Pattern Anal Applic (2006) 8: 332–344
DOI 10.1007/s10044-005-0015-5

THEORETICAL ADVANCES

Mahamed G. H. Omran · Ayed Salman
Andries P. Engelbrecht

Dynamic clustering using particle swarm optimization with application in image segmentation

Received: 9 June 2005 / Accepted: 25 September 2005 / Published online: 24 November 2005
© Springer-Verlag London Limited 2005

Abstract A new dynamic clustering approach (DCPSO), based on particle swarm optimization, is proposed. This approach is applied to image segmentation. The proposed approach automatically determines the "optimum" number of clusters and simultaneously clusters the data set with minimal user interference. The algorithm starts by partitioning the data set into a relatively large number of clusters to reduce the effects of initial conditions. Using binary particle swarm optimization the "best" number of clusters is selected. The centers of the chosen clusters is then refined via the K-means clustering algorithm. The proposed approach was applied on both synthetic and natural images. The experiments conducted show that the proposed approach generally found the "optimum" number of clusters on the tested images. A genetic algorithm and random search version of dynamic clustering is presented and compared to the particle swarm version.

Keywords Unsupervised clustering · Clustering validation · Particle swarm optimization · Image segmentation

1 Introduction

Data clustering is the process of identifying natural groupings or clusters, within multidimensional data, based on some similarity measure (e.g., Euclidean distance) [1, 2]. Clustering algorithms are used in many applications, such as data mining [3], compression, [4] image segmentation [5–7], machine learning [8], etc. A cluster is usually identified by a cluster center (or centroid) [9]. Data clustering is a difficult problem as the clusters in data may have different shapes and sizes [2]. Furthermore, it is usually not known how many clusters should be formed [10].

Most clustering algorithms are based on two popular techniques known as hierarchical and partitional clustering [11, 12]. In hierarchical clustering, the output is "a tree showing a sequence of clustering with each clustering being a partition of the data set" [12]. Such algorithms have the following advantages [11]:

- The number of clusters need not be specified a priori
- They are independent of the initial conditions

However, hierarchical clustering techniques suffer from the following drawbacks:

- They are static, i.e., data points assigned to a cluster cannot move to another cluster.
- They may fail to separate overlapping clusters due to a lack of information about the global shape or size of the clusters.

On the other hand, partitional clustering algorithms partition the data set into a specified number of clusters. These algorithms try to minimize certain criteria (e.g., a square error function) and can therefore be treated as optimization problems. The advantages of hierarchical algorithms are the disadvantages of the partitional algorithms and vice versa. Partitional clustering techniques are more popular than hierarchical techniques in pattern recognition [2], hence, this paper will concentrate on partitional techniques.

Partitional clustering aims to optimize cluster centers, as well as the number of clusters [10]. Most clustering algorithms require the number of clusters to be specified in advance [9, 10]. Finding the "optimum" number of clusters in a data set is usually a challenge since it requires a priori knowledge, and/or ground truth about the data, which is not always available. The problem of finding the optimum number of clusters in a data set has

M. G. H. Omran · A. P. Engelbrecht (✉)
Department of Computer Science,
School of Information Technology,
University of Pretoria, Pretoria 0002, South Africa
E-mail: engel@cs.up.ac.za
Tel.: +27-12-4203578
Fax: +27-12-3625188

A. Salman
Department of Computer Engineering, Kuwait University,
Kuwait, Kuwait

been the subject of several research efforts [13, 14], however, despite the amount of research in this area, the outcome is still unsatisfactory [15].

This paper proposes a new approach called dynamic clustering using a particle swarm optimization algorithm (DCPSO). The approach uses some of the ideas presented by Kuncheva and Bezdek [16] which uses genetic algorithms as a nearest prototype classifier. DCPSO automatically determines the "optimum" number of clusters and simultaneously clusters the data set with minimal user interference. The algorithm starts off by partitioning the data set into a relatively large number of clusters in order to reduce the effects of initial conditions. Using binary particle swarm optimization (PSO), the "optimal" number of clusters is selected. The centers of the chosen clusters are then refined using the K-means clustering algorithm. The binary PSO is applied again on the cluster centers to find a new "optimal" number of clusters in the data set. This process is repeated until convergence is reached.

The remainder of the paper is organized as follows: Sect. 2 surveys related work. The DCPSO algorithm is presented in Sect. 3, while an experimental evaluation of DCPSO in the area of image segmentation is provided in Sect. 4, using synthetic and natural images. Furthermore, DCPSO is compared to other clustering algorithms. Finally, Sect. 5 concludes the paper.

