had difficulties. Bhowan and McCloskey (2015) proposed two GP-based approaches to evolve a set of features, which was used directly in the Watson system, an intelligent open-domain question answering system. The first approach extracted all features, which were used in the best-of-run evolved GP tree. The second approach considers all evolved trees. Particularly, from the set of GP trees, the top N features with the most frequency were chosen as extracting features. Two values of N being used in this paper were 10 and 20. The experiment results showed that, the set of features selecting from the best GP tree only worked well when the number of selected features was small. Meanwhile, selecting top N features from the whole set of trees produces good resulted on both small and large datasets. However, as other ranking features selection algorithms, this algorithm did not consider the interaction between features, especially between redundant features.

2.3.3 PSO-based feature selection methods

Many EC algorithms have been used for feature selection, such as GAs, GP or PSO. PSO is preferable because it is easier to implement and uses fewer parameters than GAs and

GP. Wang et al. (2007) proposed a filter PSO-based feature selection algorithm, which applied rough set theory to evaluate feature subset. In the proposed algorithm, the fitness function was the combination of the classification quality of the feature subset calculated by rough set theory and the proportion of the selected features. The experimental results showed that the proposed algorithm could find the optimal solution in a smaller amount of time than a GA using rough sets. Chakraborty and Chakraborty (2013) proposed a filter PSO-based feature selection approach, which used fuzzy set to calculate the fitness value for each particle. Specifically, the membership value of fuzzy set theory was used with more than one thresholds to decide whether or not an instance is consistent according to the selected features. The proportion of consistent instances over the total number of instances was the consistent or fitness measure of the current feature subset. Two PSO-based filter feature selection algorithms were proposed in Cervante et al. (2012), where mutual information and entropy were used in the fitness function to evaluate the relevance and redundancy of the selected feature subset. The experiments showed that the proposed methods significantly reduced the number of features whilst achieved similar or better classification than using all features.

In PSO, premature convergence was a common problem, in which the swarm converged quickly to a local optima. To avoid premature convergence, Chuang et al. (2008) proposed a new *gbest* updating mechanism, which resets *gbest* elements to zero if it maintained the same value after several iterations. However, the performance of this algorithm was not compared with other PSO-based algorithms. Later, Tran et al. (2014) also applied this resetting mechanism cooperating with local searches on *pbest* to simultaneously reduce the number of selected features and improve the classification performance. In addition, the new fitness calculation was proposed, which based on the changed features (selected to not selected or vice versa). The proposed algorithm achieved smaller feature subset with lower classification error than (Chuang et al. 2008). Another binary PSO-based algorithm, which also aimed to avoid premature convergence, was proposed by Bin et al. (2012). At each iteration of this algorithm, the swarm was divided into two groups, named "leaders" and "followers". The "leaders" had better fitness values. The "followers" updated their positions and velocities based on leaders' update. The experimental results showed that the proposed update strategy better utilizes the social behaviour phenomenon than the standard binary PSO.

Xue et al. (2014) proposed three new initialization mechanisms, which mimic the sequential feature selection approach. While the small initialization used about 10 % of original features to initialize the particles, particles in the large initialization were constructed based on 50 % of original features. These two initialization mechanisms were combined in the mixed initialization, which used the small initialization for most of particles and the large initialization for the rest. In addition, three new updating mechanisms for *pbest* and *gbest* were proposed in the paper. The experimental results showed that the new initialization and updating mechanisms led to smaller feature subsets with better classification performance than the standard PSO and two-stage binary PSO algorithm.

Vieira et al. (2013) proposed a new representation for binary PSO, which simultaneously performed feature selection and optimized the SVM kernel parameters. Particularly, each bit string corresponded to an original feature or a kernel parameter, which resulted in the length of the new representation was equal to the total number of features and kernel parameters. Experimental results showed that the proposed algorithm achieved better classification performance than other binary PSO-based feature selection algorithms (Chuang et al. 2008; Lee et al. 2008) and selected smaller feature subsets than GA-based feature selection algorithm (Huang and Wang 2006). This representation was also applied to continuous encoding (Lin et al. 2008) and a mixture of binary and continuous encoding (Boubezoul and Paris 2012). Lane et al. (2013) applied statistical clustering, which groups similar features into one cluster. In particular, the proposed method arranged the features in the same cluster together and selected a single feature from each cluster based on the velocity. Lane et al. (2014) further improved his work by applying Gaussian distribution to select multiple features from each cluster. Later, Nguyen et al. (2014b) also applied statistical clustering to propose a new representation, in which each bit string belonged to a certain feature cluster and presented a feature index from the cluster. However, in the new representation a small change of the position might not lead to any different feature subset. Therefore, Nguyen et al. (2015) applied Gaussian distribution to propose a new transformation rule, which could form a smoother fitness landscape than the representation in Nguyen et al. (2014b).

Recently, filter and wrapper approaches have been combined to form hybrid approaches, which take the advantages of both filter and wrapper methods. Nguyen et al. (2014a) proposed a wrapper PSO-based feature selection approach, in which *gbest* is improved by a local search using a filter-based measure. The local search mimicked the typical backward feature selection method to remove features from *gbest* according to the relevant and redundant measure calculated by mutual information. The experimental results showed that the proposed algorithms selected much smaller number of features while still achieved similar or better classification performance than the other PSO-based algorithms. Although the proposed algorithm had to perform an extra task for local search, its computation cost is still cheaper than other PSO-based algorithms because of the smaller number of selected features. An extensive review about EC-based feature selection algorithms was conducted in Xue et al. (2015a).

Feature selection problem can be seen as a multi-objective problem, because its two objectives usually conflict with each other. However, most of wrapper feature selection approaches used only classification performance as the fitness function (Mohemmed et al. 2009; Zhang et al. 2015) or combined the classification accuracy and the number of selected features into a single fitness function (Xue et al. 2012b; Huang and Dun 2008). Xue et al. (2012a) proposed two multi-objective PSO algorithms for feature selection problems. The first algorithm applied the idea of non-dominated sorting-based multi-objective genetic algorithm II (NSGAII) into PSO for feature selection. The other algorithm based on the idea of crowding, mutation and dominance (CMDPSOFS) to evolve the Pareto front solutions. According to the experimental results, both algorithms could select a small number of features while achieving better classification performance than using all features. However, the above algorithms did not propose any specific design for feature selection problems. Therefore, this work will propose a new local search technique for MOPSO, which is specifically designed for feature selection problems. The proposed algorithm is then compared with the current state-of-the-art multi-objective PSO-based algorithm, CMDPSOFS.

3 PSO-based multi-objective feature selection algorithm: ISRPSO

In this section, we investigate a new approach to feature selection using multi-objective PSO, with two objectives to explore the Pareto front of feature subsets. Initially, standard PSO was proposed to solve single-objective problems, in which each particle remembers the best position discovered by its neighbours and itself so far, called $gbest$. To develop a multi-objective PSO algorithm, Li (2003) introduces the idea of an archive set, which is used in NSGA. The archive set contains all non-dominated solutions which have been discovered by the swarm so far. With the archive set, instead of using the individual $gbest$ to update velocity, each particle selects an archive member as its $gbest$. Therefore, the whole archive set guides the swarm through the search space. It is expected that the better archive sets would allow the swarm to construct better solutions, which select a small number of features and maintain or even improve the classification accuracy. In this paper, we investigate a novel PSO-based multi-objective feature selection algorithm (ISRPSO), which uses local search techniques to improve the archive set with a better Pareto front.

3.1 Representation and fitness function for ISRPSO

In this study, a continuous PSO is used to solve feature selection problems. Each particle's position representation is a vector of n real numbers, where n is the total number of features. Each position entry x_i corresponds to the ith feature in the original feature set. The value x_i varies between 0 and 1 and represents the confidence that feature ith should be included in the solution subset. A threshold θ is used to decide whether or not a feature is selected: the ith feature is selected if and only if $\theta < x_i$.

As stated above, each feature selection problem has two objectives: minimizing the classification error rate and minimizing the number of selected features. These objectives are calculated according to Eqs. 5 and 6, respectively. Notice that both $ErrorRate$ and $FeatureRate$ varies in the interval [0,1], which guarantees the fairness between two objectives.

$$\text{ErrorRate} = \frac{FP + FN}{TP + TN + FP + FN} \tag{5}$$

where TP, TN, FP and FN are true positives, true negatives, false positives and false negatives, respectively.

$$\text{FeatureRate} = \frac{\#Selectedfeatures}{\#Allfeatures} \tag{6}$$

3.2 Important set concepts from the archive set

Before describing the novel multi-objective PSO, it is helpful to clearly define some concepts about feature sets being built from the archive set. As mentioned above, the archive set contains all non-dominated solutions, which are not dominated by any other solutions discovered so far. Therefore, the features being selected by these archive members can be considered the important features. These features are contained in one feature set, denoted as S.

