Soft Comput (2008) 12:1073–1080
DOI 10.1007/s00500-007-0274-8

ORIGINAL PAPER

Self-organising swarm (SOSwarm)

Michael O'Neill · Anthony Brabazon

Published online: 29 January 2008

Abstract This paper introduces a novel version of the particle swarm optimisation (PSO) algorithm which we call self-organising swarm *SOSwarm*. SOSwarm can be used for unsupervised learning. In the algorithm, input vectors are projected into a lower-dimensional map space producing a visual representation of the input data in a manner similar to a self-organising map (SOM). In SOSwarm, particles react to input data during the learning process by modifying their velocities using an adaptation of the PSO velocity update function. SOSwarm is successfully applied to ten benchmark problems drawn from the UCI Machine Learning repository. The paper also demonstrates how the canonical SOM can be explored within the PSO paradigm. Illustrating this linkage between the heretofore distinct literatures of SOM and PSO opens up several new avenues of research for the development of novel self-organising algorithms.

Keywords Self-organising swarm · Self-organising map · Particle swarm algorithm

1 Introduction

Clustering is a commonly encountered scenario in many data-mining applications, the objective being to uncover a structure in a collection of unlabeled data (e.g., Jiang et al. 2006; Yue et al. 2007; Chung et al. 2006; Yang and Yi 2007; Wang et al. 2007; Karakasidis and Georgiou 2004). In cluster analysis, similar objects should be grouped into the same clusters, with dissimilar items being grouped into different clusters. Real-world applications of clustering include the mining of customer databases, the classification of plants and animals, gene clustering, fraud detection, data compression and image analysis. Over the years, a wide variety of algorithms have been developed for clustering purposes, including K-means (MacQueen 1967), fuzzy C-means (Dunn 1973), hierarchical clustering (Johnson 1967), and mixture of Gaussians (Dempster et al. 1977). In addition to these traditional clustering algorithms, several biologically-inspired clustering algorithms have recently been developed. These algorithms are derived from a variety of sources of inspiration including evolutionary processes [genetic algorithm clustering applications include Franti et al. (1997), Maulik and Bandyopadhyay (2000), Tseng and Yang (2001) and Garai and Chaudhuri (2004); genetic programming clustering applications include De Falco et al. (2005) and De Falco et al. (2006)] and social systems [examples of ant-based clustering include Deneubourg et al. (1991), Lumer and Faieta (1994) and Bonabeau et al. (1999)]. Perhaps the best-known family of biologically-inspired clustering algorithms is the self-organising map (SOMs). Since the development of SOMs (Kohonen 1982, 1990) a considerable literature has developed on the application of SOMs for the purposes of clustering. A particular feature of SOMs is that they typically reduce multi-dimensional data to a low-dimensional map (or grid) of nodes. This makes SOMs useful tools for data visualisation (Kohonen 1998).

Earlier studies have applied PSO (Kennedy and Eberhart 1995; Kennedy et al. 2001) to refine the weight vectors for a SOM after an initial application of a standard SOM training methodology (Xiao et al. 2004, 2003). In contrast, SOSwarm adopts a modified Particle Swarm Algorithm (PSA) using unsupervised learning. In this study the resulting map is

M. O'Neill (✉) · A. Brabazon
Natural Computing Research and Applications Group,
University College Dublin, Dublin, Ireland
e-mail: m.oneill@ucd.ie

A. Brabazon
e-mail: anthony.brabazon@ucd.ie

applied to a series of benchmark classification problems. This second phase can be skipped if clustering alone is desired.

It seems natural to consider the adaptation of a PSA to perform automated clustering tasks such as those typically implemented using SOM, due to the manner in which particles of a swarm cluster to similar regions, over time in a PSA. We investigate whether it is indeed possible to successfully modify a PSA to perform clustering tasks. Therefore, in this paper we introduce the novel Self-Organising Swarm (SOSwarm) algorithm, which adopts unsupervised learning using a PSA framework, and demonstrate its potential by applying it to a series of benchmark problems.

1.1 Motivation for study

This paper is motivated by two primary issues. The practical importance of clustering and classification tasks implies that there is a continuing demand for effective and efficient software tools for these tasks. Although the PSA paradigm has been widely applied over the past decade, it has not yet been widely applied for the purposes of unsupervised learning [some examples can be found in Omran et al. (2005) and Omran et al. (2006)].

