A Hybrid GA-GP Method for Feature Reduction in Classification

Hoai Bach Nguyen(✉), Bing Xue, and Peter Andreae

School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand
{Hoai.Bach.Nguyen,Bing.Xue,Peter.Andreae}@ecs.vuw.ac.nz

Abstract. Feature reduction is an important pre-processing step in classification and other artificial intelligent applications. Its aim is to improve the quality of feature sets. There are two main types of feature reduction: feature construction and feature selection. Most current feature reduction algorithms focus on just one of the two types because they require different representations. This paper proposes a new representation which supports a feature reduction algorithm that combines feature selection and feature construction. The algorithm uses new genetic operators to update the new representation. The proposed algorithm is compared with two conventional feature selection algorithms, a genetic algorithms-based feature selection algorithm, and a genetic programming-based algorithm which evolves feature sets containing both original and high-level features. The experimental results on 10 different datasets show that the new representation can help to produce a smaller number of features and improve the classification accuracy over using all features on most datasets. In comparison with other feature selection or construction algorithms, the proposed algorithm achieves similar or better classification performance on all datasets.

1 Introduction

One of the most important tasks of machine learning is classification [1] which assigns a class label to an instance. The classification is based on the instance's properties, known as features. The quality of the feature set significantly affects the classification performance. However, in many real-world problems, there are many irrelevant and redundant features that not only reduce the classification accuracy, but also increase the complexity of the learned classifier and the training time.

Feature reduction is typically a pre-processing step, which aims to create an informative feature set from the original one to improve the classification performance. In feature reduction, there are two main approaches including feature selection and feature construction. While feature selection aims at selecting a good feature subset from the original features, feature construction builds one or more high-level features with better discrimination ability. Since feature selection does not produce any new feature, it maintains the meanings of the original features in each problem. On the other hand, feature construction aims to combine original features to achieve better classification performance.

Although feature reduction is a useful step, it is not an easy task. Given N original features, the task of feature selection is to find an optimal subset among 2^N possible feature subsets. So the search space of feature selection increases exponentially with respect to the number of features. Besides the large search space, the complex interactions between features make feature selection a challenging task. For example, two relevant features may provide the same information about the class label, which means it is redundant to select both features. Meanwhile, selecting two weakly relevant features may significantly improve the classification performance if they are complementary features [2]. In order to achieve feature construction, the first step is to select a good feature subset, based on which the new high-level features are built. Therefore the search space of feature construction is even larger than feature selection since it also needs to consider how to combine features. In summary, feature reduction has two main difficulties: large search spaces and complex feature interactions which are usually addressed by two key factors: the search technique and the evaluation criteria, respectively.

According to the evaluation criteria, feature reduction can be divided into two main categories: wrappers and filters [3]. Wrapper approaches usually evaluate feature subsets by a classification algorithm. On the other hand, filters do not involve any classification algorithm during the evaluation process. In filters, the goodness of a feature set is measured by characteristics of datasets. Filters are usually more efficient and result in more general feature sets. However, wrappers usually achieve higher classification accuracies. Therefore, in this work feature reduction is achieved by wrapper approaches.

Evolutionary computation (EC) has been widely applied to feature reduction because of its potential global search ability. EC-based feature construction is usually achieved by genetic programming (GP) [4] since GP can automatically evolve mathematic formulas without any assumptions about the structure of the formulas. In addition, the tree-like representation of GP is quite flexible, which makes it easy to express complex solutions with different kinds of operators and functions. For EC-based feature selection, genetic algorithms (GAs) and particle swarm optimization (PSO) are the most popular approaches [2]. Among EC algorithms, GAs were the earliest search mechanism applied to feature selection because of its natural representation of a binary string.

Most existing feature reduction works focus on either feature selection or feature construction. However, Tran et al. [5] showed that the combination of new high-level features and the original features appearing in the high-level features achieves a better classification performance than using only selected features or only constructed features. This suggests that it is promising to do feature construction and feature selection together so that we can take the advantages of both methods. This was partially done by Tran et al. [5] but only constructed features were evolved during the evolutionary process while the selected features were chosen at the end, which meant that the interactions between constructed and selected features were ignored. In this work, a new representation, which can be seen as a combination of bit vector and tree representations, is proposed to simultaneously perform feature selection and feature construction while taking into account the interactions between the two kinds of features.

