A Self-adaptive Multi-objective Feature Selection Approach for Classification Problems

Yu Xue [a,b], Haokai Zhu [a] and Ferrante Neri [c,1]

[a] *School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China*

[b] *Engineering Research Center of Digital Forensics, Ministry of Education, Nanjing University of Information Science and Technology, Nanjing, China*

[c] *COL Laboratory, School of Computer Science, University of Nottingham, Nottingham, UK*

Abstract. In classification tasks, feature selection (FS) can reduce the data dimensionality and may also improve classification accuracy, both of which are commonly treated as the two objectives in FS problems. Many meta-heuristic algorithms have been applied to solve the FS problems and they perform satisfactorily when the problem is relatively simple. However, once the dimensionality of the datasets grows, their performance drops dramatically. This paper proposes a self-adaptive multi-objective genetic algorithm (SaMOGA) for FS, which is designed to maintain a high performance even when the dimensionality of the datasets grows. The main concept of SaMOGA lies in the dynamic selection of five different crossover operators in different evolution process by applying a self-adaptive mechanism. Meanwhile, a search stagnation detection mechanism is also proposed to prevent premature convergence. In the experiments, we compare SaMOGA with five multi-objective FS algorithms on sixteen datasets. According to the experimental results, SaMOGA yields a set of well converged and well distributed solutions on most data sets, indicating that SaMOGA can guarantee classification performance while removing many features, and the advantage over its counterparts is more obvious when the dimensionality of datasets grows.

Keywords. Feature selection, self-adaptive, multi-objective genetic algorithm, stagnation detection, classification

1. Introduction

As a crucial branch of machine learning, classification has received great attention [1,2,3]. The models used for classification are often referred to as classifiers [4,5]. In fact, one instance, which serves as the input of the classifiers, is always composed of a set of features and its label. To better solve the classification problems, a large amount of features are often included, however, most of them are irrelevant or redundant in many cases, resulting in the possibility of reduction of classification accuracy, model complexity, etc [6,7,8]. Classification is a fundamental task in machine learning [9,10] which enables a range of applications such as medicine [11,12,13,14] with numerous application in diagnostics based on electroencephalogram [15,16,17]. Other studies, more broadly, focus on neurobiology [18,19].

In the early days, researchers either optimized only one objective, i.e., classification accuracy, or aggregated multiple objectives into a single objective for optimization [20]. These practices often cause some weaknesses such as inferior optimization results. Recently, FS has been considered as a multi-objective optimization problem (MOP), see [21,22]. Meanwhile, metaheuristic techniques go viral, and they have been applied in many fields [23,24,25,26,27]. Metaheuristic techniques include for example evolutionary computation (EC) techniques [28] such as genetic algorithm (GA) [29], ant colony optimization [30], particle swarm optimization (PSO) [31,32,33], differential evolution [34], artificial bee colony [35], ant colony optimization [36,37], pattern search [38] etc. The main reasons for their popularity are: (i) there is no need of prior knowledge of the problem; (ii) the population-

[1] Corresponding Author: F. Neri, School of Computer Science, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK; Email: ferrante.neri@nottingham.ac.uk

Y. Xue and H. Zhu equally contributed to this work and should be considered *co-first* authors

based search approach is capable of obtaining multiple solutions in a single run; (iii) they specialize in global search. Benefit from these advantages, there have been many attempts to solve FS with EC techniques [39,40]. Among these methods, GA is often used due to its simple encoding scheme, excellent generalization and global search strategy [41].

Generally, most of the multi-objective FS methods based on GA are using non-dominated sorting genetic algorithm II (NSGA-II) [42,43]. However, most of existing studies only use the original algorithm to solve FS, or just study the population size, crossover probability, and mutation probability of the algorithm. In particular, only a single crossover operator with a single mutation operator is usually used in the search process. As the number of features (NF) increases, the search space accordingly grows at a tremendous rate [44,45,46]. For instance, suppose NF is n, the number of all possible feature subsets is $2^n - 1$. If only a single crossover operator is used for the entire evolutionary process, then the superior genes of the parents may not necessarily be retained to the offspring in some cases, which leads the algorithm to get stalled and end up finding the local optima. Indeed, selection, crossover, and mutation operations in genetic algorithms play a very important role, and all of these three operations contribute to the final optimization result. In this paper, we only focus on the crossover operations and try to improve the behavior of the crossover operation to get better performance of the algorithm.

In this work, we treat FS as a MOP, where classification error and solution size are considered as two objective functions. To solve it, a self-adaptive multi-objective genetic algorithm (SaMOGA) is proposed. Different from previous studies based on NSGA-II for FS, in SaMOGA, five crossover operators with different search characteristics work in conjunction with one mutation operator by applying a self-adaptive mechanism. Concretely speaking, the performance of each crossover operator in the evolutionary process is recorded, and at different evolutionary stages, the self-adaptive mechanism selects the currently preferred crossover operator for the crossover operation based on their previous performance, followed by the mutation operation. Besides, a search stagnation detection mechanism (SSDM) is also proposed to detect the search stagnation, with which the exploration can be more effectively. To verify the efficiency of SaMOGA, we conducted experiments on sixteen datasets varying widely in dimensionality, and the obtained results are also compared with five multi-objective FS algorithms.

The contributions of this work are listed as follows:

- The most appropriate crossover operator is used at different stages using an self-adaptive mechanism
- The self-adaptive mechanism and SSDM are combined to improve search capabilities.

2. Related Work

In this section, the introduction of the concept of multi-objective optimization and NSGA-II is firstly given. Subsequently, a brief review of FS approaches using multi-objective GAs is provided.

2.1. Multi-objective Optimization Problems

Some problems have multiple objectives to be optimized [47], and these objectives are often contradictory to each other, in other words, they cannot obtain the best optimization results for each of them at the same time. Such problems are called MOPs [48,49,50, 51]. The mathematical expression of a rough multi-objective minimization problem is shown as follows:

$$\begin{aligned} &min F(X) = (f_1(X), f_2(X), ..., f_m(X)) \\ &s.t.\ X \in \Omega \subseteq R^n \end{aligned} \tag{1}$$

where $X = (x_1, x_2, ..., x_n)$ is a solution to the problem that falls within the search space Ω, x_i means the i^{th} decision variable, and $F(X)$ represents the objective functions, see [52,53,54].

2.2. NSGA-II

NSGA-II is one of the most classic evolutionary algorithms and is an improved version of traditional genetic algorithms for tackling MOPs [55]. It incorporates the Pareto dominance relationship and the crowding distance mechanism to drive the evolution of pop-

ulation so that a set of trade-off solutions, which are also named as Pareto front (PF), can be obtained. The key flow of the algorithm is simply described in the following paragraph.

Let's assume that the size of parent population in t^{th} iteration P_t is N. After undergoing the genetic operations, i.e., crossover and mutation, N offspring are generated that make up the offspring population Q_t. Then, these solutions are combined together to form the combination population R_t and the next generation of individuals are selected from R_t according to the following steps. Firstly, individuals in R_t are classified into multiple hierarchies by applying non-dominated sorting procedure. Then, individuals are selected into the next generation in a hierarchical order. As can be seen from Figure 1, individuals in H_1 - H_3 have been added into the next generation. When H_4 is about to be added, the size of population would exceed the predetermined one, in which case, the individuals with the bigger crowded distance have priority in entering the next generation P_{t+1}.

The non-dominated sorting, crowding distance calculation are summarized as follows:

2.2.1. *Non-dominated sorting*

Firstly, for each solution p in the population, two properties are calculated, one is domination count n_p, i.e., the number of solutions which dominate the solution p, another one is S_p, i.e., a set of solutions that dominated by the solution p. Since the domination counts of the solutions in the first non-dominant front equal to 0, for each solution p with $n_p = 0$, visit each member q in its S_p and reduce 1 from its domination count, if the domination count of q becomes 0, it is put in a list Q. Thus, Q includes the solutions belong to the second non-dominated front. After that, the process is repeated for the solutions in queue Q till the third front is found. This process continues till all the fronts are found.

2.2.2. *Crowding distance calculation*

To begin with, the population is sorted according to each objective function values in ascending order. After that, for each objective function, the boundary solutions are assigned an infinite distance value, and each mediate solution is assigned a distance value equal to

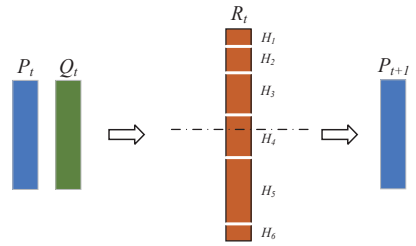


Figure 1. The Brief Evolution Process of NSGA-II

the absolute normalized difference of two adjacent solutions. This calculation process is repeated with other objective functions. The crowding distance value is calculated as the sum of individual distance of each objective.

2.3. *Multi-objective GAs for FS*

FS is now commonly considered as a MOP in many works, and GAs are highly preferred to solve it in these studies. In [56], the fault classification error is minimized, and the accuracy of dissolved gas analysis and diagnosis of power transformer is improved by selecting the optimal feature subset and the optimal feature number. Labani *et al.* [57] considered the relevance of the text features to the target class and the correlation between the features as two objectives in text FS, and proposed a multi-objective algorithm, namely MORDC. Karasu and Saraç [58] used NSGA-II to find the optimal solutions for two different fitness functions, i.e., NF and classification accuracy. In [59], NSGA-II are used to obtain a set of Pareto-optimal solutions in different pattern recognition domains, the number of used features and the classification error are set as two objectives, see [60]. Das *et al.* [61] proposed a multi-objective GA with mutation pool to solve FS problem. Two objective functions are based on rough set theory and multivariate mutual information so that the most precise and informative feature subsets can be obtained. In [62], the application of MOGA to FS based on different filter importance criteria are evaluated. Bouraoui *et al.* [63] introduced a novel approach to optimize the proper kernel function, its parameters, SVM parameters and FS for SVM classification at the same time based on NSGA-II. These works have verified the feasibility of using MOGA to solve FS, and have also achieved relatively decent results, yet most

of them simply applied the NSGA-II framework without making any improvement. Therefore, this paper aims at improving NSGA-II for solving FS problems, so that the improved algorithm has better performance in both improving the classification performance and cutting the features.

3. The Self-adaptive Multi-objective Genetic Algorithm (SaMOGA)

This section describes the main framework of SaMOGA, as well as the details of some important procedures.

