Combining Global Optimization Algorithms with a Simple Adaptive Distance for Feature Selection and Weighting

Adélia C. A. Barros and George D. C. Cavalcanti

***Abstract*— This work focuses on a study about hybrid optimization techniques for improving feature selection and weighting applications. For this purpose, two global optimization methods were used: *Tabu Search*(TS) and *Simulated Annealing* (SA). These methods were combined to k-Nearest Neighbor (k-NN) composing two hybrid approaches: SA/k-NN and TS/k-NN. Those approaches try to use the main advantage from the global optimization methods: they work efficiently in searching for solutions in the global space. In this study, the methodology is proposed by [4]. In the referred work, a hybrid TS/k-NN approach was suggested and successfully applied for feature selection and weighting problems. Based on the later, this analysis indicates a new SA/k-NN combination and compares their results using the classical *Euclidean Distance* and a *Simple Adaptive Distance* [8]. The results demonstrate that feature sets optimized by the studied models are very efficient when compared to the well-known k-NN. Both accuracy classification and number of features in the resultant set are considered in the conclusions. Furthermore, the combined use of the *Simple Adaptive Distance* improves even more the results for all datasets analyzed.**

I. Introduction

The feature set choice is one of the earliest and most important decisions in classification tasks. At this point, defined attributes will be considered in a certain class of objects in order to make easier their recognition. However, this choice is not simple: the more complex is the classification problem, the harder it is to find a satisfactory group of discriminant features.

Furthermore, the feature size set aims at being more compact once it directly affects the classifier performance and design. Larger feature sets imply in more complex data structures and computations. Despite the supracited advantages of features set careful choice, this process is made by either through an empirical or a subjective way at times.

As one may conclude, the desired solution is the smallest feature set sufficient to preserve the quality of discriminant information between classes. In this case, we can deal with features selection as an optimization problem whose objective function could be, for instance, to improve the accuracy rate as well as to reduce the feature set.

While feature selection leads to the reduction of the number of features, feature weighting is a method that preserves the original size of the set, with the singularity of associating weight values to each feature according to its discriminatory ability. Through feature weighting it is possible to observe the relevance of each feature from the group. The use of feature selection is suggested when the problem inquires if a feature is or not relevant.

Two well-known optimization methods could be used to find the optimal set of features: Tabu Search(TS)[1] [2] and Simulated Annealing(SA)[3]. Both methods are particularly efficient since they search in the space for the global minima solution avoiding local minima. Tabu Search was previously used for simultaneous feature selection and feature weighting in [4]. In the referred work, a new hybrid TS/k-Nearest Neighbor algorithm was proposed in which TS searches for solutions in the space and k-NN[5] evaluates these solutions. Despite being an expensive solution, its results overcomes those obtained with well-known feature selection methods. Feature selection and/or weighting have been useful for improving classification accuracy in combination with global optimization algorithms [6] [7].

This paper presents a study of simultaneous feature selection and feature weighting through global optimization techniques. Moreover, a new algorithm based on [4] is proposed. It combines Simulated Annealing (SA) with k-NN, once SA requires less computational time than the TS method. Both methods TS/k-NN and SA/k-NN are used here to evaluate the improvement in the classification accuracy rate.

In addition to that, a new distance measure was combined to the hybrid TS and SA approaches: the *Simple Adaptive Distance* [8]. This distance is considered interesting due to the fact it has improved the k-NN results in almost all datasets analyzed in that work. Its main idea is to rescale the distance between a query pattern and a training sample according to a radius value, get from the largest sphere centered on the training sample that excludes all patterns from other classes. By doing it, the boundary samples - frequently where the noise level is high - do not affect so much the k-NN results as we have with the use of the *Euclidean Distance*, for example, where the boundaries directly determine the classification.

