Building Nearest Prototype Classifiers Using a Michigan Approach PSO

Alejandro Cervantes* and Inés Galván† and Pedro Isasi‡
Department of Computer Science
Universidad Carlos III de Madrid, 28911 Legan´es, Madrid, Spain
Tel: +34-91-624-8879
E-mail: *alejandro.cervantes@uc3m.es †ines.galvan@uc3m.es ‡pedro.isasi@uc3m.es

***Abstract*— This paper presents an application of Particle Swarm Optimization (PSO) to continuous classification problems, using a Michigan approach. In this work, PSO is used to process training data to find a reduced set of prototypes to be used to classify the patterns, maintaining or increasing the accuracy of the Nearest Neighbor classifiers. The Michigan approach PSO represents each prototype by a particle and uses modified movement rules with particle competition and cooperation that ensure particle diversity. The result is that the particles are able to recognize clusters, find decision boundaries and achieve stable situations that also retain adaptation potential. The proposed method is tested both with artificial problems and with three real benchmark problems with quite promising results.**

I. Introduction

The Particle Swarm Optimization algorithm (described in [1]) is a biologically-inspired algorithm motivated by a social analogy. The algorithm is based on a set of potential solutions which evolves to find the global optimum of a real-valued function (fitness function) defined in a given space (search space).

The PSO algorithm uses a population of particles which move in a multidimensional space that represents the space of solutions for the problem. Particles have memory, thus retain part of their previous state. There is no restriction for particles to share the same point in the search space, but in any case their individuality is preserved.

The basic PSO uses a real-valued multidimensional space as search space, and evolves the position of each particle in that space using (1) and (2).

$$v_{id}^{t+1} = \chi(w \cdot v_{id}^{t} + c_1 \cdot \psi_1 \cdot (p_{id}^{t} - x_{id}^{t}) + c_2 \cdot \psi_2 \cdot (p_{gd}^{t} - x_{id}^{t})) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

Where the meanings of symbols are: v_{id}^{t}, component in dimension d of the i^{th} particle velocity in iteration t; x_{id}^{t}, same for the particle position; c_1 ,c_2:, constant weight factors; p_i, best position achieved so far by particle i; p_g, best position found by the neighbors of particle i; ψ_1 ,ψ_2, Random factors in the [0,1] interval; w, inertia weight; and χ, constriction factor.

The neighborhood of the particle may either be composed of the whole swarm ("gbest" topology) or only a subset of the swarm ("lbest" topologies). Also, some versions of PSO use dynamic neighborhoods, where the relationship between particles changes over time: Suganthan [2] proposed a swarm with a local neighborhood whose size was gradually increased; Brits [3] justifies the introduction of topological neighborhoods when searching for multiple solutions to multimodal problems using niching techniques. A dynamic neighborhood has been proposed in [4] for multi-objective optimization.

There are also some good theoretical studies of PSO [5][6][7] which address the topics of convergence, parameter selection and and trajectory analysis.

The purpose of this paper is to use the continuous PSO for the selection of prototypes in nearest neighbor learning. Not much work has been done concerning PSO with classification problems. In [8], PSO is used to extract induction rules to classify data; the standard PSO algorithm is run several times, extracting a single rule each time and using only unclassified patterns for the subsequent iterations. In [9][10], induction rules are encoded using the binary version of PSO and both a single and a iterated version of the algorithm.

In this work we use Nearest Neighbor (NN or 1-NN) classification. NN is a lazy learning method where the class assigned to a pattern is the class of the nearest pattern known to the system, measured in terms of a distance defined on the feature (attribute) space.

In 1-NN with Euclidean distance, the regions associated to a pattern (called Voronoi regions) are delimited by linear borders. This can be modified if the distance is changed, or if a K-NN strategy is used, where the class assigned to a pattern is the most frequent class among its K nearest neighbors. K-NN strategy can approximate non-linear borders of the Voronoi regions.

Both computational reasons and presence of noise in data led to the development of techniques that reduce the number of patterns evaluated without increasing the classification error. These methods calculate a reduced set of prototypes, that are the only ones used for classification. In instance selection methods [11] prototypes are a subset of the pattern set, but this is not true for all the prototype methods [12].