2 Related work

The most widely used partitional algorithm is the iterative K-means approach [17]. The K-means algorithm starts with K cluster *centroids*, which are initialized to random values or values derived from a priori information. Furthermore, each point in the data set is assigned to the cluster closest to it (i.e., closest centroid). The centroids are then recalculated according to their associated points. This process is repeated until convergence is reached [17]. The K-means algorithm suffers from the following drawbacks [18]:

- The algorithm is data dependent.
- It is a greedy algorithm that depends on the initial conditions, which may result in the algorithm converging to suboptimal solutions.
- The number of clusters needs to be specified in advance.

The K-means is a crisp (i.e., hard) clustering algorithm with each data point being assigned to only one cluster. Bezdek [19, 20] has developed a fuzzy version of K-means, called Fuzzy C-Means (FCM), in which each data point belongs to each cluster with some degree of membership.

Another popular clustering algorithm is the expectation-maximization (EM) algorithm [21–23]. EM is used for parameter estimation in the presence of some unknown data [24]. EM partitions the data set into clusters by determining a mixture of Gaussians fitting the data set. Each Gaussian has a mean and covariance matrix [25]. Zhang et al. [26] used EM along with a hidden Markov random field (HMRF) for the segmentation of brain magnetic resonance (MR) images.

Zhang et al. [27, 28] proposed a novel algorithm called K-harmonic means (KHM) with promising results. KHM computes the harmonic mean of the distance of each cluster center to every data point and updates the cluster centers accordingly. KHM is less sensitive to the initial conditions (contrary to K-means) and does not have the problem of collapsing Gaussians exhibited by EM [25].

Recently, Omran et al. [29] proposed a PSO-based clustering algorithm. The algorithm finds the centroids of a user-specified number of clusters, where each cluster groups together similar patterns. According to conducted experiments, the proposed approach outperforms K-means, FCM and KHM.

The above clustering algorithms (along with most partitional clustering algorithms [10]) require the number of clusters, K, to be given in advance [30]. Knowing the number of clusters within a data set, in advance, is a difficult problem by itself [10].

2.1 Unsupervised clustering algorithms

ISODATA, proposed by Ball and Hall [31], is an enhancement of the K-means algorithm, with addition of the possibility of merging classes and splitting elongated classes. An alternative approach to ISODATA is SYNERACT [32]. SYNERACT combines K-means with hierarchical descending approaches to overcome the three drawbacks of K-means mentioned previously. Three concepts used by SYNERACT are:

- A hyperplane to split up a cluster into two smaller clusters and compute their centroids
- Iterative clustering to assign pixels into available clusters
- A binary tree to store clusters generated from the splitting process

According to Huang [32], SYNERACT is faster than and almost as accurate as ISODATA. Furthermore, it does not require the number of clusters and initial location of centroids to be specified in advance. Another improvement to the K-means algorithm is proposed by Rosenberger and Chehdi [15], which automatically finds the number of clusters in an image set by using intermediate results. Furthermore, Pelleg and Moore [33] proposed a K-means based algorithm, called X-means, that uses model selection. X-means searches over a range of values of K and selects the value with the best Bayesian information criterion (BIC) [34] score. Recently, Hamerly and Elkan [10] proposed another wrapper around K-means called G-means. G-means starts with a small value for K, and with each iteration splits up the clusters whose data do not fit a Gaussian distribution. Between each round of splitting, K-means is applied to the entire data set in order to refine the current solution. According to Hamerly and Elkan [10] and Hamerly [35], G-means works better than X-means, however, it works only for data having spherical and/or elliptical clusters. G-means is not designed to work for arbitrary-shaped clusters [35]. A program called *snob* [36, 37] uses various methods to assign objects to classes in an intelligent manner [38]. After each assignment, a means of model selection called the Wallace information measure [also known as the minimum message length (MML)] [39, 40] is calculated and based on this calculation the assignment is accepted or rejected. Snob can split/merge and move points between clusters, thereby allowing it to determine the number of clusters in a data set. Bischof et al. [41] proposed another algorithm based on K-means which uses a minimum description length (MDL) framework. The algorithm starts with a large value for K and proceeds to remove centroids when this results in a reduction of the description length. K-means is used between the steps that reduce K.

