Efficient Multi-dimensional Data Clustering using Particle Swarm Optimization

Edgar A. Garcia-Martinez, Salvador Godoy-Calderon,
Ricardo Barron-Fernandez, and Javier Arellano-Verdejo

Laboratorio de Inteligencia Artificial,
Centro de Investigación en Computación (CIC),
Instituto Politécnico Nacional (IPN), México[1]
eagarciam@ipn.mx, sgodoyc@cic.ipn.mx, rbarron@cic.ipn.mx,
jarellanob10@sagitario.cic.ipn.mx

Abstract. Clustering is a very important process that plays a key role in data analysis and data mining that has been effectively used for diverse procedures in artificial intelligence. On the other hand, Particle Swarm Optimization is a family of high-performance meta-heuristic techniques successfully used for solving multi-objective optimization problems. This paper presents a modified particle swarm algorithm for optimizing the localization of cluster centroids. The results yielded by several experiments show this algorithm to be reliable and efficient.

Keywords: Particle Swarm Optimization, Pattern Recognition, Clustering

1 Introduction

Clustering is a non-supervised technique used for uncovering the underlying structure on a dataset [6]. A precise knowledge about the structure of a particular dataset turns out to be essential for image segmentation [7]-[19], data mining [13] and data compression [1] problems among others. Since there is no supervision information available when clustering data, the only reliable source of information is the similarity or dissimilarity among patterns. For measuring that similarity or dissimilarity a problem-specific user-proposed metric called Pattern Analogy function is generally used. Each cluster is defined by its centroid or by the pattern with the biggest average similarity sometimes called medoid or holotype [16]. Clustering a dataset is a difficult problem since clusters can show a great diversity of sizes and shapes [12]. Besides, the optimum number of clusters for the dataset is not known a priori and automatically finding it is still an open research problem [11], so a great majority of algorithms require it as a parameter supplied by the user.

[1] The authors wish to thank the financial support of COFAA-IPN, SIP-IPN, CIC-IPN, ICyT-DF, and SNI-CONACYT; particularly through grants of projects SIP-20121154, SIP-20120820 and ICyT-PICCO10-113

Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 45–57
Paper Recived 29-09-2012 and Acepted 01-11-2012

Particle Swarm Optimization (PSO) is a meta-heuristic algorithm, designed by Kennedy and Eberhart in 1995, modeled following the social behavior of bird flocks and fish schools [14]-[15]. In this algorithm, a group (called swarm) of candidate solutions for an optimization problem (called particles) moves across the problem space, searching for a global optimum solution. Each particle's position is updated considering its fitness (its knowledge about the environment), the position of its neighborhood's best fitted particle (social influence) and the best position that particle itself has visited (cognitive influence). PSO's ability to search huge spaces with very few particles, and the rich set of possibilities it offers for defining inter-neighborhood and inter-particle collaborative search has contributed to consider it as one of the most efficient evolutionary algorithms available today.

Some partitional and density-based clustering algorithms (like K-Means [20] and DBSCAN [10]) strongly rely upon a set of initial problem conditions. When these conditions are not sufficiently satisfied the algorithm may converge to sub-optimal solutions. PSO algorithms, because of their particle updating dynamics, are believed to be much less sensitive to initial problem conditions, so using PSO algorithms for solving clustering problems is a sound strategy with higher probabilities of finding global optimum solutions. This paper shows a modified PSO-based algorithm that efficiently clusters data patterns by optimizing centroid or holotype positions of each cluster.

2 Theoretical Background

2.1 PSO Algorithm

This algorithm initializes its swarm with randomly generated solutions and a randomly selected speed factor for each particle. Each particle includes a field called *pbest* for storing its best previous value and its position in the search space. Globally known to all particles is the best global value found so far by any particle (*gbest*) and its position. *Pbest* and *gbest* are used for changing a particles' speed after each iteration (or generation) of the swarm search.