Algorithm 1 SaMOGA

Input:
$maxFEs$: Maximum number of fitness evaluations
N: Population size
D: Number of raw features
Q: Number of operators in the crossover set
C: Number of fitness evaluations for updating SPs
Output: PF

1: $Pop \leftarrow initializePop(N,D)$
2: $Pro \leftarrow initializePro(Q)$
3: Initialize R, P according to Eq. (4)
4: $nFE \leftarrow 0$
5: $c \leftarrow 0$
6: $Pop_{new} \leftarrow \emptyset$
7: **while** $nFE < maxFEs$ **do**
8: $stagFlag \leftarrow 0$
9: Select non-dominated solutions in Pop as PF_Lg
10: **for** i = 1 to $N/2$ **do**
11: Randomly select two individuals as parents: Par
12: $Idx \leftarrow rouletteWheelSelection(Pro)$
13: $Ch\_c \leftarrow crossover(Par, Idx)$
14: $Ch\_m \leftarrow uniformMutation(Ch\_c)$
15: $nFE \leftarrow nFE + 2$
16: $\{R,P\} \leftarrow crossoverOperatorsEvaluation(Par, Ch\_m, Idx)$
17: Add Ch_m to Pop_{new}
18: **end for**
19: $Rec \leftarrow Pop \cup Pop_{new}$
20: $Pop \leftarrow environmentalSelection(Rec)$
21: Select non-dominated solutions in Pop as PF
22: $c \leftarrow c + 1$
23: **if** $c \times N = C$ **then**
24: **if** $PF\_Lg = PF$ **then**
25: $stagFlag \leftarrow 1$
26: **end if**
27: $Pro \leftarrow updatePro(R, P, stagFlag)$
28: $c \leftarrow 0$
29: **end if**
30: **end while**
31: **return** PF

3.1. Procedure of SaMOGA

The multi-objective FS algorithm proposed in this study, i.e., SaMOGA, embeds the self-adaptive mechanism and SSDM to improve the performance of NSGA-II for solving FS problems. The flowchart of SaMOGA is shown in Figure 2. Meanwhile, Algorithm 1 demonstrates the pseudo-code of SaMOGA. Firstly, vector R and P used to record the performance of different operators in the crossover operator set, as well as N discrete-encoded individuals are initialized. Then, the probability corresponding to each crossover operator is initialized (see line 1-3 from Algorithm 1). Assuming Q crossover operators are used, each of them is assigned a probability of $1/Q$. After that, the following procedures keep repeating until the stop criteria is satisfied. It is worth noting that the second, fifth and the last steps are where SaMOGA distinguishes itself from the genetic algorithm in terms of innovation.

- Retention of PF of current population. The SSDM proposed in this paper is based on the change of PF of two adjacent generations of populations during the evolution process, so before evolving the current population, PF of current population is stored using PF_Lg. Meanwhile, the stagnation marker $stagFlag$ is initialized to 0, 0 means the search is not stagnated while 1 is the opposite (see line 8-9 from Algorithm 1).
- Crossover operation. Based on the current probability assigned to each crossover operator, one of them is first selected using roulette wheel selection. Then, two parent individuals are randomly selected from the population Pop and the crossover operation is performed using the selected crossover operator. Note that the individuals already selected are not selected again in the current generation (see line 11-13 from Algorithm 1).
- Mutation operation. The uniform mutation operator is performed on two offspring individuals generated by the crossover operation (see line 14 from Algorithm 1).
- Fitness evaluation. The obtained two offspring individuals are evaluated for fitness. Then, the number of fitness evaluation is increased by 2 (see line 15 from Algorithm 1).

- Operator evaluation. The purpose of this step is to evaluate the crossover operator used for this evolution by comparing the two generated offspring with the parents (see line 16 from Algorithm 1). If the generated offspring are decent, the crossover operator is rewarded, otherwise a penalty is given. Both the reward and penalty serve to update the probability of the corresponding crossover operator being assigned. The specific procedures are illustrated in Section 3.6.
- Elitist selection. If all individuals in the current generation have undergone reproduction, and offspring population Pop_{new} are produced. The two populations are combined together into population Rec, the non-dominated sorting and crowding distance calculation are then performed on it. After that, N individuals are selected according to the non-dominance hierarchy and crowding distance. After that, PF of new generation is stored to PF (see line 19-21 from Algorithm 1).
- Update of the selection probability (SP) assigned to each crossover operator. After C fitness evaluation, it is necessary to determine that whether PF_Lg is the same as PF, if not, $stagFlag$ is assigned to 1. Thereafter, SPs are updated using R, P and $stagFlag$. Variable c is then initialized to 0 after the update (see line 22-30 from Algorithm 1).

3.2. Chromosome Encoding

GAs encode a solution to the problem as a vector, which is also known as chromosome. In the application of FS, there are roughly two general encoding schemes [64], i.e., continuous encoding scheme and discrete encoding scheme. Since in the former, it is often necessary to set a threshold value for conversion, which is often difficult to determine, in this paper, we adopt the discrete encoding scheme to represent a chromosome. The length of each chromosome is the same as NF in the dataset and each locus can be either a value of 0 or 1, where 0 means the feature is unselected and 1 means the feature is selected.

Suppose that six features are included in a dataset, if the coded chromosome is represented as 101000, then it denotes that the first and third feature are selected while the remaining ones are unselected.

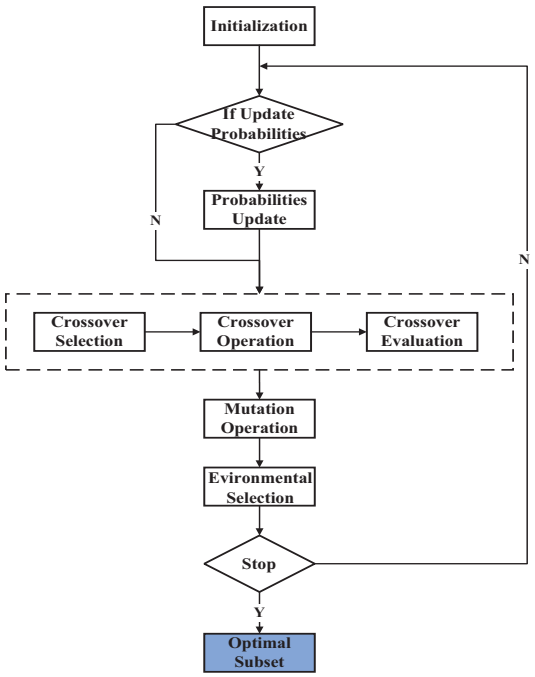


Figure 2. Flowchart of SaMOGA

3.3. Objective Function

The objective function is the basis for evaluating the goodness of the individual. In this study, we use the wrapper-based FS approach, so the classification error is used as one of the objective functions, while NF, i.e., solution size, is used as the second objective function in order to remove useless features.

3.3.1. Classification error

By calculating the quotient of the misclassified instances and the total instances, the classification error can be obtained as follows:

$$minf_1(X) = \frac{1}{h}\sum_{l=1}^{h}\frac{N_{Err}}{N_{All}} \tag{2}$$

where X is a solution, h is the parameter in h-fold cross-validation, N_{Err} and N_{All} are the number of misclassified instances and all instances, respectively.

3.3.2. Solution size

$$min f_2(X) = \sum_{i=1}^{D} x_i \tag{3}$$

where x_i means the i^{th} decision variable, D denotes the number of raw features.

3.4. Crossover Operator Set

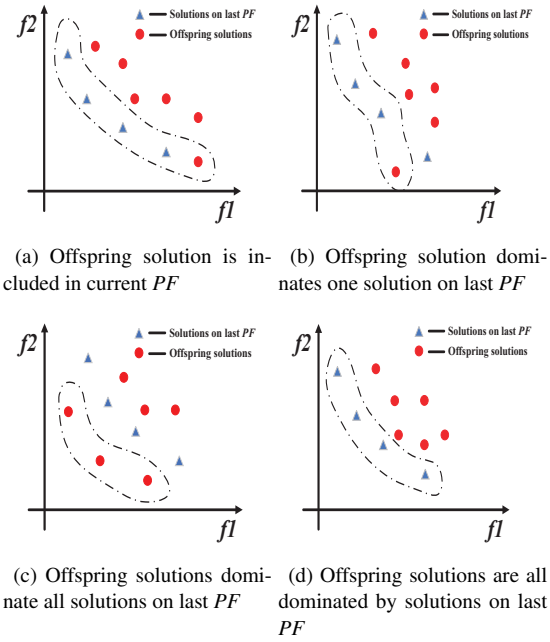
Offspring inherit genes from the parents through crossover operations, and promising genes are highly expected to be inherited. Thus, the selection of the appropriate crossover operator is critical to the performance of the offspring in terms of fitness values. However, picking out the appropriate crossover operator is time-consuming. In this paper, we intend to apply crossover operators with different properties to compensate for this deficiency, so five popular and commonly used crossover operators, i.e., single-point crossover operator [65], two-point crossover operator [66], uniform crossover operator [67], shuffle crossover operator [68] and reduced surrogate crossover operator [69] are adopted to form a crossover operator set for the crossover operation. A detailed description of them can be obtained in their referred papers.

By applying different crossover operators at different stages of evolution process, superior genes in the parents have a greater chance of being passed on to the offspring, resulting in the better performance of offspring individuals.

3.5. Search Stagnation Detection Mechanism

In the process of searching for a set of more optimal solutions, search stagnation frequently occurs, i.e., the PF is not updated after one generation of evolution, which affects the convergence rate of the population. Therefore, the SSDM is proposed, which lies in detecting the change between the PF obtained in the previous generation and the PF obtained in the current generation. Take the two-objective minimization problem as an example, in total, four different scenarios exist, and the specific examples are shown in Figure 3.

With each generation of the population, we expect the PF to move in the direction of the coordinate origin, or to complement itself. In Figure 3(b)-(c), solution(s) on last PF is/are dominated by offspring solution(s), and in Figure 3(a) that there exits offspring solution(s) complementing the solutions on last PF. It is favorable in these three cases since PF is updating. When it comes to Figure 3(d), last PF neither moved nor is complemented, so it can be assumed that the search of this generation is stagnant. As mentioned in Section 3.1, SSDM is used as a complementary technique to the update of SPs, mainly because it is normal for PF to remain unchanged for several generations at a late stage of population evolution, and using this technique alone to determine whether the search being stagnated can disrupt the search direction of the population, leading to a reduction in the speed of convergence as well as in the likelihood of searching for the global optimal.



(a) Offspring solution is included in current *PF*

(b) Offspring solution dominates one solution on last *PF*

(c) Offspring solutions dominate all solutions on last *PF*

(d) Offspring solutions are all dominated by solutions on last *PF*

Figure 3. Search Stagnation Detection Mechanism

3.6. Crossover Operators Evaluation

The multiple crossovers are handled in an ensemble fashion [70] by means of a success-based adaptation similar to that of hyperheuristics [71].

Assume that Q operators are included in the crossover operator set, and the q^{th} one is selected for the crossover operation. After undergoing crossover and mutation operations, it is intended to evaluate

the selected crossover operator based on the performance of children. The selected crossover operator is rewarded if the produced children is promising, otherwise a penalty is given. We therefore employ two vectors for this purpose and they are implemented as follows:

$$\begin{aligned} R &= \left(r_1 \; r_2 \; \cdots \; r_Q \right) \\ P &= \left(p_1 \; p_2 \; \cdots \; p_Q \right) \end{aligned} \tag{4}$$

where R and P are initialized with all elements at zero.