Goal: The overall goal of this paper is to develop a new scheme for feature reduction to evolve small new feature sets with better classification performance than using all features. The new representation scheme and genetic operators are designed so that feature selection and feature construction can be performed simultaneously. The proposed feature reduction algorithm is then compared with a GA-based feature selection algorithm, a GP-based feature construction algorithm, and two conventional feature selection algorithms on 10 datasets with different numbers of features, classes and instances. Specifically, we will investigate:

- whether the new algorithm can reduce the number of features while increasing the classification performance over using all features,
- whether the new algorithm can evolve smaller feature sets with better classification accuracy than the GA-based feature selection algorithm,
- whether the new scheme helps to build a small feature set containing both high-level and original features that achieves better performance than the high-level feature and/or selected features evolved by GP,
- whether the new algorithm can outperform two traditional (non-EC) feature selection algorithms.

2 Background

2.1 Genetic Algorithms (GAs) and Genetic Programming (GP)

GAs [6] are one of the first EC algorithms, which are inspired by the natural selection from the Darwinian theory of evolution. In GAs, an optimization problem is solved by a population of candidate solutions. Each candidate solution is represented by a fixed-length bit string which is also known as a chromosome. The algorithm starts with a number of random chromosomes. During the evolutionary process, each candidate solution is evaluated by a fitness function. The *selection* scheme ensures that the chromosomes with better fitness values have higher chance to be selected or survival in the next generation. Some genetic operators such as *crossover* and *mutation* are applied to the selected chromosomes to produce new chromosomes for the next generation. It is expected that exchanging information between two good parents can result in better children. In addition, the fittest chromosomes is possibly preserved by the *elitism* mechanism. The new generation is then evaluated and enhanced in the following iteration. The algorithm terminates when a maximum number of iterations is reached, and/or a satisfactory fitness value has been achieved. It can be seen that GAs use a vector representation, which is a natural representation for feature selection. Specifically, in the binary vector representation, “1” shows the corresponding feature is selected and “0” means not selected.

Similar to GAs, GP [7] is a population-based optimization algorithm in which the candidate solutions are evolved by a number of genetic operators such as *selection*, *crossover*, *mutation* and *elitism*. The difference between GAs and GP is mainly on their representations. In GP, each candidate solution is usually represented by a tree, where the decision variables/features are the leaf nodes and each internal node is a function selected from a predefined function set. So each candidate solution, presented by a tree representation, can be seen as a high-level function, which maps from a number of original features to a high-level feature. GP is a domain-independent method since it does not require any domain knowledge such as any assumption about the model. Therefore, GP is usually used to achieve feature construction.

2.2 Related Work on Feature Reduction

The easiest way to achieve feature selection was to consider all possible feature subsets, which guaranteed to produce an optimal feature subset. However, this method was very computationally intensive and impossible when there was a large number of features. In order to reduce the computation cost, two sequential searches including forward (SFS) and backward (SBS) selection approaches were proposed [8]. Starting from an empty or full set of features, in each iteration SFS or SBS added or removed one feature from the current feature subset. The algorithms terminated when a pre-defined number of features was reached. However, the decision on a feature could not be changed once the feature was added or removed. The issue was addressed in two floating sequential forward (SFFS) and backward (SBFS) selection methods [8], which performed an additional step to remove or select features after each of the forward or backward step.