The second motivation stems from the observation that the past twenty years has seen the development and application of a wide array of soft computing approaches such as artificial neural networks, fuzzy systems, evolutionary algorithms, social swarm algorithms and probabilistic reasoning in order to solve real-world problems. Each of these methodologies has its own strengths and weaknesses, hence it is important that we understand the similarities and differences between these methods in order to design robust problem solvers. Two major paradigms in soft computing include SOMs and PSO. In Sect. 6, this paper illustrates that a close linkage can be drawn between these two paradigms and this opens the door for future work which will explore this linkage.

1.2 Format of paper

In Sect. 2 we introduce the Particle Swarm Optimisation (PSO) algorithm upon which SOSwarm is based. We then introduce the fundamental concepts of the SOM in Sect. 3. Following a detailed exposition of SOSwarm in Sect. 4, we apply the algorithm to ten benchmark problems from the UCI Machine Learning repository (Sect. 5). In Sect. 6 we describe the linkages between the SOM and PSO paradigms and finally we draw conclusions and outlining some possible directions for future investigations with SOSwarm in Sect. 7.

2 Particle swarm optimisation

In the context of PSO, a swarm can be defined as '... a population of interacting elements that is able to optimise some global objective through collaborative search of a space.' (Kennedy et al. 2001, p. xxvii). The nature of the interacting elements (particles) depends on the problem domain and typically each particle encodes a potential solution to the problem of interest. These particles move (fly) in an n-dimensional search space, in an attempt to uncover ever-better solutions.

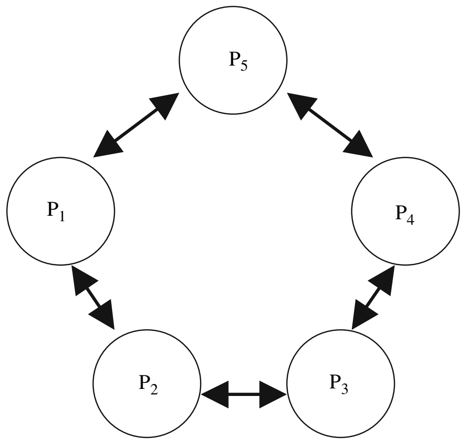

Fig. 1 Ring topology where l_{best} is defined using a three-node neighbourhood (each node is a neighbour of itself and two other nodes)

Each of the particles has two associated properties, a current position and a velocity. Each particle also has a memory of the best location in the search space that it has found so far (p_{best}) and knows the best location found to date by all the particles in the population (g_{best}) or in an alternative version of the algorithm, the best solution found in a local neighborhood around each particle (l_{best}). In the local version of the algorithm, each particle is considered to be linked to a subset of the population of particles, and this linkage structure is fixed at the beginning of the optimisation process and remains unchanged during it (see Fig.1). Whether the local or global communication version is implemented, at each step of the algorithm, particles are displaced from their current position by applying a velocity vector to them.

The velocity magnitude/direction is influenced by the velocity in the previous iteration of the algorithm (this simulates *momentum* and is also an implicit form of particle history), and the location of a particle relative to its p_{best} and g_{best} (or l_{best}). Therefore, at each step, the size and direction of each particle's move is a function of its own history and the social influence of its peer group. A number of variants of the PSA exist. The following paragraphs provide a description of a canonical continuous version of the algorithm.

(i) Initialise each particle in the population by randomly selecting values for its location and velocity vectors.
(ii) Calculate the fitness value of each particle. If the current fitness value for a particle is greater than the best fitness value found for the particle so far, then revise p_{best}.

(iii) Determine the location of the particle with the highest fitness and revise g_{best} if necessary.
(iv) For each particle, calculate its velocity according to Eq. 1.
(v) Update the location of each particle according to Eq. 3.
(vi) Repeat steps ii–v until stopping criteria are met.

