Journal of Mathematical Sciences: Advances and Applications
Volume 3, Number 2, 2009, Pages 199-227

HYBRID TS-IBPSO FOR FEATURE SELECTION

CHENG-HUEI YANG[1], LI-YEH CHUANG[2] and CHENG-HONG YANG[3]

[1] Department of Electronic Comm. Eng.
National Kaohsiung Marine University
Kaohsiung
Taiwan 811

[2] Department of Chemical Engineering
I-Shou University
Kaohsiung
Taiwan 80041

[3] Department of Electronic Engineering
National Kaohsiung University of Applied Sciences
Kaohsiung
Taiwan 80708

Abstract

The feature selection process can be considered a problem of global combinatorial optimization in machine learning, which reduces the number of features, removes irrelevant, noisy and redundant data, and results in acceptable classification accuracy. Feature selection is of great importance in the fields of data analysis and information retrieval processing, pattern classification, and data mining applications. In this paper, we propose to combine tabu search (TS) and improved binary particle swarm optimization (IBPSO) for feature selection. IBPSO serves as a local optimizer each time TS has been executed for a single generation. The *K*-nearest neighbor (*K*-NN) method with leave-one-out cross-validation (LOOCV)

2000 Mathematics Subject Classification: 90C27

Keywords and phrases: feature selection, tabu search, improved particle swarm optimization, *K* -nearest neighbor, leave-one-out cross-validation.

Received May 6, 2009

serves as an evaluator of the TS and IBPSO procedures. The proposed method is applied and compared to ten data classification problems taken from the literature and the results are compared to other feature selection methods. Experimental results show that the proposed method simplifies features effectively and either obtains higher classification accuracy or uses fewer features compared to other feature selection methods.

1. Introduction

For many pattern classification problems, a high number of selected features does not necessarily translate into high classification accuracy. In some cases, the performance of algorithms devoted to speed and predictive accuracy of the data characterization can even decrease because some features may be irrelevant or misleading, or due to spurious correlations. These factors can negatively affect the classification process during the learning stage. Feature selection can serve as a pre-processing tool of great importance before solving classification problems. The purpose of feature selection is to reduce the maximum number of irrelevant features, while at the same time maintaining acceptable classification accuracy. A good feature selection method can reduce the cost of feature measurement, as well as increase classifier efficiency and classification accuracy. Scientific interest in feature selection has been on the rise for several reasons. New applications handling vast amounts of data have been developed and are used in data analysis, data mining, information retrieval, (i.e., medical data processing, etc.) and pattern classification.

Feature selection algorithms can usually be categorized into two different classes based on the information to be extracted from the training data and the type of induction algorithm. They can be implemented independently from the performance of a specific learning algorithm. In order to optimize feature selection, a criterion function has to be either maximized or minimized. The effectiveness, i.e., the predictive accuracy of the feature selection model is directly dependent on the performance of the learning algorithms.

Several methods have previously been used to perform feature selection of training and testing data, for example, genetic algorithms

[25], branch and bound algorithms [24], [37], sequential search algorithms [27], mutual information [28], neural networks [2], tabu search [38], and hybrid genetic algorithms [25].

Several common methods of feature selection are described below. The sequential forward selection method (SFS) begins with a feature subset and sequentially adds or removes features until some termination criterion is met. SFS suffers from a nesting effect. Since the algorithms do not examine all possible feature subsets, there is no guarantee that they produce an optimal result [38]. Furthermore, features that have been discarded cannot be re-selected and the features selected cannot be removed later. The sequential backward selection method (SBS) is the backward analogue. The Plus-l take-away-r (PTA(l, r)) algorithm go forward l stages (by adding l features by SFS) and go backward r stages (by deleting r features by SBS) and repeat this process [19]. PTA was proposed to solve the nesting effect. However, there is no theoretical guidance to determine the appropriate values of l and r [38]. Sequential forward floating selection (SFFS) is the floating version of PTA(l, r). Unlike PTA(l, r), SFFS can backtrack an unlimited number of times as long as backtracking finds a better feature subset than the feature subset obtained so far at the same size [19]. However, SFFS suffers from getting trapped in local optimal solutions, when applied to large number features problems [38]. Simple genetic algorithms (SGA) have rather poor search ability in the near local optimum region and the phenomenon of premature convergence in SGAs seems to be difficult to avoid. Furthermore, parameters in SGA have to be carefully selected and tested [38].

Classification problems can be divided into two types: binary classification (two class labels) and multiclass classification (more than two class labels). Several methods have been applied to binary classification problems. Some examples are the discriminant analysis [12], [20], decision trees [29], the K-nearest neighbor method [4], [6], backpropagation neural networks [22], and support vector machines [33]. Other methods have been successfully applied to multiclass classification,

for example, the support vector machines: (1) one-versus-rest and one-versus-one [18], (2) DAGSVM [26], (3) the method by Weston and Watkins [13], [35], and (4) the method by Crammer and Singer [13], [5]. Generally speaking, multiclass classification problems are much more complex, and therefore more difficult to solve than binary ones.

In our study, we used a combination of tabu search (TS) and improved binary particle swarm optimization (IBPSO) [3] to implement feature selection. IBPSO was embedded in the TS procedure and served as a local optimizer after each generation. The *K*-nearest neighbor method (*K*-NN) with leave-one-out cross-validation (LOOCV) based on Euclidean distance calculations served as an evaluator of the TS and IBPSO for ten classification problems taken from the literature.

2. Methods

2.1. Tabu search

Tabu search (TS) is a meta-heuristic method and a general iterative procedure that was introduced by Glover [10], [11]. Tabu search directs a subordinate search method in order to avoid entrapment at a local optimum. It controls the operations of its embedded heuristic in a manner, which allows it to continue the search, rather than terminate it, upon reaching a local optimum. The objective is accomplished by strategically constraining the search process. Moves which take the solution in the next iteration, to points in the solution space previously visited are classified as forbidden (hence "tabu"). This restriction frees up the search process to explore new regions of the search space [10], [11].

TS starts with an initial solution and repeatedly constructs new solutions by searching the neighborhood of the current solution. The move evaluations correspond to the evaluation function produced by the moves. Tabu lists record the most recent moves; these moves should not be chosen for a certain number of subsequent iterations [10], [11]. In the beginning, the tabu list contains no entries. Initially, the tabu search makes a rough evaluation of the solution space. This process is known as "diversification". As candidate solutions are gradually identified, the search focuses increasingly on the production of local optimal solutions in

a process called "identification". The candidate move with the smallest fitness at each iteration will usually be considered admissible. However, when the candidate move with the smallest fitness already exists in the tabu list, the following situations must be considered. If the fitness value is greater than the aspiration value, then the next best move will be tested for admissibility. Otherwise, the move is still considered admissible. The process continues until an admissible candidate has been found. The tabu list is then updated to reflect the move and its associated aspiration value.

TS has successfully been applied to problems in a variety of areas, amongst them telecommunications [7], VLSI design [36], scheduling [1], pattern classification [32], optimization problems [23], and biomedical analysis [31], [34]. Further, details about TS mechanisms can be found in Glover [10], [11].