Gath and Geva [42] proposed an unsupervised clustering algorithm based on the combination of FCM and fuzzy maximum likelihood estimation. Lorette et al. [43] proposed an algorithm based on fuzzy clustering to dynamically determine the number of clusters in a data set. This approach, however, requires a parameter to be specified, which has a profound effect on the number of clusters generated (i.e., not fully unsupervised). Similarly, Boujemaa [44] proposed an algorithm based on a generalization of the competitive agglomeration clustering algorithm introduced by Frigui and Krishnapuram [45].

The algorithms found within the above paragraph try to modify the objective functions of FCM. These approaches are still sensitive to initialization and other parameters [30]. Frigui and Krishnapuram [30] proposed a robust competitive clustering algorithm, based on the process of competitive agglomeration. Initialization does, however, have a significant effect on the result of this algorithm.

Kohonen's self-organizing maps (SOM) [46–48] can be used to automatically find the number of clusters in a data set. SOM combines competitive learning (in which different nodes in the Kohonen network compete to be the winners when an input pattern is presented) with a topological structuring of nodes, such that adjacent nodes tend to have similar weight vectors (this is done via lateral feedback) [47, 48]. SOM suffers from being dependent on the order in which the data points are presented. To overcome this problem, the choice of data points can be randomized during each epoch [48].

Lee and Antonsson [9] used an evolution strategy (ES) to dynamically cluster a data set. The proposed ES implemented variable length genomes to search for both the centroids and K.

2.2 Clustering validation techniques

The main subject of cluster validation is "the evaluation of clustering results to find the partitioning that best fits the underlying data" [13]. Hence, cluster validity approaches are approaches used to quantitatively evaluate the result of a clustering algorithm [13]. These approaches have representative indices, called *validity indices*. The traditional approach to determine the "optimum" number of clusters is to run the algorithm repetitively using different input values and select the partitioning of data resulting in the best validity measure [49].

Two criteria that have been widely considered sufficient in measuring the quality of partitioning a data set into a number of clusters are: [13]

- *Compactness*: samples in one cluster should be similar to each other and different from samples in other clusters. An example of this would be the variance of a cluster.
- *Separation*: clusters should be well-separated from each other. An example of this criterion is the Euclidean distance between the centroids of clusters.

There are several relative validity indices; for a thorough survey in this field refer to Halkidi et al. [13]. Dunn [50] proposed a cluster validity index that identifies compact and well-separated clusters. The main goal of Dunn's index is to maximize inter-cluster distances (i.e., separation) while minimizing intra-cluster distances (i.e., increase compactness). Dunn's index suffers from two problems in that it is computationally expensive and sensitive to the presence of noise [13]. Another well-known index, proposed by Davies and Bouldin [51] minimizes the average similarity between each cluster and the one most similar to it. Recently, Turi [38] proposed an index incorporating a multiplier function (to penalize the selection of a small number of clusters) to the ratio between intra-cluster and inter-cluster distances, with some promising results. Two recent validity indices are *S_Dbw* [49] and *CDbw* [52]. In *S_Dbw*, the compactness of a data set is measured by the cluster variance, whereas the separation is measured by the density between clusters. Halkidi and Vazirgiannis [49] showed that, in tested cases, *S_Dbw* successfully found the "optimal" number of clusters whereas other well-known indices often failed to do so. However, *S_Dbw* does not work properly for arbitrary shaped clusters [49]. To address this problem, Halkidi and Vazirgiannis [52] proposed a multirepresentative validity index, *CDbw*, in which each cluster is represented by a user-specified number of points, instead of one representative as is done in *S_Dbw*. Furthermore, *CDbw* uses intra-cluster density to measure the compactness of a data set, and uses the density between clusters to measure its separation [52].

2.3 Particle swarm optimization

Particle swarm optimizers (PSO) are population-based optimization algorithms modeled after the simulation of social behavior of bird flocks [53, 54]. PSO is generally considered to be an evolutionary computation (EC) paradigm. Other EC paradigms include genetic algorithms

(GA), genetic programming (GP), evolutionary strategies (ES), and evolutionary programming (EP) [55]. These approaches simulate biological evolution and are population based. In a PSO system, a swarm of individuals (called *particles*) fly through the search space. Each particle represents a candidate solution to the optimization problem. The position of a particle is influenced by the best position visited by itself (i.e., its own experience) and the position of the best particle in its neighborhood (i.e., the experience of neighboring particles). When the neighborhood of a particle is the entire swarm, the best position in the neighborhood is referred to as the global best particle, and the resulting algorithm is referred to as a *gbest* PSO. When smaller neighborhoods are used, the algorithm is generally referred to as a *lbest* PSO [56]. The performance of each particle (i.e., how close the particle is from the global optimum) is measured using a fitness function that varies depending on the optimization problem.