Another important concept relates to each individual of the archive set. In particular, a set S_i consists of all features which are selected by the ith archive member. Similarly, a set US_i consists of all features, which are selected by all archive members except the ith particle. It can be seen that $S = S_i \bigcup US_i$ or $US_i = S \backslash S_i$.

Let us consider an example. Assume the original feature set contains 5 features, $F = \{f_1, f_2, f_3, f_4, f_5\}$. The archive set contains three members $Archive = \{A_1, A_2, A_3\}$, in which the features selected by each member are defined as below:

- 1th member (A_1): $S_1 = \{f_1, f_2\}$
- 2nd member (A_2): $S_2 = \{f_1, f_3\}$
- 3rd member (A_3): $S_3 = \{f_2, f_3, f_4\}$

So feature f_5 is not selected by any archive members, which might indicate that f_5 is an irrelevant or a redundant feature. According to the above definition, the set of all features being selected by this archive set is $S = \{f_1, f_2, f_3, f_4\}$. For each archive member, the set US_i is defined as below

- $US_1 = S \backslash S_1 = \{f_3, f_4\}$
- $US_2 = S \backslash S_2 = \{f_2, f_4\}$
- $US_3 = S \backslash S_3 = \{f_1\}$

3.3 Local search to improve the archive set

As stated above, this work investigates using local search techniques to improve the quality of the archive members, which hopefully results in a better Pareto front. In particular, three operations, including *Inserting (I), Swapping (S), Removing (R)* are proposed to enhance the archive set. All of these operators use three sets S, S_i, US_i to search around each archive member to find better solutions. These operations are described in the following sections.

3.3.1 Inserting

As mentioned above, for the ith archive member, the set US_i contains all features that are selected by all archive members except the ith members. Since the features being selected by archive members are usually important, it would be possible to improve the classification accuracy of an archive member by adding one feature from the set US_i to the current set of selected features S_i.

The features in each set US_i are sorted according to their single classification accuracy, which is the classification performance when only feature ith is used to perform classification. The better features, which have higher accuracy are in front of the list. After that, each feature in US_i is temporarily added into S_i to form a new feature set, called S_i'. If the accuracy of S_i' is better than the accuracy of S_i, the inserting process stops. The new solution, called SO_{in}, whose selected features are S_i', is built and passed as an input to the next step ($Removing$). So after the inserting step, the new solution SO_{in} will select at most one feature more than the original archive member, A_i.

Continuing the example from the above section, in which the first archive member A_1 selects $S_1 = \{f_1, f_2\}$ and its $US_1 = \{f_3, f_4\}$. Suppose that $C(\{f_1, f_2, f_3\}) = 0.75$, $C(\{f_1, f_2, f_4\}) = 0.82$, $C(\{f_1, f_2\}) = 0.80$, where $C(.)$ is the classification of a feature set. Also suppose that the single classification accuracy of f_3 is better than f_4. The inserting process will start with f_3. At the first step, the "inserting" process will temporarily add feature f_3 to set S_1, to form a new set $\{f_1, f_2, f_3\}$. However, since $C(\{f_1, f_2, f_3\}) < C(\{f_1, f_2\})$, feature f_3 will be discarded. After that feature f_4 is temporarily added to the set S_1 to form a new feature set, $\{f_1, f_2, f_4\}$. This time, since $C(\{f_1, f_2, f_4\}) > C(\{f_1, f_2\})$, a new solution SO_{in} is created, which uses $\{f_1, f_2, f_4\}$ as its selected features. This solution will be further improved by $Removing$ operation, which will be discussed in the next section.

3.3.2 Removing

In the $Inserting$ step, at most one feature might be added to the feature selected set of an archive member to form a new solution SO_{in}, which is guaranteed to be similar or better than the archive member in terms of classification accuracy. In contrast, $Removing$ step tries to improve SO_{in} by removing at most one feature from SO_{in}. The $Removing$ process works in the same way as backward selection algorithm. Firstly, all features selected by SO_{in} are sorted according to their single classification accuracies. In the sorted list, the features with less accuracy will be at the top of the list; therefore, they are more likely to be removed. After that, each feature from the sorted list will be temporarily removed from the selected set until there is an improvement in terms of the classification accuracy. The new solution, called SO_{re} is built and passed as an input to the next step ($Swapping$). So after the removing step, the new solution SO_{re} will contain at most one feature less than the solution SO_{in}, provided by the $Inserting$ step.

Continuing the example from the above section, the *Inserting* process provides $SO_{in} = \{f_1, f_2, f_4\}$ as an input to the $Removing$ process. Suppose that $C(f_2) < C(f_1) < C(f_4)$. After sorting features according to their classification accuracy, the order of removing features is $f_2 \rightarrow f_1 \rightarrow f_4$. Suppose that $C(\{f_1, f_4\}) = 0.80$, $C(\{f_2, f_4\}) = 0.85$, and $C(\{f_1, f_2\}) = 0.75$. Firstly, f_2 is temporarily removed from SO_{in}, which results in a feature set $\{f_1, f_4\}$. However, since $C(\{f_1, f_4\}) = 0.80 < C(\{f_1, f_2, f_4\}) = 0.82$, feature f_2 restored. Next feature f_1 is temporarily removed from SO_{in}, which produces another feature set,$\{f_2, f_4\}$. This time, as $C(\{f_2, f_4\}) = 0.85 > C(\{f_1, f_2, f_4\}) = 0.82$, a new solution SO_{re} is created, which uses $\{f_2, f_4\}$ as its selected features. This solution will be improved in the last operation called $Swapping$, which will be discussed in the following section.

3.3.3 Swapping

Compared with *Inserting* and *Removing* operations, *Swapping* is a bit more complicated. For each solution, this operation acts on both the selected set S_i and unselected set US_i. In addition, instead of adding or removing features from the solution, this operation tries to find out better solutions by swapping between features in S_i and US_i. In other words, this operation will temporarily replace one feature from S_i by one feature from US_i.

Firstly, all features in S_i are sorted according to their single classification accuracies in ascending order, so the worst feature is more likely to be replaced. In contrast, all features in US_i are sorted in descending order; therefore, the feature with better single classification performance will have more chance to be added. For each feature u in US_i, this

process will try to replace with a feature in S_i. Once feature u finds a suitable feature s in S_i, which means the classification accuracy is improved when s is replaced by u, the replacing process for u is stopped. After that the swapping process will continue with the feature after feature u in US_i.

Continuing the example from the above section to explain for this operation, $\{f_2, f_4\}$, the new solution SO_{re} is fed into $Swapping$ process. Its selected and unselected sets are $S_{re} = \{f_2, f_4\}$ and $US_{re} = \{f_1, f_3\}$. Suppose that $C(f_1) < C(f_3)$ and $C(f_2) > C(f_4)$, the above two sets are sorted as following $S_{re} = \{f_4, f_2\}$ and $US_{re} = \{f_3, f_1\}$. The $Swapping$ operation starts with the first element of the set US_{re}, which is feature f_3. Firstly, f_3 will try to swap with the worst feature in the set S_{re}, which is feature f_4. Suppose that $C(\{f_3, f_2\}) > C(\{f_4, f_2\})$, then f_3 immediately replaces f_4 to create a better solution SO_{s_1}, which selects the following feature $\{f_3, f_2\}$. Note that the selected set is updated, which is $S_{s_1} = \{f_3, f_2\}$. The $Swapping$ processes will continue until all features in the unselected set US_{re} have been considered.

3.3.4 Overall ISRPSO algorithm

Algorithm 1 shows the pseudo-code of ISRPSO. To determine a leader, ISRPSO maintains a set of non-dominated solutions being discovered so far. A $gbest$ for a particle is selected from the $Archive$ set according to their objective distance and a binary tournament selection. Specifically, the binary tournament selection is used to select two solutions from the $Archive$ set, and the closest solution (to the particle) in the objective space is chosen as the $gbest$. The maximum size of the $Archive$ set is set as the number of particles in the swarm.

The local search, which aims to improve the solutions in $Archive$ set, is performed every five iterations. Particularly, three operations are used to improve the solution quality of the $Archive$ set. For each $Archive$ member, the $inserting$ operation tries to add one more feature into the member's feature set to create a new solution, called SO_{in}. If adding one feature improves the classification accuracy of the current $Archive$ member, then SO_{in} is built based on the new feature set. Otherwise, SO_{in} is exactly the same as the current $Archive$ member. After that, the solution SO_{in} is further improved by the $removing$ operation. On the contrary, $removing$ operation tries to delete at most one feature from SO_{in}, which results in similar (no feature is deleted) or higher classification accuracy. Once more, the output of *removing* operation, SO_{re}, is improved by the $swapping$ operation. Instead of removing or inserting features, this operation finds a better solution by swapping between features selected by current solution (SO_{re}) and features which are selected by the other $Archive$ members. So the $swapping$ operation might produce a new solution, SO_{sw}, which has the same number of selected features but achieves better accuracy than SO_{re}. Therefore, the local search method is able to improve the archive member's accuracy by increasing at most one feature. A new position is built based on the feature set SO_{sw}. In the new position, all position entries corresponding to features from SO_{sw} are set to 1 and all other entries are set to 0. Finally, if the new solution is not dominated by any *Archive* member, it will be inserted into the $Archive$ set. In addition, all solutions, which are dominated by the new solution, will be removed from the *Archive* set.