EC has been widely applied to feature selection. GAs [9,10] and PSO [11,12] are the two most popular techniques because of their natural representations for feature selection. Some works focused on improving the feature subset qualities by estimating good starting points for the population, such as [13,14]. The experimental results suggested that a good initialization strategy not only improved the classification performance but also shortened the training time. Representations were also modified to further enhance the feature subset qualities. For example, Vieira et al. [15] included a classifier's parameters into an individual's position in PSO to simultaneously optimize both feature subsets and the parameters. Statistical feature clustering information was also utilized to shorten the representation as in [16,17], which allowed to select the most important features in different runs consistently. Besides effectiveness, improving efficiency is also an important aspect in feature selection. Nguyen et al. [18] used a small number of training instances to estimate promising search regions before further exploring the regions using the whole training set. In addition, a local search mechanism was proposed to use information from the previous iterations to improve the current feature subsets. The proposed algorithm not only significantly reduced the computation time but also evolved better feature subsets than using the whole training set.

In terms of feature construction, GP was used to build a single high-level feature by using a single tree-representation for a given problem [19]. However, a single constructed feature usually did not contain enough information to well describe the problem. Therefore, GP was extended to construct multiple high-level features. A straight-forward way was to used a multi-tree representation [5], in which each individual consisted of more than one tree and each tree corresponded to one constructed feature. The experimental results showed that using the constructed and selected features resulted in better accuracies than using either of them. However, the selected features were chosen from features used as leaf nodes in the best constructed features in the last iteration, which were not explicitly evaluated during the evolutionary process. This means that their interactions might be ignored. For example, the selected features and constructed features might be redundant since the selected feature was already included in the constructed features. Furthermore, some original features, which were complementary with the constructed features, might not be selected since they were not used for constructing features. This work will address the above issues by proposing a new representation to achieve both feature selection and feature construction.

3 Proposed Approach

3.1 Hybrid GA-GP: A New Representation

This section describes a new representation scheme for feature reduction to conduct feature selection and feature construction simultaneously. The vector representation used in GAs is most appropriate for feature selection and the tree representation used in GP is most appropriate for feature construction. In order to achieve the two tasks at the same time, the new representation is designed as a combination of the vector and the tree representations. The reason for selecting GAs and GP over other EC algorithms such as PSO and differential evolution (DE) is that the similar evolutionary mechanisms of the two selected algorithms make it easier to combine them into a single algorithm.

In the new representation scheme, each individual includes two parts. The first component of the new representation (FS) is a bit string, as in GAs. Each bit corresponds to one original feature, so the length of the string is equal to the number of original features. A bit's value of "1" means the corresponding feature is selected, otherwise the feature is discarded. The second component of the new representation (FC) is a tree, as in GP, and represents a constructed feature. Therefore, each candidate solution is a feature set containing a subset of original features and a newly constructed feature. The essential difference between this representation and the work conducted by Tran et al. [5] is that the selected and constructed features are evaluated together as a single feature set. Therefore, the new representation is able to consider the interactions between selected features and the constructed feature. An example of the new representation with four original features, $\{f_0, f_1, f_2, f_3\}$, is given in Fig. 1. As can be seen in the figure, the FS part is a binary vector with a length of 4. The bit values indicate that

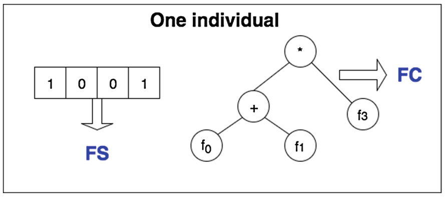

Fig. 1. An example of the new representation with a dataset including 4 features

two original features, f_0 and f_3, are selected. The FC part contains a single tree, which uses 3 original features f_0, f_1 and f_3 to construct a high-level feature, $(f_0 + f_1) * f_3$. Therefore, the individual defines a new feature set, which is $\{f_0, f_3, ((f_0 + f_1) * f_3)\}$.

3.2 Genetic Operators

The aim of the algorithm that uses this representation is to co-evolve the bit vector (FS) and the tree representation (FC) together so that the selected original features and the constructed feature in the final feature set are complementary in achieving better classification performance.

The *selection* operator of the algorithm works the same way as *selection* in traditional GAs and GP. In this work, a tournament selection is applied. Specifically, each time from the population, a number of individuals are picked, from which the fittest one is selected. This process is repeated to select a fixed number of individuals which work as parents for *crossover*, the next operator.