Algorithm 2 $crossoversOperatorEvaluation(Par, Ch, q)$

Input:
Par: List of two parents
Ch: List of two children
q: The selected crossover operator
Output: R, P

1: Compare the Pareto dominance relationship of two parents
2: **if** One dominates the other, assume $Par_1 \prec Par_2$ **then**
3: **for** $i = 1$ *to* 2 **do**
4: **if** $Par_1 \prec Ch_i$ **then**
5: $P_q \leftarrow P_q + 1$
6: **else**
7: $R_q \leftarrow R_q + 1$
8: **end if**
9: **end for**
10: **else**
11: **for** $i = 1$ *to* 2 **do**
12: **if** $Par_1 \not\prec Ch_i$ && $Par_2 \not\prec Ch_i$ **then**
13: $R_q \leftarrow R_q + 1$
14: **else**
15: $P_q \leftarrow P_q + 1$
16: **end if**
17: **end for**
18: **end if**

Based on the relationship between the children generated by the crossover operator and the corresponding parents in the objective space, we record information on the use of the crossover operator using the Pareto dominance relationship between them. Two potential scenarios can be found below.

Scenario 1: One of the parents dominates the other one. In this case, the two children merely have to compare the Pareto dominance relationship with the superior parent. If the child do not dominated by it, R_q + 1, otherwise P_q + 1 (see line 3-9 from Algorithm 2)

Scenario 2: Two parents do not non-dominate each other. Slightly different from the first case, the two children need to be compared with the two parents respectively. If the child is not dominated by two parents at the same time, R_q + 1, otherwise P_q + 1 (see line 11-17 from Algorithm 2).

After a certain number of fitness evaluation, R and P will be reinitialized again. The specific process is shown in Algorithm 2.

Algorithm 3 $updatePro(R, P, stagFlag)$

Input:
R: Reward information of different operators
P: Penalty information of different operators
Q: Number of operators in the crossover set $stagFlag$: Stagnation flag
Output: R, P

1: **if** $\exists$ 0 in Pro && $stagFlag = 1$ **then**
2: $Pro \leftarrow initializePro(Q)$
3: **else**
4: **for** $q = 1$ *to* Q **do**
5: **if** $R_q = 0$ **then**
6: $R_q \leftarrow \varepsilon$
7: **end if**
8: Update the probability of the operator q using Eq. 5
9: **end for**
10: Normalize the probabilities for all operators using Eq. 6
11: **end if**

3.7. Update of Selection Probabilities Assigned to Crossover Operators

After the number of fitness evaluations set in advance is reached, SPs are updated based on R, P and $stagFlag$, which serves as a symbol indicating whether the PF of two adjacent generations has changed. For easy implementation, number of fitness evaluations for updating SPs C is set as a multiple of the population size. Suppose the population size is N, $C = c \times N$, where c is a pre-defined constant. The specific steps are shown in Algorithm 3. First, in order to avoid the search stagnation caused by one certain crossover operator not having a chance to be selected, it is necessary to determine whether there is one or more elements in vector Pro that are 0 and $stagFlag$ is 1 at the same time. If so, Pro needs to be reinitialized the same way as mentioned before (see line 1-2 from Algorithm 3). Otherwise, SP for q^{th} crossover operator is calculated as follows:

$$Pro_q^{tmp} = \frac{R_q}{R_q + P_q} \tag{5}$$

In fact, SP of q^{th} operator is calculated by dividing the reward R_q by the sum of reward R_q and penalty P_q. However, there are cases where operator q may not be selected once in the uFE fitness evaluations, resulting in both R_q and P_q being 0. Therefore, a very small positive number ε, which is not equal to 0, is assigned to R_q to avoid the situation where the divisor is 0.

Finally, we normalize SP assigned to operator q as follows:

$$Pro_q = \frac{Pro_q^{tmp}}{\sum_{q=1}^{Q} Pro_q{}^{tmp}} \tag{6}$$

4. Experimental Settings

In this section, the experimental datasets and classifier are first introduced, followed by the comparison algorithms and the setting of relevant parameters, the metrics for evaluating the experimental results are described at last. The proposed algorithm and its counterparts are implemented using Matlab language. All the experiments are conducted on an Intel Core (TM) i5-9500 CPU with 8 GB of RAM and 1TB of hard disk.

4.1. Datasets and Classifiers

Eighteen datasets are used to train and test the proposed algorithm and its counterparts. The full descriptions of them are available in the UCI Machine Learning Repository [72]. These datasets are composed of different numbers of features (*NF* in Table 1), labels (*NL* in Table 1) and instances (*NI* in Table 1). All datasets are split into training sets (70% of the raw datasets) and test sets (30% of the raw datasets) at random. As can be seen from Table 1, the number of features ranges from 30 to 1300. In addition, the number of classes varies from 2 to 26 and the number of instances varies from 32 to 1080. Thus, the ability of the algorithms to solve the feature selection problem can be effectively evaluated with these comprehensive and complex characteristic datasets.

A learning algorithm is usually required in wrapper-based feature selection methods to evaluate the classification performance of the feature subset. In fact, many classifiers can be used for this purpose. In this paper, k-NN is applied due to its simplicity and promising classification performance. Since the division of datasets have an impact on the generalization of the model, in order to avoid this situation as much as possible, three-fold cross-validation is introduced to reduce the over-fitting and under-fitting problems [73].

Table 1. Details about Used Datasets

Datasets	NF	NL	NI
DS01	30	2	596
DS02	34	2	351
DS03	36	2	6435
DS04	41	2	1055
DS05	56	3	32
DS06	60	2	208
DS07	64	10	1000
DS08	256	10	675
DS09	301	2	1062
DS10	500	2	600
DS11	522	3	900
DS12	561	6	900
DS13	561	12	1200
DS14	617	26	1040
DS15	649	10	1000
DS16	856	9	1080
DS17	1024	2	1687
DS18	1300	2	360

*Note: NF, NL and NI represent the number of features, number of labels, and number of instances.
*DS01 to DS18 in the table represent Wdbc, Ionosphere, Satellite, QSARbiodegradation, LungCancer, ConnectionistBenchData, OpticalRecognitionofHandwritten, SemeionHandwrittenDigit, grammaticalfacialexpression01, MadelonValid, UJIIndoorLoc, Har, HAPT, Isolet5, MultipleFeaturesDigit, CNAE, QSARandrogenreceptor, MicroMass.

4.2. Comparison Algorithms and Parameter Setting

Five multi-objective algorithms for FS are adopted for comparison. They are NSGA-II [55], NSPSOFS [74], CMDPSOFS[74], SPEA2 [75] and MOEA/D [76]. All the algorithms have been implemented in Matlab by

the authors on the basis of their original description [77]. Among them, NSGA-II is one of the most classic MOEAs, and SaMOGA is an improvement on NSGA-II. NSPSOFS and CMDPSOFS are two multi-objective PSO proposed for feature selection. SPEA2 is based on the concept of Pareto dominance relationship for fitness value assignment and selection operations, and it applies the niche method and elite mechanism. Meanwhile, SPEA2 is known for its strong convergence ability. MOEA/D introduces the decomposition strategy into the evolutionary algorithm framework, and uses the decomposition strategy to divide the multi-objective optimization problem into a set of single-objective sub-problems, and then optimizes these sub-problems simultaneously. For each algorithm, 30 independent runs are performed on each dataset. Meanwhile, All algorithms run the same number of fitness evaluations (nFE). The parameter setting of SaMOGA is listed in Table 2. The parameter values of benchmark algorithms are set as in their referred papers.

Table 2. Parameter Setting of SaMOGA

Parameters	Value
$maxFEs$	300,000
N	100
M	2
Q	5
C	500
P_c	0.9
P_m	$1/D$

4.3. Performance Metrics

Since a set of non-dominated solutions are obtained in the final results, this paper uses two popular performance metrics to evaluate them, namely inverted generational distance (IGD) [78] and hypervolume (HV) [79,80,81,82]. Generally speaking, the smaller the IGD values are, the better performance an algorithm gains, which is opposite for HV. Specially, they are able to evaluate both the convergence and the diversity of the obtained Pareto solution sets.

5. Results

SaMOGA is compared with five comparison algorithms in three aspects in this section, i.e., the IGD and HV metrics, best PFs evolved on training sets and average PFs obtained on test sets. The IGD and HV are introduced to evaluate the convergence and diversity of the results achieved by these six algorithms on both training and test sets. Moreover, the Wilcoxon test[83] with a confidence level of 95% is also used on the IGD and HV metrics to check whether the performance of SaMOGA is significantly different from that of its counterparts. Since the true Pareto front is unknown, we first combine all the solutions obtained by the algorithms and compute the non-dominated solution among them as the true Pareto front to obtain the IGD and HV values.

To perform the experiments over all the datasets above, SaMOGA required approximately six hours.

5.1. Analysis of IGD and HV

Table 3 - Table 6 show the mean values and standard deviations achieved by all comparison algorithms in terms of IGD and HV on training and test sets. Best mean value for each dataset is presented in bold. In addition, symbol "W" (it stands for Wilcoxon) indicates whether there is a significant difference between the compared algorithms and SaMOGA. "+" or "-" indicates that SaMOGA is significantly better or worse than compared algorithms, "=" means there is no significant difference between them.

From Table 3, it can be observed that SaMOGA obtains the minimum IGD values on sixteen out of eighteen training sets. It also achieves much smaller values than other algorithms do on DS08 - DS18. Note that a common characteristic of these datasets is the large number of features. For DS08, NSGA-II and NSPSOFS achieve the similar results and are worse than the other four algorithms. Meanwhile, SaMOGA is slightly superior to SPEA2, followed by CMDPSOFS and MOEA/D. This is also the similar case for DS09. However, CMDPSOFS is more promising than SPEA2 on DS10 and DS11. As NF of the dataset increases, SaMOGA is always in first place, and CMDPSOFS and SPEA2 achieve similar IGD values and are in second and third place, followed by

Table 3. Results of Mean IGD Values on Training Sets

Datasets		NSGA-II		NSPSOFS		CMDPSOFS		SPEA2		MOEA/D		SaMOGA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
DS01	IGD	0.44	0.28	0.19	0.18	0.19	0.19	0.24	0.31	0.92	0.51	**0.16**	0.21
	W	+		+		+		=		+			
DS02	IGD	0.36	0.31	0.34	0.22	0.19	0.11	0.2	0.11	0.39	0.2	**0.09**	0.12
	W	+		+		+		+		+			
DS03	IGD	0.34	0.24	0.24	0.13	**0.14**	0.07	0.45	0.14	1.94	0.33	0.18	0.14
	W	+		+		+		+		+			
DS04	IGD	0.95	0.84	0.58	0.2	0.59	0.3	1.15	0.37	1.87	0.66	**0.45**	0.26
	W	+		+		+		+		+			
DS05	IGD	2.02	1.3	2.35	1.06	**0.62**	0.25	0.84	0.31	1.48	0.63	1.13	0.3
	W	+		+		-		-		+			
DS06	IGD	1.62	1.05	2.58	0.95	0.33	0.22	0.68	0.26	1.37	0.74	**0.21**	0.14
	W	+		+		+		+		+			
DS07	IGD	1.16	0.65	1.65	0.34	0.59	0.23	0.88	0.36	2.84	0.45	**0.49**	0.2
	W	+		+		+		+		+			
DS08	IGD	37.57	6.43	31.61	8.92	8.84	2.32	2.8	1.06	10.25	3.74	**1.71**	0.76
	W	+		+		+		+		+			
DS09	IGD	63.17	5.25	60.1	12.17	11.53	2.54	3.99	1.72	23.82	5.96	**3.09**	0.82
	W	+		+		+		+		+			
DS10	IGD	132.57	7.68	122.4	20.11	6.31	3.52	17.33	4.1	50.63	13.15	**0.96**	0.67
	W	+		+		+		+		+			
DS11	IGD	139.27	9.44	130.53	17.68	5.27	3.69	27.03	6.32	55.8	11.07	**0**	0
	W	+		+		+		+		+			
DS12	IGD	148.67	8.86	131.3	23.92	22.63	5.1	22.17	7.94	48.23	11.6	**0.38**	0.28
	W	+		+		+		+		+			
DS13	IGD	144.23	9.66	132.23	18.95	19.8	5.07	19.11	5.54	48.04	15.78	**0.87**	0.45
	W	+		+		+		+		+			
DS14	IGD	154.37	10.92	141.24	26.42	14.79	3.75	15.15	6.1	44.89	17.28	**0.83**	0.4
	W	+		+		+		+		+			
DS15	IGD	175.62	9.4	157.85	23.24	29.69	5.99	24.46	7.28	58.02	15.33	**0.57**	0.28
	W	+		+		+		+		+			
DS16	IGD	212.89	11.37	195.73	26.26	30.95	5.62	36.82	8.67	63.64	16.47	**8.09**	2.01
	W	+		+		+		+		+			
DS17	IGD	307.23	9.02	273.9	41.76	74.9	13.23	48.23	7.23	94.9	28.62	**1.72**	1.24
	W	+		+		+		+		+			
DS18	IGD	410.07	12.49	353.23	54.72	45.03	9.61	87.57	13.58	140.63	27.72	**1.21**	0.93
	W	+		+		+		+		+			