With these three operations, it is expected that at least the final improved solution SO_{sw} will not be dominated by any $Archive$ members. In a good case, SO_{sw} might even dominate some $Archive$ members. Therefore, this local search technique either maintains the diversity of the swarm or even improves the Pareto front by creating a better solution. Currently, since the classification accuracy is used to compare between two solutions, it is quite expensive to perform this local search. Therefore, the solutions within the $Archive$ set are improved by these three operations for every five iterations.

Algorithm 1 : Pseudo-code of ISRPSO

```
1: begin
2: divide Dataset into a Training set and a Test set;
3: initialize the swarm;
4: evaluate two objectives values for each particle;
5: add non-dominated solutions into Archive set;
6: while Maximum iterations is not reached do
7:     if iteration is divided by 5 then ▷ improve the Archive set for
           each 5 iterations
8:         for each solution i in the Archive do
9:             (SO_in, S_in, US_in) = Inserting(A_i, S_i, US_i)
10:            (SO_re, S_re, US_re) = Removing(SO_in, S_in, US_in)
11:            (SO_sw, S_sw, US_sw) = Swapping(SO_re, S_re, US_re)
12:            if SO_sw is not dominated by any archive members then
13:                insert SO_sw into Archive set
14:            end if
15:        end for
16:    end if
17:    for each particle i in the swarm do
18:        select a leader (gbest) from Archive set for each particle
               using a binary tournament selection based on the objec-
               tives distance;
19:        update the velocity and the position of particle i;
20:        evaluate two objective values for each particles;
21:        update the pbest for each particle;
22:        if the i^th particle is not dominated by any archive members
               then
23:            insert i^th particle into Archive set;
24:        end if
25:    end for
26: end while
27: calculate the testing classification error rate of the solutions in
       Archive set on the test set;
28: return the solutions in Archive;
29: return the training and test classification error rates of the solutions
       in Archive; end
```

4 Design of experiments

The proposed multi-objective PSO-based algorithm is examined and compared with a pure multi-objective PSO algorithm (MOPSO), three well-known evolutionary multi-objective algorithms (NSGAII, SPEA2 and PAES) and a state-of-the-art PSO-based multi-objective feature selection algorithm (CMDPSOFS). The comparison is performed on twelve datasets shown in Table 1, which were selected from the UCI machine learning repository (Asuncion and Newman 2007). These datasets have different numbers of features, classes and instances. For each dataset, all instances are randomly divided into a training set and a test set, which contains 70 and 30 % of the instances, respectively.

All the algorithms are wrapper approaches, which use K nearest neighbour (KNN) as their classification/learning algorithms where $K = 5$. In recent literature Xue et al. (2015b), the generality of wrapper feature selection approaches is discussed. The experimental results show that feature subsets, which are selected by wrappers using simple classification algorithms like KNN, can be general to other classification algorithms. Therefore, in this work, only KNN is used as a classification algorithm to select features and to test the performance of selected feature subset. Another reason using KNN instead of other classifier algorithms, like SVM and DT, is to make the evaluation process faster in this wrapper approach. During the training process, each particle, which represents a set of selected features, is evaluated using tenfold cross-validation. After the training process, the selected features are evaluated on the test set to obtain the testing classification error rate. For each dataset, all of the algorithms have been conducted for 50 independent runs.

In all of the PSO-based algorithms, the fully connected topology is used. The parameters are set as follows (Van Den Bergh 2006): $w = 0.7298$, $c_1 = c_2 = 1.49618$, $v_{\max} = 6.0$. The population size is 30 and the maximum number of iterations is 100. The threshold θ is set as 0.6.

In NSGAII (Deb et al. 2000), SPEA2 (Zitzler et al. 2001) and PAES (Knowles and Corne 1999), the representation of each individual is the same as the GA-based feature selection algorithms (Chakraborty 2002; Hamdani et al. 2007), where each individual is a n-bit binary string and n is the number of available features. Each bit in the representation corresponds to a feature in the original feature set. If a bit value is "1" then the corresponding feature is selected. Otherwise the bit with value "0" indicates that the corresponding feature is not selected. In these algorithms, a bit-flip mutation operator is applied and a single point crossover operator is used in NSGAII and SPEA2. The mutation rate is $\frac{1}{n}$, where n is the number of features in the original set and the crossover rate is 0.9. In these algorithms, the number of individuals is 30 and the number of generations is 100. Therefore, the total number of evaluations is 3000.

Table 1 Datasets

Dataset	#Features	#Classes	#Instances
Wine	13	3	178
Australian	14	2	178
Vehicle	18	4	846
German	24	2	1000
Ionosphere	34	2	351
Lung Cancer	56	3	32
Sonar	60	2	208
Movementlibras	90	15	360
Hillvalley	100	2	606
Musk1	166	2	476
Madelon	500	2	4400
Isolet5	617	2	1559

5 Results and discussion

For each dataset, five multi-objective algorithms, ISRPSO, CMDPSOFS, NSGAII, SPEA2, PAES and CMDPSOFS are conducted for 50 independent runs. After each run, a set of non-dominated solutions are obtained. To compare these algorithms, firstly all 50 archive sets are combined together to create an union set. In this union set, the classification error rate of feature subsets, which share the same number of features, are "averaged". A set of "average" solutions is obtained using the average classification error rate and the corresponding number of features. This average set is called the *"average" Pareto front*. The meaning of *average Pareto front* is the estimation of classification error when the number of features is predefined. In addition, for each dataset, all non-dominated solutions are selected from the union set to create a set of *"best"* solutions, called *"best" set*.

Firstly, ISRPSO is compared with a pure multi-objective PSO algorithm to illustrate how the local search can improve the Pareto front. This comparison on the training and test sets is shown in Figs. 1 and 2, respectively. Meanwhile, Figs. 3 and 4 show the comparison between ISRPSO and NSGAII, SPEA2, PAES on the training and test sets, respectively. Finally, ISRPSO is compared with the state-of-the-art multi-objective PSO-based feature selection, CMDPSOFS (Xue et al. 2013). In particular, the classification error on training set between ISRPSO and CMDPSOFS are shown in the Fig. 5, while Fig. 6 indicates these algorithms' classification performance on the test set. In these figures, "-Ave" stands for the *"average" Pareto front* resulted from the 50 independent runs. "-Best" represents the non-dominated solutions (*"best" set*) of all multi-objective algorithms. In each figure, the top line means the dataset name followed by the total number of features and the classification error rate achieved using all features.