Each particle in the swarm is represented by the following characteristics:

- $\boldsymbol{x}_i$: The *current position* of the particle.
- $\boldsymbol{v}_i$: The *current velocity* of the particle.
- $\boldsymbol{y}_i$: The *personal best position* of the particle.

The personal best position of particle i is the best position (i.e., one resulting in the best fitness value) visited by particle i so far. Let f denote the objective function that has to be minimized. Then the personal best of a particle at time step t is updated as

$$\boldsymbol{y}_i(t+1) = \left\{ \begin{array}{ll} \boldsymbol{y}_i(t) & \text{if } f(\boldsymbol{x}_i(t+1)) \geqslant f(\boldsymbol{y}_i(t)) \\ \boldsymbol{x}_i(t+1) & \text{if } f(\boldsymbol{x}_i(t+1)) < f(\boldsymbol{y}_i(t)) \end{array} \right\}. \quad (1)$$

Two main approaches to PSO exist, namely *lbest* and *gbest*, the difference being the neighborhood topology used to exchange experience among particles. For *gbest*, the best particle is determined from the entire swarm. If the position of the global best particle is denoted by the vector $\hat{\boldsymbol{y}}$, then

$$\begin{aligned}\hat{\boldsymbol{y}}(t) &\in \{\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_s\} \\ &= \min\{f(\boldsymbol{y}_0(t)), f(\boldsymbol{y}_1(t)), \ldots, f(\boldsymbol{y}_s(t))\},\end{aligned} \quad (2)$$

where s denotes the size of the swarm. For the *lbest* model, a swarm is divided into overlapping neighborhoods of particles. For each neighborhood N_j, a best particle is determined with position $\hat{\boldsymbol{y}}_j$. This particle is referred to as the *neighborhood best* particle, defined as

$$\hat{\boldsymbol{y}}_j(t+1) \in \{N_j | f(\hat{\boldsymbol{y}}_j(t+1)) = \min\{f(\boldsymbol{y}_j(t))\}, \forall \boldsymbol{y}_j \in N_j\}, \quad (3)$$

where

$$\begin{aligned}N_j = \{&\boldsymbol{y}_{i-l}(t), \boldsymbol{y}_{i-l+1}(t), \ldots, \boldsymbol{y}_{i-1}(t), \boldsymbol{y}_i(t), \\ &\boldsymbol{y}_{i+1}(t), \ldots, \boldsymbol{y}_{i+l-1}(t), \boldsymbol{y}_{i+l}(t)\}.\end{aligned} \quad (4)$$

Neighborhoods are usually determined using particle indices, [57] however, topological neighborhoods can also be used [57]. It is clear that *gbest* is a special case of *lbest* with $l = s$; that is, the neighborhood is the entire swarm. While the *lbest* approach results in a larger diversity, it is still slower than the *gbest* approach.

For each iteration of a PSO algorithm, the velocity $\boldsymbol{v}_i$ update step is specified for each dimension $j \in 1, \ldots, N_d$, where N_d is the dimension of the problem. Hence, $v_{i,j}$ represents the jth element of the velocity vector of the ith particle. Thus, the velocity of particle i is updated as using the following equation:

$$\begin{aligned}v_{i,j}(t+1) = w v_{i,j}(t) &+ c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) \\ &+ c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t)),\end{aligned} \quad (5)$$

where w is the inertia weight, [58] c_1 and c_2 are the acceleration constants and $r_{1,j} \sim U(0,1)$ and $r_{2,j} \sim U(0,1)$. Equation (5) consists of three components, namely

- The *inertia weight* term, w, which serves as a memory of previous velocities. The inertia weight controls the impact of the previous velocity: a large inertia weight favors exploration, while a small inertia weight favors exploitation [56].
- The cognitive component, $y_i(t) - x_i(t)$, which represents the particle's own experience as to where the best solution is.
- The social component, $\hat{\boldsymbol{y}}(t) - \boldsymbol{x}_i(t)$, which represents the belief of the entire swarm as to where the best solution is. Different social structures have been investigated [59, 60], with the star topology being used most.