PSO is a stochastic and iterative process operating on a particle swarm. Each particle is composed by three vectors and two fitness values as described below:

- Vector $x_i = \langle x_{i1}, x_{i2}, \ldots, x_{in} \rangle$ stores the current particle's position in the search space.
- Vector $pBest_i = \langle p_{i1}, p_{i2}, \ldots, p_{in} \rangle$ stores the position of the particle's best solution.
- Vector $v_i = \langle v_{i1}, v_{i2}, \ldots, v_{in} \rangle$ stores the direction gradient (speed) for regulating the particle's movement.
- Value $fitness\_x_i$ is the fitness of the particle's current solution.
- Value $fitness\_pBest_i$ is the fitness of the particle's best solution.

The PSO algorithm starts by setting the initial position and speed of each particle in the swarm. The initial position of each particle can be generated

randomly according to a homogeneous distribution or by using some specific initialization heuristic. Once the initial positions are set, the fitness value of each particle is calculated and the $fitness\_x_i$ and $fitness\_pBest_i$ values are updated.

The speed of each particle is randomly selected with each component within the interval $[-v_{max}, v_{max}]$, where v_{max} will be the maximum speed that each moving particle can adopt. Empirical results have shown that zero-valued speeds almost never promote good final results. Particles should move in an iterative fashion by adding their speed vector v_i to their position vector x_i , thus finding the new position vector: $x_{i+1} \leftarrow x_i + v_i$.

Once in its new position, the particle's fitness is calculated and the $fitness\_x_i$ value is updated. If the current fitness value is better than the fitness of the best found solution, then $fitness\_x_i$ and $fitness\_pBest_i$ values are also updated.

The speed vector of each particle is updated using its previous speed, the value of a cognitive component, and also the value of a social component. The resulting mathematical model is at the core of the PSO algorithm:

$$v_i{}^{k+1} = \omega v_i{}^k + \phi_1 rand_1(pBest_i - x_i{}^k) + \phi_2 rand_2(gBest - x_i{}^k) \quad (1)$$

$$x_i{}^{k+1} = x_i{}^k + v_i{}^{k+1} \quad (2)$$

Equation 1 shows the speed-vector upgrade mechanism for particle i during iteration k. The *cognitive* component of this movement is set by the term $\phi_1 rand_1(pBest_i - x_i{}^k)$ which represents the distance between the current particle's position and the position of the best known solution for the same particle. The social component is set by the term $\phi_2 rand_2(gBest - x_i{}^k)$, the distance between the current particle's solution and the best known solution for any particle in the neighborhood.

2.2 PSO-based clustering algorithms

The first PSO-based clustering algorithm was introduced by Omran et al. in [18]. The experimental results shown by Omran et al. [17]-[18] depicted that the PSO-based clustering method outperformed other well known alalgorithms like k-means, Fuzzy C-Means (FCM) and a few other state-of-the-art clustering algorithms.

Van der Merwe and Engelbrecht hybridized this approach with the k-means algorithm for clustering general datasets [21]. In their approach, a single particle of the swarm is initialized with the result of the k-means algorithm and the rest of the swarm is randomly initialized. In 2003, Xiao et al. used a new approach based on the synergism of the PSO and the Self Organizing Maps (SOM) [22] for clustering gene expression data.

Cui et al. [8] proposed a PSO based hybrid algorithm for classifying the text documents. They applied the PSO, K-means and a hybrid PSO clustering algorithm on four different text document datasets.

3 Proposed Algorithm

The proposed clustering algorithm uses a global PSO process to find the optimum position for the k cluster centroids, using a global variance measure as the fitness function. Each particle is represented by the position of one cluster centroid. For each given centroid position all patterns are assigned to the cluster defined by their nearest centroid, so that the global variance of the resulting clustering can be measured.