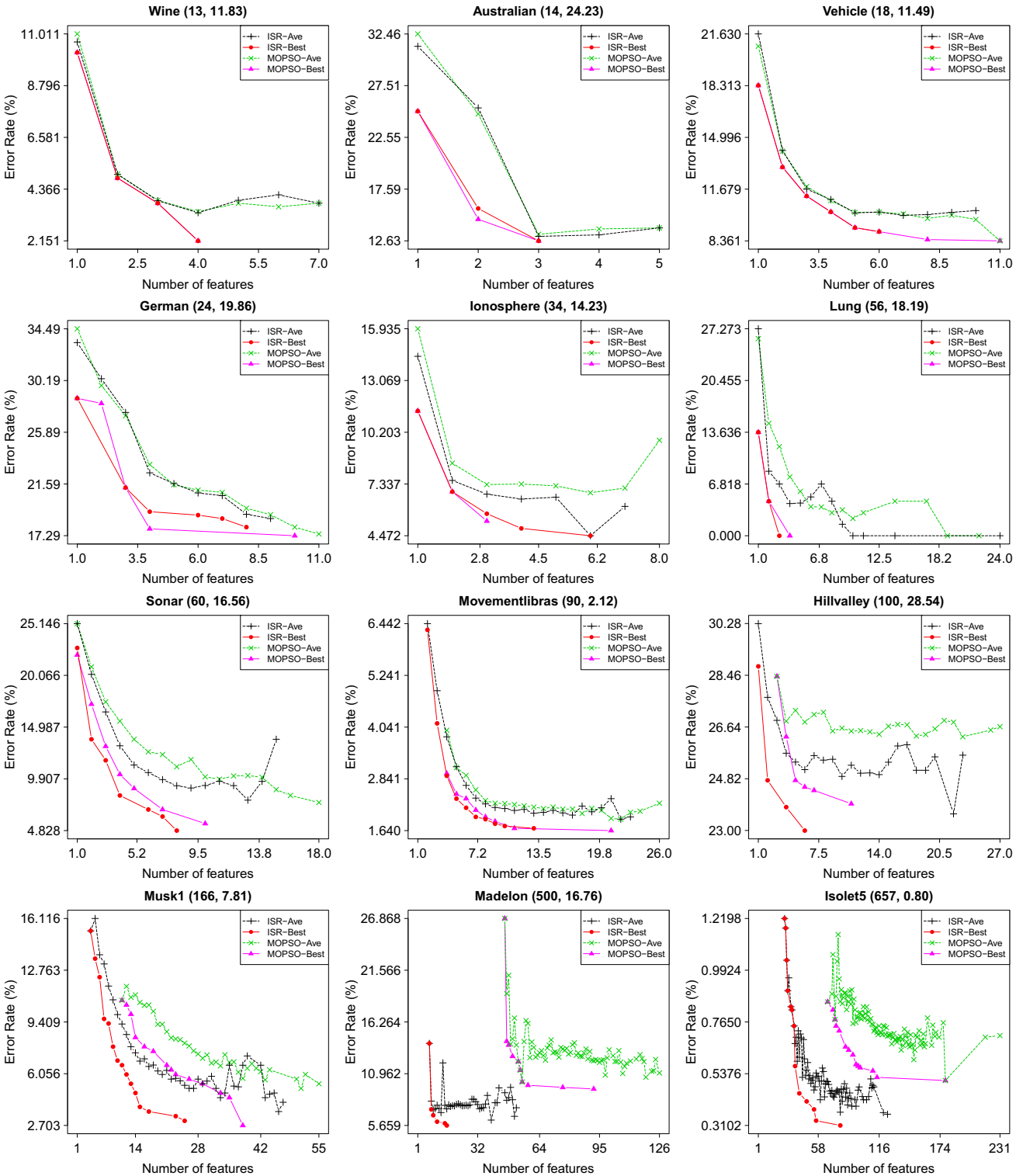

Fig. 1 Comparison between IRPSO and MOPSO on the training set

5.1 Comparison between ISRPSO and using all features

According to Fig. 2, in all cases, ISR-Ave includes three or more feature subsets, which select a smaller number of features and achieve a lower classification error rate than using all features.

As can be seen in Fig. 2, ISR-Best includes two or more solutions, which select a small number of features

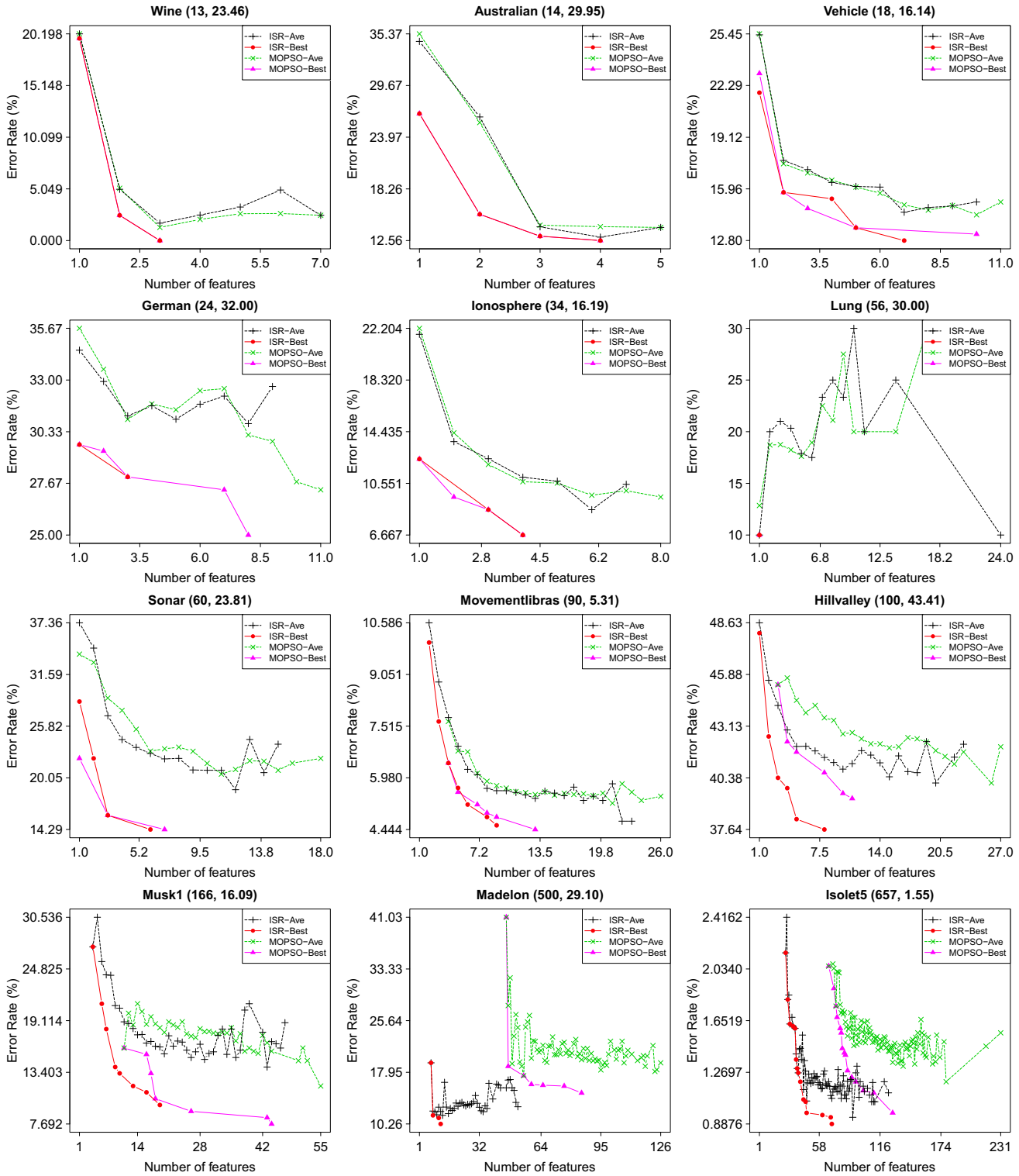

Fig. 2 Comparison between ISRPSO and MOPSO on the test set

while still achieve lower classification error rate than using all features. On all datasets, ISRPSO evolves at least one feature set, which only selects less than 10 % of the total number of features but achieves better classification performance than using all features. Specially, on Madelon dataset, despite of selecting only 5 out of 500 features, ISRPSO still can achieve around 10.05 % better than using all features.