2.2. Improved particle swarm optimization

Particle swarm optimization (PSO) is a population-based stochastic optimization technique, which was developed by Kennedy and Eberhart in 1995 [15]. PSO simulates the social behavior of organisms, such as birds in a flock or fish in a school. This behavior can be described as an automatically and iteratively updated system. In PSO, each single candidate solution is represented by a particle in the search space. Each particle makes use of its own memory and knowledge gained by the swarm as a whole to find the best solution. All of the particles have fitness values, which are evaluated by a fitness function to be optimized. During movement, each particle adjusts its position by changing its velocity based on its own experience and based on the experience of a neighboring particle, thus making use of the best position encountered by itself and its neighbor. Particles move through the problem space by following a current of optimum particles. The process is then iterated a fixed number of times or until a predetermined minimum error is achieved [15].

PSO has been successfully applied in many areas, amongst them function optimization [14], artificial neural network training [21], and fuzzy system control [8]. A comprehensive survey of PSO algorithms and

their applications can be found in [17]. However, many optimization problems occur in a space featuring discrete, qualitative distinctions between variables and between levels of variables. Kennedy and Eberhart introduced binary PSO (BPSO), which can be applied to discrete binary variables. In this study, BPSO was used, since the position of each individual particle can be given in binary form (0 or 1), which adequately reflects the straightforward "yes/no" choice of whether a feature needs to be selected or not. The changes in particle velocity can be interpreted as a change in the probability of finding the particle in one state or another. Since, this change is a probability, it is limited to a range of $\{0.0 - 1.0\}$. Further, details about BPSO can be found in Kennedy and Eberhart [16].

Feature selection may suffer from a high number of features, which lead to poor classification accuracy. In order to obtain superior classification results, we modified the simple BPSO procedure. In BPSO, each particle adjusts its position according to two fitness value, *pbest* and *gbest*, and avoids getting trapped in a local optimum by fine-tuning the inertia weight. *pbest* is a local fitness value, whereas *gbest* constitutes a global fitness value. If the *gbest* value is itself trapped in a local optimum, i.e., is not updated for three consecutive generations, the search of each particle is limited to the area, it is presently in. Under these circumstances, the search process comes to a stop and superior classification results can not be obtained. Thus, we propose an improved BPSO (IBPSO) procedure that retires *gbest* under such circumstances. The *gbest* value is reset to zero, and the search process continues on to regions outside the local optimum. In the end, better classification results can be achieved, even with a reduced number of selected features.

2.3. *K*-Nearest neighbor method

The *K*-nearest neighbor (*K*-NN) method was first introduced by Fix and Hodges in 1951, and is one of the most popular nonparametric methods [4], [9]. The purpose of the algorithm is to classify a new object based on attributes and training samples. The *K*-nearest neighbor method consists of a supervised learning algorithm, where the result of a new query instance is classified based on the majority of the *K*-nearest

neighbor category. The classifiers do not use any model for fitting and are only based on memory, which works based on a minimum distance from the query instance to the training samples to determine the *K*-nearest neighbors. Any tied results are solved by a random procedure.

The *K*-NN method has been successfully applied in various areas, e.g., statistical estimation, pattern recognition, artificial intelligence, categorical problems, and feature selection. The advantage of the *K*-NN method is that, it is simple and therefore easy to implement. *K*-NN is not negatively affected by large training data sets and is indifferent to noisy training data [4]. In this study, the feature subset was measured by the leave-one-out cross-validation of one nearest neighbor (1-NN). Neighbors are calculated using their Euclidean distance. The 1-NN classifier does not require any user-specified parameters and the classification results are implementation independent.

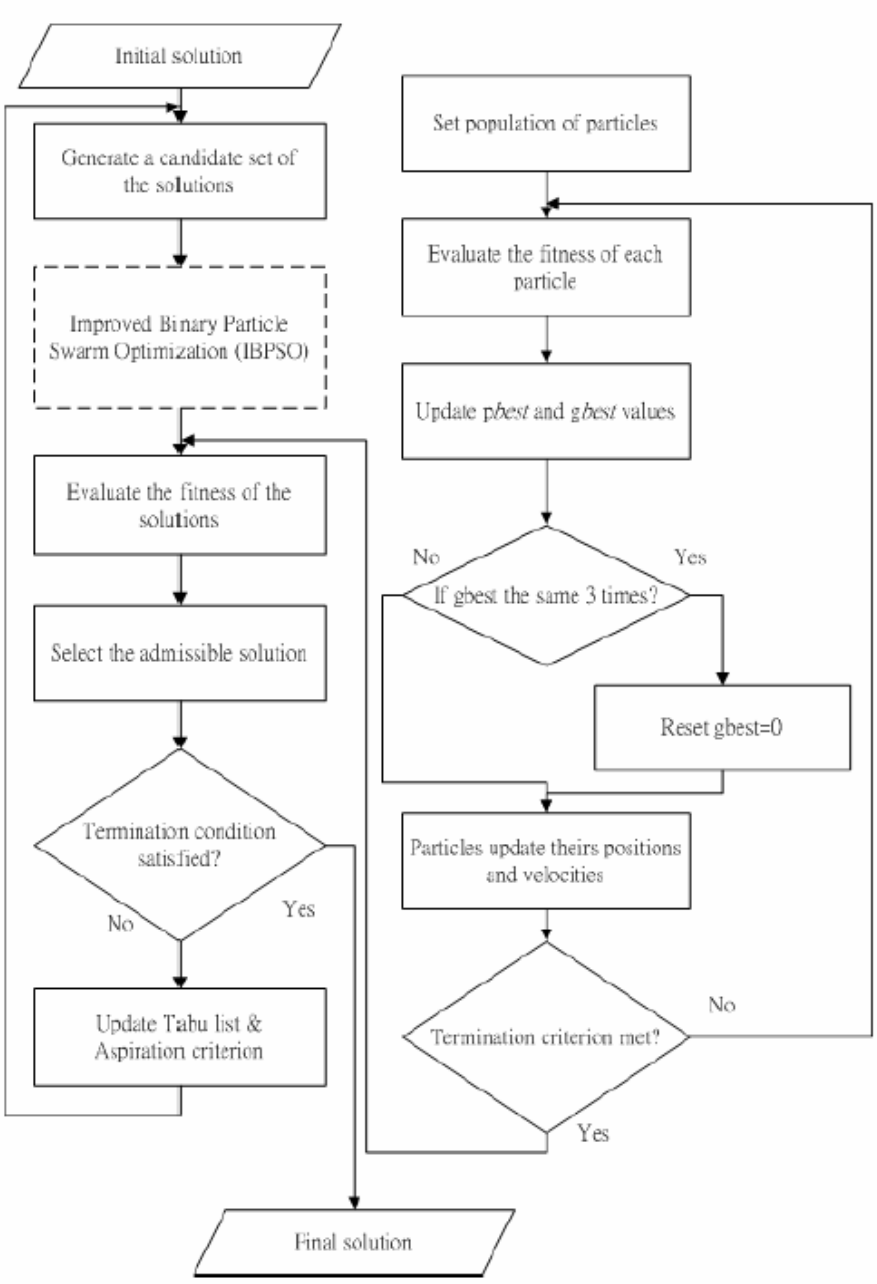

Figure 1. Flowchart of the hybrid TS-IBPSO.

