A Cooperative Coevolution-Based Pittsburgh Learning Classifier System Embedded with Memetic Feature Selection

Yun Wen, Hua Xu

***Abstract*—Given that real-world complex classification tasks always have irrelevant or noisy features which degrade both prediction accuracy and computational efficiency, feature selection is an effective data reduction technique showing promising performance. This paper presents a cooperative coevolution framework to make the feature selection process embedded into the classification model construction within the genetic-based machine learning paradigm. The proposed approach utilizes the divide-and-conquer strategy to manage two populations in parallel, corresponding to the selected feature subsets and the rule sets of classifier respectively, in which a memetic feature selection algorithm is adopted to evolve the feature subset population while a Pittsburgh-style learning classifier system is used to carry out the classifier evolution. These two coevolving populations cooperate with each other regarding the fitness evaluation and the final solution is obtained via collaborations between the best individuals from each population. Empirical results on several benchmark data sets chosen from the UCI repository, together with a non-parametric statistical test, validate that the proposed approach is able to deliver classifiers of better prediction accuracy and higher stability with fewer selected features, compared with the original learning classifier system. In addition, the incorporated feature selection process is shown to help improve the computational efficiency as well.**

I. Introduction

With the rapid development of computer and database technologies, data sets with hundreds and even thousands of features are now ubiquitous in pattern recognition, data mining, and machine learning [1]. Since the real-world data of high dimensionality always contain redundance and noise, the presence of irrelevant features degrades not only the computational efficiency but also the prediction accuracy, which poses traditional machine learning techniques with the problem of dimensionality curse. As one of the most important data reduction techniques, feature selection addresses this problem by removing the irrelevant, redundant, or noisy data while selecting the most discriminative features to drive the learning process. In addition to defying the curse of dimensionality to improve the prediction performance, feature selection techniques also help reduce the computational cost and provide better understandings of the underlying structure of data sets [2]. Though feature selection can be applied to both supervised (classification) and unsupervised (clustering) learning, we focus here on the classification problems.

In the context of classification, feature selection techniques generally fall into three categories depending on how they combine the feature selection search with the classification model construction: filter methods, wrapper methods, and embedded methods [2]. Nevertheless, no matter which category it belongs to, feature selection consists in identifying the optimal feature subset retaining maximally useful information of the target concept without sacrificing the prediction accuracy of the learned classifier, which is in essence a combinatorial optimization problem. Therefore, a bunch of well-known metaheuristics, including Tabu Search (TS) [3], Simulated Annealing (SA) [4], and Genetic Algorithm (GA) [5], have been adopted to manage the search for optimal feature subset, among which GA receives the most immense popularity [5], [6]. As a general-purpose global optimization technique, GA has shown promising performance for a wide range of complex optimization problems including feature selection. However, with the feature size increasing, GA may take a long time to locate the local optimum in the region of convergence due to its inherent lack of local guidance. In recent years, there exists a trend of incorporating GA with some local search operators that are capable of fine-tuning individuals in the population, which is expected to obtain better balance between global exploration and local exploitation in the evolutionary process. This hybrid form of evolutionary computation (EC) is nowadays referred to as Memetic Algorithm (MA) [7]. Compared with its conventional counterparts, MA is claimed to improve not only the solution quality but also the computational efficiency.

On the other hand, over the past few decades, numerous approaches belonging to different paradigms of machine learning have been developed to tackle the classification problem, among which the Learning Classifier System (LCS), also named Genetic-based Machine Learning (GAML) systems, is nowadays experiencing rising popularity [8], [9]. LCS, firstly developed by Holland [10], typically maintains a population of condition-predication rules, namely *classifiers*, to represent the current description of the target concept. Traditionally, LCSs can be broadly divided into two main classes, Michigan-style and Pittsburgh-style, with respect to the meaning of every individual in the population [8]. Each individual in Michigan LCSs only acts in some parts of the problem domain while the whole population is made up of a single rule set representing a complete solution to the problem at hand. In contrast, each individual in Pittsburgh LCSs is just a rule set consisting of a variable number of rules, which encodes an entire candidate solution. To sum up, the Pittsburgh LCS is much simpler than the Michigan-style one, for the former is indeed an

optimization tool applied to learning tasks that uses an EC technique as its main driving force while the latter involves the cooperation of several learning modules, one of them being some EC method [11].