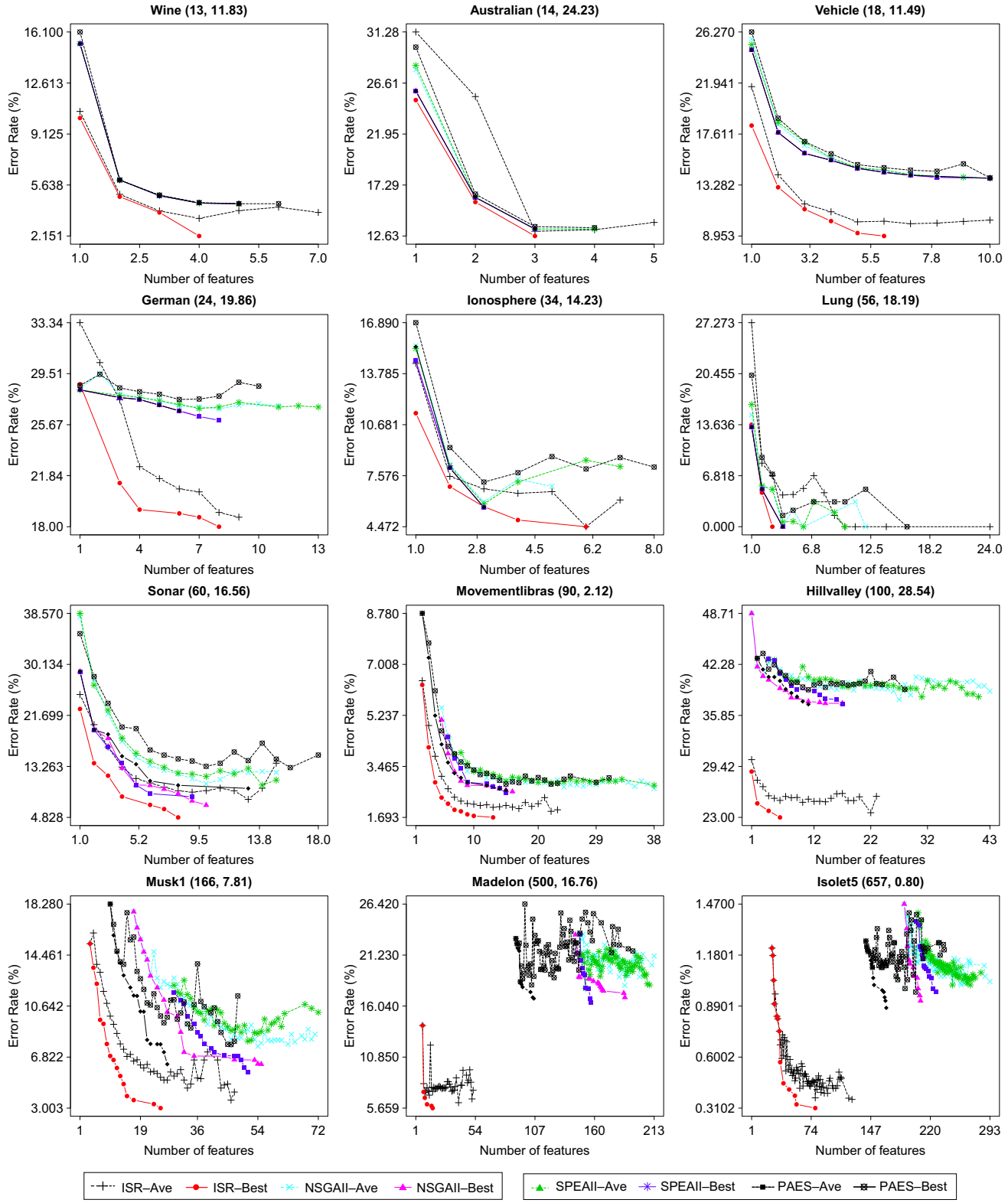

Fig. 3 Comparisons between NSGAII, SPEA2, PAES and ISRPSO on the training set

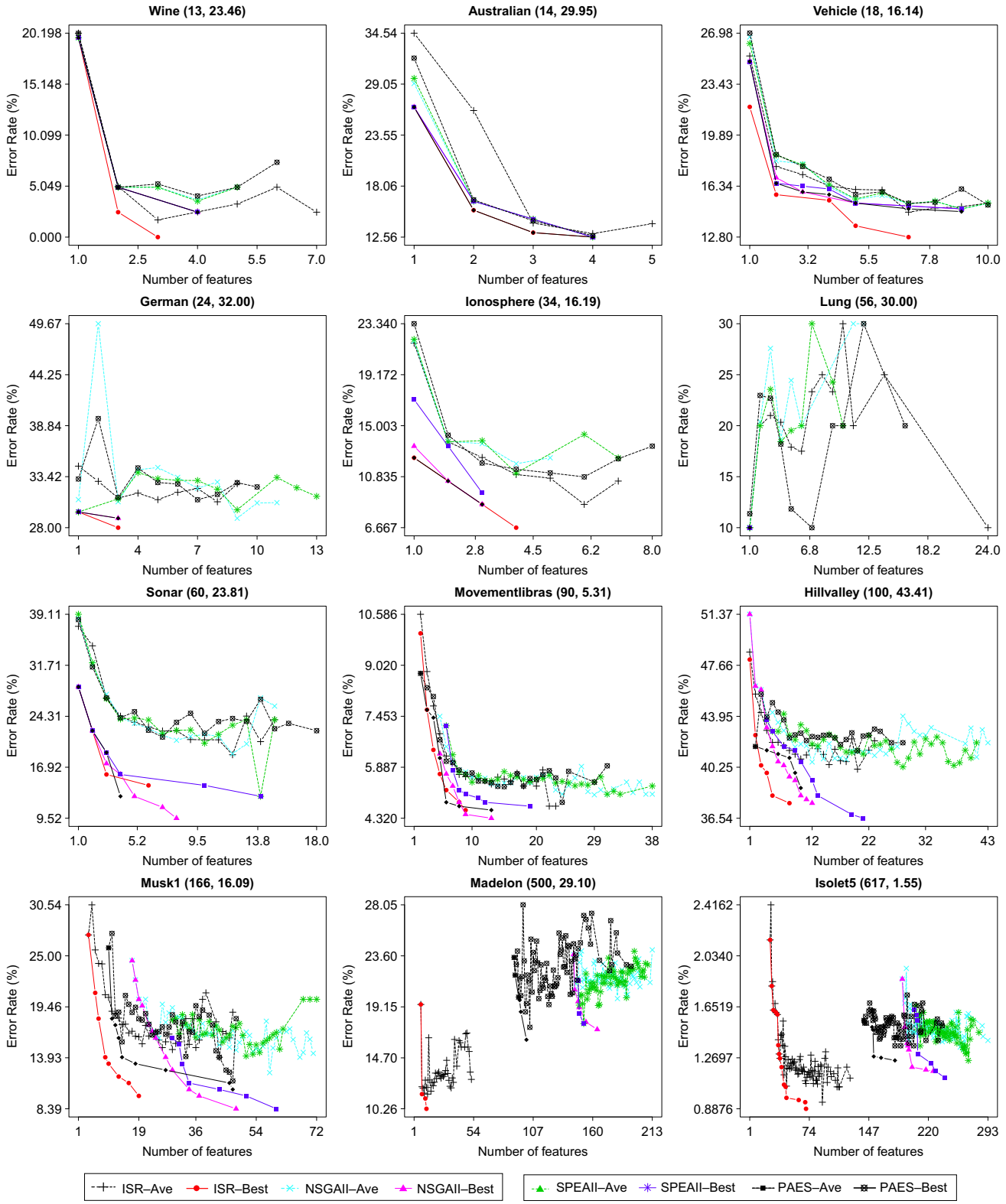

Fig. 4 Comparisons between NSGAII, SPEA2, PAES and ISRPSO on the test set

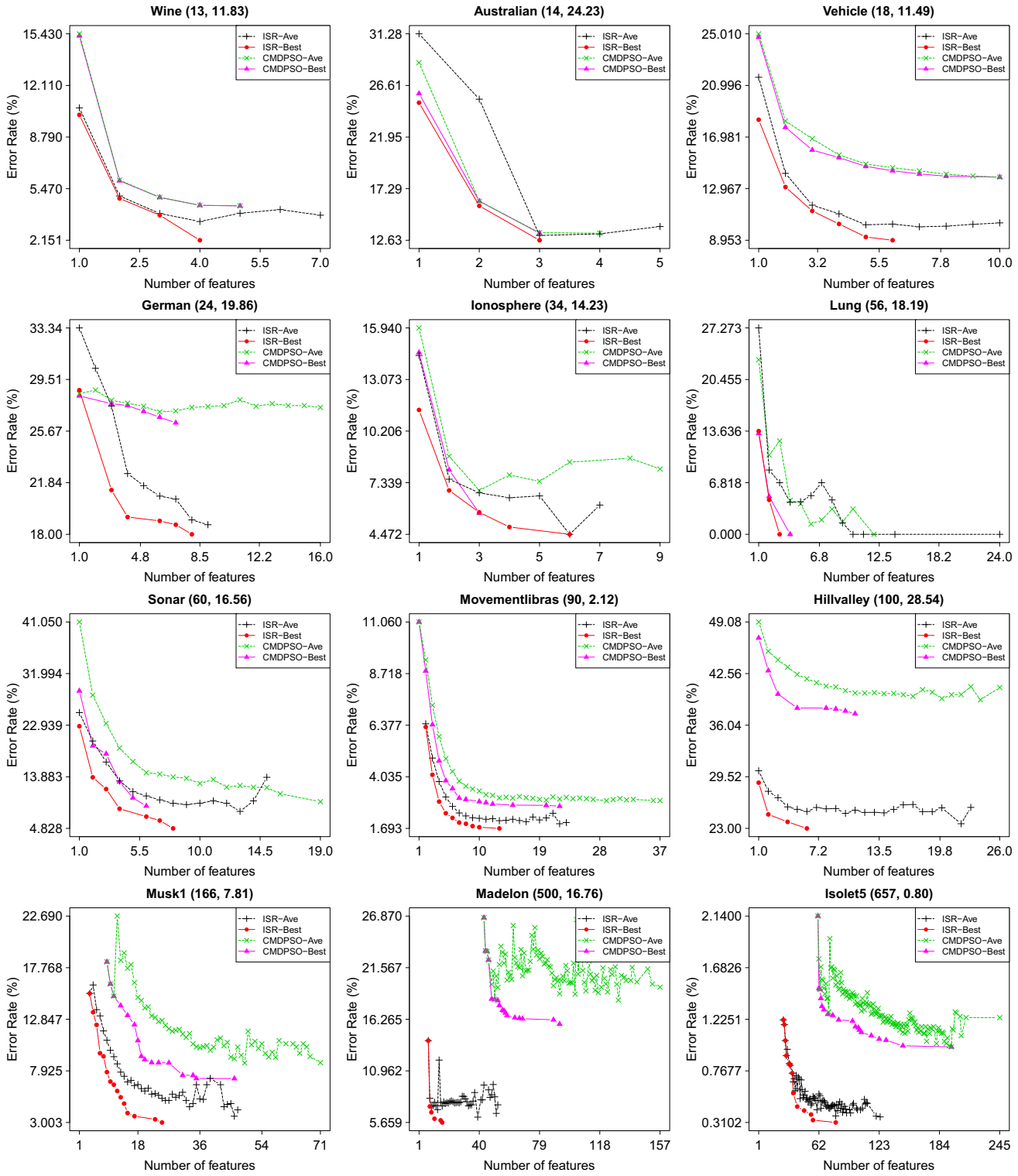

Fig. 5 Comparison between ISRPSO and CMDPSOFS on the training set

The results suggest that ISRPSO with three operations applied on the *Archive* set can effectively explore the Pareto front, which can select a small number of features and achieve better classification accuracy than using all features.