The *Euclidean Distance* is such a well-known literature reference and for that reason, it does not require further comments. It is defined in equation 1.

$$d(\vec{x}, \vec{x_i}) = \left(\sum_{j=1}^{d} |x^j - x_i^j| \right)^{\frac{1}{2}} \quad (1)$$

where $\vec{x}$ and $\vec{x_i}$ are the input vectors and d is the number

This work was supported in part by the Brazilian National Research Council CNPq (Proc. 478534/2006-0).

Authors are with Center of Informatics (CIn), Federal University of Pernambuco (UFPE), P.O. Box 7851, Cidade Universitária, Cep: 50.740-540, Recife-PE, Brazil (corresponding authors to provide phone: 55-81-21268430; e-mails: acab@cin.ufpe.br and gdcc@cin.ufpe.br).

978-1-4244-1821-3/08/$25.00

of features.

The *Adaptive Distance* is briefly described below. Its definition is given as follows in equation 2.

$$d_{adaptive}(\vec{x}, \vec{x_i}) = \frac{d_{adaptive}(\vec{x}, \vec{x_i})}{r_i} \quad (2)$$

where r_i is the radius of the sphere which is defined in 3.

$$r_i = \min d(\vec{x_i}, \vec{x_l}) - \epsilon \quad (3)$$

in what the labels of $x_i \neq x_l$ and $\epsilon > 0$ is an arbitrarily small number.

By definition, the radius is achieved from the largest sphere whose central point is in the training pattern x_i and which does not embrace any pattern labeled differently from x_i. Notice that boundary samples tends to have smaller radii than the ones situated in the middle. So, the division by the radius in 2 decreases the influence of the boundary and, on the other hand, increases the magnitude of more centralized patterns.

We intend to compare the results obtained by k-NN, TS/k-NN and the proposed SA/k-NN when combined with the classical *Euclidean Distance* with those generated from the combination to the *Adaptive Distance*. Through this comparative study it is expected to show that: 1) the new hybrid SA/k-NN technique performs good results (feature set reduction is included in the conception of performance evaluation) and 2) the *Adaptive Distance* can, even more, increase the optimization algorithms performance.

Section II introduces the global optimization methods studied here: TS/k-NN and SA/k-NN. In Section IV, we present the experimental results of these hybrid techniques combined with the *Euclidean* and *Adaptive* distances. Finally, important remarks are presented in the Conclusion section V.

II. Global optimization Algorithms

Before mentioning the description of the global optimization algorithms, it is important to discuss the difference between local search and global search. There are in the literature methods based on the gradient information, they use this local information to move in the direction of a function minima point. Backpropagation[9] is a classical example of local search algorithms. It is highly efficient to determine the appropriate direction and magnitude for moving iteratively from a coordinate in the space to a local minima point.

On the other hand, global optimization algorithms search for the global minima point based on the information of the function surface as a whole[10]. Global methods can be applied to any function due to do not be dependent of gradient. Both methods *Tabu Search* and *Simulated Annealing* are considered global algorithms. They have as goal to find the function global minima point discarding the local minima ones when necessary.

A. TS/k-NN

As mentioned above, *Tabu Search* is a global search algorithm. A set of new solutions is created from the current solution and the best one (that whose cost function is the less) is always accepted as the current. The fact of always admit the new solution avoids the fall into local minima.

In order to prevent cycles in the search trajectory, the lately visited solutions are stored in a list called *tabu*. These solutions are prohibited with the intention of avoiding the algorithm to look them up again. In other words, it prevents trajectory cycles.

However, the cost of storing all the visited solutions is too high. So, only the T (size of tabu list) last solutions keep on list. The tabu list acts as a first-in-first-out (FIFO): when its capacity is reached, the oldest solution gives place to a new one.

Tabu Search keeps in memory the best of all visited solutions apart of being the current. So, even if it "pass" the best solution over the execution, it still will be available.

The algorithm is given in Algorithm 1. The 8^{st} step of the algorithm refers to find the best solution in the neighborhood. In this step the cost of each solution if evaluated by a k-NN classifier. So, the solution whose number of correct classified patterns is choose to be the current solution, in this work.