Table 4. Results of Mean IGD Values on Test Sets

Datasets		NSGA-II		NSPSOFS		CMDPSOFS		SPEA2		MOEA/D		SaMOGA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
DS01	IGD	0.67	0.43	**0.39**	0.27	0.71	0.34	0.93	0.17	1.12	0.47	0.86	0.26
	W	=		-		=		=		+			
DS02	IGD	0.67	0.38	0.65	0.29	0.68	0.06	0.61	0.18	**0.45**	0.27	0.66	0.16
	W	+		+		+		+		-			
DS03	IGD	0.71	0.26	0.36	0.05	0.44	0.25	**0.24**	0.15	1.07	0.25	0.27	0.12
	W	+		+		+		-		+			
DS04	IGD	1.07	0.69	0.99	0.36	0.71	0.5	0.8	0.59	0.79	0.44	**0.55**	0.31
	W	+		+		+		+		+			
DS05	IGD	2.48	1.67	3.01	1.3	**0.9**	0.36	1.11	0.39	1.69	0.94	1.21	0.53
	W	+		+		-		=		+			
DS06	IGD	2.37	1.32	3.43	1.16	**0.67**	0.27	0.9	0.41	1.99	0.96	0.73	0.28
	W	+		+		=		=		+			
DS07	IGD	1.44	0.8	2.29	0.53	0.56	0.19	**0.39**	0.15	1.43	0.28	0.4	0.16
	W	+		+		=		=		+			
DS08	IGD	36.66	6.08	30.94	8.4	11.8	2.33	5.66	2.44	12.67	3.56	**4.78**	2.1
	W	+		+		+		+		+			
DS09	IGD	69.23	5.25	66.17	12.17	15.67	2.83	5.2	2.43	28.9	6.88	**0.33**	0.21
	W	+		+		+		+		+			
DS10	IGD	136.17	7.68	126	20.11	9.27	4.21	20.93	4.1	54.23	13.15	**0.42**	0.26
	W	+		+		+		+		+			
DS11	IGD	139.27	9.44	130.53	17.68	5.27	3.69	27.03	6.32	55.8	11.07	**0**	0
	W	+		+		+		+		+			
DS12	IGD	142.59	8.86	125.22	23.92	21.04	4.19	21.34	6.35	43.11	10.65	**6.23**	1.71
	W	+		+		+		=		+			
DS13	IGD	150.77	9.66	138.77	18.95	26.03	5.3	25.22	5.96	54.57	15.79	**1.48**	0.72
	W	+		+		+		+		+			
DS14	IGD	162.08	10.92	148.95	26.42	21.21	4.37	21.39	7.13	52.58	17.32	**1.94**	0.91
	W	+		+		+		+		+			
DS15	IGD	173.33	9.4	155.57	23.24	31.77	5.44	28.21	6.52	57.94	13.91	**5.25**	0.78
	W	+		+		+		+		+			
DS16	IGD	235.69	11.37	218.49	26.38	41.71	6.95	42.35	10.8	77.66	19.68	**10**	2.92
	W	+		+		+		+		+			
DS17	IGD	251.83	9.02	218.5	41.76	64.17	10	72.77	1.3	71.1	12.37	**56.93**	11.65
	W	+		+		+		+		+			
DS18	IGD	410.99	12.49	354.16	54.72	45.96	9.61	88.49	13.58	141.56	27.72	**1.7**	1.06
	W	+		+		+		+		+			

Table 5. Results of Mean HV Values on Training Sets

Datasets		NSGA-II		NSPSOFS		CMDPSOFS		SPEA2		MOEA/D		SaMOGA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
DS01	HV	0.04	0.02	0.01	0	0.05	0	**0.06**	0.01	0.02	0.01	**0.06**	0.01
	W	+		+		+		=		+			
DS02	HV	0.12	0.04	0.05	0.02	0.18	0	**0.18**	0.01	0.09	0.03	**0.18**	0.01
	W	+		+		+		=		+			
DS03	HV	1.26	0.04	1.17	0.01	1.3	0	1.36	0	1.05	0.08	**1.37**	0
	W	=		+		=		=		+			
DS04	HV	0.8	0.21	0.66	0.04	1.11	0.02	1.19	0.02	0.75	0.13	**1.2**	0.02
	W	+		+		+		=		+			
DS05	HV	0.69	0.42	0.07	0.09	1.88	0.06	**1.9**	0.17	0.89	0.41	1.57	0.45
	W	+		+		-		-		+			
DS06	HV	0.18	0.12	0.01	0.01	0.57	0.05	0.69	0.05	0.2	0.14	**0.76**	0.03
	W	+		+		+		+		+			
DS07	HV	17.69	1.24	15.63	0.57	20	0.23	20.56	0.06	17.83	0.79	**20.69**	0.03
	W	+		+		+		+		+			
DS08	HV	17.61	4.43	20.48	5.8	41.1	2.19	51.05	1.49	36.05	4	**55.97**	0.36
	W	+		+		+		+		+			
DS09	HV	0	0	0	0	1.5	0.2	2.63	0.25	0.46	0.36	**3.69**	0.11
	W	+		+		+		+		+			
DS10	HV	0	0	0	0	0.54	0.61	0	0	0	0	**3.26**	0.05
	W	+		+		+		+		+			
DS11	HV	0	0	0	0	0	0	0	0	0	0	**1.00E-25**	5.84E-41
	W	+		+		+		+		+			
DS12	HV	0	0	0	0	0.01	0.05	0.05	0.15	0	0	**7.55**	0.23
	W	+		+		+		+		+			
DS13	HV	0	0	0	0	3.8	2.09	4.3	2.54	0.17	0.55	**17.79**	0.32
	W	+		+		+		+		+			
DS14	HV	0	0	0	0	18.83	3.37	19.68	5.91	2.15	3.45	**40.9**	1.39
	W	+		+		+		+		+			
DS15	HV	0	0	0	0	0.34	0.64	1.79	2.43	0	0	**18.6**	0.49
	W	+		+		+		+		+			
DS16	HV	0	0	1.17	4.97	101.4	4.95	100.72	7.65	55.97	11.86	**142.46**	3.31
	W	+		+		+		+		+			
DS17	HV	0	0	0	0	0	0	0	0	0	0	**2.27**	0.22
	W	+		+		+		+		+			
DS18	HV	0	0	0	0	0	0	0	0	0	0	**1.75**	0.5
	W	+		+		+		+		+			

Table 6. Results of Mean HV Values on Test Sets

Datasets		NSGA-II		NSPSOFS		CMDPSOFS		SPEA2		MOEA/D		SaMOGA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
DS01	HV	0.05	0.06	0.04	0.04	0.12	0.03	**0.14**	0.03	0.03	0.04	0.13	0.03
	W	+		+		=		=		+			
DS02	HV	0.02	0.04	0.01	0.02	**0.13**	0.03	0.12	0.04	0.1	0.04	0.11	0.03
	W	+		+		-		=		=			
DS03	HV	0.56	0.03	0.54	0.04	0.6	0.02	0.6	0.01	0.58	0.07	**0.62**	0.01
	W	=		=		=		=		=			
DS04	HV	0.02	0.04	0.02	0.02	0.07	0.05	0.07	0.06	**0.2**	0.07	0.08	0.05
	W	+		+		=		=		-			
DS05	HV	0.05	0.16	0.01	0.03	0.13	0.15	**0.16**	0.25	0.14	0.15	0.14	0.26
	W	=		+		=		=		=			
DS06	HV	0.02	0.04	0	0.01	0.14	0.09	0.12	0.09	0.07	0.11	**0.15**	0.11
	W	+		+		=		=		+			
DS07	HV	9.69	1	8.2	0.53	11.96	0.22	12.07	0.19	10.12	0.7	**12.19**	0.11
	W	+		+		+		+		+			
DS08	HV	16.92	3.58	19.71	4.61	35.29	1.71	41.09	1.37	37.09	3.79	**43.86**	1.36
	W	+		+		+		+		+			
DS09	HV	0	0	0	0	0	0	0	0.01	0	0	**0.05**	0.02
	W	+		+		+		+		+			
DS10	HV	0	0	0	0	0.01	0.05	0	0	0	0	**0.65**	0.11
	W	+		+		+		+		+			
DS11	HV	0	0	0	0	0	0	0	0	0	0	**8.33E-26**	3.79E-26
	W	+		+		+		+		+			
DS12	HV	0	0	0	0	13.82	2.03	13.39	3.09	3.37	2.63	**26.08**	0.58
	W	+		+		+		+		+			
DS13	HV	0	0	0	0	0.01	0.04	0.16	0.5	0	0	**9.4**	0.37
	W	+		+		+		+		+			
DS14	HV	0	0	0	0	13.44	3.07	13.75	5.11	0.9	2.07	**32.17**	1.24
	W	+		+		+		+		+			
DS15	HV	0	0	0	0	30.14	3.22	32.84	3.84	12.96	6.83	**50.94**	0.65
	W	+		+		+		+		+			
DS16	HV	0	0	0	0	2.07	2.77	3.17	3.53	0	0	**27.55**	2.99
	W	+		+		+		+		+			
DS17	HV	0	0	0	0	6.45	1.27	6.89	1.07	6.92	1.5	**11.23**	2.31
	W	+		+		+		+		+			
DS18	HV	0	0	0	0	0	0	0	0	0	0	**1.37**	0.52
	W	+		+		+		+		+			

MOEA/D. NSGA-II and NSPSOFS are also at almost the same level, but the IGD values of them are the worst among all algorithms. It is also worth noting that SaMOGA is significantly better than all compared algorithms on DS08 - DS18. When it comes to training sets with relative small number of features, i.e., DS01 - DS07, SaMOGA achieves the smallest IGD values on all of them except on DS03 and DS05. CMDPSOFS can obtain better or similar IGD values on DS01 - DS07 when compared to SPEA2. On DS01, there is no significant difference among SaMOGA, NSPSOFS, CMDPSOFS and SPEA2. On DS04, it is significantly better than the other five algorithms. On DS05, CMDPSOFS obtain the smallest IGD value, followed by SPEA2, and SaMOGA is in the third place.