2.4. Hybrid TS-IBPSO (IBPSO nested in a TS)

The hybrid TS-IBPSO procedure used in this study combines TS and IBPSO for feature selection, as shown in Figure 1. It adheres to the following pattern. A randomly generated initial solution is coded to a binary string $S = F_1F_2 \ldots F_n$, $n = 1, 2, \ldots, m$, m-dimensional data set. F represents a single feature; the bit value $\{1\}$ represents a selected feature, whereas the bit value $\{0\}$ represents a non-selected feature. The candidate set of solutions is generated by adding or deleting a feature. For example, when a 10-dimensional data set $(n = 10)$ $S_n = F_1F_2F_3F_4F_5F_6F_7F_8F_9F_{10}$ is analyzed, any number of features smaller than 10 can be selected, i.e., 6 random features $(S_n = F_1F_3F_5F_7F_9F_{10})$. TS candidate sets of the solutions may be 1000101011 (feature deleted), 1110101011 (feature added), etc.

IBPSO is embedded within the TS and serves as a local optimizer in order to improve the TS performance with each iteration. Each candidate of the solutions in TS represents a single particle of the IBPSO. The position of each particle is represented by $X_p = \{x_{p1}, x_{p2}, \ldots, x_{pd}\}$ and the velocity of each particle is represented by $V_p = \{v_{p1}, v_{p2}, \ldots, v_{pd}\}$. The particle is updated at each iteration by following two "best" (optimum) values, called *pbest* and *gbest*. Each particle keeps track of its coordinates in the problem space; the coordinates are associated with the best solution (fitness) the particle has achieved so far. This fitness value is stored and represents the position called *pbest*. When a particle takes the whole population as its topological neighbor, the best value is a global optimum value called *gbest*.

Once the adaptive values *pbest* and *gbest* are obtained, the features of the *pbest* and *gbest* particles can be tracked with regard to their position and velocity. Each particle is updated according to the following equations [16].

$$v_{pd}^{new} = w \times v_{pd}^{old} + c_1 \times rand_1 \times \left(pbest_{pd} - x_{pd}^{old}\right) + c_2 \times rand_2 \times \left(gbest_d - x_{pd}^{old}\right), \quad (1)$$

$$\text{if } v_{pd}^{new} \notin (V_{\min}, V_{\max}), \text{ then } v_{pd}^{new} = \max(\min(V_{\max}, v_{pd}^{new}), V_{\min}), \quad (2)$$

$$S\left(v_{pd}^{new}\right) = \frac{1}{1 + e^{-v_{pd}^{new}}}. \quad (3)$$

$$\text{If } \left(rand < S\left(v_{pd}^{new}\right)\right), \text{ then } x_{pd}^{new} = 1; \text{ else } x_{pd}^{new} = 0. \quad (4)$$

In these equations, w is the inertia weight, c_1 and c_2 are acceleration (learning) factors, and $rand$, $rand_1$ and $rand_2$ are random numbers between 0 and 1. Velocities v_{pd}^{new} and v_{pd}^{old} are those of the updated particle and the particle before being updated, respectively, x_{pd}^{old} is the original particle position (solution), and x_{pd}^{new} is the updated particle position (solution).

In Equation 2, particle velocities of each dimension are tried to a maximum velocity $V_{\max}$. If the sum of accelerations causes the velocity of that dimension to exceed $V_{\max}$, then the velocity of that dimension is limited to $V_{\max}$. $V_{\max}$ and $V_{\min}$ are user-specified parameters (in our case $V_{\max} = 6$, $V_{\min} = -6$).

The new velocity of particles is produced by the product of the inertia weight and the old particle velocity; the cognition-only model of particles, and the social-only model of particles can be applied. The cognition-only model of particles belongs to the local optimum, and the social-only model of particles belongs to the global optimum.

The BPSO converges rapidly during the initial stages of a search, but then often slows considerably due to particles being trapped in a local optimum. In order to avoid this entrapment in a local optimum, the *gbest* value has to be evaluated before each particle position is updated. If *gbest* has the same value for a preset number of times (in our case three times), the particle could conceivably be trapped in a local optimum. In such a case, the *gbest* position is reset to zero in the fitness function (classification accuracy), meaning that zero features are selected, while

pbest is kept. In the next iteration, particles in the neighborhood of the local optimum will adjust their position by congregating towards the *gbest* position.

The features after updating are calculated by the function $S(v_{pd}^{new})$ (Equation 3) [16], in which v_{pd}^{new} is the updated velocity value. If $S(v_{pd}^{new})$ is larger than a randomly produced disorder number, that is, within $\{0.0 \sim 1.0\}$, then its position value S_n, $n = 1, 2, \ldots, m$ is represented as $\{1\}$ (meaning this feature is selected as a required feature for the next update). If $S(v_{pd}^{new})$ is smaller than a randomly produced disorder number, that is, within $\{0.0 \sim 1.0\}$, then its position value F_n, $n = 1, 2, \ldots, m$ is represented as $\{0\}$ (meaning this feature is not selected as a required feature for the next update) [16].

In IBPSO, the predictive accuracy of a 1-nearest neighbor (1-NN) determined by the leave-one-out cross-validation (LOOCV) method is used to measure the fitness of each particles. In the LOOCV method, a single observation from the original sample is selected as the validation data, and the remaining observations serve as training data. This is repeated so that each observation in the sample is used once as the validation data. Essentially, this is the same as *K*-fold cross-validation, where *K* is equal to the number of observations in the original sample. The obtained classification accuracy is an adaptive functional value.

The TS was configured to contain 10 candidates and was run for 30 iterations in each trial or until the termination criterion was satisfied (fitness value=1.0). The tabu list size, set to 7 in our case, is the same value Glove proposed [10]. The number of particles used was 10. The acceleration factors c_1 and c_2 were set to $c_1 = c_2 = 2$. The inertia weight w was 1.0. The above values for c_1, c_2, and w were taken from Shi and Eberhart [30]. The maximum number of iterations used in our IBPSO procedure was 30. The pseudo-code of the proposed method is given below.

Pseudo-code for hybrid TS-IBPSO procedure

```
1.  begin
2.      Randomly generate an initial solution
3.      while (number of iterations, or the termination criterion is not met)
4.       Generate the candidate set of solutions by Tabu Search( )
5.          Execute Improved Binary Particle Swarm Optimization( )
6.              for c = 1 to number of candidate set
7.              Evaluate fitness value of candidate set by 1-Nearest Neighbor( )
8.              next c
9.          Execute Tabu Search ( )
10.     next iterations until termination criterion
11. end
```

Pseudo-code for Tabu Search procedure

```
1.  begin
2.      Randomly generate initial solution s
3.      while (number of iterations, or the termination criterion is not met)
4.          Generate the candidate set N(x)
5.          Evaluate fitness value in N(x) by 1-Nearest Neighbor( )
6.          Find the best solution y in N(x)
7.            if y is in Tabu list and x does not satisfy the aspiration criterion, then
8.                N(x)-{y} and go to 6:
9.            end if
10.         Otherwise s = y  if y is better than s
11.         Update tabu list and aspiration criterion
12.     next iterations until termination criterion
13. end
```