Based on the aforementioned fact that there exists a rising trend of applying EC techniques to deal with feature selection problems and classification problems separately, it is natural for us to consider the possibility of integrating both feature selection and classification within the genetic-based machine learning paradigm. In this paper, a Cooperative Coevolution-based Pittsburgh Learning Classifier System embedded with Memetic Feature Selection (CoCoLCS_MFS) is proposed, in which the feature selection search is built into the classifier evolution process of LCS. The proposed CoCoLCS_MFS approach concurrently manages two subpopulations, corresponding to the classifiers and the feature subsets respectively, in attempt to complete the classification model construction and the feature selection simultaneously. More specifically, the proposed algorithm adopts a memetic feature selection algorithm to manage the feature subset search process while taking advantage of GAssist [11], a Pittsburgh-style LCS, to conduct the classifier evolution. Although these two populations are evolved separately with different genetic mechanisms, they cooperate with each other in terms of fitness evaluation, in which one population is evaluated via collaborations with an elitist individual selected from the other population. The final solution to the target problem is delivered by merging the best individuals in both populations. Empirical study conducted on some benchmark data sets from the UCI repository [12] shows that our proposed CoCoLCS_MFS approach outperforms the original GAssist in terms of both classification accuracy and computational efficiency.

The reminder of the paper is organized as follows. Section II formulates the studied problem while Section III briefly covers some background materials. The proposed approach is detailed in Section IV. The experimental results are presented in Section V. Finally, the paper is summarized in Section VI.

II. Feature Selection Problem for Classification

Feature selection for classification can be viewed as an optimization problem to search for the optimal feature subset on which the classifier is constructed to deliver the best prediction performance. Note that feature selection only chooses a feature subset from existing features, rather than construct new ones in feature extraction or construction [13].

Given a training data set $\mathcal{D}$ with d labeled instances $\{(X_1, Y_1), (X_2, Y_2), ..., (X_d, Y_d)\}$, each of which is described as an assignment of values $X_i = (x_{i1}, x_{i2}, ..., x_{iN})$ to the full set of N features $F = (F_1, F_2, ..., F_N)$ and a class label Y_i, the task of classification is to induce a classifier $\mathcal{C}$ that as accurately as possible predicts the labels of novel instances in test data set $\mathcal{T}$ based on their feature values. The feature selection problem involves selecting a minimal subset of M features $S = (S_1, S_2, ..., S_M)$ from the original feature set $F = (F_1, F_2, ..., F_N)$, where $M \leq N$ and $S \subseteq F$, so that the feature dimensionality is optimally reduced while the prediction accuracy is maximized.

$$Acc(\mathcal{C}, S_{opt}) = \max_{|Z|=i, Z \subseteq F} Acc(\mathcal{C}, Z), i = 1, 2, ..., N \quad (1)$$

An exhaustive approach to this problem would require examining all $\sum_{i=1}^{N} \binom{N}{i} = 2^N - 1$ possible subsets of the original feature set F. Since feature selection for classification is a NP-hard problem [14], the search for the optimal solution is computationally intractable even with a medium feature size. Hence, it is appropriate to utilize metaheuristics to find suboptimal solutions within a reasonable amount of time.

III. Related Works

A. Feature Selection Approaches

With regard to the approach in which the feature selection search interacts with the classifier construction, existing algorithms can be classified into three categories: filter methods, wrapper methods and embedded methods [2]. In general, filter techniques merely assess the relevance of features based on the intrinsic properties of data. In most cases, a feature relevance score is calculated for each feature and low-scoring features are removed in one pass, which is independent of any classification algorithm. Hence, the scoring criterion plays an important role in the design of filter methods. Most of existing filters employ univariate criteria to evaluate each feature separately [15], thereby ignoring feature dependencies, which may lead to worse classification performance. In order to overcome this problem, some multivariate filter techniques have been introduced [16]. Wrapper methods, on the other hand, search in the space of all possible feature subsets and various feature subsets are generated and evaluated. The evaluation of a given feature subset involves training and testing a classification model to calculate the prediction accuracy, which renders the selected feature subset tailored to a specific classification algorithm. To sum up, filter methods are computationally much more efficient, but usually do not perform as well as wrapper methods. In recent years, there are several reported works concerning with the hybridization of both filter and wrapper approaches [17], [18], which is expected to balance between solution quality and computational efficiency. Typically, MA is often used to organize this hybridization, in which the EA component is utilized to manage the feature subset search of wrapper while the local search operator is guided by the scoring criterion used in filter.

Besides wrappers and filters, the embedded methods are a third category of feature selection techniques, in which the search for an optimal feature subset is built into the classifier construction, thus specific to a given learning algorithm [2]. Compared with the wrapper counterparts, embedded techniques have the advantage that they avoid retraining a classifier from scratch for every feature subset investigated, thus achieving substantial savings of computational cost.