5.2 Comparison between ISRPSO and a pure multi-objective PSO (MOPSO)

To test the effect of the local search applied in ISRPSO, ISRPSO is compared with a pure multi-objective PSO algo-

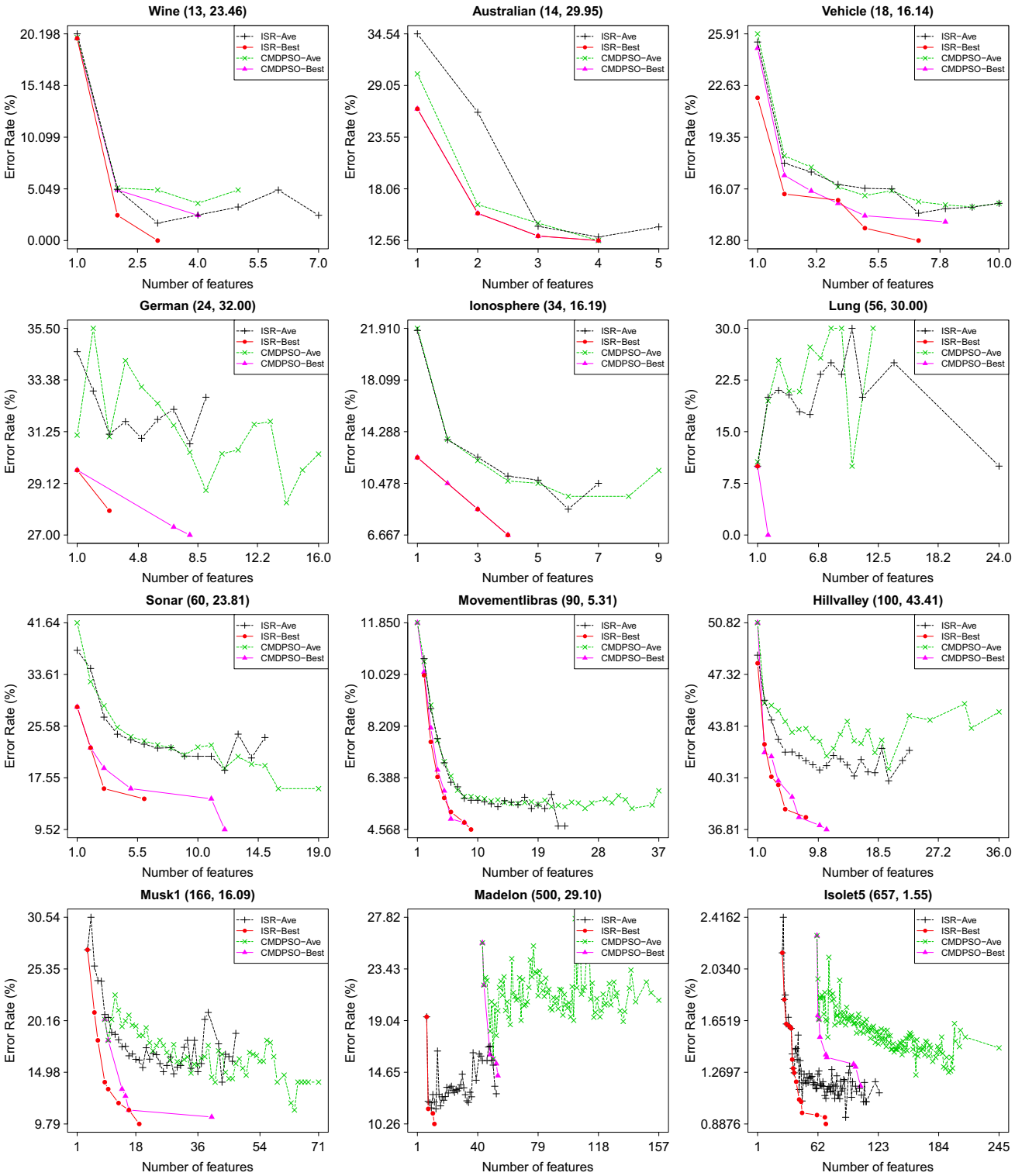

Fig. 6 Comparison between ISRPSO and CMDPSOFS on the test set

rithm (MOPSO) for feature selection. The comparisons on the training and the test set are shown in Figs. 1 and 2, respectively.

As can be seen from two figures, on the first six datasets in which the number of features is quite small, the Pareto fronts evolved by these two algorithms are similar. It is clear that

on small datasets, MOPSO is able to effectively explore the search space to produce an optimal or near optimal Pareto front without any additional help. However, the big differences between these two algorithms occur on the other six datasets, which have larger number of features than the first six datasets. As can be seen from two Figs. 1 and 2, in both training and test set, ISRPSO outperforms MOPSO in terms of the number of selected features as well as the classification performance. For example, in the Hillvalley dataset, even selecting the same number of features, ISRPSO always achieves 2–3 % better classification accuracy than MOPSO. Similarly, in the Musk1 dataset, ISRPSO only selects around 7 features to achieve the same performance as 20 features selected by MOPSO. The biggest difference can be seen in the Madelon dataset. ISRPSO selects only 10 features but still achieves 5 % better performance than 90 features selected by MOSPO. Similarly, in the Isolet5 dataset, ISRPSO is able to evolve at least three solutions with very low classification error, which cannot be achieved by MOPSO even when a large number of features are selected.

The larger the number of features, the more complicated the search space. Therefore, in the datasets with a large number of features, MOPSO tends to stick at local optima. In these cases, applying local search techniques in MOPSO will help the particle to exploit better solutions, which results in a better Pareto front. Specifically, three proposed operations try to improve the classification performance of each archive member while maintaining or even reducing the number of selected features using the set of promising features. The idea is by improving the archive set, the whole swarm will be guided to better positions. The experimental results show that the proposed local search technique is helpful for MOPSO to better exploit the Pareto front.

5.3 Comparison between ISRPSO and NSGAII, SPEA2, PAES

To test the performance of ISRPSO, it is compared with three popular evolutionary multi-objective algorithms, namely, NSGAII, SPEA2 and PAES. Comparisons between ISRPSO, NSGAII, SPEA2 and PAES on training and test set are shown in Figs. 3 and 4, respectively.

According to Fig. 3, on the training set, in most cases, the average Pareto fronts evolved by ISRPSO are much better than the other multi-objective algorithms. Similarly, ISRPSO's best Pareto fronts provide better solutions than NSGAII, SPEA2 and PAES. Particularly, on the small datasets (the first six datset in Table 1), the ISRPSO's Pareto fronts are quite similar to other algorithms' Pareto front, except in Vehicle dataset, ISRPSO's accuracy is about 9 % better than other algorithms' accuracy while they select the same number of features. On the other big dataset, ISRPSO's Pareto fronts are significantly better than NSGAII and SPEA2. With the same number of selected features, ISRPSO's accuracy is about 2–15 % better than the best accuracy of NSGAII, SPEA2 and PAES. Especially, on large dataset like Madelon and Isolet5, ISRPSO is able to find out solutions with a very small number of features, which cannot be found by either NSGAII, SPEA2 or PAES. In addition, on Madelon dataset, the worst solution of ISRPSO significantly dominates the best solution of NSGA, SPEA2 and PAES algorithms.

As can be seen in Fig. 4, in most cases, ISRPSO's Pareto front is similar or better than other multi-objective algorithms. Particularly, ISRPSO usually selects a smaller number of features to achieve the same accuracy with NSGA, SPEA or PAES. However, in the first nine small datasets, sometimes ISRPSO sacrifices the classification accuracy to select a small number of features. For example, in Hillvalley dataset, ISRPSO cannot find out the solution which achieves higher classification performance than the best solution of PAES. On the other datasets (big datasets), ISRPSO's Pareto front is significantly better than the other three algorithms. For example, in the Isolet5 dataset, the lowest classification error achieved by ISRPSO is 0.88 % with 74 selected features. Meanwhile, the lowest classification error achieved by NSAII is 1.2 %, which requires more than 220 features. In addition, on the Madelon dataset, the lowest classification accuracy, achieved by ISRPSO, is 80.5 % with only two selected features. Meanwhile, 80.5 % is also the best performance that NSGAII can achieve but NSGAII needs to select a large feature set, which contains more than 160 features.