In this paper, we use PSO to find a collection of prototypes that do not belong to the problem data, but represent the training pattern set and can classify the patterns according to the class of the nearest prototype in the collection.

In the standard PSO, we would have decided the number of prototypes to use and we would have encoded the positions of

1-4244-0708-7/07/$20.00 ©2007 IEEE

all the prototypes and their classes in each particle. However, we want to validate a different approach that we call the Michigan Approach; this term is borrowed from the area of genetic classifier systems ([13][14]).

The Michigan approach was applied to PSO in [10], where the binary version of PSO was used to discover a set of induction rules for discrete classification problems.

In this approach, each particle represents a single prototype. The solution is a collection of particles instead of a single particle. For this approach to work, the standard PSO has to be modified to ensure that the particles do not converge, but instead the swarm evolves towards a configuration where each particle may be considered part of a collective solution. The advantages of the Michigan approach versus the conventional PSO approach are: a) scalability and computational cost, as particles have much lower dimension; and b) flexibility and reduced assumptions, as the number of prototypes in the solution is not fixed.

This paper is organized as follows: section 2 shows how the solution to the classification problem is encoded in the particles of the swarm; section 3 details the modifications made to the original PSO algorithm to implement the Michigan approach and adapt it to classification problems; section 4 describes the experimental setting and results of experimentation; finally, section 5 discusses our conclusions and future work related to the present study.

II. PSO Algorithm Encoding Prototypes

In the Michigan-approach PSO we propose, each particle represents a potential prototype to be used to classify the patterns using the nearest neighbor rule. The particle position will be interpreted as the position of a prototype. Each particle has also a class; this class does not evolve following the PSO rules, but remains fixed for each particle since its creation.

For a given problem, a particle swarm is created with the same dimension of the problem (number of attributes) and with an equal share of particles of each of the classes present in the training set. The swarm runs until a stopping criterion is met, and then the resulting particles positions are interpreted as the positions of the prototypes of the nearest neighbor classifier.

In nearest neighbor classifiers, the distance used to determine closeness has to be defined. In our work, we have consistently used the Euclidean distance.

This selection means that the Voronoi regions associated to the prototypes will have lineal boundaries, which may or may not be adequate to the problem.

III. PSO Algorithm Modifications

A. Sociality Terms for the Michigan PSO

For the success of the Michigan approach, the algorithm has to avoid convergence towards a single high-fitness particle, because this particle can only represent a very limited solution (only one prototype).

In the standard PSO, the attractive sociality term in (1) tries to move particles close to positions with good fitness. To change this behavior, in the Michigan approach we introduce several modifications that are based on using the particle class to define its neighborhood and divide the swarm into competing and non-competing particles:

- For each particle of class C_i, non-competing particles are all the particles of classes $C_j \neq C_i$ that classify at least one pattern of class C_i.
- For each particle of class C_i, competing particles are all the particles of class C_i that classify at least one pattern of that class (C_i).

When the movement for each particle is calculated, that particle is both:

1) Attracted by the closest (in terms of Euclidean distance) non-competing particle in the swarm, which becomes the "attraction center" for the movement. In this way, non-competing particles guide the search for the patterns of different class.
2) Repelled by the closest competing particle in the swarm, which becomes the "repulsion center" for the movement. In this way, competing particles retain diversity and push each other to find new patterns of their class in different areas of the search space.

The rules above achieve the following results:

- A particle is only attracted by those particles that misclassify patterns. In that case, it is only affected by the closest particle that meets that criterion. Hence, particles may cluster around different points of the search space instead of converging towards a single position.
- The way repulsion is defined means that, when several particles of a class are close to a cluster of patterns of that class, instead of converging towards the cluster center (or elsewhere the position that maximizes the fitness function), they can stay near the border of the cluster. This allows more accurate classification, as particles are able to locate the decision frontier where classification is harder.