B. Coevolutionary Algorithm

In the context of evolutionary computation, the coevolutionary algorithm (CEA) is a recently devised paradigm in which

two or more populations interact with each other regarding fitness assignment. The coexistence of multiple interacting populations allows CEAs to employ a divide-and-conquer strategy to split the target problem into different parts, each of which is solved by a separate population. Thus, a full candidate solution is formed by integrating individuals chosen from each interacting subpopulation. With respect to the type of interaction, existing CEAs can be categoried into two classes, competitive CEA (ComCEA) and cooperative CEA (CoCEA). In the ComCEA, individuals are rewarded at the expense of those with whom they interact, which is analogous to the notion of an arms race [19]. Whereas in the CoCEA, interacting populations cooperate with each other, which is demonstrated by the fact that an increase in the collaborative fitness value are shared among individuals of all the populations [20]. Generally, the ComCEA is significantly more complicated than the CoCEA. Some theoretical works have shown that coevolution is able to beat the well-known No-Free-Lunch (NFL) [21] barrier subject to most of optimization techniques [22]. It means that CEAs are possible to perform better than other EAs when averaged over all functions, if the interaction between several coevolving populations is managed in a suitable way [23].

One important design point in CoCEA is how to select the members of each population (collaborators) that will be used to evaluate the fitness function. One exhaustive approach is to evaluate an individual against every single collaborator in the other population [24], which guarantees the performance but significantly increases the computational cost in terms of fitness evaluation. To improve the efficiency, there are other options, such as the use of just a random individual or the best individual from the previous generation [25], the latter of which is adopted by the proposed approach.

Some most recent works utilize CoCEAs to solve feature selection tasks [26], [27], in which the CoCEA serves as a wrapper framework for just feature selection while the classification task is left to some separate classifier, such as the nearest neighbor rule in [27]. However, in our proposed approach, the CoCEA is used to carry out both feature selection and classifier construction simultaneously, which is more similar to a embedded method. Our preliminary empirical study validates that the integration of both feature selection and classifier evolution helps improve the prediction accuracy without introducing too much extra running cost, but even increasing the computational efficiency.

IV. The Proposed Cooperative Coevolution-based LCS with Memetic Feature Selection

In this section, various characteristics of the proposed CoCoLCS_MFS are presented, including the cooperative coevolutionary framework, the memetic feature selection approach, and the classifier construction in LCS, as well as an analysis of the computational complexity.

A. Cooperative Coevolution Framework

As mentioned in the Introduction section, the proposed CoCoLCS_MFS approach concurrently manages two separate

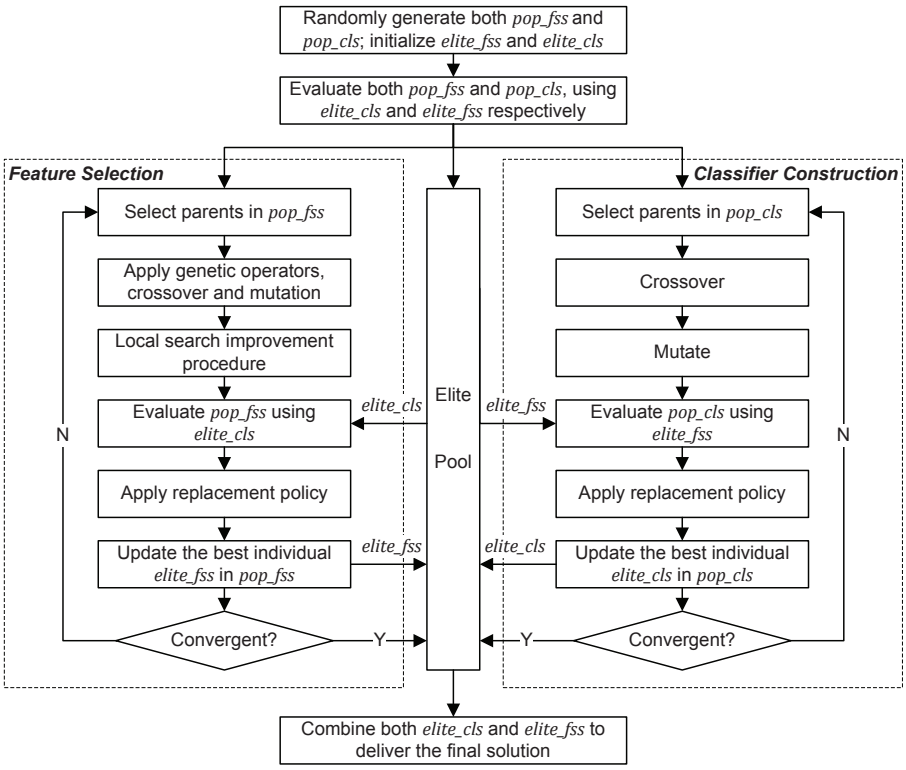

Fig. 1. Flow chart of the proposed CoCoLCS_MFS approach.

populations, corresponding to the feature subsets and the classifiers respectively.