Algorithm 1 Pseudocode of the proposed PSO algorithm

```
Population ← initialize_population();
while not_stop_condition() do
  for i ← 1 to size(Popultation) do
    Evaluate each particle x_i in Population
    if fitness(x_i) is better than fitness(pBest_i) then
      pBest_i ← x_i
      fitness(pBest_i) ← fitness(x_i)
    end if
    if fitness(pBest_i) is better than fitness(gBest) then
      gBest ← pBest_i
      fitness(gBest) ← fitness(pBest_i)
    end if
  end for
  for i ← 1 to size(Popultation) do
    v_i ← ωv_i + φ_1 rand_1(pBest_i − x_i) + φ_2 rand_2(gBest − x_i)
    x_i ← x_i + v_i
  end for
end while
```

3.1 Particle representation

Each particle p_i is represented as:

$$p_i = \begin{bmatrix} C_1 \\ C_2 \\ \dots \\ C_K \end{bmatrix}$$

Where each component $C_j = \langle x_1, x_2, \dots, x_D \rangle$ in the particle corresponds to the position for the centroid of cluster j. D is the search-space dimension and K the number of clusters.

3.2 Fitness Function

Several cluster validation indexes are available to be used as the fitness function for a PSO-based clustering algorithm. The source code library used for experimentation with the proposed algorithm was carefully parametrized so that any suitable function can be used as the fitness function for each run. Although the choice depends on the topologic properties of the dataset, a few of the most widely used indexes are available, like the $I - index$ [2], Davies-Bouldin [9] and Xie-Beni [23]. For the sake of clarity all the experimental results shown in this paper were achieved using a relatively simple global variance expression as fitness. So the fitness function for the experimental results section below is given by:

$$f(x) = \frac{1}{K} \sum_{j=1}^{K} \sum_{i=1}^{N} D_E(o_i, C_j) \tag{3}$$

Where K is the number of clusters, N the number of patterns in the j cluster, D_E is a standard similarity/dissimilarity metric (Euclidian distance for the experiments shown in this paper), o_i is the ith pattern in the j cluster, and C_j is the centroid for cluster j.

4 Experimental Results

To test the proposed algorithm several experiments were performed using widely known synthetic dataset. The results of proposed PSO algorithm were compared with the VGA-clustering algorithm proposed by Bandyaopadhyay and Maulik in 2001 [2]. The fitness function used in VGA-clustering algorithm is computed through the $I - index$ described in [2]. The datasets, labeled $Data\_5\_2$ [2]-[3]-[5], $Data\_4\_3$ [2]-[3]-[5], $Data\_6\_2$ [3], $Data\_9\_2$ [2]-[3]-[5], $Data\_10\_2$ [4]-[5] were taken from `http://www.isical.ac.in/~sanghami/data.html`. Each experiment was performed 100 times to reliably test its performance and accuracy. As is traditional when testing meta-heuristic procedures, configuration parameters for each experiment were empirically set and are shown in tables 1, 2, 3, 4 and 5. The considered performance metrics were execution time and number of iterations. All experiments were run on a AMD Athlon ^{TM} II X2-220 processor at 2.8 Ghz, with 6 GB Ram. Table 6 shows the average result from the 100 executions of each kind of experiment using the proposed PSO and VGA-clustering algirithms. Lastly, figures 1, 2, 3, 4 and 5 show the best clustering found for each dataset and each algorithm.