Other authors have already used the idea of repulsion in PSO in different ways. For instance, in [15] and [16], repulsion is used to increase population diversity in the standard PSO and allow the swarm to dynamically adapt to change. In [9] a repulsive force is introduced to achieve particle diversification in a binary PSO.

B. Social Adaptability Factor

In PSO, the social term is an attraction towards the best particle in the neighborhood. In our version of PSO we use the closest particle to select which is the particle that exerts attraction or repulsion. In preliminary experimentation, this meant that particles with very good fitness moved far from their position due to the effect of particles with much lesser fitness. This does not happen in standard PSO because the best particle in the neighborhood is never influenced by the rest.

To correct this effect we have generalized the influence of fitness in the sociality terms by introducing a new term in the PSO equations, called "Social Adaptability Factor" (S_f),

that depends inversely on the fitness value of the particle. In particular, we have chosen plainly the expression in (3).

$$Sf_i = 1/(\text{Local Fitness}_i + 1.0) \tag{3}$$

With the current Local Fitness function this value is in the range $[0.25, 1]$.

This equation assumes Local Fitness is greater than -1.0. This factor also contributes to the swarm stability because, as particles evolve towards better positions, they are less likely to walk away from those positions, as the individual component of the particle velocity is not affected by this factor (see 4).

C. Modified PSO pseudo code and movement

The factors above replace the standard PSO implementation. The overall procedure can be described as follows:

1) Load training patterns
2) Initialize swarm; dimension equals number of attributes.
3) Insert N particles of each class in the training patterns.
4) Until max. number of iterations reached or success rate is 100%,
   a) Calculate which particles are in the competing and non-competing sets of particles for every class.
   b) For each particle,
      i) Calculate Local Fitness.
      ii) Calculate Social Adaptability Factor.
      iii) Find the closest particle in the non-competing set for the particle class (attraction center).
      iv) Find the closest particle in the competing set for the particle class (repulsion center).
      v) Calculate the particle's next position based on its previous velocity, previous best position, attraction center and repulsion center.
   c) Move the particles
   d) Assign classes to the patterns in the training set using the nearest particle.
   e) Evaluate the swarm classifi cation success
   f) If the swarm gives the best success so far, record the particles' current positions as "current best swarm".
5) Delete, from the best swarm found so far, the particles that can be removed without a reduction in the classifi cation success value.
6) Evaluate the swarm classifi cation success over the validation set and report result.

Note that a reduction algorithm is applied after the swarm reaches its maximum number of iterations. Particles are removed one at a time, starting with the particle with the worse local fitness value, if this action does not reduce the swarm classification success rating over the training set. The "clean" solution is then evaluated using the validation set.

D. Modified PSO Equation

In order to take into account the modifications previously described, the equation that determines the velocity at each iteration in the basic PSO algorithm (1) has to be modified. In our work the velocity change is given by (4).

$$\begin{aligned} v_{id}^{t+1} = \chi(w \cdot v_{id}^t + c_1 \cdot \psi_1 \cdot (p_{id}^t - x_{id}^t) + \\ c_2 \cdot \psi_2 \cdot sign(a_{gd}^t - x_{id}^t) \cdot Sf_i + \\ c_3 \cdot \psi_3 \cdot sign(x_{id}^t - r_{gd}^t) \cdot Sf_i) \end{aligned} \tag{4}$$

Where the meanings of the new symbols are: c_3, new constant repulsion weight factor; a_g, attraction center for particle i ; r_g, repulsion center for particle i ; Sf_i, Social Adaptability Factor, inversely dependent on the particle fitness; and ψ_3, random factor in the [0,1] interval.

Note that the repulsion term is weighted by a random factor (ψ_3) and a fixed weight (c_3). This weight becomes a new parameter for the algorithm.

If a_g or r_g don't exist their respective terms are ignored.