The results in Figs. 3 and 4 suggest that ISRPSO can effectively improve the Pareto front. In most cases, ISRPSO outperforms NSGAII, SPEA2 and PAES in both objectives. However, on Hillvalley dataset, ISRPSO sacrifices the classification accuracy to achieve smaller number of features. The reason is that the proposed algorithm lacks exploration ability. Since the local search operations aim to add or remove at most one feature from the current feature set, the operations can only find out the better subset which are close to the current position but better performance than the current feature set. Therefore, the positions, which correspond to a much higher number of features than the current solution, are too far and hard to be discovered by the local search operations. However, with the same number of selected features, the local search operations ensure that the better feature combination will be discovered. Therefore, ISRPSO usually achieves better performance than other multi-objective algorithms when the same number of features are selected.

5.4 Comparison between ISRPSO and CMDPSOFS

Figure 5 indicates the performance of ISRPSO and CMDPSOFS on the training set. As can be seen from Fig. 5, on all

datasets, ISRPSO selects a smaller number of features and still achieves better classification performance than CMDPSOFS. With the same number of selected features, ISRPSO outperforms CMDPSOFS in terms of classification accuracy. For example, in the Hillvalley, with seven selected features, ISRPSO's performance is about 14 % better than CMDPSOFS's classification accuracy. In most cases, the ISRPSO's average Pareto front is even better than the best Pareto front evolved by CMDPSOFS. In Madelon dataset, despite of selecting only five features, ISRPSO still achieves better performance than the performance of 100 features selected by CMDPSOFS. To achieve same classification accuracy as ISRPSO, CMDPSOFS usually needs to select much more features than ISRPSO. For example, in Movementlibras dataset, CMDPSOFS selects from 2 to 8 times more features than ISRPSO.

Figure 6 shows the differences between ISRPSO and CMDPSOFS on the test set. In the first ten datasets, the average Pareto fronts of ISRPSO and CMDPSOFS are similar in terms of classification accuracy but ISRPSO usually selects smaller number of features than CMDPSOFS. Similarly, on the first ten datasets, ISRPSO's best Pareto fronts are almost identical to the best Pareto fronts from CMDPSOFS. Although, in the best Pareto fronts, ISRPSO still selects slightly smaller number of features than CMDPSOFS. However, on some datasets like Sonar or German, ISRPSO cannot find out the solution which selects a large number of features but achieves low classification rate. Once more, ISRPSO sacrifices the classification accuracy to select a small number of features. On the other big datasets, namely Madelon and Isolet5, ISRPSO outperforms CMDPSOFS in terms of the number of selected features and the classification performance. For example, in Madelon dataset, to achieve 19 % classification error rate, ISRPSO selects only 5 features while CMDPSOFS needs to select more than 40 features. In Isolet5, with 63 selected features, ISRPSO's classification error rate is 0.88 % which is much smaller than CMDPSOFS's error rate (1.3 %).

The results in Figs. 5 and 6 show that in most cases ISRPSO evolves similar or better Pareto fronts than CMDPSOFS. Specially, in the datasets with a large number of features, ISRPSO significantly outperforms CMDPSOFS in terms of classification accuracy and number of selected features. This results illustrate that for a complicated search space (high dimension) the proposed local search technique, which is specifically designed for feature selection, provides a better support than crossover and mutation operators being used in CMDPSOFS. Particularly, CMDPSOFS uses a general mutation operator on the whole swarm to improve both global and local search abilities of the algorithm. However, the mutation operator in CMDPSOFS is not specifically designed for feature selection problems. In our algorithm, the local search technique uses the sequential forward/backward idea to add/remove features from the current feature set. In addition, the local search technique also uses a good feature set (all features selected by the archive set) to improve the current feature subsets in the archive set, which is likely to be faster and more effective than using all features, like in CMDPSOFS. Furthermore, our local search concentrates on improving the quality of the archive set rather than the whole swarm like CMDPSOFS. The reason is a good archive set is expected to guide the whole swarm to discover better feature subsets and evolve a more optimal Pareto front. However, the improvement over CMDPSOFS made by ISRPSO on the training set is much better than on the test set, which might be a result of overfitting problem.

Table 2 Comparisons on computational times (in minutes)

	ISRPSO	CMD	NSGA2	PAES	SPEA2
Wine	0.19	0.25	0.24	0.21	0.20
Australian	2.41	4.0	3.99	3.95	3.24
Vehicle	7.6	6.59	6.59	6.35	5.53
German	7.75	9.3	9.32	8.85	7.64
Ionosphere	0.67	1.54	1.06	1.04	0.87
Lung Cancer	0.01	0.01	0.01	0.01	0.01
Sonar	0.49	0.49	0.49	0.48	0.43
Movement	3.54	2.08	2.37	1.97	2.09
Hillvalley	121.39	23.49	28.88	24.88	30.23
Musk1	38.21	6.02	7.46	5.73	6.69
Madelon	5166.41	523.52	713.4	609.38	716.09
Isolet5	4090.83	198.67	323.94	283.24	327.78

5.5 Comparisons on computational time

Table 2 shows the average computational time (in minutes) used by ISRPSO, CMDPSOFS (CMD), NSGA2, PAES and SPEA2 in one run.

It can be seen that on the first seven datasets, ISRPSO uses similar or less time than CMDPSOFS, NSGA2, PAES and SPEA2. However, on the datasets having a large number of features, ISRPSO is more expensive than the other algorithms. The reason is that three local search operators' cost depends on the total number of features. Particularly, suppose that in the worst case, the archive set (A) selects all N original features. For each archive member, the inserting and removing operators, respectively, consider the features, which are selected and not selected by the archive member. In the worst case when all features are considered to be added/removed, the number of evaluations in the two operators equals to the total number of features, which is N. The swapping operator tries to swap each pair of selected and unselected feature. In the worst case, all pairs are tested, which leads to the

number of evaluations $|U| * |US|$, where $|U| + |US| = N$. When $|U| = |US| = \frac{N}{2}$, the number of evaluations in the swapping operator reaches the highest (worst) value, which is $\frac{N^2}{4}$. In the worst case, the local search technique needs to evaluate $N + \frac{N^2}{4}$ times. So in general, for each iteration the number of evaluations caused by local search operators is: $|A| * (N + \frac{N^2}{4})$, where $|A|$ is the size of archive set. Meanwhile, in CMDPSOFS, the mutation operators are applied to each individual, so in each iteration the number of evaluations caused by the mutation operator is equal to the population size. Therefore, when the number of original features is much more than the swarm size, ISRPSO becomes more expensive than CMDPSOFS. In general, ISRPSO achieves better classification accuracy than the other algorithms, it is efficient only on the small datasets. On the large datasets, although ISRPSO selects a smaller number of features than other algorithms to reduce the evaluation cost, it is still expensive when there is a large number of original features. We will investigate how to reduce the computation cost of the local search operators.

6 Conclusions and future work

The goal of this study was to develop a PSO-based multi-objective feature selection approach, which evolved a set of non-dominated feature with high classification performance. The algorithm ISRPSO was proposed to simultaneously minimize the number of features and the classification error rate. In ISRPSO, three operations, namely *inserting, removing* and *swapping* are applied on each archive member to better explore the Pareto front. The experiments on 12 datasets show that ISRPSO successfully evolved a set of non-dominated solutions, which reduced the number of features and achieved better classification performance than using all features. In comparison with a pure multi-objective PSO (MOSPO), ISRPSO usually evolved better Pareto front especially on the datasets with a large number of features, which indicated that the proposed local search was useful for feature selection problems. Furthermore, the results showed that ISRPSO outperformed three well-known evolutionary multi-objective algorithms (NSGAII, SPEA2, PAES) in terms of both classification accuracy and the number of selected features. Compared with CMDPSOFS, ISRPSO achieved similar or better classification accuracy while selected smaller number of features. Especially, for the datasets with a large number of features, ISRPSO significantly outperformed all other algorithms.