Pseudo-code for Improved Binary Particle Swarm Optimization procedure

1. **begin**
2. **while** (number of iterations, or the termination criterion is not met)
3. Evaluate fitness value of particle swarm by **1-*Nearest Neighbor*( )**
4. **for** $p = 1$ to number of particles
5. **if** fitness of X_p is greater than the fitness of $pbest_p$, **then**
6. $pbest_p = X_p$
7. **end if**
8. **if** fitness of any particle of particle swarm is greater than *gbest* **then**
9. *gbest = position of particle*
10. **end if**
11. **if** fitness of *gbest* is the same Max times **then** give up and reset *gbest*
12. **end if**
13. **for** $d = 1$ to number of features of each particle
14. $v_{pd}^{new} = w \times v_{pd}^{old} + c_1 \times rand_1 \times \left(pbest_{pd} - x_{pd}^{old}\right) + c_2 \times rand_2 \times \left(gbest_d - x_{pd}^{old}\right)$
15. **if** $v_{pd}^{new} \notin (V_{\min}, V_{\max})$ **then** $v_{pd}^{new} = \max(\min(V_{\max}, v_{pd}^{new}), V_{\min})$
16. **end if**
17. $S\left(v_{pd}^{new}\right) = \dfrac{1}{1 + e^{-v_{pd}^{new}}}$
18. **if** $\left(rand < S\left(v_{pd}^{new}\right)\right)$ **then** $x_{pd}^{new} = 1$ **else** $x_{pd}^{new} = 0$
19. **end if**
20. **next *d***
21. **next *p***
22. **next generation until termination criterion**
23. **end**

Pseudo-code for 1-Nearest Neighbor procedure

```
1.  begin
2.      for i = 1 to sample number of classification problem
3.        For j = 1 to sample number of classification problem
4.          for k = 1 to dimension number of classification problem
5.              dist_i = dist_i + (data_ik − data_jk)^2
6.          next k
7.          if dist_i < nearest then
8.              class_i = class_j
9.              nearest = dist_i
10.         end if
11.       next j
12.     next i
13.     for i = 1 to sample number of classification problem
14.         if class_i = real class of testing data then correct = correct + 1
15.         end if
16.     next i
17.     Fitness value = correct / number of testing data
18. end
```

3. Results and Discussion

The data sets, we used in this study were obtained from the UCI Repository, with the number of features being greater than 10 [24]. The data was arranged in the format shown in Table 1. Three types of classification problems were tested. If the number of features was between 10 and 19, the sample groups were considered small; these sample groups included the glass, vowel, wine, letter, vehicle and segmentation problems. If the number of features was between 20 and 49, the sample group test problems were of medium size. These problems

included the WDBC, ionosphere, and satellite problems. If the number of features was greater than 50, the test problems were large sample group problems; this group included the sonar problem. The evaluation method used was the 1-NN method with leave-one-out cross-validation (LOOCV) for all data sets. For the test problem wine, the attribute values have been normalized.

Table 1. Format of classification test problems

Data sets	Number of samples	Number of classes	Number of features	Evaluation method
Glass	214	7	10	1-NN
Vowel	990	11	10	1-NN
Wine	178	3	13	1-NN
Letter	15000/5000	26	16	1-NN
Vehicle	846	4	18	1-NN
Segmentation	210/2100	7	19	1-NN
WDBC	569	2	30	1-NN
Ionosphere	201/150	2	34	1-NN
Satellite	4435/2000	6	36	1-NN
Sonar	104/104	2	60	1-NN

Legend. x /y indicates the number of testing and training samples, respectively.

Table 2. Classification accuracy for the tested data sets

Data sets	d[a]	SFS	PTA	SFFS	SGA	HGA(1)	HGA(2)	HGA(3)	HGA(4)	TS		TS-IBPSO	
										d[b]	A(%)	d[b]	A(%)
Glass (D = 10)	2	99.07	99.07	99.07	99.07	99.07	99.07	NA	NA	4	**100**	4	**100**
	4	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**				
	6	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**				
	8	**100**	**100**	**100**	**100**	**100**	**100**	**100**	NA				
Vowel (D = 10)	2	62.02	62.02	62.02	62.02	62.02	62.02	NA	NA	8	**99.70**	7	**99.70**
	4	92.63	92.83	92.83	92.83	92.83	92.83	92.83	92.83				
	6	98.28	98.79	98.79	98.79	98.79	98.79	98.79	98.79				
	8	**99.70**	**99.70**	**99.70**	**99.70**	**99.70**	**99.70**	**99.70**	NA				
Wine (D = 13)	3	93.82	93.82	93.82	93.82	93.82	93.82	93.82	NA	6	93.81	6	**99.44**
	5	94.38	94.38	94.94	95.51	95.51	95.51	95.51	95.51				
	8	95.51	95.51	95.51	95.51	95.51	95.51	95.51	95.51				
	10	92.13	92.13	92.70	92.70	92.70	92.70	92.70	92.70				
Letter (D = 16)	3	47.09	47.09	47.09	47.09	47.09	47.09	47.09	NA	12	93.85	12	**96.58**
	6	86.20	87.60	87.60	87.60	87.60	87.60	87.60	87.60				
	10	96.12	96.35	96.35	96.35	96.35	96.35	96.35	96.35				
	13	96.42	96.42	96.42	96.42	96.42	96.42	96.42	96.42				
Vehicle (D = 18)	4	62.77	64.78	69.15	69.50	69.74	69.74	69.74	69.74	10	72.34	8	**75.54**
	7	69.15	70.09	73.52	72.97	73.52	73.52	73.52	73.52				
	11	69.50	71.75	71.87	71.84	72.46	72.46	72.46	72.46				
	14	68.20	70.80	70.80	70.80	70.80	70.80	70.80	70.80				

Segmentation (D = 19)	4	92.81	92.81	92.81	92.81	92.81	92.81	92.81	92.81	5	96.88	7	**97.92**
	8	92.95	92.95	92.95	92.95	92.95	92.95	92.95	92.95				
	11	92.95	92.95	92.95	92.95	92.95	92.95	92.95	92.95				
	15	92.57	92.57	92.57	92.57	92.57	92.57	92.57	92.57				
WDBC (D = 30)	6	93.15	93.15	94.20	93.67	94.90	94.90	93.99	93.99	8	93.85	12	**97.68**
	12	92.62	92.97	94.20	94.38	94.38	94.38	94.38	94.38				
	18	94.02	94.20	94.20	93.85	94.20	94.20	94.20	94.20				
	24	92.44	93.50	93.85	93.85	93.85	93.85	93.85	93.85				
Iono-sphere (D = 34)	7	93.45	93.45	93.45	95.44	**95.73**	**95.73**	**95.73**	**95.73**	14	92.88	15	**95.73**
	14	90.88	92.59	93.79	94.87	**95.73**	**95.73**	**95.73**	**95.73**				
	20	90.03	92.02	92.88	94.30	94.30	94.30	94.02	94.30				
	27	89.17	91.17	90.88	91.45	91.45	91.45	91.45	91.45				
Satellite (D = 36)	7	86.85	88.20	88.55	88.10	88.55	88.55	88.55	88.55	21	91.06	22	**91.73**
	14	89.45	89.85	90.10	90.85	90.95	91.00	90.95	90.95				
	22	90.45	91.10	91.45	91.45	91.45	91.45	91.45	91.45				
	29	90.40	90.70	91.25	91.25	91.25	91.25	91.25	91.25				
Sonar (D = 60)	12	87.02	89.42	92.31	93.75	94.71	95.67	95.19	95.67	25	93.75	26	96.52
	24	89.90	90.87	93.75	95.67	96.63	96.63	**97.12**	**97.12**				
	36	88.46	91.83	93.27	95.67	96.15	96.15	96.15	96.15				
	48	91.82	92.31	91.35	92.79	92.79	93.27	93.27	93.27				

Legend. D: total number of features, d^a: selected number of features, SFS: sequential forward search, PTA: plus and take away, SFFS: sequential forward floating search, SGA: sequential genetic algorithm, HGA: hybrid genetic algorithm (Oh et al. [25]), d^b: optimal selected number of features, TS: tabu search, TS-IBPSO: the proposed method, A(%): classification accuracy in %. Highest values are in bold-type.