E. Local Fitness Function

In the Michigan approach, a local fitness has to be calculated for each particle. For this purpose, we use (5).

$$\text{Local Fitness} = \begin{cases} G_f = \sum_{\{g\}} \frac{1}{d_{g,i}+1.0} \\ B_f = \sum_{\{b\}} \frac{1}{d_{b,i}+1.0} \\ \frac{G_f}{Total} + 2.0 & \text{if } \{b\} = \emptyset \\ \\ \frac{G_f - B_f}{G_f + B_f} + 1.0 & \text{otherwise} \end{cases} \tag{5}$$

Where $\{g\}$ is the set of patterns of the same class classified by the particle; $\{b\}$ is the set of patterns of different class classified by the particle; $d_{g,i}, d_{b,i}$ are the Euclidean distances between the patterns and the particle; and $Total$ is the number of patterns in the training set,

This fitness function gives higher values (greater than $+2.0$) to the particles that only classify patterns whose class matches the class of the particle, and assigns values in the range $[0.0, +2.0]$ to particles that classify some patterns of a wrong class.

By using the distance between the pattern and the particle we give higher fitness to particles close to the patterns they classify, so they can move closer to the area where the patterns are located. This tendency may be compensated by the sociality terms.

In the lowest range, the particles only take into account local information (the proportion of good to bad classifications made by itself). In the highest range, the particle fitness uses some global information (the total number of patterns to be classified), to be able to rank the fitness of particles with a 100% accuracy (particles for which $\{b\} = \emptyset$).

For particles that classify no patterns, local fitness is 0.

F. Global Swarm Evaluation

The local fitness function, defined above, is used for the particles movement. However, to evaluate the goodness of the swarm as a classifier system, we use the classification success rate (6).

$$\text{Swarm Evaluation} = \frac{Good\ classifications}{Total\ patterns} \cdot 100 \tag{6}$$

Given the nearest neighbor criterion for classification, unclassified patterns are not possible, as every pattern is assigned the class of the nearest prototype.

TABLE I
PROBLEMS USED IN THE EXPERIMENTS

Name	Dim	Classes	Validation
Clusters	2	2	Train set only
Diagonal	2	2	Train and validation sets
Diabetes	8	2	10-fold cross validation
Bupa	6	2	10-fold cross validation
Glass	9	6	10-fold cross validation

TABLE II
PARAMETERS USED FOR EACH OF THE PROBLEMS

Problem	Iterations	Particles	w	χ	c_1	c_2	c_3
Cluster	100	20	0.1	1.0	0.5	0.15	0.05
Diagonal	300	20	0.1	1.0	0.5	0.15	0.05
Diabetes	300	20	0.1	1.0	1.0	1.0	0.25
Bupa	300	20	0.1	1.0	1.0	1.0	0.25
Glass	300	60	0.1	1.0	1.0	1.0	0.25

The system stores the best swarm evaluation obtained when performing classification on the training set. This function is also used to evaluate the final best swarm success rate over the validation set.

IV. EXPERIMENTATION

A. Problems' Description

We perform experimentation on the problems summarized in Table I. The first two are artificial problems and they are used to understand the new algorithm's properties; the rest are well-known real problems taken from the UCI collection, used for comparison with other classification algorithms.

- **Cluster Problem.** This is a very simple problem with five different clusters, randomly generated and clearly separated. The problem has 75 training patterns with two attributes and two classes.
- **Diagonal Problem.** This is a bi-dimensional problem that is very simple for linear classifiers. We generate 500 random training patterns and 1500 validation patterns with coordinates in the $[0, 1]$ range. Patterns where $x > y$ are assigned class 1, and the rest are assigned class 0.
- **UCI problems** The Pima Indians Diabetes Database, Bupa Liver Disorder data, and Glass Classification data all come from the UCI data repository. The Diabetes database includes 768 patterns, of which 500 are from class 0 and 268 from class 1 (representing cases tested positive for diabetes). The Bupa Liver Disorder database has 345 instances, 200 of class 1 and 145 of class 0. The Glass Identification data is composed of 214 instances of six different classes. Success rates from several classification algorithms over these problems can be found in [12].

B. Parameter Selection

In all the experiments, the attributes' values are normalized to the $[0, 1]$ interval. This means that the significant part of the search space is restricted to the corresponding hypercube, and the values used for the parameters correspond to this size of the search space. Velocity was clamped to the interval $[-1.0, +1.0]$.