Algorithm 1 Tabu Search

1: s_0: initial solution
2: s_{best}: best solution
3: I: number of iterations
4: V: set of neighbors solutions
5: Insert s_0 in tabu list
6: **for** $i = 0$ to $i = I - 1$ **do**
7: Generate V neighbors solutions
8: Find the best $s' \in V$
9: **if** s' is not in tabu list **then**
10: $s_{i+1} \leftarrow s'$
11: Update tabu list
12: Update s_{best}
13: **end if**
14: $i = i + 1$
15: **end for**
16: Return s_{best}

1) Encoding solution: the encoding solution proposed by [4] consists of 3 parts as shown in Figure 1. The first consists of real weight values associated to each feature. These weights will be modified through *feature weighting*. The second part, or the binary part, consists of 0 or 1 values. And, finally, the third part consists of the *k* value extracted from *k-NN*.

2) Cost function: the cost function used in this work is the total number of correct classified patterns as shown in equation 4. So, we have as objective function to maximize this number.

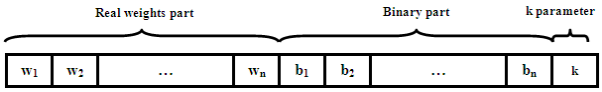

Fig. 1. Encoding vector solution

$$Cost = \sum_{i=1}^{n} C_i \tag{4}$$

where n is the number of classes and C_i is the number of correct classified patterns in each class. It is important to emphasize that *k-NN* was used to evaluate the cost in this work.

3) Neighborhood: there is a step in the *Tabu Search* algorithm that consists in the neighborhood creation. In this approach, a total of *M* x *N* + *P* new solutions are created at each iteration. In which *M* x *N* new solutions, or neighbors, are created by assigning *M* random weights to *N* different features. *P* new solutions are generated from turning on/off the bit of the encoding second part. In this approach, turning a bit off means deleting a feature. Each neighbor solution is evaluated and the best one is chosen to be the next current solution.

In order to clarify, creation through examples will be depicted the neighborhood creation through examples. The *M* parameter indicates that we want to add *M* random values to one specific feature. For example, if *M* is set to 2, it implies in adding 2 random values to the selected feature f_x. By doing so, we have created initially 2 new neighbors from the matrix solution.

We can also determine which feature will be modified through the *N* parameter. We can, as example, set *N* as 2, it indicates that 2 random features will be selected. Lets assume they are f_2 and f_5. Until this point we have a total of 4 solutions created. The first solution adds a random number to the feature f_2, the second adds a different random value to f_2, the third and fourth do the same with feature f_5.

Finally, extra solutions comes from inverting the current value of *P* features (see the binary part of Figure 1). To complete our example, lets assume *P* as 3. So, 3 solutions will be created, each one with a random selected feature turned on/off. As could be seen, a total of *M* x *N* + *P* neighbors.

As have been explained above, random numbers in the interval of [-1 1] are added to some features. In this work, after creation of the neighbors (7_th step in Algorithm 1), they are normalized by feature in order to avoid disparate intervals.

4) Termination criteria: the algorithm stops if the fixed number of iterations is reached or if after some iterations the objective function does not change.

5) Tabu rule: if a solution feature has been recently modified and it is in the tabu list, it is then prohibited. It means that, to accept an current solution, that feature could not have been modified before.

B. SA/k-NN

The method *Simulated Annealing* consists in, at each iteration, generate one new solution from the current solution. When a new solution is created, its cost is evaluated to decide if it can be accepted as current. If the cost is less than the current, it is immediately accepted. Otherwise, it can be accepted with a certain probability, known as the *Metropolis Criteria* [11]. According to this criteria, a random number δ between 0 and 1 is generated. If $\delta \leq e^{(-\Delta C/t)}$, then the solution is accepted. Where C is the cost function variation and t is an parameter called temperature. Cooling schemas are responsible to define an initial temperature value, as well as, a rule to iteratively decreases this value. In this work the cooling schema adopted was the *Geometric* one [14].