- The feature subset population pop_fss performs a feature selection process, to which a memetic algorithm is applied (to be detailed in Section IV-B).
- The classifier population pop_cls conducts an evolution process of classifier construction, just as in the Pittsburgh-style LCS (to be detailed in Section IV-C).

From now on, these two populations will be referred to as FSS population and CLS population respectively. In the context of feature selection, the prediction accuracy of a classifier can be determined only when a specific feature subset is selected. It means that the two coevolving populations interact with each other in terms of fitness calculation, since the prediction accuracy is a part of the fitness function used in each population.

The overall flow of the proposed CoCoLCS_MFS approach is depicted in Fig. 1. After randomly initializing both pop_fss and pop_cls, these two populations are evolved separately, to which different genetic mechanisms are applied. However, fitness of individuals from each population are calculated in collaboration with an elite individual selected from the other population. In the end of each generation, the best individual of each population ($elite\_fss$ and $elite\_cls$) is updated, which serves as the collaborator for individuals of the other population in the next iteration. Once upon convergence, the proposed CoCoLCS_MFS delivers the final solution by integrating the elites from both populations. Note that the initial elite of the FSS population $elite\_fss$ is a chromosome with all bits set to 1, based on which the CLS population is initially evaluated and the initial $elite\_cls$ is selected. Moreover, the quality of a combination of $elite\_fss$ and $elite\_cls$ relies solely on their obtained classification accuracy on the training data set, which acts as the evaluating indicator used in the updating phase. The next two sections are devoted to presenting the proposed memetic feature selection approach and the adopted LCS for classifier evolution respectively.

B. Memetic Feature Selection

Traditional GAs are proved to be good at shuffling the search space but fail to effectively exploit the regions with promising solutions. As a result, the memetic algorithm framework is devised to overcome this weakness, in which a local search process is incorporated to enable the self-refinement of individuals. The proposed CoCoLCS_MFS approach adopts a MA to manage the search process for the optimal feature subset, the pseudocode of which is shown in Fig. 2.

The next four subsections will detail the memetic feature selection approach from different aspects, including the chromosome representation, the fitness calculation, the genetic operators, and the incorporated local search procedure.

1) Encoding Mechanism and Population Initialization: In this memetic feature selection framework, we adopt a binary chromosome representation to encode a candidate feature subset, as shown in Fig. 3. The length of the binary chromosome is equivalent to the total number of features N, with each

1. $pop\_fss \leftarrow$ RandomIntializeGA();
2. InitialEvaluate(pop_fss);
3. **repeat**
4. $parent \leftarrow$ Select(pop_fss);
5. $elite\_cls \leftarrow$ BestIndividual(pop_cls);
6. $offspring \leftarrow$ Crossover($parent$);
7. $offspring \leftarrow$ Mutate($offspring$);
8. $offspring \leftarrow$ LocalSearch($offspring$);
9. Evaluate($offspring, elite\_cls$);
10. $pop\_fss \leftarrow$ Replace($pop\_fss, offspring$);
11. **until** the termination condition is met
12. Output the best solution ever found $elite\_fss$.

Fig. 2. Algorithmic flow of the proposed memetic feature selection.

bit encoding a single feature. A bit set to 1 (0) indicates the corresponding feature is selected (excluded).

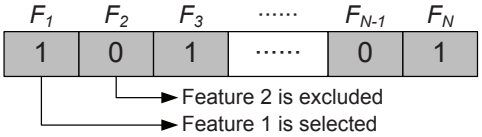

Fig. 3. Encoding chromosome of the FSS population.

The FSS population of MA is initialized with $popsize\_fss$ randomly generated individuals, in which each bit is randomly set to 1 or 0 with equal probability, 0.5.

2) Fitness Function: In general, the objective of feature selection is twofold, minimizing the number of selected features without loss of prediction accuracy. The proposed MA uses a fitness function composed of two components: the prediction accuracy of the classifier given a selected feature subset, $Acc(cls, fss)$, and the reduction rate of the given feature subset over all the features, $Red(fss)$. As a result, a classifier from the CLS population is needed to compute the $Acc(cls, fss)$ subcomponent of the fitness function. In order to avoid the exhaustive pairing with every individual of the CLS population, the best classifier in the previous generation $elite\_cls$ is recorded to evaluate all the candidate feature subsets in the concurrent iteration. The $Acc(cls, fss)$ is defined as follows:

$$Acc(cls, fss) = \frac{\#Instances\ correctly\ classified}{d} \quad (2)$$

where d is the total number of instances in the training data set. On the other hand, for a given chromosome fss, the $Red(fss)$ value can be directly calculated as follows:

$$Red(fss) = 1.0 - \frac{\#Feature\ selected}{N} \quad (3)$$

where $\#Feature\ selected$ is the number of genes set to 1 in the given chromosome while N denotes the total number of features.