The values of the swarm parameters were selected after some preliminary experimentation as shown in Table II, that also shows the number of experiments to be performed ($10x10$ means ten cross-validation experiments with 10 folds each).

We have used 10 particles per class for all the experiments.

For the artificial problems, we used small values for the PSO parameters (c_1, c_2, and c_3) which lead to slower evolution but more accurate results.

For the real problems we used the same parameters for each of the problems, with no further tuning, except that we experimentally found that a small value of w lead to better results in all cases. The number of iterations (300) was selected after checking that number was roughly equal to double the average iteration in which the best result was achieved.

After the given number of iterations is reached, the best solution found for the training set is cleaned (useless particles are removed) and the resulting swarm is used to classify the validation set.

We performed 100 runs of the algorithm for the Cluster problem and Diagonal problems, and 10 runs of the problems that use 10-fold cross validation, which gives a total of 100 runs over each.

The success rate is averaged over the number of runs; we also provide the best result over the set of experiments. Wherever cross-validation was used we provide the best of the 10 different runs.

C. Experimental Results

The first two experiments are used to show the behavior of the swarm with the proposed modifications. This can be easily shown plotting some of the solutions found by the swarm for these problems. For both the "Cluster" and "Diagonal" problem, sample figures are provided.

In Fig.1 (left) we represent one of the solutions for the "Cluster" problem. It shows how the attraction and repulsion rules make the particles group around the different clusters but also how the particles of the same class remain separated (due to the repulsion force). Once the swarm reaches this kind of configuration, attraction forces are null (no particle classifies patterns of the wrong class), and repulsion forces are compensated if they move the particles towards positions of lower local fitness, so the particles remain almost stable in positions near the optimum. In this problem, the swarm easily finds and separates the clusters every time.

In the figure we can also see that a particle can be placed in an nonproductive area of the space. As the particle does not classify any pattern, its local fitness is 0. This particle would be removed by the removal algorithm.

To the right (in Fig.1) we represent one of the solutions to the "Diagonal" problem. It shows how the particles move

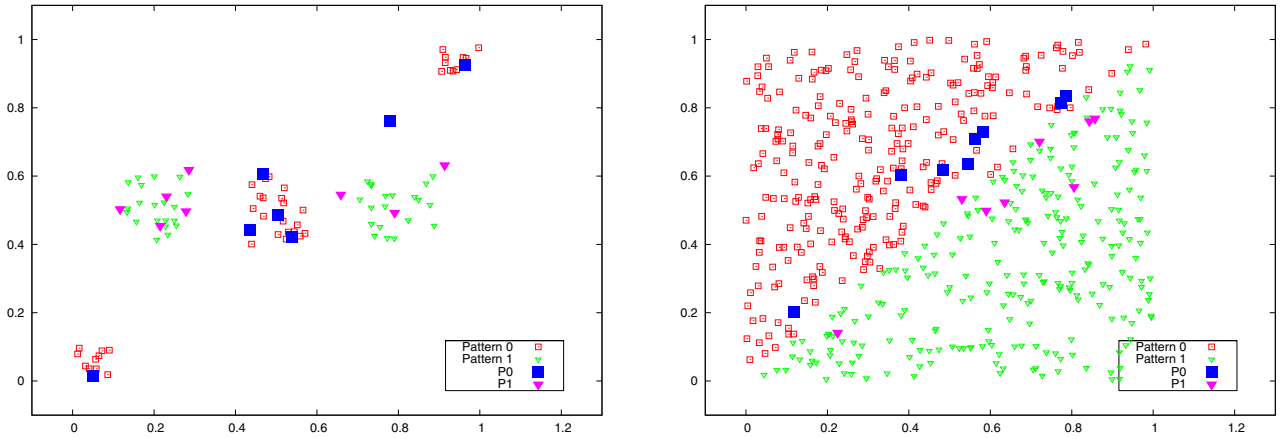

Fig. 1. Sample solutions for the problems "Cluster" (left) and "Diagonal" (right), before removal of useless particles. Patterns are outlined, particles are solid.

towards the diagonal which is the boundary between the classes. Also, if the sequence of the particles' positions is observed, the particles are sometimes grouped in pairs (one from each class) like the pair in the lower left part of the figure.