Table 1 illustrates the format of the ten classification problems. Table 2 compares experimental results obtained by other methods from the literature [25] with TS and the proposed method. The proposed method obtained the highest classification accuracy for the glass, vowel, wine, letter, vehicle, segmentation, WDBC, ionosphere, and satellite classification problems. The classification accuracies of the wine and segmentation classification problems obtained by the proposed method are 99.44% and 97.92%, respectively. This constitutes a considerable increase of 4% classification accuracy compared to the other methods shown in Table 2. The increase in accuracy could be achieved in spite of the fact that the number of selected features was usually lower (but in any case never higher) than in the methods from the literature. This clearly, indicates that not all features are necessary to achieve higher classification accuracy. The classification accuracies achieves for the sonar classification problems were slightly lower than the classification accuracies of some of the other feature selection methods. Yet, they are very close in value and were sometimes achieved with fewer features selected. These results indicate that the proposed method (TS-IBPSO) can serve as a useful pre-processing tool and help optimize the feature selection process, thus improving classification accuracy.

In Oh et al. [25], optimal classification accuracy was obtained by exhaustive search for various numbers of features. Classification accuracies were measured by four values (D/5, 2D/5, 3D/5, and 4D/5) of the total number of features (D). In fact, the optimal number of features for each test problem is unknown. The method of Oh et al. is time-consuming compared to the method, we propose here, in which an optimal member of features is determined by the TS with IBPSO as an evaluator. Even if the accuracy results obtained cannot be directly compared, it seems safe to imply that they are at least on par, if not better (where the same or similar number of features has been used), and that, this accuracy has been obtained in a fraction of the time. For the test problem letter, the classification accuracy obtained by the proposed method was higher than the results in Oh et al., while the number of features selected was lower. For the test problems glass and vowel, the classification accuracies obtained by the proposed method was the same as the results obtained in

Oh et al., while the number of features used was lower than Oh et al., For the test problems wine, segmentation, and satellite, classification accuracies obtained by the proposed method were higher than the results obtained in Oh et al. [25], even though at the same time the number of features used was lower. Table 3 shows the time complexity of the proposed method compared to the other methods. The number of the initial feature sets is denoted D. In the Table, $\Theta(\ )$ denotes a tight estimate of complexity (exact, except for a multiplicative constant) and $O(\ )$ denotes an estimate of complexity for which only an upper bound is known. Time complexities under a typical setting of parameters are shown in parentheses. These time complexities are only an indication, when used in these algorithms [19]. Search types can be divided into two cases, called *sequential* and *parallel*.

Table 3. Time complexity of existing methods

Existing methods	Time complexity	Search types
SFS	$\Theta(D^2)$	Sequential
PTA	$\Theta(D^2)$	Sequential
SFFS	$O(2^D)$	Sequential
SGA	$\Theta(D)$	Parallel
HGAs	$\Theta(D)\Theta(D)$	Parallel
TS-IBPSO	$\Theta(D)\Theta(D)$	Parallel

Legend. SFS: sequential forward search, PTA: plus and take away, SFFS: sequential forward floating search, SGA: sequential genetic algorithm, HGA: hybrid genetic algorithm, TS-IBPSO: the proposed method.

Figures 2-11 show the number of iterations vs. classification accuracy for the tested data sets.

The red colour indicates the TS-IBPSO method and the blue colour indicates TS. For the glass data set (Figure 2), the proposed method obtained 100% classification accuracy before reaching the maximum

number of iterations. Table 2 and Figure 2 show that in cases where the number of features selected is identical, the classification accuracy can still be increased since the actual features selected can be different even, if the total number of features selected is identical. For this reason, good feature selection should not only decrease the number of features, but also choose features that are beneficial to improving classification accuracy. It also demonstrates that a low number of features selected is not necessarily detrimental to accuracy. Hence, as long as the chosen features contain enough feature classification information, higher classification accuracy can be achieved.

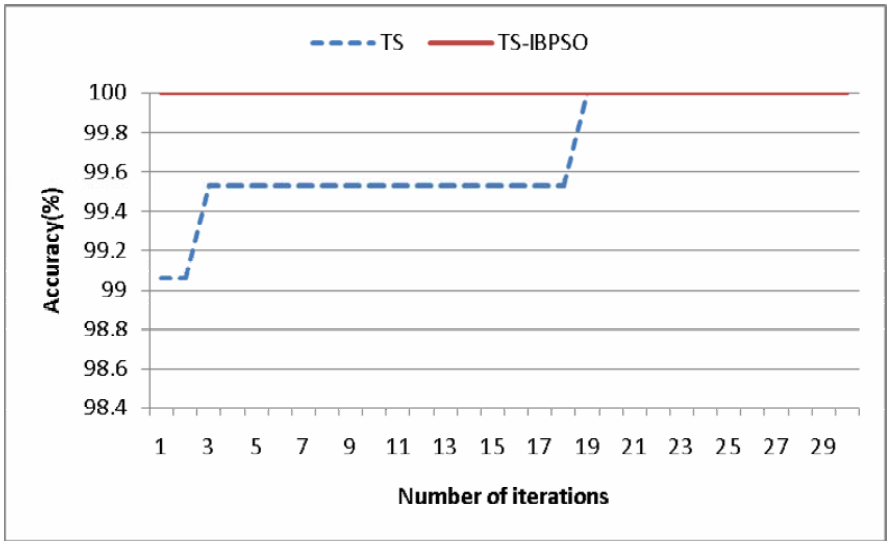

Figure 2. Glass data set-Number of iterations vs. accuracy.

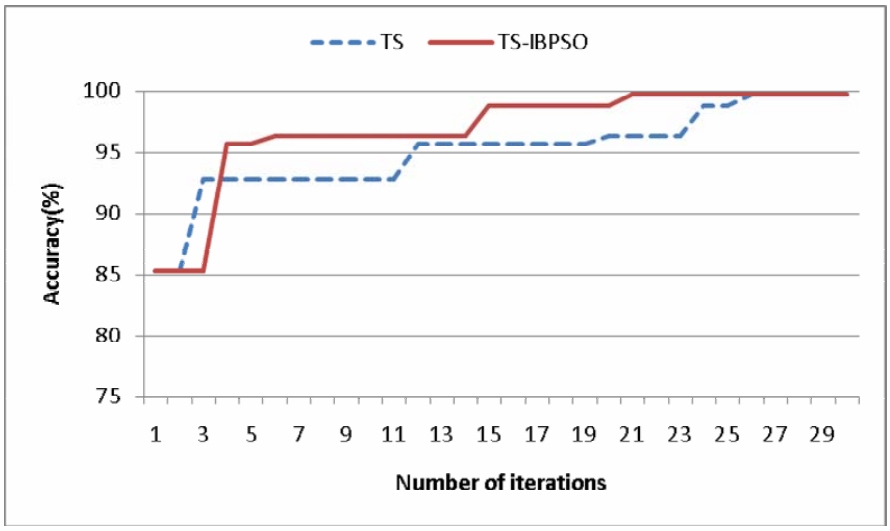

Figure 3. Vowel data set-Number of iterations vs. accuracy.