Algorithm 2 Simulated Annealing

1: s_0: initial solution
2: I: number of iterations
3: **for** $i = 0$ to $i = I - 1$ **do**
4: Generate solution s'
5: **if** $Cost(s') \leq Cost(s_i)$) **then**
6: $s_{i+1} \leftarrow s'$
7: **else**
8: **if** $random \geq e^{(-[Cost(s')-Cost(s_i)]/t_{i+1})}$ **then**
9: $s_{i+1} \leftarrow s'$
10: **end if**
11: **end if**
12: $i = i + 1$
13: Update temperature
14: **end for**
15: Return s_i

The encoding solution, neighborhood generation, termination criteria and tabu rules of acceptance are the same described in the Section II-A.

The *Simulated Annealing* algorithm is given in Algorithm 2.

III. Experiments

This section intends to describe the experiments performed within the combination of a simple *Adaptive Distance*[8] and two hybrid algorithms already described in previous section: TS/k-NN and SA/k-NN. It is important to emphasize that this combination (with the *Adaptive Distance*) is not part of the original experiments presented in [4]. The mentioned work have performed its research using the *Euclidean Distance*. Another point to consider is the use of *Simulated Annealing* instead of *Tabu Search* in the hybrid SA/k-NN model proposed here, a cheaper solution in terms of processing time. We consider it an interesting alternative to TS approach.

The aim of this section is to show that through the *Adaptive Distance* use, it is possible to improve the performance

of TS/k-NN and SA/k-NN, focusing in the particular case of feature sets selection and weighting. More details about how the experiments were carried out as well as the datasets description will be shown throughout this section.

A. Combined Methods

Six methods, called here "combinations", were compared in the following experiments, they are:

1) ***k-NN* + Euclidean Distance***:* the well-known *K-Nearest Neighbor* classifier which uses Euclidean Distance as similarity measure. This is a traditional combination in the literature.

2) ***k-NN* + Adaptive Distance***:* in this approach, the *Euclidean Distance* has been replaced by the *Adaptive* one proposed in [8].

3) ***SA/k-NN* + Euclidean Distance***:* here, the Euclidean Distance is used in combination with the hybrid method constituted by the k-NN classifier and the global optimization technique *Simulated Annealing* described in Section II-B.

4) ***SA/k-NN* + Adaptive Distance***:* the same hybrid method mentioned above, however with the *Adaptive Distance* in place of the *Euclidean*.

5) ***TS/k-NN* + Euclidean Distance***:* original method proposed in [4].

6) ***TS/k-NN* + Adaptive Distance***:* the hybrid model proposed in [4] with the particularity of the *Adaptive Distance* use.

B. Datasets

The experimental results were performed on several benchmarks datasets available from the UCI Machine Learning Repository [12] and the Statlog Project [13]. Table I shows more information about the used datasets.

Data were randomly divided into training set and testing set. The training set consists of 70% of the total number of prototypes and the test set consists in 30% of the data. Hould-out is used in the training phase. To compute the results, 5 runs of each algorithm were simulated and their classification accuracy were used to obtain the final average rate results. The experiments done with k-NN are the only exception, in this case only the number of iterations constitutes the average results.

TABLE I
DATASETS DESCRIPTION

Dataset	#Prototypes	#Features	#Classes
Liver	345	6	2
Pima	768	8	2
Heart	270	13	2
Australian	690	14	2
Ionosphere	351	34	2
Sonar	208	60	2

C. Settings

There were run 5 simulations of the k-NN method varying the number of k, for each database. The k values experimented are in the interval from 1 to 5. The one which has presented best results was adopted. As mentioned above, the final rate is the average of results collected in each iteration. In this work, k-NN iterations number was set to 100.

The method TS/k-NN has its parameters described here as well. The number of iterations is 100 for all datasets. In addition to it, were performed 5 trial runs, in other words, each run has the average result of 100 iterations and the final result is the average of the 5 trials.