In terms of IGD values on the test sets, as presented in Table 4, SaMOGA still obtains the minimum IGD value among all compared algorithms and it is also significantly better than all compared algorithms on DS08 - DS18. On DS01, NSPSOFS achieves the smallest IGD value and it is significant better than SaMOGA. On DS02, MOEA/D obtains the best result, and SaMOGA achieves result similar to NSGA-II, NSPSOFS, CMDPSOFS and SPEA2. CMDPSOFS performs well on DS05 and DS06. It can be observed that SaMOGA is significantly better than CMDPSOFS and SPEA2 on training sets of DS06 and DS07, but is not significantly different from CMDPSOFS and SPEA2 on the test sets of DS06 and DS07.

From Table 5, SaMOGA obtains the maximum HV values on fourteen out of sixteen training sets. SaMOGA performs well on DS08 - DS18. CMDPSOFS and SPEA2 are ranked in the second and third place on DS12 - DS16, followed by MOEA/D, NSGA-II and NSPSOFS. On DS01, DS02 and DS05, SPEA2 achieves the biggest HV values among all comparison algorithms. Although SaMOGA achieves the biggest HV values on DS03 and DS04, it is not significantly different from SPEA2. From DS01 - DS02, we can observe that SPEA2 is very promising.

Table 6 presents the HV values on test sets. From Table 6, SaMOGA obtains slightly worse HV values than that on the training sets but it is still able to achieve the biggest HV values on twelve out of sixteen test sets. In particular, its performance remains stable on test sets with a large number of features. On DS01 - DS05, SPEA2 achieves the biggest HV values for two times while CMDPSOFS and MOEA/D achieve the biggest HV values once each. SaMOGA is significantly superior to NSGA-II, NSPSOFS and MOEA/D on DS06 and it obtains slightly bigger HV value than all compared algorithms on DS07.

From IGD and HV values on both training and test sets presented in Table 3 - Table 6, we can find that SaMOGA wins on most training and test sets and it never lose on datasets with relative large number of features. Meanwhile, on datesets with relative small number of features, CMDPSOFS performs well in terms of IGD metric while SPEA2 is good in terms of HV metric. From these results, we can conclude that SaMOGA outperforms NSGA-II, NSPSOFS, CMDPSOFS, SPEA2 and MOEA/D on most datasets in terms of IGD and HV values, especially for those with large number of features.

5.2. Analysis of Best PFs Evolved on Training Sets

In this subsection, we present the best PFs evolved on training sets to analyze the convergence and diversity of the obtained solutions by different algorithms. For each algorithm, we first merge all PFs obtained from 30 independent runs to form the PF ensemble, and then perform non-dominated sorting operation to it and select the non-dominated solutions to obtain the best PF. Considering the length of the article, we only present the plots of the best Pareto front for the six training sets here, which are named as DS01_Tr, DS02_Tr, DS07_Tr, DS08_Tr, DS10_Tr and DS11_Tr in Figure 4, the remaining ones are shown in the Appendix. Meanwhile, the number of raw features and the classification error obtained by using them for classification on each training sets is presented at the bottom of the corresponding subfigures.

It can be seen from DS01_Tr and DS02_Tr that the final solutions obtain by SaMOGA are well-converged. Meanwhile, it achieves the similar results to SPEA2 and CMDPSOFS on most cases. Among all comparison algorithms, the final solutions obtained by NSPSOFS are inferior. Despite the wide distribution of solutions in the results obtained by NSPSOFS, it is difficult to obtain better convergence. On DS07_Tr, all comparison algorithms perform well and achieve the same results. It is also noticeable that NSGA-II and

NSPSOFS tend to obtain solutions with small classification error and big solution size.

Meanwhile, SaMOGA outperforms other five comparison algorithms on DS08_Tr, DS10_Tr and DS11_Tr. Notably, SaMOGA is good at searching out the solution that has small solution size and high classification error. Perhaps it is with these solutions that SaMOGA can ensure population diversity and promote population convergence to prevent stagnation. On DS08_Tr, which has 256 raw features, the gap between CMDPSOFS, SPEA2, MOEA/D and SaMOGA is not so obvious. However, MOEA/D starts to become quite inferior as NF increases. It can be also observed that CMDPSOFS and SPEA2 have better search ability, CMDPSOFS outperforms SPEA2 on DS10_Tr, DS11_Tr while SPEA2 outperforms CMDPSOFS on the other eight training sets.

From best PFs evolved on the training sets, we notice that the best PF obtained by SaMOGA is great. Besides, better final solutions are obtained when training on the datasets with a larger number of features. Therefore, we can conclude that SaMOGA has strong search capability and the advantage over its counterparts is more obvious when NF in the dataset is larger.

5.3. Analysis of Average PFs on Test Sets

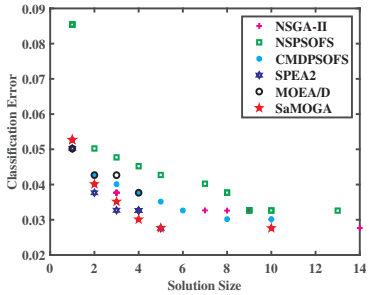
To evaluate the performance of different algorithms on the test sets, the average PFs obtained on the test sets are analyzed. Considering the length of the article, we only present the plots of the average Pareto front for the six test sets here, which are named as DS01_Te, DS02_Te, DS07_Te, DS08_Te, DS10_Te and DS11_Te in Figure 4, the remaining ones are shown in the Appendix. Since each algorithm is run 30 times on each dataset, 30 obtained PFs invariably contain solutions with the same solution size but different classification errors. To obtain the average PF of each algorithm on each dataset, we average the classification errors of solutions with the same solution size so that one solution size will correspond to one classification error. The PF obtained in this way is called average PF. Meanwhile, the number of raw features and the classification error obtained by using them for classification on each test sets is presented at the bottom of the corresponding subfigure.

According to DS01_Te, DS02_Te and DS07_Te, all the comparison algorithms achieve the similar results, i.e., small classification error and small solution size, and they can obtain a feature subset with only one feature on all low-dimensional datasets except for NSPSOFS. In addition, on these three test sets, SaMOGA can obtain solutions with smaller or slightly higher classification errors than using all features, but with more feature cuts, e.g., 93% of features on DS01_Te and 94% of features on DS02_Te.

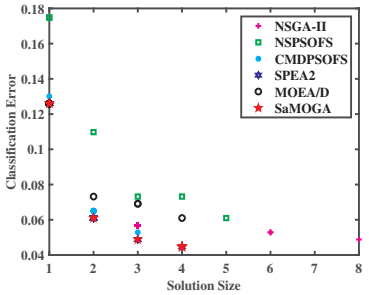
In terms of the classification error, it can be observed that MOEA/D wins on DS08_Te, however, it has a deteriorated performance on the other datasets, achieving similar or worse results than NSGA-II and NSPSOFS. In almost all cases, SaMOGA obtains similar or smaller classification errors than that of CMDPSOFS and SPEA2, achieving smaller classification errors than it does without using the FS operation. From the perspective of solution size, NSPSOFS is able to obtain smaller solution size than that of NSGA-II, yet inferior to MOEA/D. On most test sets, CMDPSOFS and SPEA2 obtain feature subsets with similar solution size. Furthermore, SaMOGA is capable of achieving feature subsets with minimum solution size among all comparison algorithms on all test sets, especially when NF increases, its strength in cutting features becomes stronger. The most outstanding one is that SaMOGA gets a classification error of 0 with only one feature on DS11_Tr. Overall, SaMOGA is great at removing irrelevant or redundant features and ensuring a low classification error.

On the basis of the results of this study, a future direction of our research will include the hybridization of the proposed SaMOGA with modern ingenious paradigms in neural systems and classification such as Enhanced Probabilistic Neural Network [84], Neural Dynamic Classification Algorithm [85], Dynamic Ensemble Learning Algorithm [86], and Finite Element Machine for Fast Learning [87].

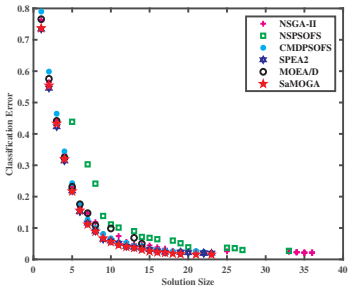
Another direction of future research is the application of the proposed self-adaptive multi-objective logic to other algorithmic structures for optimization such as distributed neural dynamic algorithms [88], spiral dynamic algorithms [89], harmony search algorithm [90], water drop algorithm [91], and central force metaheuristic optimization.



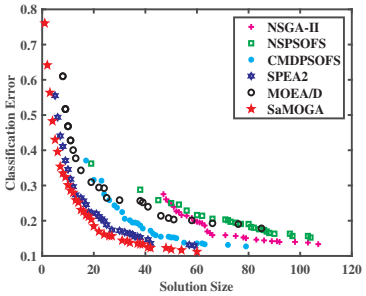
DS01_Tr (30, 0.06)



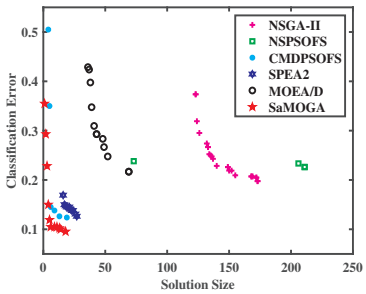
DS02_Tr (34, 0.15)



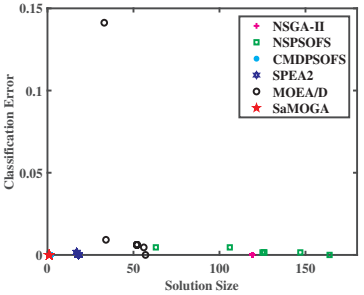
DS07_Tr (64, 0.05)



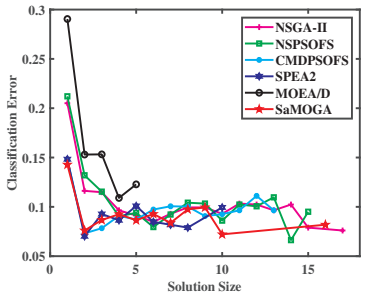
DS08_Tr (256, 0.18)



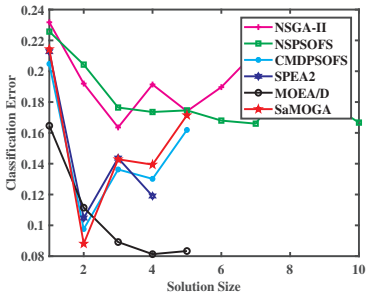
DS10_Tr (500, 0.35)