For this problem, the learning bias of the algorithm is not favoring the accuracy of the solution. A very good solution involves just one prototype of each class, located at the central position of each area; the solutions found by our method are worse than that trivial solution due to the fact that it explicitly searches near the classes boundary (the diagonal) where a solution is more difficult to find.

In Table III we summarize the results for all the experiments. In that table, "Valid. Set" is the best classification rate on the validation set (training set for the "Cluster" problem), averaged over all the runs of the experiment and all the folds when 10-fold validation is used; "Average Iter." is the average number of iterations needed to find the best result for each individual run; "Best Exp." is the best success rate from all the runs of each experiment; and "Prototypes" is the average number of prototypes kept after cleaning the best swarm.

TABLE III
EXPERIMENT RESULTS FOR THE PSO WITH MICHIGAN APPROACH, RESULTS IN %

Problem	Train Set	Validation Set	Average Iter.	Best Exper.	Proto-types
Cluster	100	N/A	18.9	100	12.21
Diagonal	94.88	94.05± 0.37	161.1	98.27	13.37
Diabetes	74.66	74.14± 0.59	155.7	75.95	10.47
Bupa	64.76	63.29± 0.85	145.9	65.15	10.89
Glass	82.76	82.04± 1.01	177.0	84.33	10.3

The results of our PSO (MichPSO) on all the UCI data sets is shown in Table IV; in this table we compare our results (MichPSO) with the results of our own tests with other algorithms, using WEKA with the default parameters for each of the algorithms. In the table, IBK(k=1) means plain NN, and IBK(k=3), 3-NN classification.

TABLE IV
EXPERIMENT RESULTS (SUCCESS RATE) FOR THE UCI DATA SETS, USING SEVERAL CLASSIFIERS

Domain	Mich PSO	PART	Naive Bayes	IBK (K=1)	IBK (K=3)	SMO
Diabetes	74.14	74.18	75.69	70.44	73.88	76.63
Bupa	63.29	63.08	55.63	62.90	63.16	57.95
Glass	82.04	73.79	47.25	71.99	70.58	57.10

The success rate in terms of accuracy for MichPSO is comparable or clearly better than the result of IBK with K=1 in all cases. This means that the patterns are correctly represented with the particles in the solution swarm.

For the Diabetes data set, it is known that higher K values in K-NN classification, and non-Euclidean distances achieve better results than 1-NN. The result of MichPSO also as good as with 3-NN classification. The best result found in literature using nearest neighbor classifiers is obtained with K=23 and Manhattan distance [17] (76.7 $\pm$ 4.0 %) .

For the Bupa data set, the algorithm performs basically as well as all the other algorithms, and again bettern than plain 1-NN classification.

The algorithm significantly improved the accuracy of all the rest in the Glass problem. Good performance on this problem is possible due to the fact that non-euclidean distances seem not to improve the results of Euclidean distance in this case (see [12]). Our algorithm seems to significantly improve the results of all the other classifiers, being to our knowledge the best result over this data set.

V. CONCLUSIONS

The purpose of this paper is to study the performance of PSO in classification problems with continuous attributes. We will use PSO to determine a set of prototypes that represent the training patterns and can be used as a classifier using the nearest neighbor (1-NN) rule.

A straightforward encoding of a set of prototypes in each particle would produce a search space of high dimension that might hinder the swarm success. For this reason, we propose a Michigan Approach PSO, in which each particle represents a single prototype (not a set of prototypes).

For this approach to work, the PSO rules of movement have to be changed. Also, the local fitness function used to guide the particles' search may differ from the global evaluation function used to evaluate the swarm as a whole.

The Michigan PSO rules of neighborhood and social interaction have been adapted to achieve the following objectives:

- Particles must retain diversity, because the solution has to be a set of particles in different positions.
- Particles must be able to locate clusters of patterns.
- Particles must be able to locate decision boundaries between sets of patterns of different classes, where classification is harder.