One important parameter of *Tabu Search* is the length T of tabu list. Some criterion are suggested in the literature in order to help in the choice of T. The tabu list size was calculated, in this work, by:

$$T = ceil(\sqrt{F}) \tag{5}$$

where T and F are, respectively, the size of tabu list and the number of features.

Other parameters introduced by the model proposed in [4] are M, N and P. Just to review, M x N + P neighbors are generated at each TS/k-NN iteration. Where M are the number of patterns, N the number of feature weights to be modified and, finally, P is the number of solutions generated by turning on/off some features. These parameters values were set according to [4]. In that work, the value of P was obtained through equation (5), M was set to 10 and N set to 2.

The same values of M, N and P used in TS/k-NN, were used in the SA/k-NN model. In the *Simulated Annealing* rule, we have seen that even if a new solution has inferior performance when compared to the current solution, it could be accepted to avoid local minima. Follows the acceptance criterion: one random value is tested to be less than a prior defined probability, if the test succeeds, the new solution replaces the current one. Otherwise, it is discarded. In this work, the prior defined probability was determined by:

$$p = \frac{exp(-(E_n - E_a))}{t} \tag{6}$$

where p is the probability, t is the temperature, E_a is the classification error of the current solution and E_n the error of the new solution.

The choice of a cooling schema is another configuration required by the *Simulated Annealing* method. A cooling schema has to define an initial temperature as well as the rule for updating its value. One common cooling schema is the *Geometric* one [14], where the new temperature is computed by the product of the current temperature with a reduction factor. In this study, the initial temperature was set to 1 and the reduction factor set to 0.9. The temperature is updated following this schema at every 10-multiple iteration.

IV. RESULTS AND DISCUSSION

Table II presents the accuracy rate in % for the algorithms k-NN, SA/k-NN and TS/k-NN when using *Euclidean Distance*. Notice that the results of the proposed model SA/k-NN outperforms the simple k-NN results to all datasets presented. Furthermore, the TS/k-NN performance is the highest of all.

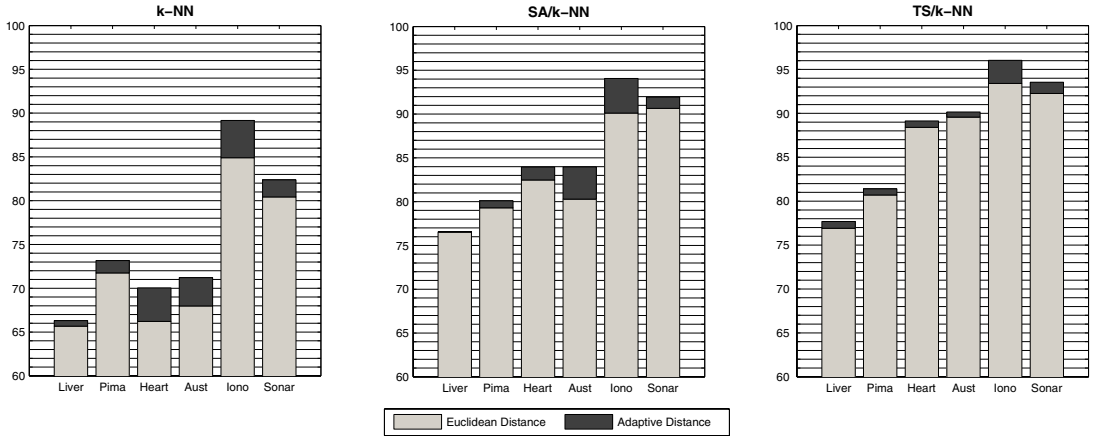

Fig. 2. k-NN, TS/k-NN and SA/k-NN accuracy classification (in %) for all datasets analyzed. Both distance approaches are compared here, the Euclidean and the Adaptive.

Another point to observe is the number of features resultant. The k-NN does not reduces the features number, so we have to consider the total number of features in this case.