The update algorithm for particle i's velocity vector v_i is

$$v_{i,d}(t+1) = w \times v_{i,d}(t) + (c_1 \times r_1 \times (y_{i,d} - x_{i,d}(t))) + (c_2 \times r_2 \times (g_{\text{best}_{i,d}} - x_{i,d}(t))) \quad (1)$$

where

$$w = w_{\max} - ((w_{\max} - w_{\min})/\text{iter}_{\max}) \times \text{iter} \quad (2)$$

In Eq. 1, y_i (p_{best}) is the location of the best solution found to date by particle i, g_{best} is the location of the global-best solution found by all particles to date, c_1 and c_2 are the weights associated with the p_{best} and the g_{best} terms in the velocity update equation, x_i is particle i's current location, and r_1 and r_2 are randomly drawn from $U(0, 1)$. The parameter w represents a momentum coefficient which is reduced according to Eq. 2 as the algorithm iterates. In Eq. 2, $\text{iter}_{\max}$ and iter are the total number of iterations the algorithm will run for, and the current iteration value respectively, and $w_{\max}$ and $w_{\min}$ set the upper and lower boundaries on the value of the momentum coefficient. The velocity update on any dimension is constrained to a maximum value of $v_{\max}$. Once the velocity update for particle i is determined, its position is updated (Eq. 3), and p_{best} is updated if necessary (Eqs. 4, 5), where y_i represents the location of p_{best}.

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (3)$$

$$y_i(t+1) = y_i(t) \quad \text{if } f(x_i(t)) \le f(y_i(t)) \quad (4)$$

$$y_i(t+1) = x_i(t) \quad \text{if } f(x_i(t)) > f(y_i(t)) \quad (5)$$

After the location of all particles have been updated, a check is made to determine whether g_{best} needs to be updated (Eq. 6).

$$\hat{y} \in (y_0, \ldots, y_{n-1}) | f(\hat{y}) = \max(f(y_0), \ldots, f(y_{n-1})) \quad (6)$$

In each iteration of the algorithm, a particle is stochastically accelerated towards its previous best position and towards a global (or neighbourhood) best position, thereby forcing particles to continually search in the most promising regions found so far in the solution space. The weight coefficients c_1 and c_2 control the relative impact of the p_{best} and g_{best} locations on the velocity of a particle. Low values for c_1 and c_2 allow each particle to explore far away from already uncovered good points (there is less emphasis on past learning), high values of the parameters encourage more intensive search of regions close to these points. The random coefficients r_1 and r_2 ensure that the algorithm is stochastic. A practical effect of r_1 and r_2, is that neither the individual nor the social learning terms are always dominant.

The neighbourhood structure plays an important role in determining the nature of the communication between particles during the search process. If the neighbourhood is set at 1 (each particle only communicates with itself), then each particle searches independently of all other particles. If the neighborhood $= N$ (the number of particles in the swarm) all particles can communicate with each other and we have the g_{best} version of the PSO algorithm.

Particle Swarm algorithms have been successfully applied to a diverse range of optimisation problems including financial modelling (Brabazon and O'Neill 2006), the automatic generation of programs (O'Neill and Brabazon 2004; O'Neill et al. 2004; O'Neill and Brabazon 2006) using a grammatical representation borrowed from Grammatical Evolution (O'Neill and Ryan 2003), assembly line sequencing (Rahimi-Vahed et al. 2007), and the optimisation of weights in SOM structures (Xiao et al. 2004).

3 Self-organising map

Self-organising maps (Kohonen 1982, 1990, 1998) are a form of artificial neural network (NN) which can cluster data using unsupervised learning. The SOM acts to project (compress) input data vectors onto a low-dimensional space, typically a two-dimensional grid structure, thereby producing a visual representation of the input data. The unsupervised learning process is based on measures of similarity amongst the input data vectors. During the training process, the network undergoes self-organisation as like input data patterns are grouped or clustered together on the grid structure. SOMs have been utilised for a variety of clustering and classification problems including speech recognition and medical diagnosis (Gurney 1997). The SOM bears similarities with the traditional statistical technique of principal component analysis (PCA). However, unlike PCA the projection of the input data is not necessarily restricted to be linear.

The SOM consists of two layers, the input layer (a holding point for the input data), and the *mapping* layer (see Fig. 2). The input layer has as many nodes as there are input variables. The two layers are fully connected to each other and each of the nodes in the hidden layer has an associated weight vector, with one weight for each connection with the input layer.