Dataset *Data_5_2* contains 250 patterns in $\mathbb{R}^2$ and 5 clusters. The configuration parameters for the PSO procedure used during this experiment were:

Table 1. Configuration parameters for experiments with the *Data_5_2* dataset

Parameter	Value
Swarm size (in particles)	5
Search-space dimensions	2
Inertia factor	0.9
Cognitive factor weight	1.8
Social factor weight	1.6
Search-space limits	[5.0 16.0]
Maximum speed	1.5

The *Data_4_3* dataset contains 400 patterns in $\mathbb{R}^3$ and 4 clusters. Configuration parameters for experiments with this data set were:

Table 2. Configuration parameters for experiments with the *Data_4_3* dataset

Parameter	Value
Swarm size (in particles)	20
Search-space dimensions	3
Inertia factor	0.9
Cognitive factor weight	1.8
Social factor weight	1.2
Search-space limits	[−2.0 17.5]
Maximum speed	2.0

The *Data*_6_2 dataset contains 300 patterns in $\mathbb{R}^2$ and 6 clusters. Configuration parameters for experiments with this data set were:

Table 3. Configuration parameters for experiments with the *Data*_6_2 dataset

Parameter	Value
Swarm size (in particles)	20
Search-space dimensions	2
Inertia factor	0.9
Cognitive factor weight	1.2
Social factor weight	1.4
Search-space limits	[−2.0 21.0]
Maximum speed	1.2

The *Data*_10_2 dataset contains 500 patterns in $\mathbb{R}^2$ and 10 clusters. Configuration parameters for experiments with this data set were:

Table 4. Configuration parameters for experiments with the *Data*_10_2 dataset

Parameter	Value
Swarm size (in particles)	15
Search-space dimensions	2
Inertia factor	0.9
Cognitive factor weight	1.8
Social factor weight	1.2
Search-space limits	[−18.0 18.0]
Maximum speed	2.0

The *Data*_9_2 dataset (sometimes referred to as the *st*900_2_9 dataset) contains 900 patterns in $\mathbb{R}^2$ and 9 clusters. Configuration parameters for experiments with this data set were:

Table 5. Configuration parameters for experiments with the *Data_9_2* dataset

Parameter	Value
Swarm size (in particles)	25
Search-space dimensions	2
Inertia factor	0.9
Cognitive factor weight	1.8
Social factor weight	1.2
Search-space limits	[−3.5 3.5]
Maximum speed	1.5

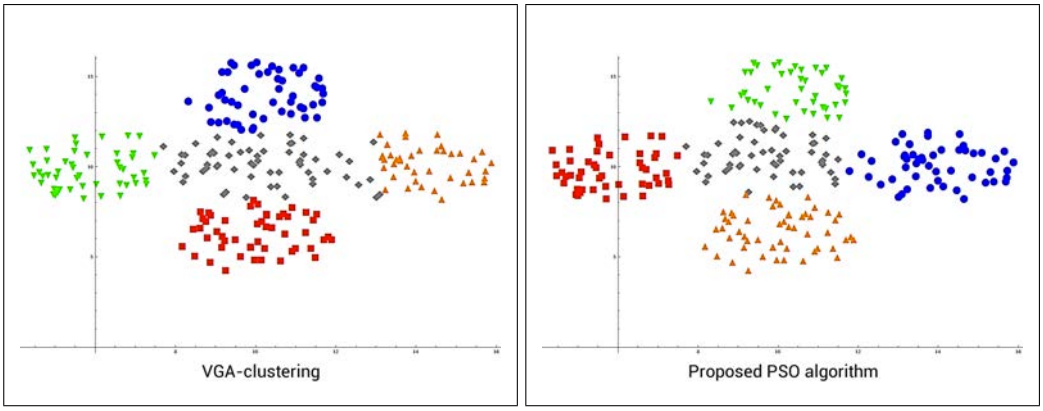

Fig. 1. Best clustering results of each algorithm for all *Data_5_2* experiments.