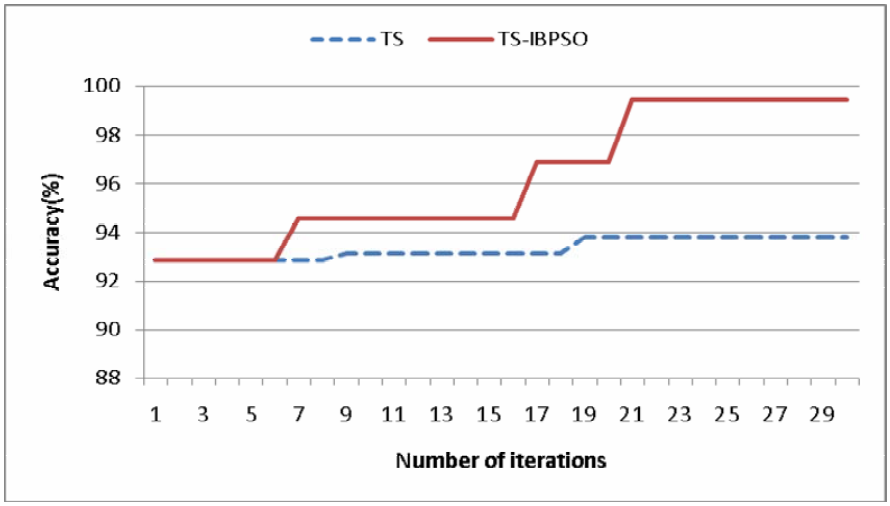

Figure 4. Wine data set-Number of iterations vs. accuracy.

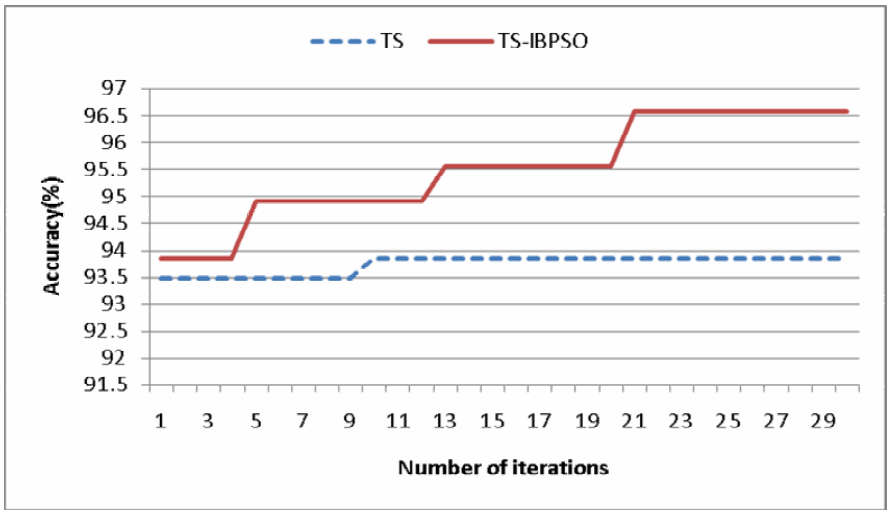

Figure 5. Letter data set-Number of iterations vs. accuracy.

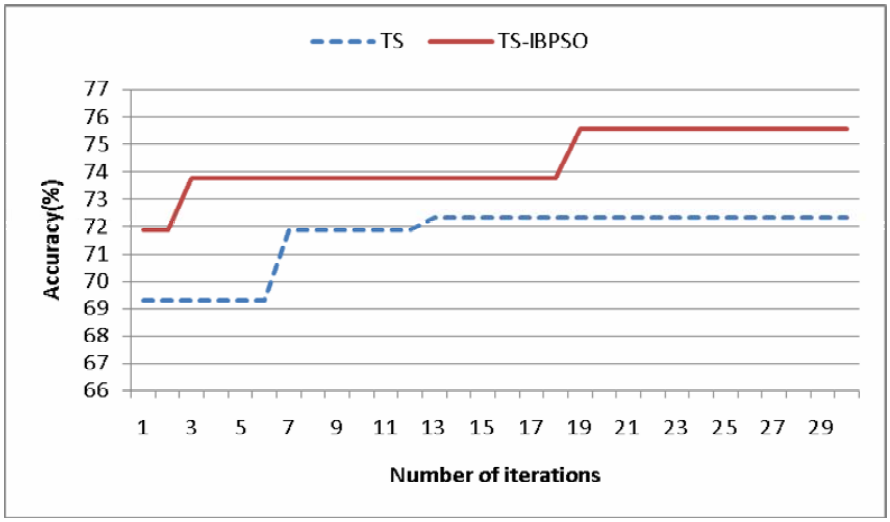

Figure 6. Vehicle data set-Number of iterations vs. accuracy.

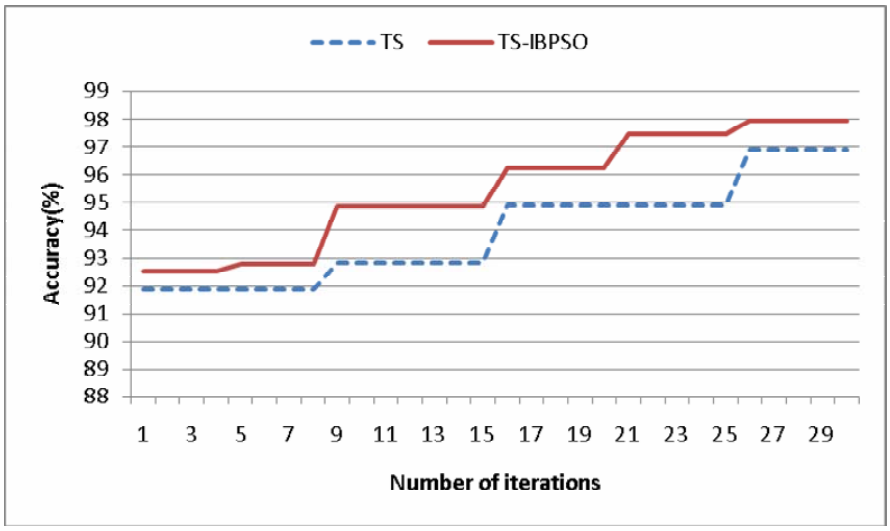

Figure 7. Segmentation data set-Number of iterations vs. accuracy.

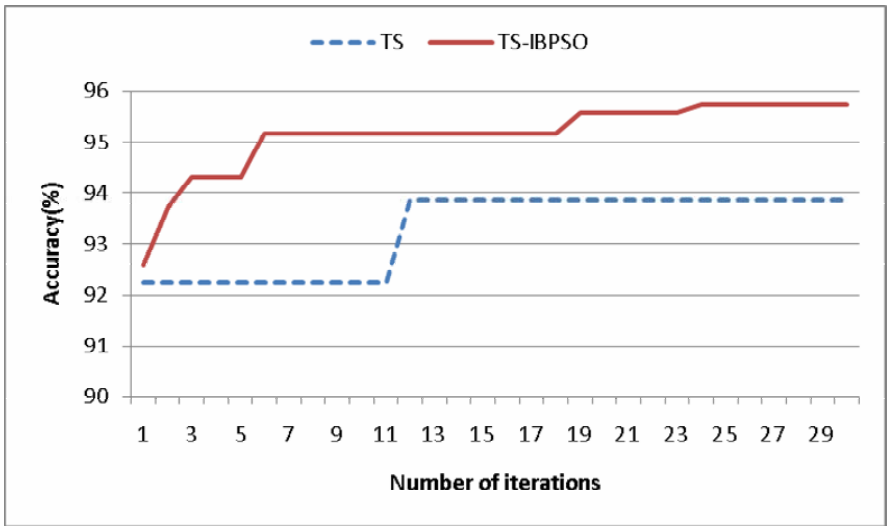

Figure 8. WDBC data set-Number of iterations vs. accuracy.