The aim of the SOM is to group like input data-vectors together on the mapping layer. Therefore the method is *topology preserving* as items which are close in the input space are also close in the mapping space. During training the data vectors are presented to the SOM through the input layer one at a time. The nodes in the mapping layer *compete* for the input data vector. The winner is the mapping node whose vector of incoming connection weights most closely resembles the

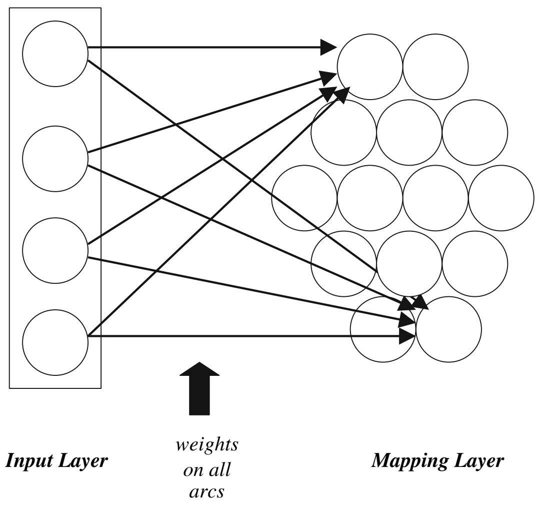

Fig. 2 A SOM with a 2-d mapping layer. On grounds of visual clarity, only the connections between the input layer and two of the mapping layer nodes are shown

components of the input data vector. The winner has the values of its weight vector adjusted to move them towards the values of the input data vector, and the mapping layer nodes in the neighbourhood of the winning node also have their weight vectors altered to become more like the input data vector (a form of *co-operation*, or peer-learning, between the neighbouring nodes). As more input data vectors are passed through the network, the weight vectors of the mapping layer nodes will self-organise. By the end of the training process, different parts of the mapping layer will respond strongly to specific regions of input space. Once training of the network is complete, the clusters obtained can be examined in order to gain better insight into the underlying dataset (for example, what input items have been grouped together, what are the typical values for each input in a specific cluster).

3.1 PSO-SOM hybrids

There are several ways that a PSO-SOM hybrid could be constructed. In Xiao et al. (2003, 2004) a PSO algorithm was used to refine the weight vectors for a SOM obtained after an initial application of a standard SOM training methodology. In this approach each particle consisted of a complete set of weights for the SOM, and the object was to improve the initial clustering result by applying PSO to the population of weight vectors. The approach in our study differs fundamentally from the above and is outlined in the next section.

4 Self-organising swarm

The SOSwarm operates in a similar fashion to a SOM with the adoption of a 2-d mapping layer but the components of this layer are particles from a PSA. Instead of simply adjusting node weights in the map space with respect to the training input vectors, the particles in the mapping layer adjust their values using an adapted form of a PSA velocity update function.

The canonical form of the PSA update embeds two key elements:

- a history, and
- a social influence.

History is embedded in the PSA via the momentum term and the p_{best} components of the velocity update equation. The social influence is embedded via the influence of either g_{best} or l_{best} in the velocity update.

Section 6 provides a further discussion on the algorithm adopted. In undertaking our experiments in Sect. 5 we apply the output from an unsupervised SOSwarm learning process for classification purposes. An outline of the SOSwarm algorithm used for this purpose is presented below.

```
0  initialise particles in mapping layer randomly
1  for( max number of iterations )
2     for( each input training vector in turn )
3         set gbest to be the input vector
4         find particle with closest match to gbest
5         denote this particle as the firing particle
6         update firing particle's velocity
          and position vectors
7         if new position better than pbest update pbest
          update firing particle's neighbours
8     endfor
9  endfor
10 assign class to each particle using training data
11 calculate classification accuracy using test data
```

In order to determine the firing particle (the particle that is the closest match to an input vector) a simple distance calculation is adopted

$$\text{Firing particle} = \arg\min_i \| V - P_i \| \tag{7}$$

where V corresponds to the input vector, P_i is the ith particle's position vector. A number of alternative distance functions could be adopted and we adopt Euclidean distance as outlined in Eq. 8 (where d is the dimensionality of the vector or particle).