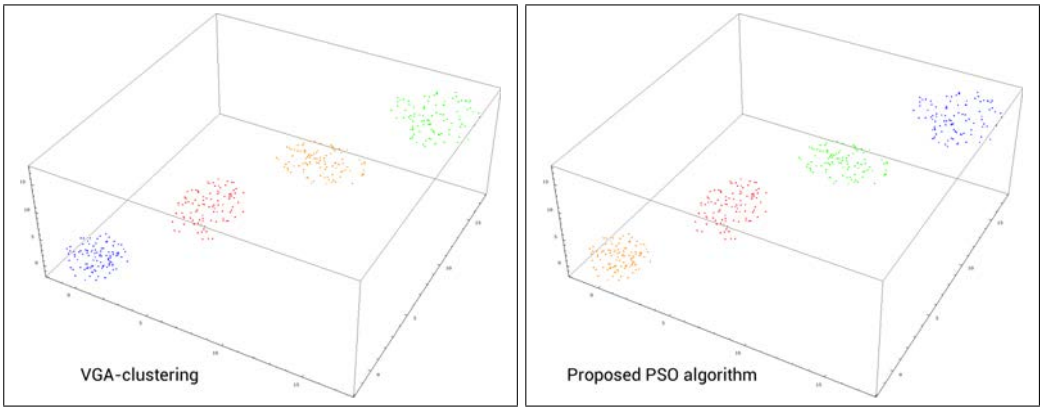

Fig. 2. Best clustering results of each algorithm for all *Data_4_3* experiments.

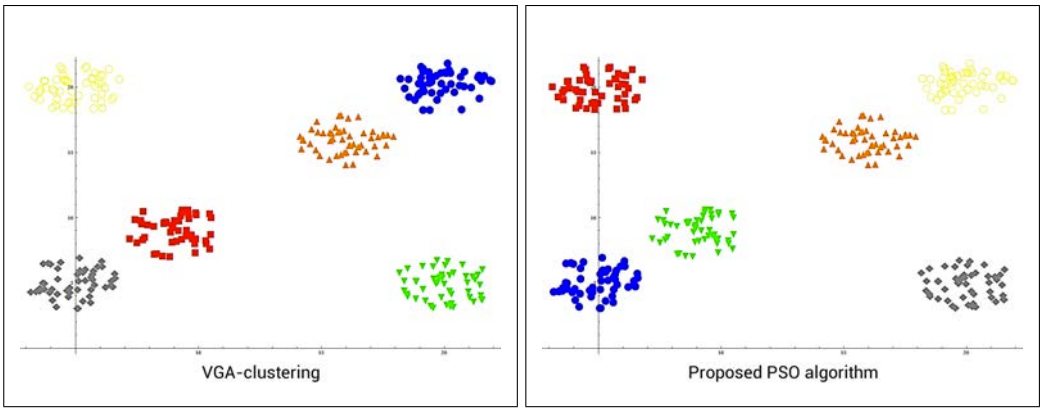

Fig. 3. Best clustering results of each algorithm for all *Data*_6_2 experiments.

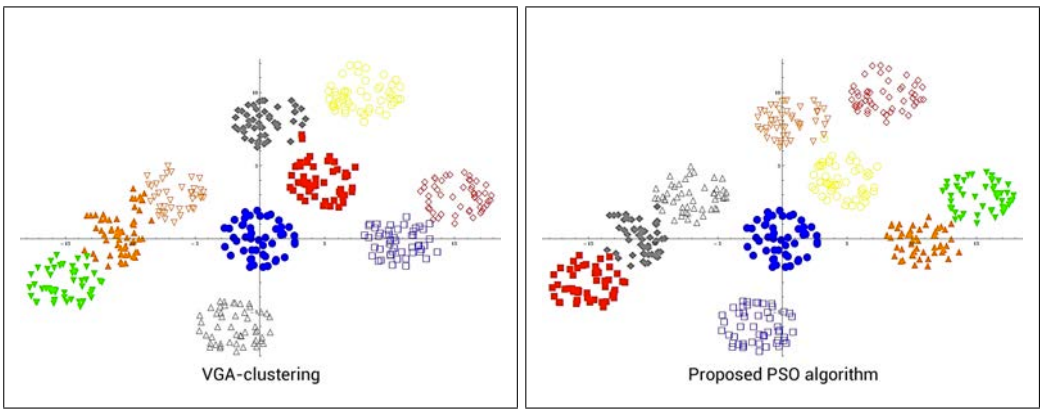

Fig. 4. Best clustering results of each algorithm for all *Data*_10_2 experiments.