The position of particle i, $\boldsymbol{x}_i$, is then updated using the following equation:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1). \quad (6)$$

The reader is referred to Van den Bergh [61] and Van den Bergh et al. [62] for a study of the relationship between the inertia weight and acceleration constants, in order to select values which will ensure a convergent behavior. Velocity updates can also be clamped through a user defined maximum velocity, $V_{\max}$, which would prevent them from exploding, thereby causing premature convergence [61].

The PSO algorithm performs the update equations above, repeatedly, until a specified number of iterations have been exceeded, or velocity updates are close to zero. The quality of particles is measured using a fitness function which reflects the optimality of a particular solution.

Kennedy and Eberhart [63] have adapted PSO to search in binary space. In binary PSO, the component values of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are restricted to the set {0, 1}. The velocity, $\boldsymbol{v}_i$ is interpreted as a probability to change a bit from 0 to 1, or from 1 to 0 when updating the position of particles. This can be done using a sigmoid function, defined as

$$\mathrm{sig}(x) = \frac{1}{1 + \mathrm{e}^{-x}}. \quad (7)$$

Hence, the equation for updating positions (6) is replaced by the probabilistic update equation, namely [61]:

$$x_{i,j}(t+1) = \begin{cases} 0 & \text{if } r_j(t) \geq \mathrm{sig}(v_{i,j}(t+1)) \\ 1 & \text{if } r_j(t) < \mathrm{sig}(v_{i,j}(t+1)) \end{cases}, \tag{8}$$

where $r_j \sim U(0,1)$. This paper makes use of a binary PSO to optimize the number of clusters.

2.4 Image segmentation

Image segmentation is a fundamental component in many computer vision applications [1]. Image segmentation is defined as the process of subdividing an image into its constituent parts and extracting the desired parts [38]. There are many techniques for image segmentation in the literature [64, 65], of which one of the simplest techniques is to use a clustering algorithm (e.g., K-means). In this paper, the proposed algorithm is applied in the field of image segmentation in order to show its potential.

3 The DCPSO algorithm

This section presents the DCPSO algorithm. For this purpose define the following symbols:

- N_c is the maximum number of clusters.
- N_p is the number of data vectors to be clustered.
- N_d is the dimension of the data set (i.e., the number of attributes of each vector).
- $\boldsymbol{Z} = \{z_{j,p} \in \Re | j = 1, \ldots, N_d \text{ and } p = 1, \ldots, N_p\}$ is the set of data points.
- $\boldsymbol{M} = \{m_{j,k} \in \Re | j = 1, \ldots, N_d \text{ and } k = 1, \ldots, N_c\}$ is the set of N_c cluster centroids.
- $\boldsymbol{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_s\}$ is the swarm of s particles such that $\boldsymbol{x}_i$ indicates particle i, with $x_{ik} \in \{0,1\}$ for $k=1,\ldots, N_c$ such that if $x_{ik}=1$ then the corresponding $\boldsymbol{m}_k$ in $\boldsymbol{M}$ has been chosen to be part of the solution proposed by $\boldsymbol{x}_i$. Otherwise, if $x_{ik}=0$ then the corresponding $\boldsymbol{m}_k$ in $\boldsymbol{M}$ is not part of the solution proposed by $\boldsymbol{x}_i$.
- n_i is the number of clusters used by the clustering solution represented by particle $\boldsymbol{x}_i$ such that

$$n_i = \sum_{k=1}^{N_c} x_{ij}, \text{ with } n_i \leq N_c.$$

- $\boldsymbol{M}_i$ is the clustering solution represented by particle $\boldsymbol{x}_i$ such that $\boldsymbol{M}_i = (\boldsymbol{m}_k)\ \forall\, k : x_{i,k} = 1$ with $\boldsymbol{M}_i \subseteq \boldsymbol{M}$.
- n_τ is the number of clusters used by the clustering solution represented by the global best particle $\hat{\boldsymbol{y}}$ such that