Since each individual has both a bit string (FS) and a tree (FC), the two operators *crossover* and *mutation* must be modified to cope with the new representation. In *crossover*, firstly two parents are randomly selected. After that, the *crossver* is performed on both FS and FC components, simultaneously. Specifically, a single crossover point on the two bit strings is selected and two crossover nodes in two trees are randomly selected. All bits beyond the crossover point are swapped and the two subtrees rooted at the two crossover nodes are exchanged to form two new trees. The results of the *crossover* operator is two new offspring and each of them contains a new bit string and a new tree. In Fig. 2, the vertical green dash-line shows the crossover point between the two bit strings and the 4^{th} bits of the two strings are exchanged. The crossover node and its corresponding subtree are marked in green.

As in *crossover*, *mutation* is also changed so that both components in the new representation are mutated. Given a parent randomly selected from the population, its bit string is mutated by firstly selecting some bits randomly and then flipping the selected bits (change 1 to 0 and 0 to 1). Similarly, the tree from the FC part is mutated by replacing a random subtree by a small newly generated tree. Therefore, the *mutation* operator replaces a selected parent by an offspring, which also has both a bit string and a tree. An example of *mutation*

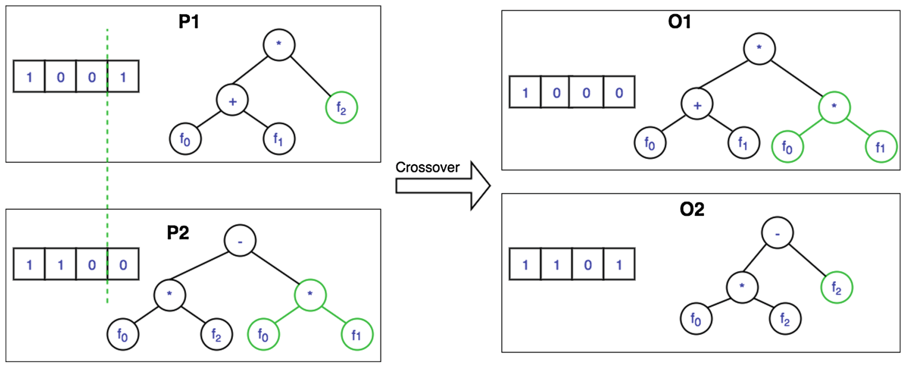

Fig. 2. An example of crossover for the new representation (Color figure online)

is shown in Fig. 3, where the bits and subtree selected for *mutation* are filled in green. Specifically, the subtree $\{f_2\}$ is replaced by another small tree, $\{f_0 * f_1\}$.

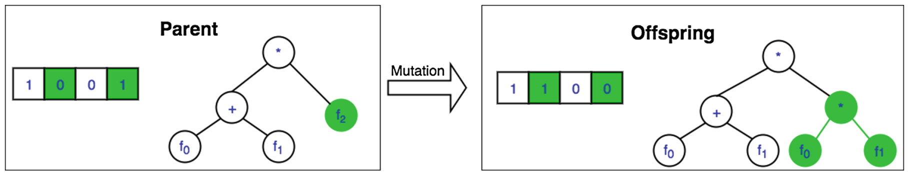

Fig. 3. An example of mutation for the new representation (Color figure online)

3.3 Fitness Function

In feature reduction, there are two main objectives: to minimize the classification error and to minimize the number of features. The algorithm combines the two objectives into a single fitness function, as shown in the following equation:

$$fitness = \alpha * ErrorRate + (1 - \alpha) * \frac{\#manipulatedFeatures}{\#originalFeatures} \tag{1}$$

where *ErrorRate* is the classification error of the new feature set, *#manipulatedFeatures* and *#originalFeatures* represent the number of features in the new feature set and the total number of original features, respectively. α is used to control the contribution of the two objectives. The task of a feature reduction algorithm is to minimize the fitness value calculated by Eq. (1). The classification error is calculated based on the new feature set including selected original features from the bit string (FS) and constructed features from the tree (FC), which means that the interaction between the two parts of the new representation is considered.