Finally, the fitness of a given feature subset can be obtained by a linear combination of both $Acc(cls, fss)$ and $Red(fss)$,

employing a real-valued weighting factor α:

$$Fitness(fss) = \alpha * Acc(cls, fss) + (1-\alpha) * Red(fss) \quad (4)$$

3) Genetic Operator: The proposed MA adopts a generational GA to explore the search space of all possible feature subsets. All the genetic operators used are described as follows.

- *A Binary Tournament Selection*. Rather than value-based selection operators such as Roulette Wheel Selection, this rank-based selection alleviates the dominant effect of some individuals with especially high fitness value, which to some extent enhance the robustness of GA.
- *A Random-One-Point Crossover*. The candidates selected from parent generation are grouped into pairs, each of which will undergo the crossover operator with a certain probability, $crossover\_rate$. A crossover point is randomly chosen for both parents. The segments to the right of the crossover point are exchanged to form two offspring. Moreover, parents that do not crossover may still undergo mutation.
- *A Bit-wise Flipping Mutation*. The mutation rate $mutation\_rate$ gives the probability that a bit in the given chromosome will be reversed, which means that the corresponding feature is changed from selected to excluded or conversely. As a result, the expected number of mutations per chromosome is equal to the mutation rate multiplied by the length of an chromosome, N.
- *A Elitism Replacement Policy*. The previous generation is totally replaced by the reproduced offspring, with the exception that the best chromosome is directly copied to new generation. This elitism policy encourages a more explorative search without the risk of losing the best found solution.

4) Local Search Procedure: Typically, local search improvement procedures are adopted by MA to enhance the local exploitation ability of GA, in which domain-specific knowledge is used to fine-tune a given individual. In our proposed approach, a local search procedure similar to the one used in [18] is adopted, in which ReliefF [15] is utilized as the filter ranking method that guides the *Add* and *Del* local search operators [18]. However, rather than deterministically choosing the first w elitists among the population to undergo the local search procedure, our proposed approach takes advantage of a randomized technique that samples the population with a relatively low probability, $sampling\_rate$, while setting the local search length l exactly to 1. Since elites in the population usually denote local optimum, especially when the evolutionary process is close to convergence, applying local search procedure to these individuals hardly improves their quality, which results in a waste of computation capacity. The proposed local search procedure is outlined in Fig. 4. For more details about both Add and Del operators, refer to [18].

```
1. for each chromosome fss in pop_fss do
2.   if rand() < sampling_rate then
3.     fss' ← Add(fss);
4.     fss' ← Del(fss');
5.     if fitness(fss') > fitness(fss) then
6.       fss ← fss';
7.     end if
8.   end if
9. end for
```

Fig. 4. The proposed local search improvement procedure.

C. Classifier Evolution in LCS

Generally, the proposed CoCoLCS_MFS takes advantage of a Pittsburgh-style LCS, namely GAssist [11], to evolve the CLS population. GAssist adopts a near-standard generational GA as its optimization tool, in which each individual consists of an ordered, variable-length rule set that by itself is a complete solution to the classification problem at hand. A fitness function based on the Minimum Description Length (MDL) principle is used, which helps balance the complexity and accuracy of the rule set. Moreover, in order to help control the bloat effect, an explicit default rule mechanism, as well as a rule deletion operator, is adopted, which is believed to deliver more compact and accurate solutions. For more implementation details about GAssist, refer to [11].

The main difference lies in that the classification accuracy calculation in GAssist is adapted to accommodate the situation of feature selection. That is to say, the accuracy of each classifier cls is calculated based on a given feature subset fss, which has been defined in Eq. (2). More specifically, when a rule is applied to match a specific instance, only those attribute values corresponding to the selected features in the given feature subset are concerned. For the same reason explained in Section IV-B2, the best feature subset in the previous generation of the FSS population $elite\_fss$ is recorded to evaluate all the candidate classifiers in the concurrent generation of the CLS population.

D. Computational Complexity

This section presents a rough analysis of the computational complexity of the proposed CoCoLCS_MFS approach. Since the filter-based feature ranking computation is a one-time offline cost that is negligible compared with the fitness evaluation cost in every iteration, we define the computational cost of a single fitness evaluation as the basic unit of computational cost in our analysis.