TABLE II
ACCURACY CLASSIFICATION RATE (IN %) USING *Euclidean Distance*

	k-NN			SA/k-NN				TS/k-NN			
	$\bar{x}$		k	$\bar{x}$		f	k	$\bar{x}$		f	k
Liver	65.67	±1.55	5	76.54	±1.54	6	9	**76.92**	±0.59	3	1
Pima	71.75	±1.45	5	79.30	±0.86	7	9	**80.70**	±0.71	7	9
Heart	66.21	±2.35	3	82.47	±2.58	9	9	**88.40**	±1.20	5	7
Australian	67.98	±1.65	5	80.29	±2.59	14	7	**89.57**	±0.53	6	9
Ionosphere	84.91	±1.75	1	90.11	±1.10	30	1	**93.41**	±0.78	26	1
Sonar	80.43	±2.63	1	90.65	±1.55	53	1	**92.26**	±0.67	45	1

TABLE III
ACCURACY CLASSIFICATION RATE (IN %) USING *Adaptive Distance*

	k-NN			SA/k-NN				TS/k-NN			
	$\bar{x}$		k	$\bar{x}$		f	k	$\bar{x}$		f	k
Liver	66.31	±1.77	5	76.54	±1.21	5	9	**77.69**	±0.53	3	1
Pima	73.16	±1.35	5	**80.09**	±0.74	8	5	80.00	±0.41	5	9
Heart	70.04	±2.17	3	83.95	±2.58	13	5	**89.14**	±0.28	6	9
Australian	71.21	±1.08	5	83.96	±2.43	12	5	**88.99**	±0.34	7	1
Ionosphere	89.16	±1.37	1	94.07	±0.49	25	1	**96.04**	±0.30	20	7
Sonar	82.37	±2.03	1	91.94	±0.81	52	1	**93.55**	±0.57	46	3

According to the table, the use of hybrid SA/k-NN and TS/k-NN reduced significantly the datasets number of features. The only exception happened to SA/k-NN which does not minimize the number of features in Liver and Australian bases. Figure 3 clarify this argument. It shows the resultant number of features for all algorithms (including a comparison between distances) and all datasets.

The results showing the accuracy classification through the *Adaptive Distance* are presented in Table III. Observing the information demonstrated in this table, the hybrid optimization algorithms have higher performance than k-NN, even with the use of a new distance. One more time, TS/k-NN performs better than the proposed SA/k-NN except for Pima database.

Also, the number of features was significantly reduced when compared with the original sets size (see Figure 3).

However, one important point to pay attention in is: the gain obtained with SA/k-NN when compared with the simple k-NN is greater than the gain between SA/k-NN and TS/k-NN. If we think it over the SA performance in terms of processing time, it is less than the TS method. So we consider the SA/k-NN a good option which gives us good accuracy rate and smaller feature sets.

In this way, we could see that the use of hybrid techniques reduces significantly the number of features in all databases analyzed. This fact instigates more study in the way of optimization techniques to Features Selection and Weighting.

Furthermore, when comparing both Tables II and III we have very clear that the use of *Adaptive Distance* gives us much better results than simply the utilization of *Euclidean Distance*. Figure 2 contrasts these information. Through this Figure it is easy to perceive that for all approaches (k-NN, SA/k-NN and TS/k-NN) *Adaptive Distance* has got superior rates. The eminence of the algorithms combined with *Adaptive Distance* is evident also while comparing the features subsets size.

V. CONCLUSION

In this paper, a new hybrid Simulated Annealing/k-Nearest Neighbor method is proposed to simultaneous feature selection and weighting based on [4]. This method outperformed the results gotten by with k-NN for all datasets analyzed. It includes both: the smaller number of features in the resultant subset and the classification accuracy.