Note that in a GA-based feature selection algorithm, the new feature set is actually a subset of features selected from the original feature set. So the features inside the new feature set are **not** new features. On the other hand, in the proposed feature reduction algorithm with the new representation, the new feature set contains a new constructed feature, along with a subset of the original features.

3.4 Overall Algorithm

Algorithm 1 shows the pseudo-code of the proposed algorithm, which combines GAs and GP together to perform feature selection and feature construction simultaneously. The proposed algorithm is called HGAGP. In the initialization, each bit in a bit string is randomly assigned to 0 or 1. Trees in FC components are initialized using the ramped half-and-half strategy.

Algorithm 1. Hybrid GA-GP for feature reduction (HGAGP)

```
1: Initialize the population;
2: Evaluate each individual according to Eq. (1);
3: while Maximum number of iterations is not reached do
4:    Perform selection operator;
5:    Perform crossover operator;
6:    Perform mutation operator;
7:    Evaluate the new population according to Eq. (1);
8: end while
9: Return the best individual including the selected and constructed features with its
   corresponding training and testing accuracies.
```

4 Experiment Design

To examine the performance of the proposed algorithm HGAGP, two traditional feature selection methods, which are floating sequential forward and backward selection (SFFS and SBFS) [8] and two feature reduction algorithms based on GAs and GP are used as benchmark techniques. Note that for the GP based algorithm, GP is used to construct a single feature. The feature set evolved by HGAGP is compared with a single constructed feature (GP), and the combination of the constructed feature with the selected original features used as leaf nodes in the constructed feature (GPWS).

The algorithms are compared on 10 different datasets selected from the UCI machine learning repository [20]. The datasets have different numbers of features, classes and instances (Table 1). For each dataset, the instances are divided into training and test sets with the proportions of 70% and 30%, respectively. The division ensures that the class distribution are roughly maintained in the two instance subsets. On each dataset, each algorithm is ran 50 independent times.

Table 1. Datasets

Dataset	No. of features	No. of classes	No. of instances
Wine	13	3	178
Vehicle	18	4	946
WBCD	30	2	569
Ionosphere	34	2	351
Sonar	60	2	208
Movementlibras	90	15	360
Hillvalley	101	2	606
Musk1	166	2	476
Arrhythmia	279	16	452
Isolet5	617	5	7797

All feature reduction algorithms in this work are wrapper approaches, in which each candidate solution is evaluated by the K-nearest neighbor (KNN) classification algorithm. K is set to 5 so that KNN can avoid the noisy instances while maintaining its efficiency. During the training process, 10-fold cross validation is applied to measure the classification error of a feature set to be used in the fitness function. The α value in Eq. (1) is set to 0.9 so that the search focuses more on improving the classification performance.

For all algorithms, the population size is set to three times the total number of original features since the search space size increases exponentially with respect to the number of features. However, the population size is limited to 100. The maximum initial depth of a tree is set to 7 and the maximum tree depth is 17 to avoid the bloating problem in GP. The crossover and mutation rates for GAs, GP and HGAGP are 0.8 and 0.2, respectively, which follows the parameter settings by Tran et al. [5] to ensure a fair comparison. In this work, followed the implementation by the DEAP package [21], *elitism* is not implemented. However the best solution evolved during the evolutionary process is recorded, which is returned at the end of each algorithm. All algorithms stop after 50 generations. Different algorithms are compared by a statistical significance test, Wilcoxon signed rank test with the significance level being set to 0.05.