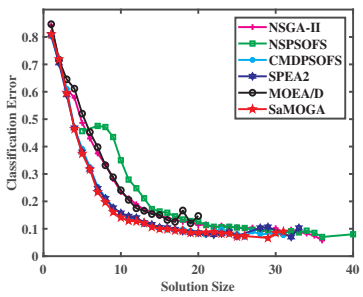
DS11_Tr (522, 0.00)



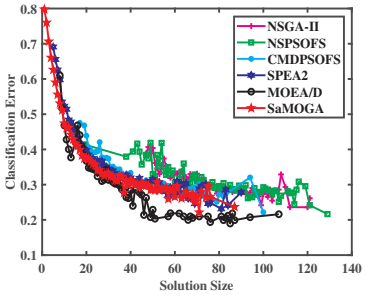
DS01_Te (30, 0.12)



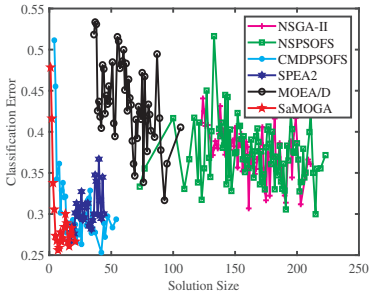
DS02_Te (34, 0.18)



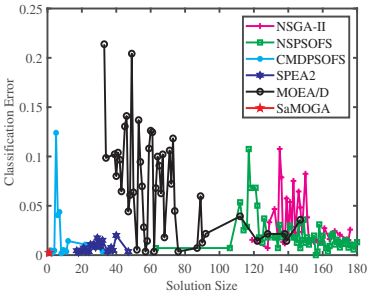
DS07_Te (64, 0.08)



DS08_Te (256, 0.25)



DS10_Te (500, 0.40)



DS11_Te (522, 0.00)

Figure 4. Best and Average PFs Evolved on Training and Test Sets

6. Conclusions

This paper considers FS as a MOP, and proposes SaMOGA to handle it. By adopting the self-adaptive mechanism and SSDM simultaneously, SaMOGA commits to a sufficient search in the search space to yield a set of solutions with the small classification error and solution size without ending up finding local optima. Experiments are conducted on sixteen datasets with the number of feature from 30 to 1300. In addition, five multi-objective optimization algorithms for FS are adopted for comparison. The results reveal that SaMOGA is able to obtain a lower classification error using fewer features than the comparison algorithms. Meanwhile, SaMOGA is able to obtain better results than other comparison algorithms on most training and test sets in terms of IGD and HV. The success of the proposed SaMOGA over the other multi-objective approaches resides in its flexibility due to the integrated set of crossover operators in an ensemble fashion.

In the future work, we intend to investigate whether individuals can be encoded with variable length to narrow the search space, and then apply SaMOGA to further improve the classification performance.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (61876089, 61876185, 61902281), the Opening Project of Jiangsu Key Laboratory of Data Science and Smart Software (No.2019DS301), the Natural Science Foundation of Jiangsu Province (BK20141005), the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (14KJB520025).

References

[1] Burns A, Adeli H, Buford JA. Upper Limb Movement Classification Via Electromyographic Signals and an Enhanced Probabilistic Network. Journal of Medical Systems. 2020;44(10):1–12.

[2] Maeda K, Takahashi S, Ogawa T, Haseyama M. Convolutional sparse coding-based deep random vector functional link network for distress classification of road structures. Computer-Aided Civil and Infrastructure Engineering. 2019;34(8):654–676.

[3] Guo J, Wang Q, Li Y. Semi-supervised learning based on convolutional neural network and uncertainty filter for façade defects classification. Computer-Aided Civil and Infrastructure Engineering. 2021;36(3):302–317.

[4] Erharter GH, Oberhollenzer S, Fankhauser A, Marte R, Marcher T. Learning decision boundaries for cone penetration test classification. Computer-Aided Civil and Infrastructure Engineering. 2021;36(4):489–503.

[5] Apicella A, Isgrò F, Prevete R, Tamburrini G. Middle-level features for the explanation of classification systems by sparse dictionary methods. International Journal of Neural Systems. 2020;30(08):2050040.

[6] Kou G, Yang P, Peng Y, Xiao F, Chen Y, Alsaadi FE. Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. Applied Soft Computing. 2020;86:105836.

[7] Peng G, Nourani M, Harvey J, Dave H. Personalized EEG Feature Selection for Low-Complexity Seizure Monitoring. International Journal of Neural Systems. 2021:2150018–2150018.

[8] Sun H, Jin J, Xu R, Cichocki A. Feature Selection Combining Filter and Wrapper Methods for Motor-Imagery Based Brain-Computer Interfaces. International Journal of Neural Systems. 2011;31(09).

[9] Dias ML, Maia ÁN, da Rocha Neto AR, Gomes JP. Parsimonious Minimal Learning Machine via Multiresponse Sparse Regression. International journal of neural systems. 2020;30(05):2050023.

[10] Mishra P, Piciarelli C, Foresti GL. A Neural Network for Image Anomaly Detection with Deep Pyramidal Representations and Dynamic Routing. International Journal of Neural Systems. 2020;30(10):2050060–2050060.

[11] Liu G, Zhou W, Geng M. Automatic seizure detection based on S-Transform and deep convolutional neural network. International journal of neural systems. 2020;30(04):1950024.

[12] Lin LC, Ouyang CS, Wu RC, Yang RC, Chiang CT. Alternative diagnosis of epilepsy in children without epileptiform discharges using deep convolutional neural networks. International journal of neural systems. 2020;30(05):1850060.

[13] Feng W, Halm-Lutterodt NV, Tang H, Mecum A, Mesregah MK, Ma Y, et al. Automated MRI-Based Deep Learning Model for Detection of Alzheimer's Disease Process. International Journal of Neural Systems. 2020;30(06):2050032.

[14] Leming M, Górriz JM, Suckling J. Ensemble deep learning on large, mixed-site fMRI datasets in autism and other tasks. arXiv preprint arXiv:200207874. 2020.

[15] Hou H, Zhang X, Meng Q. Olfactory eeg signal classification using a trapezoid difference-based electrode sequence hashing approach. International journal of neural systems. 2020;30(03):2050011.

[16] Cura OK, Akan A. Classification of Epileptic EEG Signals Using Synchrosqueezing Transform and Machine Learning. International Journal of Neural Systems. 2021:2150005–2150005.

[17] Ozdemir MA, Cura OK, Akan A. Epileptic eeg classification by using time-frequency images for deep learning. Interna-

tional Journal of Neural Systems. 2021:2150026.

[18] Sánchez-Reolid R, Martínez-Rodrigo A, López MT, Fernández-Caballero A. Deep support vector machines for the identification of stress condition from electrodermal activity. International Journal of Neural Systems. 2020;30(07):2050031.

[19] Lozano A, Suárez JS, Soto-Sánchez C, Garrigós J, Martínez-Alvarez JJ, Ferrández JM, et al. Neurolight: a deep learning neural interface for cortical visual prostheses. International Journal of Neural Systems. 2020;30(09):2050045.

[20] Huang CL, Dun JF. A distributed PSO–SVM hybrid system with feature selection and parameter optimization. Applied soft computing. 2008;8(4):1381–1391.

[21] Zawidzki M, Jankowski Ł. Multiobjective optimization of modular structures: Weight versus geometric versatility in a Truss-Z system. Computer-Aided Civil and Infrastructure Engineering. 2019;34(11):1026–1040.

[22] Wei Y, Jin JG, Yang J, Lu L. Strategic network expansion of urban rapid transit systems: A bi-objective programming model. Computer-Aided Civil and Infrastructure Engineering. 2019;34(5):431–443.

[23] Chiong R, Weise T, Michalewicz Z, editors. Variants of Evolutionary Algorithms for Real-World Applications. Springer; 2012.

[24] Rodrigues D, Papa JP, Adeli H. Meta-heuristic multi- and many-objective optimization techniques for solution of machine learning problems. Expert Systems. 2017;34(6):e12255.

[25] Palacios JJ, Gonzalez-Rodriguez I, Vela CR, Puente J. Satisfying flexible due dates in fuzzy job shop by means of hybrid evolutionary algorithms. Integrated Computer-Aided Engineering. 2019;26(1):65–84.

[26] Wang Q, Liu HL, Yuan J, Chen L. Optimizing the energy-spectrum efficiency of cellular systems by evolutionary multi-objective algorithm. Integrated Computer-Aided Engineering. 2019;26(2):207–220.

[27] Liang Y, He F, Zeng X. 3D mesh simplification with feature preservation based on Whale Optimization Algorithm and Differential Evolution. Integrated Computer-Aided Engineering. 2020;(Preprint):1–19.

[28] Blum C, Chiong R, Clerc M, Jong KAD, Michalewicz Z, Neri F, et al. Evolutionary Optimization. In: Chiong R, Weise T, Michalewicz Z, editors. Variants of Evolutionary Algorithms for Real-World Applications. Springer; 2012. p. 1–29.

[29] Dong H, Li T, Ding R, Sun J. A novel hybrid genetic algorithm with granular information for feature selection and optimization. Applied Soft Computing. 2018;65:33–46.

[30] Paniri M, Dowlatshahi MB, Nezamabadi-pour H. MLACO: A multi-label feature selection algorithm based on ant colony optimization. Knowledge-Based Systems. 2020;192:105285.

[31] Xue B, Zhang M, Browne WN. Particle swarm optimization for feature selection in classification: a multi-objective approach. IEEE Trans Cybern. 2013;43(6):1656–1671.

[32] Hossain SI, Akhand M, Shuvo M, Siddique N, Adeli H. Optimization of university course scheduling problem using particle swarm optimization with selective search. Expert Systems with Applications. 2019;127:9–24.

[33] Hu Y, Zhang Y, Gong D. Multiobjective Particle Swarm Optimization for Feature Selection With Fuzzy Cost. IEEE Transactions on Cybernetics. 2021;51(2):874–888.

[34] Zhang Y, Gong Dw, Gao Xz, Tian T, Sun Xy. Binary differential evolution with self-learning for multi-objective feature selection. Information Sciences. 2020;507:67–85.

[35] Hancer E, Xue B, Zhang M, Karaboga D, Akay B. Pareto front feature selection based on artificial bee colony optimization. Information Sciences. 2018;422:462–479.

[36] Ke L, Feng Z, Xu Z, Shang K, Wang Y. A multiobjective ACO algorithm for rough feature selection. In: 2010 Second Pacific-Asia Conference on Circuits, Communications and System. vol. 1; 2010. p. 207–210.

[37] Khan A, Baig AR. Multi-objective feature subset selection using mRMR based enhanced ant colony optimization algorithm (mRMR-EACO). Journal of Experimental & Theoretical Artificial Intelligence. 2016;28(6):1061–1073.

[38] Neri F, Triguero I. A Local Search with a Surrogate Assisted Option for Instance Reduction. In: Castillo PA, Laredo JLJ, de Vega FF, editors. Applications of Evolutionary Computation - 23rd European Conference, EvoApplications 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings. vol. 12104 of Lecture Notes in Computer Science. Springer; 2020. p. 578–594.