We have performed experimentation on simple artificial problems to validate the swarm behavior and determine the correct rules to be used. Also, we have tested the resulting swarm in three well-known benchmark problems and we have found that the results are close or better than the standard nearest neighbor classifier (1-NN) with the limited number of prototypes found by the algorithm. This proves that PSO can be used to produce a representative set of prototypes.

The results on the benchmark problems indicate that the set of prototypes matches or outperforms 1-NN classifiers on these problems. Accuracy is better or close to the other algorithms used for comparison in the Pima Diabetes and Bupa Liver Disorder problems, and is significantly better than any other tested algorithm on the Glass Identification data set.

Closer observation of the MichPSO behavior also shows that this approach obtains a swarm with some characteristics which may be useful in other fields of application:

- Particles in the swarm are able to locate different areas with high values of the local fitness function and perform local search in those areas. This can be seen as multimodal optimization of the local fitness function, and can be of interest in that field.
- Particles are able to find equilibrium situations which can be altered by the influence of intruders, forcing them to adapt to the new situation. This behavior can be of interest for dynamically-adapting swarms.

Given that we have not yet introduced k-NN classification (with $k > 1$), attribute processing, nor any hybridization, we think these results are quite promising. However either attribute weighting or non Euclidean distances are likely to be required to compete in other problems.

Some issues that will be addressed in further work are analysis of the swarm sensitivity to the parameters' values, and inclusion of a mechanism to adaptively adjust the number of particles and their class distribution.

ACKNOWLEDGMENTS

This article has been financed by the Spanish founded research MEC project OPLINK::UC3M, Ref: TIN2005-08818-C04-02 and CAM proyect UC3M-TEC-05-029.

REFERENCES

[1] J. Kennedy, R.C. Eberhart, and Y. Shi. *Swarm intelligence*. Morgan Kaufmann Publishers, San Francisco, 2001.

[2] P. N. Suganthan. Particle swarm optimiser with neighbourhood operator. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1958–1962, 1999.

[3] Riaan Brits. Niching strategies for particle swarm optimization. Master's thesis, University of Pretoria, Pretoria, 2002.

[4] X. Hu and R.C. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimisation. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1677–16, 2002.

[5] Maurice Clerc and James Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evolutionary Computation*, 6(1):58–73, 2002.

[6] Frans van den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, South Africa, 2002.

[7] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Inf. Process. Lett.*, 85(6):317–325, 2003.

[8] Tiago Sousa, Arlindo Silva, and Ana Neves. Particle swarm based data mining algorithms for classifi cation tasks. *Parallel Comput.*, 30(5-6):767–783, 2004.

[9] Alejandro Cervantes, Pedro Isasi, and In´es Galv´an. Binary particle swarm optimization in classifi cation. *Neural Network World*, 15(3):229–241, 2005.

[10] Alejandro Cervantes, Pedro Isasi, and In´es Galv´an. A comparison between the pittsburgh and michigan approaches for the binary pso algorithm. In *Proceedings of the 2005 IEEE Congress on Evolucionary Computation, CEC 2005*, pages 290–297, 2005.

[11] Henry Brighton and Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2):153–172, 2002.

[12] Fernando Fern´andez and Pedro Isasi. Evolutionary design of nearest prototype classifi ers. *Journal of Heuristics*, 10(4):431–454, 2004.

[13] J.H. Holland. Adaptation. *Progress in theoretical biology*, pages 263–293, 1976.

[14] Steward W. Wilson. Classifi er fi tness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[15] T. M. Blackwell and Peter J. Bentley. Don't push me! collision-avoiding swarms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1691–1696, 2002.

[16] T. M. Blackwell and Peter J. Bentley. Dynamic search with charged swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference 2002 (GECCO)*, pages 19–26, 2002.

[17] Wlodzislaw Duch. Datasets used for classifi cation: comparison of results. http://www.phys.uni.torun.pl/kmk/projects/datasets.html.