5 Results and Discussions

5.1 Results on the Training Set

The average training accuracies of the feature sets evolved by the four algorithms over 50 independent runs are shown in Table 2. In the table, "Full" means that all the original features are used for classification. The significance test results are shown in the brackets, where "+"/"−" means that the corresponding algorithm is significantly better/ worse than the proposed algorithm, HGAGP; "="

Table 2. Training accuracies

Datset	Full	GAs	GP	GPWS	HGAGP
Wine	96.74 (−)	98.01 (−)	97.34 (−)	97.41 (−)	99.10
Vehicle	84.26 (−)	83.99 (−)	77.19 (−)	79.52 (−)	85.72
WBCD	97.22 (−)	97.73 (−)	98.13 (=)	97.68 (−)	97.96
Ionosphere	92.24 (−)	95.64 (−)	95.77 (−)	94.68 (−)	96.70
Sonar	91.66 (−)	97.13 (=)	88.73 (−)	89.93 (−)	97.08
Movementlibras	89.64 (−)	89.92 (=)	64.31 (−)	75.15 (−)	90.22
Hillvalley	78.15 (−)	80.77 (−)	99.57 (=)	95.80 (−)	99.49
Musk1	92.77 (−)	95.64 (=)	85.56 (−)	87.40 (−)	95.86
Arrhythmia	69.84 (−)	77.18 (−)	71.82 (−)	73.19 (−)	78.55
Isolet5	91.19 (−)	94.41 (=)	57.53 (−)	78.36 (−)	94.58

means not significantly different. As can be seen in the table, HGAGP achieves better training accuracies than using all features on all datasets. Especially on Hillvalley, the accuracy of HGAGP is 20% higher than using all features. Similarly, HGAGP is significantly better than GPWS on all datasets. In comparison with GAs and GP, HGAGP significantly outperforms them on a majority of the datasets. For example, on Isolet5, HGAGP's training accuracy is 94.58% which is almost twice higher than the one obtained by GP.

5.2 Results on the Test Set

Feature sets evolved by the four algorithms were evaluated on unseen instances from the test set. The average testing accuracies are shown in Table 3. As can

Table 3. Testing accuracies

Datset	Full	GAs	GP	GPWS	HGAGP
Wine	98.14 (+)	93.48 (−)	91.33 (−)	94.14 (−)	96.62
Vehicle	66.14 (−)	72.99 (−)	62.07 (−)	65.95 (−)	74.09
WBCD	98.83 (+)	95.99 (=)	96.10 (=)	96.37 (=)	95.82
Ionosphere	78.09 (−)	84.55 (=)	84.49 (=)	83.75 (=)	84.74
Sonar	77.77 (−)	79.65 (=)	64.76 (−)	69.84 (−)	80.66
Movementlibras	75.00 (=)	74.96 (=)	27.00 (−)	47.22 (−)	74.77
Hillvalley	58.24 (−)	61.76 (−)	99.13 (=)	90.72 (−)	98.92
Musk1	76.92 (−)	86.65 (=)	64.40 (−)	70.61 (−)	85.93
Arrhythmia	53.67 (−)	59.41 (−)	58.86 (−)	60.44 (−)	62.52
Isolet5	75.42 (−)	84.79 (−)	28.18 (−)	60.74 (−)	85.14

be seen from the table, on 7 out of the 10 datasets, HGAGP improves the testing accuracies over using all features. Especially on the Hillvalley dataset, in comparison with using all features, HGAGP is almost two times more accurate. Compared with GP and GPWS, on most datasets HGAGP performs significantly better. For example, on Isolet5, the largest dataset, HGAGP's accuracy is almost 60% and 25% higher than GP and GPWS, respectively. HGAGP is significantly better than GAs on 5 datasets and achieves similar performance on the other datasets. The most significant difference between the two algorithms is on Hillvalley, where HGAGP's accuracy is almost 40% better than that of GAs. Therefore, on all datasets the proposed algorithm is never significantly worse than GAs, GP or GPWS.

It can be seen that GAs, GP and GPWS do not perform consistently well on all datasets. On some datasets, feature selection with GAs is more suitable. On the other datasets, constructing features with GP is a better method. However, HGAGP shows that it can adapt with different datasets to consistently produce good features on all datasets since it does both feature selection and feature construction at the same time. Although GPWS also combines a constructed feature with original features, it does not consider the interaction between the two kinds of features, which results in its worse performance than HGAGP.