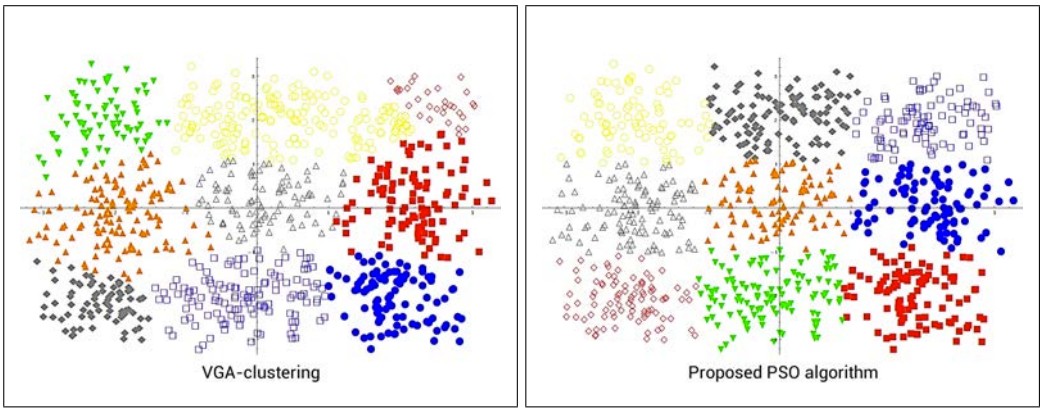

Fig. 5. Best clustering results of each algorithm for all *Data_9_2* experiments.

Table 6. Average execution time and number of iterations of the proposed PSO and VGA-clustering algorithms.

Dataset	Algorithm	Number of patterns	Execution time	Number of iterations
Data_5_2	*Proposed PSO*	250	2.70 seconds	194
Data_5_2	$VGA - clustering$	250	4.61 seconds	395
Data_4_3	*Proposed PSO*	400	12.74 seconds	196
Data_4_3	$VGA - clustering$	400	21.36 seconds	274
Data_6_2	*Proposed PSO*	300	55.85 seconds	912
Data_6_2	$VGA - clustering$	300	68.74 seconds	954
Data_10_2	*Proposed PSO*	500	52.88 seconds	497
Data_10_2	$VGA - clustering$	500	66.02 seconds	519
Data_9_2	*Proposed PSO*	900	54.57 seconds	196
Data_9_2	$VGA - clustering$	900	101.98 seconds	273

5 Conclusions

Particle Swarm Optimization offers some particular advantages over other similar meta-heuristics. It requires only a small number of particles and yet it can explore very large search-spaces, therefore it poses as a good candidate for data-mining and huge data volume problems. Besides, PSO is not vulnerable to poorly set initial conditions which make it ideal for clustering and classification tasks.

This paper presented a modified PSO algorithm which uses a global variance measure for solving multivariate data clustering problems. Five different and widely known datasets were used for testing the proposed algorithm. Each experiment was run 100 times and average results (in table 6) show it to be precise and efficient. In all datasets the proposed PSO algorithm shows better clustering results than the VGA-clustering algorithm.

Experimental results show how a small number of particles (or candidate solutions) can successfully solve any data clustering problem. Also, as shown by those results, even a dramatic increase in the dataset size can only induce a very small increase in the number of particles needed for a specific problem, and the observed variation between the execution time with the smaller and the bigger dataset is non-relevant.

References

[1] Abbas, H.M., Fahmy, M.M.: Neural Networks for Maximum Likelihood Clustering, Signal Processing, vol. 36(1), 111-126 (1994).