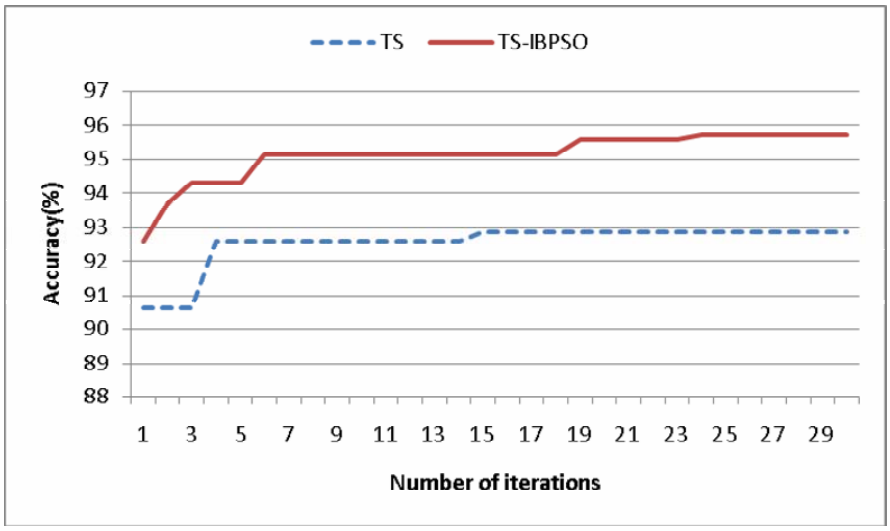

Figure 9. Ionosphere data set-Number of iterations vs. accuracy.

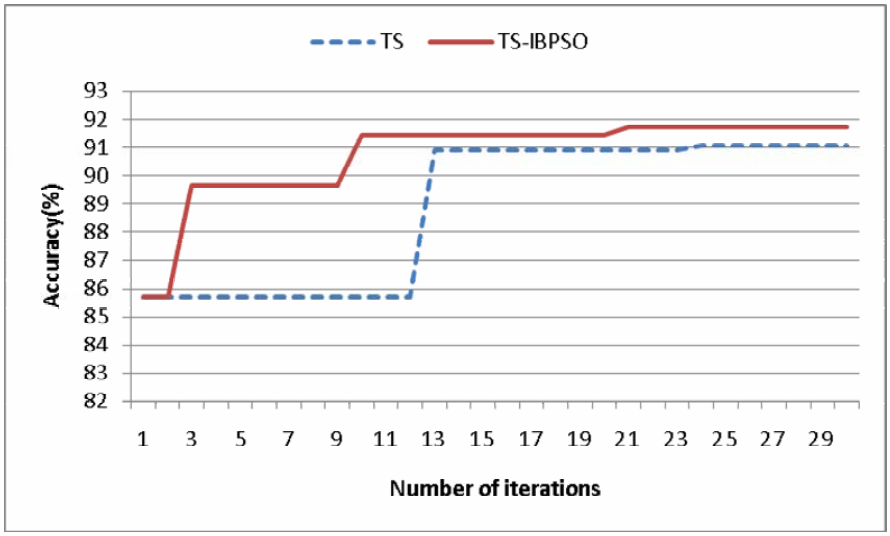

Figure 10. Satellite data set-Number of iterations vs. accuracy.

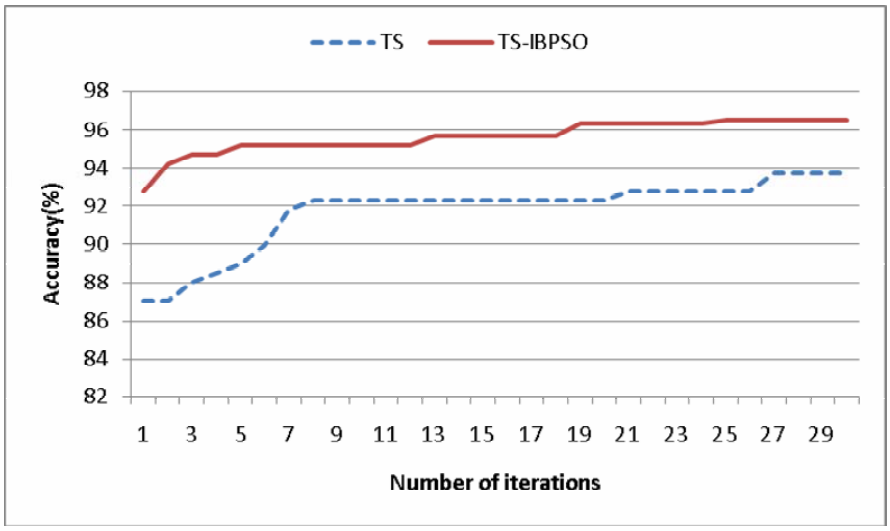

Figure 11. Sonar data set-Number of iterations vs. accuracy.

TS randomly generated an initial solution, which later on by way of shifting and increased characteristics, results in a combination of neighbor solutions. Some or all candidate sets are, then chosen from the neighbor solutions. The solution of particle movement in TS is not the same as the calculation of a greedy algorithm. After applying TS to a generation, a temporarily best solution is produced. TS itself can solve the calculation, but it can not guarantee that the best candidate improves the accuracy of classification. Since, TS can always solve the calculation, it won't be interrupted by particles stuck in a local optimum. These characteristics of TS ensure that a global best solution can be found.

TS-IBPSO is a combination of the TS and IBPSO methods used to improve progressive characteristics. In general, the TS candidate list is established by deleting or adding characteristics. However, for a relative high number of characteristics $(D > 13)$, TS results are usually poor. We therefore, decided to improve the TS by combining it with IBPSO. IBPSO was added to the TS after each generation. The IBPSO carries out a local search inside the TS. IBPSO can thus improve the TS candidate set, which in turn leads to better overall solutions. Through, the combination of these two different methods their respective advantages can both be

used. TS prevent particles from getting trapped in a local optimum, whereas IBPSO allows for a search through all possible solution spaces in a short time.

When analyzing the experimental results in Table 2, we see that when only TS is used the rate of correct classification is lower or equal to TS-IBPSO, except in the cases of the glass and vowel data sets. The number of dimensions of both, the glass and vowel data sets, is low (D = 10). For these cases, the results for TS and TS-IBPSO are identical. We suppose that for a smaller dimension of characteristics, the incorporation of IBPSO does not result in improved candidate sets. However, for wine (D = 13), letter (D = 16), vehicle (D = 18), segmentation (D = 19), WDBC (D = 30), ionosphere (D = 34), satellite (D = 36) and sonar (D = 60), all data sets with higher dimensionality, TS-IBPSO improved the rate of correct classification considerably. The results prove that TS-IBPSO can improve the accuracy of classification significantly for data sets with a high number of characteristics.

Conclusion

We used a hybrid of TS and improved binary PSO (TS-IBPSO) to perform feature selection. The *K*-NN method with LOOCV served as an evaluator of the TS and IBPSO fitness functions. Experimental results show that TS-IBPSO simplified feature selection by reducing the total number of features needed effectively, and that, it obtained higher classification accuracy compared to other feature selection methods for most test data sets. The classification accuracy obtained by the proposed method had the highest classification accuracy in nine of the ten data set test problems, and is comparable to the classification accuracy of the tenth test problems. TS-IBPSO can serve as a valuable pre-processing tool to help optimize the feature selection process, since it either increases the classification accuracy, reduces the number of necessary features for classification or does both. The proposed TS-IBPSO method could conceivably be applied to problems in other areas in the future.