Another important contribution from this analysis is the combination of the *Adaptive Distance* with all the algorithms implemented, they are: k-NN, TS/k-NN and SA/k-NN. These combinations were compared with the ones composed with

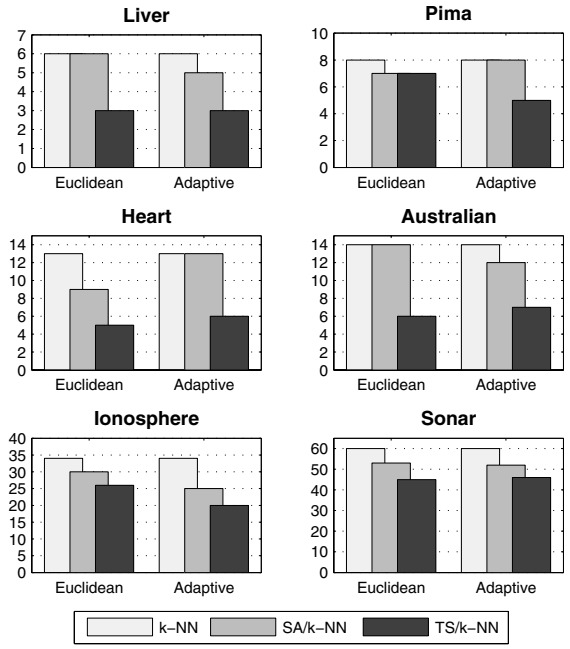

Fig. 3. The Y axis presents the resultant number of features obtained with both *Euclidean* and *Adaptive* distances combined to k-NN, SA/k-NN and TS/k-NN.

the *Euclidean Distance*. Was observed the superiority of the *Adaptive Distance* usage over the *Euclidean* in all datasets.

Furthermore, were detected very common problems in the literature (feature selection and feature weighting). Thus we aim at contributing to this field of study.

We consider that the methodology proposed by [4] has been more consolidate since it was perfectly adapted for Simulated Annealing and we think it could be used in many others successful combinations with optimization methods.

REFERENCES

[1] F. Glover,"Future paths for integer programming and links to artificial intelligence". Computers and Operation Research, vol. 13, pp. 533-549. 1986.

[2] P. Hansen, "The steepest ascent mildest descent heuristic for combinatorial programming." Conf. on Numerical Methods in Combinatorial Optimisation. 1986.

[3] S. KirkPatrick, C. D. Gellat JR. and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671-680. 1983.

[4] M. A. Tahir, A. Bouridane and F. Korugollu, "Simultaneous feature selection and feature weighting using Hybrid Tabu Search/K-nearest neighbor classifier," *Pattern Recognition Letters,* vol. 28, pp. 438–446. 2007.

[5] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. Inform. Theory,* vol. IT-13, no. 1, pp. 21–27. 1967.

[6] M. L. Raymer et al., "Dimensionality reduction using Genetic algorithms," *IEEE Trans. Evolution. Comput.,* vol. 4(2), pp. 164–171. 2000.

[7] H. Zhang and G. Sun, "Features Selection using Tabu Search method," *Pattern Recognition Letters,* vol. 35, pp. 701–711. 2002.

[8] J. Wang, P. Neskovic andL. N. Cooper, "Improving nearest neighbor rule with a simple adaptive distance measure," *Pattern Recognition Letters,* vol. 28, pp. 207–213. 2007.

[9] A. K. Jain, J. Mao and K. M. Mohiuddin. "Artificial Neural Networks: A Tutorial," IEEE Computer, pp. 31–44. 1996.

[10] X. Yao, "Evolving artificial neural netwoks". *Proc.* of the IEEE. pp. 1423 – 1447. September 1999.

[11] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, "Equation of state calculations by fast computing machines", Journal of Chem. Phys., vol. 21, no. 6, pp. 1087–1092. 1953.

[12] A. Asuncion and D. J. Newman. "UCI Machine Learning Repository" [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

[13] D. Michie, D.J. Spiegelhalter and C.C. Taylor, "Machine Learning, Neural and Statistical Classification," Ellis Horwood. 1994.

[14] D. T. Pham and D. Karaboga, "Introduction , Intelligent Optimisation Techniques (Edited by D. T. Pham and D. Karaboga), pp. 1–50, Springer-Verlag. 2000.