[2] Bandyopadhyay, S., Maulik, U.: Nonparametric genetic clustering: Comparison validity indices, IEEE Transactions on Systems, Man and Cybernetics, Part C, vol. 31, no. 1, 120-125 (2001).

[3] Bandyopadhyay, S., Maulik, U.: Genetic Clustering for Automatic Evolution of Clusters and Application to Image Classification, Pattern Recognition, vol.35, 1197-1208 (2002).

[4] Bandyopadhyay, S., Murthy, C. A., Pal, S. K.: Pattern Classification Using Genetic Algorithms, Pattern Recognition Letters, vol. 16, 801-808 (1995).

[5] Bandyopadhyay, S., Pal, S. K.: Classification and Learning Using Genetic Algorithms: Applications in Bioinformatics and Web Intelligence, Springer, Heidelberg (2007).

[6] Cai, W., Chen, S., Zhang, D.: A simultaneous learning framework for clustering and classification, Pattern Recognition 42 (7), 1248-1259 (2009).

[7] Coleman, G.B., Andrews, H.C.: Image Segmentation by Clustering, Proc.IEEE, vol. 67, 773-785 (1979).

[8] Cui, X., Potok, T.E.: Document clustering analysis based on hybrid PSO + K-means algorithm. Journal of Computer Sciences (Special Issue), 27-33 (2005) ISSN 1549-3636.

[9] Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1, 224âĂŞ227 (1979).

[10] Ester, M., Kriegel, H.P., Sander, J., and Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, OR, 226-231 (1996).

[11] Hamerly, G., Elkan, C.: Learning the K in K-means, 7th Annual Conference on Neural Information Processing Systems (2003).
[12] Jain, A.K., Duin, R., Mao, J.: Statistical Pattern Recognition: A Review, IEEE Transactions on Pattern Analysis and Machine Intellgence, vol. 22 (1), 4-37 (2000).
[13] Judd, D. , Mckinley, P., Jain, A.K.: Large-scale Parallel Data Clustering, IEEE Transactions on Pattern Analysis and Machine Intellgence, vol. 20 (8), 871-876 (1998).
[14] Kennedy, J., Eberhart, R.: Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, vol. 4, 1942-1948 (1995).
[15] Kennedy, J., Eberhart, R.: Swarm Intelligence, Morgan Kaufmann (2001).
[16] Lee, C.-Y., Antonsson, E.K.: Dynamic Partitional Clustering Using Evolution Strategies, In The Third Asia-Pacific Conference on Simulated Evolution and Learning (2000).
[17] Omran, M., Engelbrecht, A.P., Salman, A.: Particle swarm optimization method for image clustering. International Journal of Pattern Recognition and Artificial Intelligence 19(3), 297-322 (2005).
[18] Omran, M., Salman, A., Engelbrecht, A.P.: Image classification using particle swarm optimization. In: Conference on Simulated Evolution and Learning, vol. 1, pp. 370-374 (2002).
[19] Ray, S., Turi, R.H.: Determination of Number of Clusters in K-Means Clustering and Application in Colour Image Segmentation, Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99), Calcutta, India, 137-143 (1999).
[20] Tou, J.T., Gonzalez, R.C.: Pattern Recognition Principles, Addison-Wesley, Reading, MA (1974).
[21] Van der Merwe, D.W., Engelbrecht, A.P.: Data clustering using particle swarm optimization. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, Piscataway, NJ, pp. 215-220 (2003).
[22] Xiao, X., Dow, E.R., Eberhart, R.C., Miled, Z.B., Oppelt, R.J.: Gene clustering using self-organizing maps and particle swarm optimization. In: Proc. of the 17th International Symposium on Parallel and Distributed Processing (PDPS 2003). IEEE Computer Society, Washington (2003).
[23] Xie, X., Beni, G.: Validity measure for fuzzy clustering. IEEE Trans. Pattern Anal. Machine Learning 3, 841-846 (1991)