[39] Tran B, Xue B, Zhang M. Variable-length particle swarm optimization for feature selection on high-dimensional classification. IEEE Transactions on Evolutionary Computation. 2018;23(3):473–487.

[40] Chen K, Xue B, Zhang M, Zhou F. Hybridising particle swarm optimisation with differential evolution for feature selection in classification. In: 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2020. p. 1–8.

[41] Kociecki M, Adeli H. Two-phase genetic algorithm for topology optimization of free-form steel space-frame roof structures with complex curvatures. Engineering Applications of Artificial Intelligence. 2014;32:218–227.

[42] Mukhopadhyay A, Maulik U. An SVM-wrapped multiobjective evolutionary feature selection approach for identifying cancer-microRNA markers. IEEE transactions on nanobioscience. 2013;12(4):275–281.

[43] Tan CJ, Lim CP, Cheah YN. A multi-objective evolutionary algorithm-based ensemble optimizer for feature selection and classification with neural network models. Neurocomputing. 2014;125:217–228.

[44] Hancer E, Xue B, Zhang M. Differential evolution for filter feature selection based on information theory and feature ranking. Knowledge-Based Systems. 2018;140:103–119.

[45] Wijnands JS, Zhao H, Nice KA, Thompson J, Scully K, Guo J, et al. Identifying safe intersection design through unsupervised feature extraction from satellite imagery. Computer-Aided Civil and Infrastructure Engineering. 2021;36(3):346–361.

[46] Luo X, Li H, Yu Y, Zhou C, Cao D. Combining deep features and activity context to improve recognition of activities of workers in groups. Computer-Aided Civil and Infrastructure

Engineering. 2020;35(9):965–978.

[47] Yu G, Jin Y, Olhofer M. A multi-objective evolutionary algorithm for finding knee regions using two localized dominance relationships. IEEE Transactions on Evolutionary Computation. 2020.

[48] Rodrigues D, Papa JP, Adeli H. Meta-heuristic multi- and many-objective optimization techniques for solution of machine learning problems. Expert Syst J Knowl Eng. 2017;34(6).

[49] Han H, Liu Z, Hou Y, Qiao J. Data-Driven Multiobjective Predictive Control for Wastewater Treatment Process. IEEE Transactions on Industrial Informatics. 2019;16(4):2767–2775.

[50] Thurnhofer-Hemsi K, López-Rubio E, Roé-Vellvé N, Molina-Cabello MA. Multiobjective optimization of deep neural networks with combinations of Lp-norm cost functions for 3D medical image super-resolution. Integr Comput Aided Eng. 2020;27(3):233–251.

[51] Judt D, Lawson C, van Heerden ASJ. Rapid design of aircraft fuel quantity indication systems via multi-objective evolutionary algorithms. Integr Comput Aided Eng. 2021;28(2):141–158.

[52] Bai Q, Miralinaghi M, Labi S, Sinha KC. Methodology for analyzing the trade-offs associated with multi-objective optimization in transportation asset management under uncertainty. Computer-Aided Civil and Infrastructure Engineering. 2021;36(4):381–401.

[53] Civera M, Pecorelli ML, Ceravolo R, Surace C, Zanotti Fragonara L. A multi-objective genetic algorithm strategy for robust optimal sensor placement. Computer-Aided Civil and Infrastructure Engineering. 2021.

[54] Xue Y, Jiang P, Neri F, Liang J. A Multiobjective Evolutionary Approach Based on Graph-in-graph for Neural Architecture Search of Convolutional Neural Networks. International Journal of Neural Systems.

[55] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation. 2002;6(2):182–197.

[56] Peimankar A, Weddell SJ, Jalal T, Lapthorn AC. Evolutionary multi-objective fault diagnosis of power transformers. Swarm and Evolutionary Computation. 2017;36:62–75.

[57] Labani M, Moradi P, Jalili M. A multi-objective genetic algorithm for text feature selection using the relative discriminative criterion. Expert Systems with Applications. 2020;149:113276.

[58] Karasu S, Saraç Z. Investigation of power quality disturbances by using 2D discrete orthonormal S-transform, machine learning and multi-objective evolutionary algorithms. Swarm and Evolutionary Computation. 2019;44:1060–1072.

[59] Hamdani TM, Won JM, Alimi AM, Karray F. Multi-objective feature selection with NSGA II. In: International conference on adaptive and natural computing algorithms. Springer; 2007. p. 240–247.

[60] Gaspar-Cunha A, Recio G, Costa L, Estébanez C. Self-adaptive MOEA feature selection for classification of bankruptcy prediction data. The Scientific World Journal. 2014;2014.

[61] Das AK, Das S, Ghosh A. Ensemble feature selection using bi-objective genetic algorithm. Knowledge-Based Systems. 2017;123:116–127.

[62] Spolaôr N, Lorena AC, Lee HD. Multi-objective genetic algorithm evaluation in feature selection. In: International Conference on Evolutionary Multi-Criterion Optimization. Springer; 2011. p. 462–476.

[63] Bouraoui A, Jamoussi S, BenAyed Y. A multi-objective genetic algorithm for simultaneous model and feature selection for support vector machines. Artificial Intelligence Review. 2018;50(2):261–281.

[64] Yang W, Li D, Zhu L. An improved genetic algorithm for optimal feature subset selection from multi-character feature set. Expert Systems with Applications. 2011;38(3):2733–2740.

[65] Deep K, Thakur M. A new crossover operator for real coded genetic algorithms. Applied mathematics and computation. 2007;188(1):895–911.

[66] Zhang Qy, Chang Sc. An improved crossover operator of genetic algorithm. In: 2009 Second International Symposium on Computational Intelligence and Design. vol. 2. IEEE; 2009. p. 82–86.

[67] Umbarkar AJ, Sheth PD. Crossover operators in genetic algorithms: a review. ICTACT Journal on Soft Computing. 2015;6(1).

[68] Caruana RA, Eshelman LJ, Schaffer JD. Representation and Hidden Bias II: Eliminating Defining Length Bias in Genetic Search via Shuffle Crossover. In: IJCAI'89: Proceedings of the 11th international joint conference on Artificial intelligence. vol. 1; 1989. p. 750–755.

[69] Picek S, Golub M. Comparison of a crossover operator in binary-coded genetic algorithms. WSEAS transactions on computers. 2010;9(9):1064–1073.

[70] Iacca G, Neri F, Caraffini F, Suganthan PN. A Differential Evolution Framework with Ensemble of Parameters and Strategies and Pool of Local Search Algorithms. In: Esparcia-Alcázar AI, Mora AM, editors. Applications of Evolutionary Computation - 17th European Conference, EvoApplications 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers. vol. 8602 of Lecture Notes in Computer Science. Springer; 2014. p. 615–626.

[71] Caraffini F, Neri F, Epitropakis MG. HyperSPAM: A study on hyper-heuristic coordination strategies in the continuous domain. Inf Sci. 2019;477:186–202.

[72] Bache K, Lichman M. UCI Machine Learning Repository; 2013.

[73] Xue Y, Xue B, Zhang M. Self-adaptive particle swarm optimization for large-scale feature selection in classification. ACM Transactions on Knowledge Discovery from Data (TKDD). 2019;13(5):1–27.

[74] Xue B, Zhang M, Browne WN. Particle swarm optimization for feature selection in classification: A multi-objective approach. IEEE Transactions on Cybernetics. 2012;43(6):1656–1671.

[75] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving

the strength Pareto evolutionary algorithm. TIK-report. 2001;103.
[76] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation. 2007;11(6):712–731.
[77] Rostami S, Neri F, Gyaurski K. On Algorithmic Descriptions and Software Implementations for Multi-objective Optimisation: A Comparative Study. SN Comput Sci. 2020;1(5):247.
[78] Van Veldhuizen DA, Lamont GB. Multiobjective evolutionary algorithm research: A history and analysis. Citeseer; 1998.
[79] Zitzler E, Thiele L. Multiobjective optimization using evolutionary algorithms—a comparative case study. In: International Conference on Parallel Problem Solving from Nature. Springer; 1998. p. 292–301.
[80] Rostami S, Neri F. Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. Integr Comput Aided Eng. 2016;23(4):313–329.
[81] Rostami S, Neri F. A fast hypervolume driven selection mechanism for many-objective optimisation problems. Swarm Evol Comput. 2017;34:50–67.
[82] Rostami S, Neri F, Epitropakis MG. Progressive preference articulation for decision making in multi-objective optimisation problems. Integr Comput Aided Eng. 2017;24(4):315–335.
[83] Wilcoxon F. Individual comparisons by ranking methods. Biometrics Bulletin. 1945;1(6):80–83.
[84] Ahmadlou M, Adeli H. Enhanced probabilistic neural network with local decision circles: A robust classifier. Integrated Computer-Aided Engineering. 2010;17(3):197–210.
[85] Rafiei MH, Adeli H. NEEWS: A novel earthquake early warning model using neural dynamic classification and neural dynamic optimization. Soil Dynamics and Earthquake Engineering. 2017;100:417–427.
[86] Alam KMR, Siddique N, Adeli H. A dynamic ensemble learning algorithm for neural networks. Neural Computing and Applications. 2020;32(12):8675–8690.
[87] Pereira DR, Piteri MA, Souza AN, Papa JP, Adeli H. FEMa: a finite element machine for fast learning. Neural Computing and Applications. 2020;32(10):6393–6404.
[88] Park HS, Adeli H. Distributed neural dynamics algorithms for optimization of large steel structures. Journal of Structural Engineering. 1997;123(7):880–888.
[89] Siddique N, Adeli H. Spiral dynamics algorithm. International Journal on Artificial Intelligence Tools. 2014;23(06):1430001.
[90] Siddique N, Adeli H. Harmony search algorithm and its variants. International Journal of Pattern Recognition and Artificial Intelligence. 2015;29(08):1539001.
[91] Siddique N, Adeli H. Water drop algorithms. International Journal on Artificial Intelligence Tools. 2014;23(06):1430002.
[92] Gutlein M, Frank E, Hall M, Karwath A. Large-scale attribute selection using wrappers. In: 2009 IEEE symposium on computational intelligence and data mining. IEEE; 2009. p. 332–339.
[93] Caruana R, Freitag D. Greedy attribute selection. In: Machine Learning Proceedings 1994. Elsevier; 1994. p. 28–36.

Appendix

In this section, two conventional wrapper feature selection methods, i.e., linear forward selection (LFS) [92] and greedy stepwise backward selection (GSBS) [93], are adopted to further examine the performance of SaMOGA.

LFS and GSBS were derived from sequential forward selection and sequential backward selection, respectively. LFS limits the number of features to be considered in every single step of forward selection so that the number of evaluations can be reduced. GSBS starts with the full set and keeps removing features until the classification error no longer decreases.

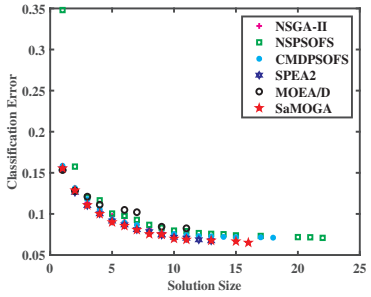
The results obtained by LFS, GSBS and SaMOGA in some of the datasets are shown in Table 7. "Size-A" and "Size-B" represent the average and smallest size of feature subsets in 30 runs, respectively. "Err-A" and "Err-B" represent the average and smallest classification error in 30 runs, respectively. For each dataset, LFS and GSBS obtain a unique feature subset. So they don't have the value smallest size of feature subsets and the smallest classification error, we use the symbol "-" to represent this.