$$n_\tau = \sum_{k=1}^{N_c} \hat{y}_k, \text{ with } n_\tau \leq N_c.$$

- $\boldsymbol{M}_\tau$ is the clustering solution represented by $\hat{\boldsymbol{y}}$ such that $\boldsymbol{M}_\tau = (\boldsymbol{m}_k) \forall k\!: \hat{y}_k = 0$ with $\boldsymbol{M}_\tau \subseteq \boldsymbol{M}$.
- $\boldsymbol{M}_r$ is the set of centroids in $\boldsymbol{M}$ which have not been chosen by $\hat{\boldsymbol{y}}$ such that $\boldsymbol{M}_r = (\boldsymbol{m}_k)\ \forall\, k\!: \hat{y}_k = 0$ with $\boldsymbol{M}_r \subseteq \boldsymbol{M}$(i.e. $\boldsymbol{M}_r \cap \boldsymbol{M}_\tau = \emptyset$; and $\boldsymbol{M}_r \cup \boldsymbol{M}_\tau = \boldsymbol{M}$).
- p_{ini} is a user-specified probability defined in Kuncheva and Bezdek [16] which is used to initialize a particle position, $\boldsymbol{x}_i$, as follows:

$$x_{i,k}(t) = \begin{cases} 0 & \text{if } r_k(t) \geqslant p_{\text{ini}} \\ 1 & \text{if } r_k(t) < p_{\text{ini}} \end{cases}, \tag{9}$$

where $r_k\ (t) \sim\ U(0,1)$. Obviously a large value for p_{ini} results in selecting most of the centroids in $\boldsymbol{M}$.

The algorithm works as follows: a pool of cluster centroids, $\boldsymbol{M}$, is randomly chosen from $\boldsymbol{Z}$. The swarm of particles, $\boldsymbol{S}$, is then randomly initialized. Binary PSO is then applied to find the "best" set of cluster centroids, $\boldsymbol{M}_\tau$, from $\boldsymbol{M}$. K-means is applied to $\boldsymbol{M}_\tau$ in order to refine the chosen centroids. $\boldsymbol{M}$ is then set to $\boldsymbol{M}_\tau \cup \boldsymbol{M}_r$ which a randomly chosen set of centroids from $\boldsymbol{Z}$ (to add diversity to $\boldsymbol{M}$). The algorithm is then repeated using the new $\boldsymbol{M}$. When the termination criteria are met, $\boldsymbol{M}_\tau$ will be the resulting "optimum" set of cluster centroids and n_τ will be the "optimum" number of clusters in $\boldsymbol{Z}$.

The DCPSO algorithm is summarized below:

1. Select $\boldsymbol{m}_k \in \boldsymbol{M}\ \forall\ k = 1, \ldots, N_c$, where $1 < N_c < N_p$, randomly from $\boldsymbol{Z}$.
2. Initialize the swarm $\boldsymbol{S}$, with $x_{i,k} \sim U\ \{0,1\}\ \forall\ i=1,\ldots, s$ and $k=1,\ldots, N_c$ using (9).
3. Randomly initialize the velocity $\boldsymbol{v}_i$ of each particle i in $\boldsymbol{S}$ such that $v_{i,k} \in [-5,5]\ \forall\ i=1,\ldots, s$ and $k=1,\ldots, N_c$. The range of $[-5,5]$ was set empirically.
4. For each particle $\boldsymbol{x}_i$ in $\boldsymbol{S}$.
   (a) Partition $\boldsymbol{Z}$ according to the centroids in $\boldsymbol{M}_i$ by assigning each data point to the closest (in terms of the Euclidean distance) cluster in $\boldsymbol{M}_i$.
   (b) Calculate the clustering validity index, VI, using one of the clustering validity indices as defined in Sect. 3.1 to measure the quality of the resulting partitioning of $\boldsymbol{Z}$ (i.e., $VI=V$, $VI=S\_Dbw$ or $VI=1/D$ since D should be maximized).
   (c) $\mathrm{f}(\boldsymbol{x}_i) = VI$
5. Apply the binary PSO velocity and position update equations (5) and (8) on all particles in $\boldsymbol{S}$.
6. Repeat steps 4 and 5 until the termination criteria are met.
7. Adjust $\boldsymbol{M}_\tau$ by applying the K-means clustering algorithm.
8. Randomly re-initialize $\boldsymbol{M}_r$ from $\boldsymbol{Z}$.
9. Set $\boldsymbol{M} = \boldsymbol{M}_r \cup \boldsymbol{M}_\tau$.
10. Repeat steps 2–9 until termination criteria are met.

The termination criteria can be a user-defined maximum number of iterations or a lack of progress in improving the best solution found so far for a user-specified consecutive number of iterations, TC. In this paper, the latter approach is used with $TC=50$ for step 6 and $TC=2$ for step 10. These values for TC were set empirically.

p_{ini}, w, c_1, c_2, N_c, and s are user-defined parameters. Large values for N_c and s are recommended to find a good solution.