In general, the computational cost of the proposed CoCoLCS_MFS is composed of two parts, one for the classifier evolution in LCS while the other for the feature selection in MFS. The computational complexity of LCS can be derived as $O(ps\_cls * g)$, where ps_cls is the size of the CLS population and g is the number of generations. On the other hand, the computational complexity of MFS can be defined as $O((1 + sr) * ps\_fss * g)$, where ps_fss is the size of the FSS population, sr denotes the sampling rate used in the local search procedure, the factor 1 stands for the generational GA while the factor sr represents the extra cost introduced by the local search procedure. To sum up, the overall computational

complexity of the proposed CoCoLCS_MFS is $O(((1+sr)*ps\_fss+ps\_cls)*g)$, which is approximate to $O(ps\_cls*g)$ given that $ps\_fss \ll ps\_cls$ and $sr \ll 1$. Moreover, in view of the reduced feature subset after feature selection, the computational complexity of classifier evaluation, a main cost of fitness calculation, will be significantly reduced. As a result, the incorporated feature selection process is expected to improve the computational efficiency, rather than increase the computational cost, of the proposed approach.

V. Experimental Study

In this section, the comparative study of the proposed CoCoLCS_MFS with the original GAssist is detailed. First, we present the benchmark data sets used to evaluate the classification performance, as well as the parameter settings of both algorithms. Next, experimental results in terms of several comparison metrics are described, which indicate the improved performance of our proposed approach. Finally, the Wilcoxon paired signed ranks test, a non-parametric pairwise statistical test, is performed to validate the existence of significant difference between the performance of both algorithms.

A. Benchmarks and Parameter Settings

In order to evaluate the performance of our method, some real-world binary classification problems obtained from the UCI Machine Learning Repository [12] are treated as the benchmark data sets. Main characteristics of these data sets are shown in Table I, including the total number of instances, features, and classes. Furthermore, all the data sets considered are partitioned using the five-fold cross validation procedure, and the numerical attributes are discretized via the Chi2 algorithm [28], in which the significant level α is set to 0.01.

TABLE I
Benchmark data sets used in the experimental study

Name	#Instance	#Feature	#Classes
diabetes	768	8	2
german	1000	24	2
onehr	1845	72	2
sonar	208	60	2
wdbc	569	32	2
mushroom	8124	22	2
parkinsons	197	23	2
spambase	4601	57	2
SPECT	267	22	2
housevotes	435	16	2
kr-kp	3196	36	2

The following parameter settings shown in Table II are empirically determined to obtain good performance in the original GAssist, most of which are assigned with the values recommended in [11]. Moreover, the same parameter settings are also used in the LCS component of the proposed CoCoLCS_MFS. On the other hand, the parameter settings used in the proposed memetic feature selection approach are summarized in Table III. Unless otherwise specified, we use the following values in our experiments.

TABLE II
Parameter settings used in the GAssist

Parameter	Value
Size of population	200
Prob. of crossover	0.6
Prob. of mutation	0.04
# Strata	2
Prob. of value 1 in initialization	0.5
Selection strategy	truncation selection
Prob. in selection	0.5
Replacement strategy	elitist replacement
Prob. in replacement	0.5
# Rules in initial individual	20
Terminal generation	4000
Generations for rule deletion	50

TABLE III
Parameter settings used in the proposed Memetic Feature Selection

Parameter	Value
Size of population	50
Crossover rate	0.8
Mutation rate	0.1
Sampling rate	0.1
Prob. of value 1 in initialization	0.5
Terminal generation	4000

B. Comparison Results

This section presents the comparison results obtained in the empirical study, employing 10*5-fold cross validation scheme (50 trials per data set). Our proposed approach is compared with the original GAssist in terms of several metrics, including prediction accuracy, feature reduction rate, and computational cost. All the algorithms are implemented in Java while all the experiments are run on a desktop with Intel Core 2 3.00 GHz processor and 2 GB RAM.

1) Classification Accuracy: First, we compare the two approaches in terms of prediction accuracy, which is the main optimization objective of both classifier evolution and feature selection. The classification accuracy obtained on the testing data set, rather than the accuracy values calculated in the training phase, is used as the comparison metric. The average accuracy rates, together with the standard deviation values, are shown in Table IV, in which the best results are shown in **bold** font. The results indicate that our proposed CoCoLCS_MFS outperforms the original GAssist for a majority of all the benchmark problems (9 out of 11 data sets). In the remaining two data sets, the proposed approach also delivers competitive performance. Furthermore, the performance of our proposed CoCoLCS_MFS shows less fluctuations within each data set, which demonstrates that the integration of feature selection helps improve not only the prediction accuracy but also the stability of the classification model.

2) Reduction Rate: Second, we investigate the feature reduction rate achieved by the proposed approach, which is one of the dual optimization objectives of feature selection. The results are summarized in Table V. It appears that the feature selection process built into our proposed CoCoLCS_MFS

consistently reduces more than half of the original features with all the benchmark data sets (all the reduction rates are greater than 0.5). Given the reduced feature subset after feature selection, the computational cost of classifier evaluation will be significantly decreased (to be discussed in the next section).