5.3 Size of New Feature Sets

The size of new feature sets evolved by the algorithms is shown in Table 4. Since the target is to select a small number of features, the smaller feature set the better the algorithm. GP necessarily produces the smallest feature set because it only constructs a single feature. It can be seen that on all datasets, HGAGP evolves feature sets containing less than 30% of the total number of original features. On the small datasets, HGAGP usually selects a similar or smaller number of features than other algorithms (except for GP). When the

Table 4. Number of features in the new feature set

Datset	Full	GAs	GP	GPWS	HGAGP
Wine	13.0 (−)	4.2 (=)	1.0 (+)	8.8 (−)	4.3
Vehicle	18.0 (−)	7.3 (=)	1.0 (+)	7.5 (=)	7.5
WBCD	30.0 (−)	6.1 (=)	1.0 (+)	10.8 (−)	5.7
Ionosphere	34.0 (−)	7.4 (=)	1.0 (+)	10.5 (−)	6.9
Sonar	60.0 (−)	19.7 (+)	1.0 (+)	20.4 (=)	21.7
Movementlibras	90.0 (−)	30.8 (=)	1.0 (+)	15.1 (+)	31.7
Hillvalley	100.0 (−)	30.4 (−)	1.0 (+)	14.5 (−)	5.1
Musk1	166.0 (−)	56.5 (=)	1.0 (+)	24.2 (+)	58.2
Arrhythmia	278.0 (−)	85.5 (+)	1.0 (+)	33.1 (+)	94.9
Isolet5	617.0 (−)	234.6 (=)	1.0 (+)	28.0 (+)	240.0

number of original features is increased, HGAGP tends to select more features to preserve its high classification performance. So the experimental results show that HGAGP can adapt with different numbers of features to maintain its high classification accuracies. Due to space limitations the computation cost is not included. In general, HGAGP is a little bit more expensive than GAs and GP since it needs to transform the original dataset using both bit strings and trees.

5.4 Further Comparison with Traditional Methods

A comparison between HGAGP and two sequential searches, SFFS and SBFS, is shown in Table 5. The number of features evolved by HGAGP is used as stopping criteria for the two sequential searches. In the table, on each dataset (each column) the best testing accuracy is marked in bold. As can be seen from the table, HGAGP achieves the best performance on 7 out of the 10 datasets. The largest difference between the three methods is on Hillvalley, where HGAGP is almost twice accurate than SFFS and SBFS. The experimental results show that the new representation helps HGAGP to better explore the search space than the two sequential algorithms.

Table 5. Results of SFFS and SBFS

Method	Wine	Vehicle	WBCD	Ionosphere	Sonar
SFFS	**98.15**	69.69	92.98	81.91	74.60
SBFS	**98.15**	68.90	92.98	84.70	**80.95**
HGAGP	96.62	**74.09**	**95.92**	**84.74**	80.66
Method	Movementlibras	Hillvalley	Musk1	Arrhythmia	Isolet5
SFFS	73.15	64.56	82.52	**64.71**	83.55
SBFS	73.15	59.34	82.52	55.88	76.92
HGAGP	**74.77**	**98.92**	**85.93**	62.52	**85.14**

6 Conclusions and Future Work

The goal of the research was to develop a new representation for feature reduction using EC, which not only performs feature construction and selection simultaneously but also considers the interaction between the two kinds of features. The goal has been achieved by combining a bit string and a tree together to form a new representation. The genetic operators were also redesigned to suit the new representation. The experimental results on 10 different datasets show that the proposed algorithm, HGAGP, can evolve smaller feature sets with better classification accuracy than using all features. Since HGAGP performs feature selection and construction at the same time, it can consistently achieve good performance on all datasets. In addition, considering the interactions between original and constructed features helps HGAGP to outperform using GAs for feature selection only, and using GP for feature construction and/or feature selection.

In the future, we will investigate on how to construct multiple features along with selecting features to further improve the classification performance. In addition, it will be interesting to exchange the information between two parts in the new representation. For example, the selected features in feature selection can be used to enhance the constructed features in feature construction and vice versa.