$$\text{Firing particle} = \arg\min_i \sqrt{\sum_1^d (V_d - P_{id})^2} \tag{8}$$

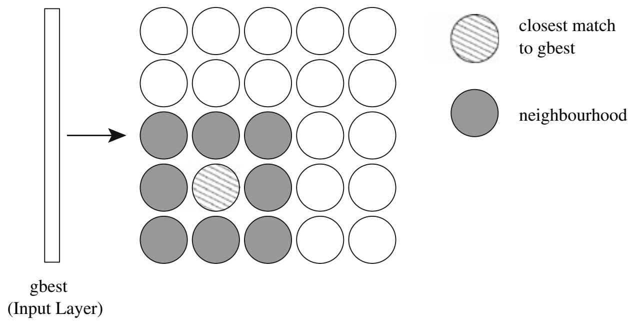

Fig. 3 A Self-Organising Swarm (*SOSwarm*) with a 2-d mapping layer

A visual representation of SOSwarm is presented in Fig. 3 with the adoption of a 2-d mapping layer. The particles are arranged a priori into a fixed neighborhood topology, a simple grid in this example. The firing particle, that is, the particle whose position vector is closest to the current input vector (designated as g_{best}) updates its position vector in a particle swarm style according to the update Eqs. (1)–(3). In addition, particles lying within a fixed neighbourhood of the firing particle also adjust their position vectors using the same equations, implicitly embedding a form of social communication between neighbouring particles.

Once the map has been trained, each node in the mapping layer is assigned a class label using the training data, based on a simple majority voting scheme. In calculating the in sample and out sample classification accuracy, the distance between each input data vector and each mapping node is calculated, with the input data vector being assigned the class label of the mapping node it is closest to.

5 Experimental setup and results

In order to assess the utility of the SOSwarm algorithm, four classification problems from the UCI Machine Learning Repository (Hettich et al. 1998) are examined. The datasets consist of varying numbers of inputs and known output classes (see Table 1). Initially, the SOSwarm clustering algorithm is applied and then the nodes on the resulting map are labeled. The labeled nodes are then used to classify both the in sample (training) data and the out sample (test) data. We then report the classification accuracies on each dataset.

The following parameters were used for the SOSwarm algorithm, $c_1 = 1.0$, $c_2 = 1.0$, $w_{\max} = 0.9$, $w_{\min} = 0.4$, $c_{\min} = 0$, $c_{\max} = 1$, and $v_{\max} = c_{\max}$. Where $c_{\min}$ and $c_{\max}$ are used to constrain dimension values to a valid range specific to each dataset. The population of particles was set at 100 (a 10×10 grid structure). The algorithm was run for a total of 10,000 iterations. In this proof of concept study the parameter values were not optimised and are set at typical values for a standard PSA. It might be expected that further performance gains could be achieved through their tuning.

Table 1 Dataset training and test set partition sizes

Dataset	Training	Test
Wisconsin	559	140
Pima	614	154
New thyroid	172	43
Glass	171	43
Iris	120	30
Wine	142	36
Australian	552	138
Bupa	276	69
Ionosphere	281	70
Waveform	4,000	1,000

As the mapping process utilises a distance metric, the input variables in each dataset were normalised independently on each dimension into the range [0, 1]. The datasets were partitioned in training and test sets of the following sizes as illustrated in Table 1. The distance metric in Eq. 8 is used to determine the particle that is the closest match. A fixed grid neighbourhood topology is adopted, with the range of the neighbourhood as illustrated in Fig. 3. Therefore a particle will have at most eight neighbours which will be subjected to updates if that particle fires.

For each of 10 recuts of a problem dataset, 30 independent runs for SOSwarm are conducted with the corresponding results presented in Table 2 for the unseen test data. The results for the training dataset are presented in Table 3. Comparing the in sample and out sample results, it is notable that the mean best out sample results are similar in quality to the mean best in sample results, indicating that the SOSwarm methodology has generated classifiers which are generalising well out of sample.

In order to provide a benchmark for the results obtained by SOSwarm, these results are compared with the best results obtained on the same datasets using other classification methods. These results are drawn from a table complied by Smith and Bull (2005) which provides benchmark information for seven of the ten datasets on which we tested the algorithm. Table 2 illustrates the highly competitive performance of SOSwarm, with it outperforming six of the seven available benchmark scores. It is worth noting that computational effort required per iteration of SOSwarm will be very similar to that of a traditional SOM with the addition of the calculation of the momentum ($w.v_{i,d}$), personal/cognitive learning ($y_{id} - x_{i,d}$), and the generation of two pseudo-random constants (r_1 and r_2) in Eq. 1.