Acknowledgements

This work is partly supported by the National Science Council in Taiwan under grants NSC94-2622-E-151-025-CC3, NSC94-2311-B037-001, NSC93-2213-E-214-037 and NSC92-2213-E-214-036.

References

[1] Imtiaz Ahmad, K. Muhammad Dhodhi and M. Faridah Ali, Tabu search based scheduling algorithm for behavioral synthesis of functional pipelines, The Computer Journal 13 (2000), 152-166.

[2] F. Brill, D. Brown and W. Martin, Fast genetic selection of features for neural network classifiers, IEEE Transactions of Neural Networks 3(2) (1992), 324-328.

[3] L. Y. Chuang, H. W. Chang, C. J. Tu and C. H. Yang, Improved binary PSO for feature selection using gene expression data, Computational Biology and Chemistry 32 (2008), 29-38.

[4] T. Cover and P. Hart, Nearest neighbor pattern classification, Proc. IEEE Trans. Information Theory IT-11 (1967), 21-27.

[5] K. Crammer and Y. Singer, On the Learn Ability and Design of Output Codes for Multiclass Problems, Proceedings of the Thirteen Annual Conference on Computational Learning Theory (COLT 2000), Stanford University, Palo Alto, CA, June 28-July 1, (2000).

[6] B. V. Dasarathy, NN concepts and techniques, nearest neighbor (NN) norms: NN Pattern classification techniques, (Ed.), IEEE Computer Society Press (1991), 1-30.

[7] C. Dimitropoulos, Tabu search for the radio links frequency assignment problem, Proceedings of the Conference on Applied Decision Technologies 2 (1995), 233-255.

[8] H. M. Feng, Particle swarm optimization learning fuzzy systems design, Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) 2 (2005), 363-366.

[9] E. Fix and J. L. Hodges, Discriminatory analysis-nonparametric discrimination: Consistency Properties, Project 21-49-004, Report 4, US Air Force School of Aviation Medicine, Randolph Field (1951), 261-279

[10] F. Glover, Tabu Search-Part I, ORSA Journal on Computing 3 (1989), 190-206.

[11] F. Glover, Tabu Search-Part II, ORSA Journal on Computing 2 (1990), 4-32.

[12] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, Prediction Springer, (2001).

[13] C. W. Hsu and C. J. Lin, A comparison of methods for multi-class support vector machines, IEEE Trans. Neural Network 12 (2002), 415-425.

[14] S. Janson and M. Middendorf, A hierarchical particle swarm optimizer for dynamic optimization problems, LNCS No.3005: Proceedings of Applications of Evolutionary Computing: EvoWorkshops 2004: EvoBIO, EvoCOMNET, EvoHOT, EvoISAP, EvoMUSART, and EvoSTOC, Coimbra, Portugal, (2004), 513-524.

[15] J. Kennedy and R. C. Eberhart, Particle swarm optimization, In proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, 4 (1995), 1942-1948.

[16] J. Kennedy and R. C. Eberhart, A discrete binary version of the particle swarm algorithm, Systems, Man and Cybernetics, 1997, Computational Cybernetics and Simulation, 1997 IEEE International Conference on Volume 5, 12-15 Oct. (1997), 4104-4108.

[17] J. Kennedy, R. C. Eberhart and Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, 2001.

[18] U. Kreßel, Pairwise classification and support vector machines, Advances in Kernel Methods, Support Vector Learning, Cambridge, MA: MIT Press (1999), 255-268.

[19] M. Kudo and J. Sklansky, Comparison of algorithms that select features for pattern classifiers Pattern Recognition 33 (2000), 25-41.

[20] T. Li, S. Zhu and M. Ogihara, Efficient multi-way text categorization via generalized discriminant analysis, Proceedings of Twelfth International Conference on Information and Knowledge Management (CIKM 2003), ACM Press, NY, (2003), 317-324.

[21] M. Meissner, M. Schmuker and G. Schneider, Optimized Particle Swarm Optimization (OPSO) and its Application to Artificial Neural Network Training, BMC Bioinformatics, in press, (2006).

[22] T. M. Mitchell, Machine Learning, McGraw-Hill, New York, USA, (1997).

[23] T. Morzy, M. Matysiak and S. Salza, Tabu search optimization of large join queries, Lecture Notes in Computer Science 779 (1994), 309-322.

[24] P. M. Murphy and D. W. Aha, UCI Repository of Machine Learning Databases, Technical Report, Department of Information and Computer Science, University of California, Irvine, Calif, (1994).
Available: http://www.ics.uci.edu/~mlearn/MLRepository.html.

[25] I. S. Oh, J. S. Lee and B. R. Moon, Hybrid genetic algorithms for feature selection, IEEE Trans. Pattern Analysis and Machine Intelligence 26(11) (2004), 1424-1437.

[26] J. C. Platt, N. Cristianini and J. Shawe-Taylor, Large margin DAGS for multiclass classification, Advances in Neural Information Processing Systems 12, MIT Press (2000), 547-553.

[27] P. Pudil, J. Novovicova and J. Kittler, Floating search methods in feature selection, Pattern Recognition Letters 15 (1994), 1119-1125.

[28] B. Roberto, Using mutual information for selecting features in supervised neural net learning, IEEE Trans. on Neural Networks 5(4) (1994), 537-550.

[29] R. S. Safavian and D. Landgrebe, A survey of decision tree classifier methodology, IEEE Trans. Systems, Man, and Cybernetics 21 (1991), 660-675.

[30] Y. H. Shi and R. C. Eberhart, A modified particle swarm optimizer, In Proceedings of the IEEE International Conference on Evolutionary Computation (1998), 69-73.

[31] M. A. Tahir, A. Bouridane, F. Kurugollu and A. Amira, A novel prostate cancer classification technique using intermediate memory tabu search, EURASIP J. Appl. Sign. Proc. 14 (2005), 2241-2249.

[32] M. A. Tahir, A. Bouridane and F. Kurugollu, Simultaneous feature selection and feature weighting using hybrid tabu search/*K*-nearest neighbor classifier, Pattern Recognition Letters 28 (2007), 438-446.

[33] V. Vapnik, Statistical Learning Theory, Wiley-Interscience, New York, USA, (1998).

[34] M. P. Wachowiak and A. S. Elmaghraby, The continuous Tabu Search as an Optimizer for 2D-to-3D Biomedical Image Registration, Lecture Notes in Computer Science, Proc. MICCAI 2001-2208, (2001), 1273-1274.

[35] J. Weston and C. Watkins, Support vector machines for multi-class pattern recognition, Proceedings of the Seventh European Symposium On Artificial Neural Networks (ESANN 99), Bruges (1999), 21-23.

[36] Habib Youssef and M. Sadiq Sait, Timing-driven global routing for standard-cell {VLSI} design, Int. J. Comp. Sys. Sci. Eng. 14 (1999), 175-185.

[37] B. Yu and B. Yuan, A more efficient branch and bound algorithm for feature selection, Pattern Recognition 26(6) (1993), 883-889.

[38] H. Zhang and G. Sun, Feature selection using tabu search method, Pattern Recognition 35 (2002), 701-711.

■