TABLE IV
ACCURACY COMPARISON ON BENCHMARK DATA SETS

Data Set	GAssist		CoCoLCS_MFS	
	Ave.	Std.	Ave.	Std.
diabetes	0.6490	0.0088	**0.7239**	**0.0007**
german	0.6697	0.0257	**0.7078**	**0.0105**
onehr	0.9609	0.0116	**0.9620**	**0.0048**
sonar	0.6960	0.0618	**0.7278**	**0.0550**
wdbc	0.9114	0.0225	**0.9413**	**0.0125**
mushroom	**0.9967**	0.0043	0.9915	**0.0029**
parkinsons	0.8923	0.0501	**0.8949**	**0.0135**
spambase	0.9151	0.0148	**0.9236**	**0.0056**
SPECT	0.6745	0.0288	**0.7519**	**0.0137**
housevotes	0.9630	0.0155	**0.9690**	**0.0036**
kr-kp	**0.9684**	0.0032	0.9654	**0.0009**

TABLE V
FEATURE REDUCTION RATES ON BENCHMARK DATA SETS

Data Set	#Original features	#Features After FS		Average reduction rate
		Ave.	Std.	
diabetes	8	1.00	0	0.875
german	24	6.44	1.53	0.732
onehr	72	24.25	1.91	0.663
sonar	60	26.33	1.12	0.561
wdbc	32	12.44	1.67	0.611
mushroom	22	7.10	1.73	0.677
parkinsons	23	7.45	0.51	0.676
spambase	57	27.60	1.78	0.516
SPECT	22	5.75	0.97	0.739
housevotes	16	2.10	0.32	0.869
kr-kp	36	14.00	2.62	0.611

3) Computational Cost: Lastly, the computational cost of both algorithms is compared. The average time elapsed (in seconds) of both algorithm shown in Table VI indicate that the proposed significantly outperforms the original GAssist in terms of computational cost, which is in accordance with the preliminary analysis presented in Section IV-D.

C. Significance Tests

Furthermore, in order to find significant differences among the results obtained by the proposed approach and the original GAssist, statistical analysis is necessary. Since the obtained results may present neither normal distribution nor homogeneity of variance, we consider the use of non-parametric tests according to the recommendations made in [29]. Specifically, the Wilcoxon signed-rank test, a pairwise non-parametric statistical test, is employed to check whether there are significant differences in the classification performance and the computational cost of both algorithms. The results are presented in Table VII, and a level of significance $\alpha = 0.05$ is used in all the tests. Since the Wilcoxon signed-rank test rejects all the hypotheses, it could be safely concluded that the proposed CoCoLCS_MFS approach is statistically better than the original GAssist regarding both classification performance and computational efficiency, with a significant level of 0.05.

TABLE VI
COMPUTATIONAL COST COMPARISON ON BENCHMARK DATA SETS (IN SECONDS)

Data Set	GAssist		CoCoLCS_MFS	
	Ave.	Std.	Ave.	Std.
diabetes	91.41	9.68	24.87	0.50
german	337.11	51.28	96.60	9.96
onehr	1296.05	719.75	469.17	63.33
sonar	137.67	14.09	114.69	9.04
wdbc	168.96	23.18	122.46	24.96
mushroom	3772.23	454.81	826.66	332.29
parkinsons	71.33	7.71	63.16	5.84
spambase	4569.21	436.674	1742.89	158.48
SPECT	193.67	39.36	42.93	7.41
housevotes	183.02	15.46	30.64	2.44
kr-kp	2649.16	510.72	738.05	150.76

TABLE VII
SIGNIFICANCE TESTS ON CLASSIFICATION ACCURACY AND COMPUTATIONAL COST WITH ALL THE BENCHMARK DATA SETS. THE p-VALUES ARE FROM ONE SIDED WILCOXON SIGNED-RANK TEST. THE LEVEL OF SIGNIFICANCE α IS SET TO 0.05

GAssist vs CoCoLCS_MFS	R^+	R^-	p-value	Hypothesis
Accuracy	7	59	0.011	Rejected
Time elapsed	66	0	0.002	Rejected

VI. CONCLUSIONS AND FUTURE WORKS

This paper presents a novel hybrid learning algorithm, namely CoCoLCS_MFS, in which a memetic feature selection search is embedded into the classifier evolution process of GAssist, a Pittsburgh-style LCS, by means of a cooperative coevolution framework. In the proposed approach, the selected feature subsets and the rule sets of classifiers in GAssist are encoded and evolved by two separate populations. These two coevolving populations cooperate with each other regarding fitness evaluation. Experimental results on several benchmark data sets from the UCI repository illustrate that the proposed CoCoLCS_MFS is capable of delivering solutions with better accuracy and higher stability, compared with the original GAssist. Moreover, the incorporated feature selection helps improve the computational efficiency as well by reducing the number of features involved in the evaluation of classifiers.