6 SOSwarm and SOM

Although, as already noted in Sect. 3.1, a number of studies have combined PSO and SOM methodologies, significantly

Table 2 A comparison of the results obtained on the unseen test data for the Self-Organising Swarm (SOSwarm) algorithm across the benchmark problems analysed averaged across the ten recuts of the dataset in each case

Problem	#Variables	#Classes	Average accuracy (SD)	Best accuracy	Best reported accuracy
Wisconsin breast cancer	9	2	0.962 (0.014)	1.0	0.96
Pima indians diabetes	8	2	0.693 (0.032)	0.80	0.78
New thyroid	5	3	0.912 (0.038)	1.0	0.97
Glass	9	6	0.579 (0.061)	0.81	0.74
Iris	4	3	0.845 (0.063)	1.0	NA
Wine	13	3	0.809 (0.064)	0.97	0.98
Australian	6	2	0.710 (0.033)	0.79	NA
Bupa	6	2	0.584 (0.054)	0.75	0.71
Ionosphere	34	2	0.871 (0.039)	0.99	0.91
Waveform	21	3	0.741 (0.015)	0.79	NA

Table 3 A comparison of the results obtained on the training data for the SOSwarm algorithm across the benchmark problems analysed averaged across the ten recuts of the dataset in each case

Problem	Mean average accuracy (Mean SD)	Mean best accuracy
Wisconsin breast cancer	0.977 (0.003)	0.980
Pima indians diabetes	0.767 (0.011)	0.782
New thyroid	0.964 (0.012)	0.971
Glass	0.876 (0.019)	0.883
Iris	0.977 (0.010)	0.983
Wine	0.970 (0.013)	0.978
Australian	0.774 (0.012)	0.780
Bupa	0.762 (0.017)	0.770
Ionosphere	0.905 (0.016)	0.916
Waveform	0.771 (0.009)	0.776

however, no previous study has examined the deeper linkages between the two methodologies. The teasing out of such linkages is important as both paradigms are well-developed and are widely used. The drawing of parallels between both paradigms opens up a door for useful cross-fertilization between each.

In the canonical SOM, the update of a firing node's weight vector is governed by:

$$x_i(t+1) = \eta(t)h(t)(x_i(t) - \beta) \tag{9}$$

where x_i is the firing node's weight vector, η is the time-varying learning rate and β is the input vector. Hence, after firing, the weight vector of the relevant node in the mapping layer, and those of its neighbours which are defined by the neighbourhood function $h(t)$, are adjusted in order to more closely resemble the input vector. Ignoring the update of neighbouring nodes, and thinking of Eq. 9 in particle swarm terms, it is apparent that it can be written as a velocity update:

$$v_i(t+1) = \eta(t)(x_i(t) - \beta) \tag{10}$$

Of course, the component $\eta(t)$ in Eq. 10, in effect a weighting term, is similar in concept to the weight parameter c_1 in Eq. 1. Hence, the canonical SOM update equation can be closely approximated by a reduced (non-momentum) version of the canonical PSA update equation.

This parallel between the SOM and a reduced form PSA suggests multiple possibilities for the creation of new hybrid algorithms for self-organisation. For example, the PSA embeds momentum, a form of personal particle history which is not included in the canonical SOM. Like the learning rate in the SOM, the momentum term in the PSA velocity update is time-varying, and it reduces over time in order to encourage particle convergence. The SOSwarm algorithm described in Sect. 4 includes momentum.

Another interesting possibility, drawing on the use of peer learning in the PSA, is the incorporation of an additional 'peer-learning' term into the SOSwarm velocity update. For example, a topology consisting of a series of small overlapping neighbourhoods could be defined between the particles before the algorithm started, with Eq. 1 being extended by the addition of the term $c_3 \times r_3 \times (y_{\text{local}} - x_i(t))$, where y_{local} is the location of a randomly selected neighbouring particle of $x_i(t)$. This social learning would tend to lessen the disruptive impact of an anomalous input vector during the learning process. This could prove especially useful in environments where training data is noisy or contains many errors.

Table 4 provides a comparison of SOSwarm and SOM again on ten recuts of each dataset. SOM adopts the same fixed, hamiltonian neighbourhood as adopted in SOSwarm, with a constant learning rate ($\eta(t) = 1.0$). To further minimise the differences between SOSwarm and SOM, the implementation adopted here can thus be represented by the following Eq. 11, where we have removed the momentum term ($w \times v_{i,d}(t)$), and the cognitive learning term

Table 4 A comparison of the results obtained on the test data for the SOSwarm algorithm versus SOM across the benchmark problems analysed averaged across the ten recuts of the dataset in each case