References

1. Lones, M.A., Smith, S.L., Alty, J.E., Lacy, S.E., Possin, K.L., Jamieson, D.S., Tyrrell, A.M.: Evolving classifiers to recognize the movement characteristics of Parkinson's disease patients. IEEE Trans. Evol. Comput. **18**(4), 559–576 (2014)
2. Xue, B., Zhang, M., Browne, W.N., Yao, X.: A survey on evolutionary computation approaches to feature selection. IEEE Trans. Evol. Comput. **20**(4), 606–626 (2016)
3. Nguyen, H.B., Xue, B., Andreae, P.: Mutual information for feature selection: estimation or counting? Evol. Intel. **9**(3), 95–110 (2016)
4. Neshatian, K., Zhang, M., Andreae, P.: A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. IEEE Trans. Evol. Comput. **16**(5), 645–661 (2012)
5. Tran, B., Xue, B., Zhang, M.: Genetic programming for feature construction and selection in classification on high-dimensional data. Memet. Comput. **8**(1), 3–15 (2015)
6. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. Mach. Learn. **3**(2), 95–99 (1988)
7. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Subprograms. MIT Press, Cambridge (1994)
8. Niu, G.: Feature selection optimization. Data-Driven Technology for Engineering Systems Health Management, pp. 139–171. Springer, Singapore (2017). doi:10.1007/978-981-10-2032-2_6
9. De Paula, L.C., Soares, A.S., de Lima, T.W., Coelho, C.J.: Feature selection using genetic algorithm: an analysis of the bias-property for one-point crossover. In: GECCO 2016 Companion, pp. 1461–1462 (2016)
10. Stefano, C.D., Fontanella, F., Marrocco, C., di Freca, A.S.: A GA-based feature selection approach with an application to handwritten character recognition. Pattern Recogn. Lett. **35**, 130–141 (2014). Frontiers in Handwriting Processing
11. Li, N.J., Wang, W.J., Hsu, C.C.J.: Hybrid particle swarm optimization incorporating fuzzy reasoning and weighted particle. Neurocomputing **167**, 488–501 (2015)
12. Mistry, K., Zhang, L., Neoh, S.C., Lim, C.P., Fielding, B.: A micro-GA embedded PSO feature selection approach to intelligent facial emotion recognition. IEEE Trans. Cybern. **47**(6), 1496–1509 (2017)
13. Bharti, K.K., Singh, P.K.: Opposition chaotic fitness mutation based adaptive inertia weight BPSO for feature selection in text clustering. Appl. Soft Comput. **43**, 20–34 (2016)
14. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms. Appl. Soft Comput. **18**, 261–276 (2014)

15. Vieira, S.M., Mendonça, L.F., Farinha, G.J., Sousa, J.M.: Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. Appl. Soft Comput. **13**(8), 3494–3504 (2013)
16. Nguyen, H.B., Xue, B., Liu, I., Andreae, P., Zhang, M.: Gaussian transformation based representation in particle swarm optimisation for feature selection. In: Mora, A.M., Squillero, G. (eds.) EvoApplications 2015. LNCS, vol. 9028, pp. 541–553. Springer, Cham (2015). doi:10.1007/978-3-319-16549-3_44
17. Nguyen, H.B., Xue, B., Liu, I., Zhang, M.: PSO and statistical clustering for feature selection: a new representation. In: Dick, G., et al. (eds.) SEAL 2014. LNCS, vol. 8886, pp. 569–581. Springer, Cham (2014). doi:10.1007/978-3-319-13563-2_48
18. Nguyen, H.B., Xue, B., Andreae, P.: Surrogate-model based particle swarm optimisation with local search for feature selection in classification. In: Squillero, G., Sim, K. (eds.) EvoApplications 2017. LNCS, vol. 10199, pp. 487–505. Springer, Cham (2017). doi:10.1007/978-3-319-55849-3_32
19. Guo, H., Nandi, A.K.: Breast cancer diagnosis using genetic programming generated feature. Pattern Recogn. **39**(5), 980–987 (2006)
20. Lichman, M.: UCI machine learning repository (2013)
21. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: evolutionary algorithms made easy. J. Mach. Learn. Res. **13**, 2171–2175 (2012)