In future research, we plan to take a closer investigation into the empirical characteristics of the proposed CoCoLCS_MFS, such as the impact of the weighting factor α to both the structure and the performance of the final solution, and the influence of the relative size of both populations on the algorithm convergence, intending to achieve the self-adaptive parametrization. Another planned future research is to incorporate the cooperative coevolution framework into the Michigan-style LCS.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (Grant No. 60875073), National Key Technology R&D Program of China (Grant No. 2009BAG12A08), and National S&T Major Projects of China (Grant No. 2009ZX02001).

REFERENCES

[1] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, pp. 245–271, December 1997.

[2] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, March 2003.

[3] M. Tahir, A. Bouridane, F. Kurugollu, and A. Amira, "Feature selection using tabu search for improving the classification rate prostate needle biopsies," in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 2, 2004, pp. 335–338.

[4] S.-W. Lin, T.-Y. Tseng, S.-Y. Chou, and S.-C. Chen, "A simulated-annealing-based approach for simultaneous parameter optimization and feature selection of back-propagation networks," *Expert System with Application*, vol. 34, pp. 1491–1499, February 2008.

[5] K. Chan, H. Zhu, C. Lau, and S. Ling, "Gene signature selection for cancer prediction using an integrated approach of genetic algorithm and support vector machine," in *2008 IEEE Congress on Evolutionary Computation*, 2008, pp. 217–224.

[6] M. Martin-Bautista and M.-A. Vila, "A survey of genetic feature selection in mining issues," in *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, vol. 2, 1999, pp. 1314–1321.

[7] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Springer New York, 2003, vol. 57, pp. 105–144.

[8] R. J. Urbanowicz and J. H. Moore, "Learning classifier systems: a complete introduction, review, and roadmap," *Journal of Artificial Evolution and Applications*, vol. 2009, pp. 1–25, January 2009.

[9] A. Orriols-Puig, J. Casillas, and E. Bernad-Mansilla, "Genetic-based machine learning systems are competitive for pattern recognition," *Evolutionary Intelligence*, vol. 1, pp. 209–232, 2008.

[10] J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithms," *ACM SIGART Bulletin*, pp. 49–49, June 1977.

[11] J. Bacardit, "Pittsburgh genetics-based machine learning in the data mining era: representations, generalization, and run-time," Ph.D. dissertation, Ramon Llull University, 2004.

[12] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[13] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, pp. 273–324, December 1997.

[14] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, pp. 237–260, December 1998.

[15] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the 9th International Workshop on Machine Learning*, San Francisco, CA, USA, 1992, pp. 249–256.

[16] D. Koller and M. Sahami, "Toward optimal feature selection," Stanford InfoLab, Technical Report 1996-77, February 1996.

[17] N. Xiong and P. Funk, "Combined feature selection and similarity modelling in case-based reasoning using hierarchical memetic algorithm," in *2010 IEEE Congress on Evolutionary Computation*, 2010, pp. 1–6.

[18] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper-filter feature selection algorithm using a memetic framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 70 –76, 2007.

[19] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, pp. 1–29, March 1997.

[20] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, pp. 1–29, March 2000.

[21] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[22] ——, "Coevolutionary free lunches," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 721–735, 2005.

[23] T. C. Service and D. R. Tauritz, "A No-Free-Lunch framework for coevolution," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2008, pp. 371–378.

[24] L. Panait, R. P. Wiegand, and S. Luke, "Improving coevolutionary search for optimal multiagent behaviors," in *Proceedings of the 18th International Joint Conference on Artificial intelligence*, San Francisco, CA, USA, 2003, pp. 653–658.

[25] L. Panait, S. Luke, and J. F. Harrison, "Archive-based cooperative coevolutionary algorithms," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2006, pp. 345–352.

[26] J. Derrac, S. García, and F. Herrera, "A First Study on the Use of Coevolutionary Algorithms for Instance and Feature Selection," in *Hybrid Artificial Intelligence Systems*, Corchado, E and Wu, X and Oja, E and Herrero, A and Baruque, B, Ed., vol. 5572, 2009, pp. 557–564.

[27] ——, "IFS-CoCo: Instance and feature selection based on cooperative coevolution with nearest neighbor rule," *Pattern Recognition*, vol. 43, pp. 2082–2105, June 2010.

[28] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proceedings of the 13th International Joint Conference on Articial Intelligence*, 1993, pp. 1022–1029.

[29] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, no. 7, pp. 1–30, 2006.