Problem	SOSwarm	SOM
Wisconsin breast cancer	**0.962**	0.956
Pima indians diabetes	0.693	0.691
New thyroid	**0.912**	0.898
Glass	0.579	0.583
Iris	**0.845**	0.826
Wine	0.809	0.814
Australian	0.710	0.707
Bupa	**0.584**	0.570
Ionosphere	0.871	**0.883**
Waveform	0.741	**0.756**

Mean average accuracy is reported with significant performance advantage highlighted in bold

$((c_1 \times r_1 \times (y_{i,d} - x_{i,d}(t))))$ from SOSwarm. We have retained the $c_2 \times r_2$ coefficients applied to the difference of the node from the input vector $((g_{\text{best}_{i,d}} - x_{i,d}(t)))$. Note $c_2 = 1.0$ in this study, so the main difference from standard SOM is the addition of a noise coefficient as represented by r_2.

$$v_{i,d}(t+1) = c_2 \times r_2 \times (g_{\text{best}_{i,d}} - x_{i,d}(t)) \tag{11}$$

Any observed differences can therefore be attributed to the momentum and cognitive learning terms that are present in SOSwarm and absent in SOM. A two-tailed t test was conducted at the 95% confidence level on the results presented in Table 4. It is found that SOSwarm significantly outperforms SOM on four out of the ten problems examined (Iris, New Thyroid, Wisconsin Breast Cancer, Bupa), with SOM significantly outperforming SOSwarm in two instances (Ionosphere and Waveform). There was no statistically significant difference at this level on the remaining four problems. This provides evidence to support the suggestion that the adoption of concepts from Particle Swarm can bring performance advantages to SOM. In particular, the results suggests that the addition of momentum and a cognitive learning term, which takes into consideration personal experience over earlier input vectors can be useful.

7 Conclusions and future work

This paper presented a novel SOSwarm algorithm, and illustrated the utility of the algorithm by applying it to a series of benchmark problems from the UCI Machine Learning repository. Classification accuracies reveal that SOSwarm produces highly competitive results with significant performance gains over an equivalent SOM algorithm on four of the instances. Given that this study represents an initial application of SOSwarm these results are very encouraging.

There are several interesting avenues of future research for SOSwarm. A variety of distance metrics could be used in calculating the distance between input vectors and each member of the swarm. In this study, we utilised a simple Euclidean distance metric but other distance metrics could be applied. Another open research avenue is to investigate the effect of differing neighbourhood topologies between the particles in the swarm. It would also be interesting to examine in what circumstances a reducing size of neighbourhood over the course of the algorithm would be beneficial. Other possible extensions of the study include the investigation of different velocity update formulations in particular, drawing on the importance of peer learning in the PSA, the exploration of what an explicit peer-learning term can contribute to the improvement of the SOSwarm algorithm. Although we have applied SOSwarm for classification purposes in this study, it could clearly be applied for clustering purposes, opening up such potential applications as gene clustering, and customer database segmentation.

Results presented here suggest that the incorporation of an explicit 'history' mechanism in the form of momentum and a cognitive learning term, as exists in the SOSwarm algorithm, might be a useful feature to adopt in SOM. The comparison between SOSwarm and SOM suggests that there is merit to further investigate the correspondence between these two approaches, focusing on the relative importance of the momentum and cognitive learning, and the various update topologies in both SOM and PSA in order to determine whether there are useful lessons which can be transplanted between the two paradigms.

Acknowledgments We would like to thank the anonymous reviewers for their constructive feedback, which has helped to improve the final draft of this paper.

References

Brabazon A, O'Neill M (2006) Biologically inspired algorithms for financial modelling. Springer, Berlin

Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, Oxford

Chung F-L, Wang S, Deng Z, Shu C, Hu D (2006) Clustering analysis of gene expression data based on semi-supervised visual clustering algorithm. Soft Comput. 10(11):981–993

De Falco I, Tarantino E, Delia Cioppa A, Gagliardi F (2005) A novel grammar-based genetic programming approach to clustering. In: Proceedings of the 2005 ACM symposium on applied computing, Santa Fe, New Mexico, pp 928–932

De Falco I, Tarantino E, Delia Cioppa A, Fontanella F (2006) An innovative approach to genetic programmingf̂9based clustering. Adv Soft Comput 55–64

Dempster A, Laird N, Rubin D (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B 39(1): 1–38

Deneubourg J, Gross S, Franks N, Sendova-Franks A, Detrain C, Chretien L (1991) The dynamics of collective sorting robot-like ants and ant-like robots. In: Meyer J, Wilson S (eds) Proceedings of 1st conference on simulation of adaptive behavior: from animals to animats (SAB 90). MIT Press, Cambridge, pp 356–365