However, there are some potential limitations of ISRPSO. Firstly, overfitting is one of ISRPSO's problem. As can be seen from the results, sometimes the significant improvement made by ISRPSO on the training set does not lead to a significant improvement in the test set. Therefore, in the future, we will further develop ISRPSO to improve its generalization. In addition, for some datasets, ISRPSO seems to sacrifice the classification accuracy to select a small number of features. It is necessary to further balance between these two objectives for ISRPSO, which is left for future work. In addition, as can be seen from the experimental results, both genetic operators such as crossover, mutation and three local search operators helps MOSPO to better explore the Pareto front. Therefore, the combination between these two ideas might even better support MOPSO, which will be investigated in the future. Additionally, ISRPSO is not tested with high-dimensional datasets mainly because of the expensive computation cost of wrapper approaches. In the future, it would be interesting to reduce the computation cost of ISRPSO so that it can be extended to deal with high-dimensional datasets such as gene datasets. Although the experimental results show that ISRPSO could evolve a good Pareto front, the computation complexity of the algorithm is not analysed in detail due to the variation in the fitness evaluation process of feature selection problems. Investigating the complexity of a feature selection algorithm is still an open issue, which is left for future work.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Asuncion A, Newman D (2007) Uci machine learning repository

Bhowan U, McCloskey D (2015) Genetic programming for feature selection and question-answer ranking in ibm watson. In: Genetic Programming. Springer, New York, pp 153–166

Bin W, Qinke P, Jing Z, Xiao C (2012) A binary particle swarm optimization algorithm inspired by multi-level organizational learning behavior. Eur J Oper Res 219(2):224–233

Boubezoul A, Paris S (2012) Application of global optimization methods to model and feature selection. Pattern Recogn 45(10):3676–3686

Cervante L, Xue B, Zhang M, Shang L (2012) Binary particle swarm optimisation for feature selection: a filter based approach. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp 1–8

Chaaraoui AA, Flórez-Revuelta F (2013) Human action recognition optimization based on evolutionary feature subset selection. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation (GECCO). ACM, pp 1229–1236

Chakraborty B (2002) Genetic algorithm with fuzzy fitness function for feature selection. In: Proceedings of the 2002 IEEE International Symposium on Industrial Electronics (ISIE), vol 1, pp 315–319

Chakraborty B, Chakraborty G (2013) Fuzzy consistency measure with particle swarm optimization for feature selection. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp 4311–4315

Chuang LY, Chang HW, Tu CJ, Yang CH (2008) Improved binary pso for feature selection using gene expression data. Comput Biol Chem 32(1):29–38

Dash M, Liu H (1997) Feature selection for classification. Intell Data Anal 1(3):131–156
Deb K, Agrawal S, Pratap A, Meyarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. Lect Notes Comput Sci 1917:849–858
Gheyas IA, Smith LS (2010) Feature subset selection in large dimensionality domains. Pattern Recogn 43(1):5–13
Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
Hamdani TM, Won JM, Alimi AM, Karray F (2007) Multi-objective feature selection with nsga ii. In: Adaptive and Natural Computing Algorithms. Springer, New York, pp 240–247
Hancer E, Xue B, Karaboga D, Zhang M (2015) A binary abc algorithm based on advanced similarity scheme for feature selection. Appl Soft Comput 36:334–348
Huang CL, Dun JF (2008) A distributed pso-svm hybrid system with feature selection and parameter optimization. Appl Soft Comput 8(4):1381–1391
Huang CL, Wang CJ (2006) A ga-based feature selection and parameters optimizationfor support vector machines. Expert Syst Appl 31(2):231–240
Kennedy J, Eberhart R et al (1995) Particle swarm optimization. Proc IEEE Int Conf Neural Netw 4:1942–1948
Knowles J, Corne D (1999) The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In: Congress on Evolutionary Computation, vol 1. IEEE
Lane MC, Xue B, Liu I, Zhang M (2013) Particle swarm optimisation and statistical clustering for feature selection. In: AI 2013: Advances in Artificial Intelligence. Springer, New York, pp 214–220
Lane MC, Xue B, Liu I, Zhang M (2014) Gaussian based particle swarm optimisation and statistical clustering for feature selection. In: Evolutionary computation in combinatorial optimisation. Springer, New York, pp 133–144
Lee S, Soak S, Oh S, Pedrycz W, Jeon M (2008) Modified binary particle swarm optimization. Prog Nat Sci 18(9):1161–1166
Li X (2003) A non-dominated sorting particle swarm optimizer for multiobjective optimization. In: Proceedings of the 5th annual conference on Genetic and Evolutionary Computation (GECCO). Springer, New York, pp 37–48
Liang D, Tsai CF, Wu HT (2015) The effect of feature selection on financial distress prediction. Knowl-Based Syst 73:289–297
Lin F, Liang D, Yeh CC, Huang JC (2014) Novel feature selection methods to financial distress prediction. Expert Syst Appl 41(5):2472–2483
Lin SW, Ying KC, Chen SC, Lee ZJ (2008) Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Syst Appl 35(4):1817–1824
Liu Y, Wang G, Chen H, Dong H, Zhu X, Wang S (2011) An improved particle swarm optimization for feature selection. J Bionic Eng 8(2):191–200
Marill T, Green DM (1963) On the effectiveness of receptors in recognition systems. IEEE Trans Inf Theory 9(1):11–17
Mohemmed AW, Zhang M, Johnston M (2009) Particle swarm optimization based adaboost for face detection. In: IEEE Congress on Evolutionary Computation (CEC), pp 2494–2501
Neshatian K, Zhang M (2009a) Dimensionality reduction in face detection: a genetic programming approach. In: IEEE 24th International Conference on Image and Vision Computing New Zealand (IVCNZ'09), pp 391–396
Neshatian K, Zhang M (2009b) Genetic programming for feature subset ranking in binary classification problems. In: Genetic programming. Springer, New York, pp 121–132
Nguyen H, Xue B, Liu I, Zhang M (2014a) Filter based backward elimination in wrapper based pso for feature selection in classification. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp 3111–3118
Nguyen HB, Xue B, Liu I, Zhang M (2014b) Pso and statistical clustering for feature selection: a new representation. In: Simulated evolution and learning. Springer, New York, pp 569–581
Nguyen HB, Xue B, Liu I, Andreae P, Zhang M (2015) Gaussian transformation based representation in particle swarm optimisation for feature selection. In: Applications of evolutionary computation. Springer, New York, pp 541–553
Oreski S, Oreski G (2014) Genetic algorithm-based heuristic for feature selection in credit risk assessment. Expert Syst Appl 41(4):2052–2064
Pudil P, Novovičová J, Kittler J (1994) Floating search methods in feature selection. Pattern Recogn Lett 15(11):1119–1125
Seo JH, Lee YH, Kim YH (2014) Feature selection for very short-term heavy rainfall prediction using evolutionary computation. Adv Meteorol 2014:203545. doi:10.1155/2014/203545
Stearns SD (1976) On selecting features for pattern classifiers. In: Proceedings of the 3rd International Conference on Pattern Recognition (ICPR 1976), pp 71–75
Tran B, Xue B, Zhang M (2014) Improved pso for feature selection on high-dimensional datasets. In: Simulated evolution and learning. Springer, New York, pp 503–515
Unler A, Murat A (2010) A discrete particle swarm optimization method for feature selection in binary classification problems. Eur J Oper Res 206(3):528–539
Van Den Bergh F (2006) An analysis of particle swarm optimizers. Ph.D. thesis, University of Pretoria
Vieira SM, Mendonça LF, Farinha GJ, Sousa JM (2013) Modified binary pso for feature selection using svm applied to mortality prediction of septic patients. Appl Soft Comput 13(8):3494–3504
Wang L (2005) A hybrid genetic algorithm-neural network strategy for simulation optimization. Appl Math Comput 170(2):1329–1343
Wang X, Yang J, Teng X, Xia W, Jensen R (2007) Feature selection based on rough sets and particle swarm optimization. Pattern Recogn Lett 28(4):459–471
Whitney AW (1971) A direct method of nonparametric measurement selection. IEEE Trans Comput 100(9):1100–1103
Xue B, Cervante L, Shang L, Browne WN, Zhang M (2012a) A multi-objective particle swarm optimisation for filter-based feature selection in classification problems. Connect Sci 2–3:91–116
Xue B, Zhang M, Browne W (2012b) New fitness functions in binary particle swarm optimisation for feature selection. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp 2145–2152
Xue B, Zhang M, Browne WN (2013) Particle swarm optimization for feature selection in classification: a multi-objective approach. IEEE Trans Cybern 43(6):1656–1671
Xue B, Zhang M, Browne WN (2014) Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms. Appl Soft Comput 18:261–276
Xue B, Zhang M, Browne W, Yao X (2015a) A survey on evolutionary computation approaches to feature selection. IEEE Trans Evol Comput (99):1–1. doi:10.1109/TEVC.2015.2504420
Xue B, Zhang M, Browne WN (2015b) A comprehensive comparison on evolutionary feature selection approaches to classification. Int J Comput Intell Appl 14(02):1550008
Yang CS, Chuang LY, Ke CH, Yang CH (2008) Boolean binary particle swarm optimization for feature selection. In: 2008 IEEE Congress on Evolutionary Computation (CEC), pp 2093–2098
Yuan H, Tseng SS, Gangshan W, Fuyan Z (1999) A two-phase feature selection method using both filter and wrapper. In: 1999 IEEE International Conference on Systems, Man, and Cybernetics (SMC), vol 2, pp 132–136

Zhang Y, Gong D, Hu Y, Zhang W (2015) Feature selection algorithm based on bare bones particle swarm optimization. Neurocomputing 148:150–157

Zhao H, Sinha AP, Ge W (2009) Effects of feature construction on classification performance: an empirical study in bank failure prediction. Expert Syst Appl 36(2):2633–2644

Zhu Z, Ong YS, Dash M (2007) Wrapper-filter feature selection algorithm using a memetic framework. IEEE Trans Syst Man Cybern 37(1):70–76

Zitzler E, Laumanns M, Thiele L (2001) Spea 2: improving the strength pareto evolutionary algorithm