From Table 7, we can see that SaMOGA achieves better classification performance and reduces more features than LFS and GSBS do. It is also worth noting that SaMOGA is able to obtain only one feature on all of the datasets. The results indicate that SaMOGA has better performance compared to the traditional feature selection method.

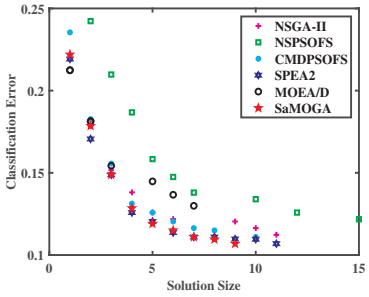
Figures 5 and 6 show the plots that were not included in the paper.

Table 7. Results of LFS, GSBS and SAMOGA

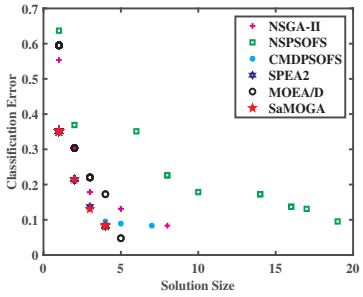
Dataset	Algorithm	Size-A	Err-A	Size-B	Err-B
Wdbc	LFS	10	11.11	-	-
	GSBS	25	16.37	-	-
	SaMOGA	3	3.56	1	2.73
Ionosphere	LFS	4	13.33	-	-
	GSBS	30	21.90	-	-
	SaMOGA	2.50	7.01	1	4.47
LungCancer	LFS	6	10	-	-
	GSBS	33	10	-	-
	SaMOGA	3	16.55	1	4.76
MadelonValid	LFS	7	35.38	-	-
	GSBS	489	48.72	-	-
	SaMOGA	4.82	15.97	1	9.54
Isolet5	LFS	24	1.66	-	-
	GSBS	560	2.84	-	-
	SaMOGA	26.40	23.17	1	9.73



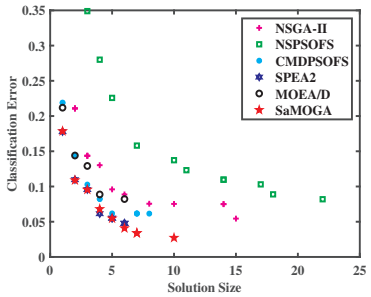
DS03 (36, 0.08)



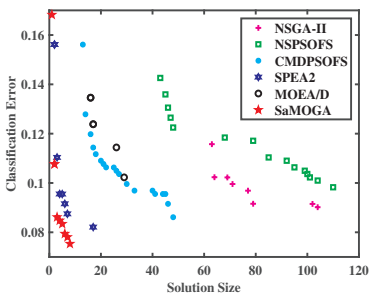
DS04 (41, 0.21)



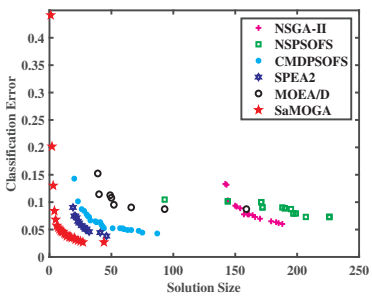
DS05 (56, 0.60)



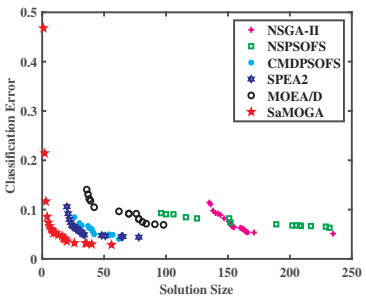
DS06 (60, 0.27)



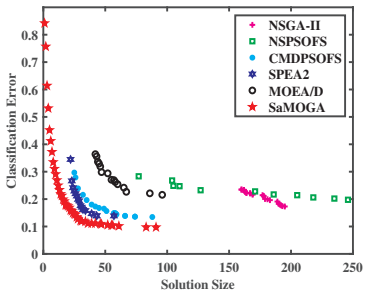
DS09 (301, 0.24)



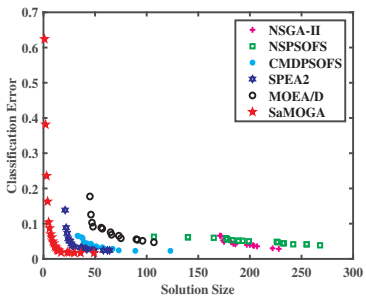
DS12 (561, 0.11)



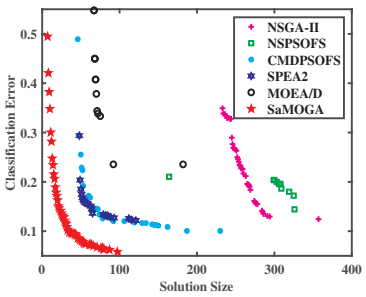
DS13 (561, 0.09)



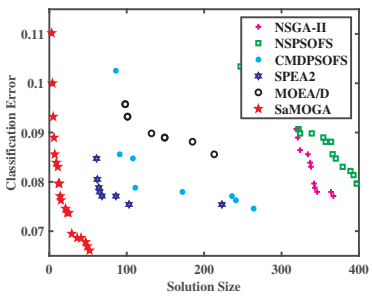
DS14 (617, 0.28)



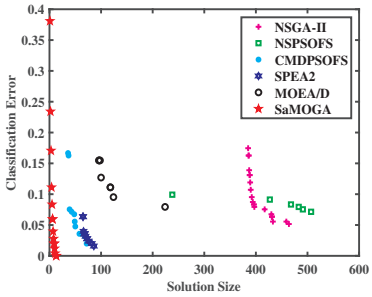
DS15 (649, 0.08)



DS16 (856, 0.20)

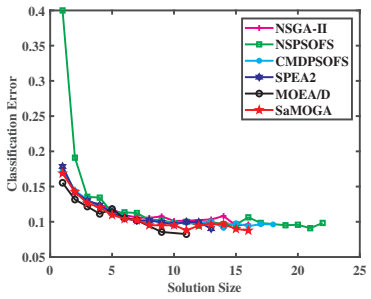


DS17 (1024,0.10)

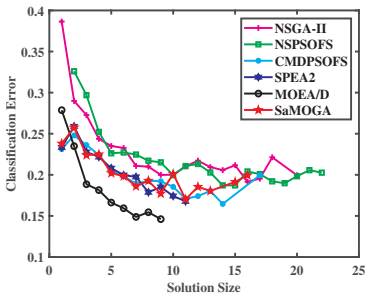


DS18 (1300,0.20)

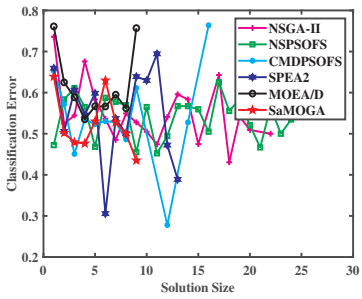
Figure 5. Best PFs Evolved on Training Sets



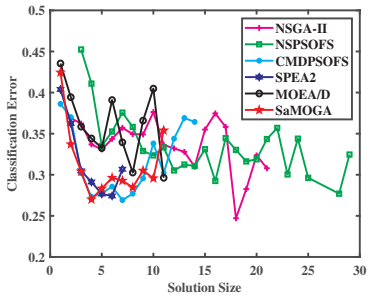
DS03 (36, 0.27)



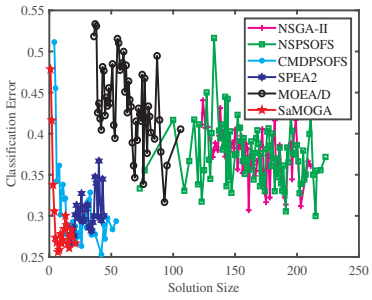
DS04 (41, 0.21)



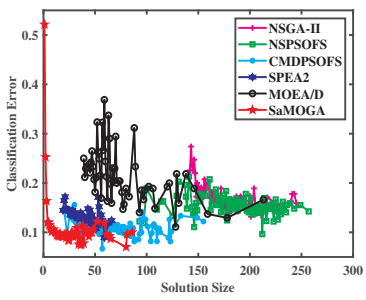
DS05 (56, 0.39)



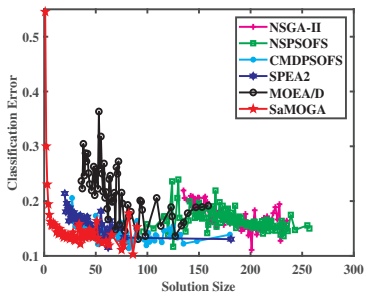
DS06 (60, 0.37)



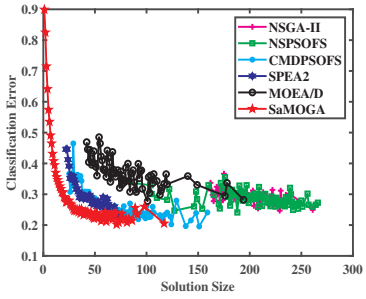
DS10 (500, 0.40)



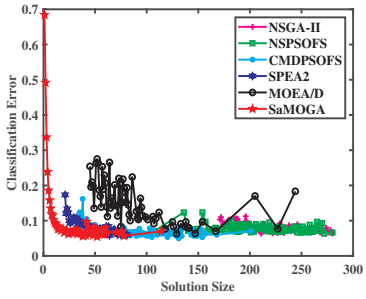
DS12 (561, 0.17)



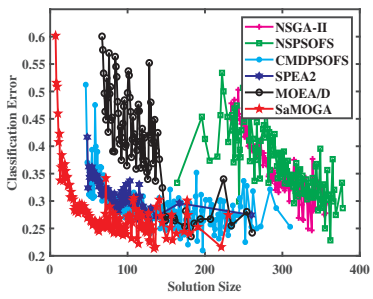
DS13 (561, 0.17)



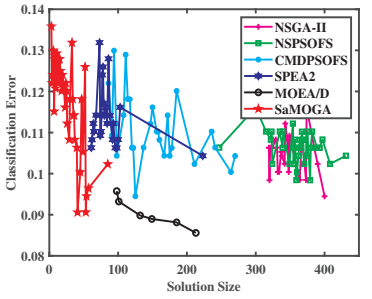
DS14 (617, 0.31)



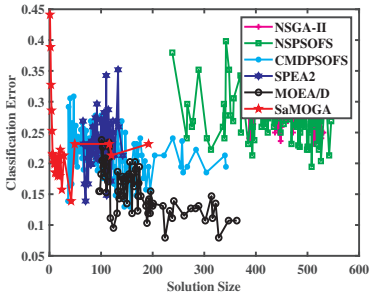
DS15 (649, 0.12)



DS16 (856, 0.25)



DS17 (1024, 0.25)



DS18 (1300, 0.31)

Figure 6. Average PFs Obtained on Test Sets