Dunn J (1973) A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J Cybern 3:32–57

Franti P, Kivijarvi J, Kaukoranta T, Nevalainen O (1997) Genetic algorithms for large scale clustering problems. Comput J 40:547–554

Garai G, Chaudhuri B (2004) A novel genetic algorithm for automatic clustering. Pattern Recognit Lett 25(2):173–187

Gurney K (1997) An introduction to neural networks. University College London Press, London

Hettich S, Blake CL, Merz CJ (1998) UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html. University of California, Department of Information and Computer Science, Irvine, CA

Jiang K, Liao Q-M, Xiong Y (2006) A novel white blood cell segmentation scheme based on feature space clustering. Soft Comput 10(1):12–19

Johnson S (1967) Hierarchical clustering schemes. Psychometrika 2:241–254

Karakasidis T, Georgiou D (2004) Partitioning elements of the Periodic Table via fuzzy clustering technique. Soft Comput 8(3):231–236

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, pp 1942–1948

Kennedy J, Eberhart R, Shi Y (2001) Swarm intelligence. Morgan Kauffman, San Mateo

Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43:59–69

Kohonen T (1990) The self-organizing map. Proc IEEE 78(9):1464–1480

Kohonen T (1998) The SOM methodology. In: Deboeck G, Kohonen TVisual explorations in finance with self-organizing maps. Springer, Berlin

Lumer E, Faieta B (1994) Diversity and adaptation in populations of clustering ants. In: Proceedings of third international conference on simulation of adaptive behaviour, pp 501–508

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley symposium on mathematical statistics and probability, vol 1. University of California Press, Berkeley, pp 281–297

Maulik U, Bandyopadhyay S (2000) Genetic algorithm-based clustering technique. Pattern Recognit 33:1455–1465

Omran M, Engelbrecht AP, Salman A (2005) Particle swarm optimization method for image clustering. Int J Pattern Recognit Artif Intell 19(3):297–322

Omran MGH, Salman A, Engelbrecht AP (2006) Dynamic clustering using particle swarm optimization with application in image segmentation. Pattern Anal Appl 8(4):332–344

O'Neill M, Ryan C (2003) Grammatical evolution: evolutionary automatic programming in an arbitrary language. Kluwer Academic Publishers, Boston Computation 5(4):349–358

O'Neill M, Brabazon A (2006) Grammatical swarm: the generation of programs by social programming. Nat Comput 5:443–462

O'Neill M, Brabazon A, Adley C (2004) The automatic generation of programs for classification using grammatical swarm. In: Proceedings of the congress on evolutionary computation CEC 2004. IEEE Press, Portland, pp 104–110

O'Neill M, Brabazon A (2004) Grammatical swarm. In: Proceedings of the genetic and evolutionary computation conference GECCO 2004. Springer, Seattle, pp 163–174

Rahimi-Vahed AR, Mirghorbani SM, Rabbani M (2007) A new particle swarm algorithm for a multi-objective mixed-model assembly line sequencing problem. Soft Comput 11(10):997–1012

Smith M, Bull L (2005) Genetic programming with a genetic algorithm for feature construction and selection. Genet Program Evol Mach 6(3):265–281

Tseng L, Yang S (2001) A genetic approach to the automatic clustering problem. Pattern Recognit 34:415–424

Wang P, Liu Z-Q, Yang S-Q (2007) Investigation on unsupervised clustering algorithms for video shot categorization. Soft Comput 11(4):355–360

Xiao X, Dow E, Eberhart R, Miled Z, Oppelt R (2003) Gene-clustering using self-organizing maps and particle swarm optimization. In: Proceedings of the IEEE international parallel and distributed processing symposium (IPDPS), 22–26 April 2003. IEEE Press, Nice

Xiao X, Dow E, Eberhart R, Miled Z, Oppelt R (2004) A hybrid self-organizing maps and particle swarm optimization approach. Concur Comput Pract Exp 16(9):895–915

Yue X, Abraham A, Chi Z-X, Hao Y-Y, Mo H (2007) Artificial immune system inspired behavior-based anti-spam filter. Soft Comput. 11(8):729–740

Yang C, Yi Z (2008) Document clustering using locality preserving indexing and support vector machines. Soft Comput (published online 17 Oct 2007, in press)