A GA version of DCPSO can easily be implemented by replacing step 5 in the above algorithm with GA evolutionary operators for selection, crossover, and mutation [66]. On the other hand, a random search (RS) version of DCPSO, as described by Kunchevea and Bezdek [16] can be implemented by removing step 5 and keeping only the best solution encountered so far.

As an illustration of the DCPSO algorithm, consider the following example.

Example 1. Let $N_p = 100$, $N_d = 1$, and $N_c = 5$.

Let $\boldsymbol{M}$ be

3	29	78	150	200

An example of a particle position, $\boldsymbol{x}_i$, is

0	1	1	0	1

Which means that cluster centers 29, 78, and 200 are chosen for this particle such that $\boldsymbol{M}_i$ is

29	78	200

In other words, all data in $\boldsymbol{Z}$ are grouped in only three clusters.

After step 6, assume the global best particle, $\hat{\boldsymbol{y}}$, is

0	1	0	1	1

Then, $\boldsymbol{M}_\tau$ is

29	150	200

Apply K-means on $\boldsymbol{Z}$ using the $\boldsymbol{M}_\tau$ centroids, then $\boldsymbol{M}_\tau$ (for example) will be

30.5	129.9	201

Then, randomly initialize the remaining $N_c - n_\tau$ (i.e., $5-3=2$) clusters, representing $\boldsymbol{M}_r$, from $\boldsymbol{Z}$ (shown below in bold). The resultant $\boldsymbol{M}$ may look as follows:

110	30.5	**8**	129.9	201

The DCPSO algorithm is then repeated using the new $\boldsymbol{M}$.

3.1 Validity index

One of the advantages of the DCPSO is that it can use any validity index. Therefore, the user can choose the validity index suitable for his/her data set. In addition, any new index can easily be integrated with DCPSO. The validity indices used in this paper are briefly described below. In the sequel, K is the number of clusters and $\boldsymbol{C}_k$ is the kth cluster.

3.1.1 Dunn's index [50]

$$D = \min_{k=1,\ldots,K}\left\{\min_{kk=k+1,\ldots,K}\left(\frac{\operatorname{dist}(\boldsymbol{C}_k,\boldsymbol{C}_{kk})}{\max\limits_{a=1,\ldots,k}\operatorname{diam}(\boldsymbol{C}_a)}\right)\right\}, \tag{10}$$

where $\operatorname{dist}(\boldsymbol{C}_k,\boldsymbol{C}_{kk})$ is the dissimilarity function between two clusters $\boldsymbol{C}_k$ and $\boldsymbol{C}_{kk}$ defined as

$$\operatorname{dist}(\boldsymbol{C}_k,\boldsymbol{C}_{kk}) = \min_{\boldsymbol{u}\in\boldsymbol{C}_k,\boldsymbol{w}\in\boldsymbol{C}_{kk}} d(\boldsymbol{u},\boldsymbol{w}),$$

where $d(\boldsymbol{u},\boldsymbol{w})$ is the Euclidean distance between $\boldsymbol{u}$ and $\boldsymbol{v}$; and $\operatorname{diam}(\boldsymbol{C})$ is the diameter of a cluster which can be defined as

$$\operatorname{diam}(\boldsymbol{C}) = \max_{\boldsymbol{u},\boldsymbol{w}\in\boldsymbol{C}} d(\boldsymbol{u},\boldsymbol{w}).$$

3.1.2 Validity index proposed by Turi [38]

$$V = (c \times \mathrm{N}(2,1) + 1) \times \frac{\text{intra}}{\text{inter}}, \tag{11}$$

where c is a user-specified parameter and N(2,1) is a Gaussian distribution with mean 2 and standard deviation of 1. The *intra* term is the average of all the distances between each data point and its cluster centroid $\boldsymbol{m}_k$ is defined as

$$\text{intra} = \frac{1}{N_p}\sum_{k=1}^{K}\sum_{\boldsymbol{u}\in\boldsymbol{C}_k}||\boldsymbol{u}-\boldsymbol{m}_k||^2.$$

This term is used to measure the compactness of the clusters. The *inter* term is the minimum distance between the cluster centroids which is defined as

$$\text{inter} = \min\left\{||\boldsymbol{m}_k - \boldsymbol{m}_{kk}||^2\right\} \ \forall\, k = 1,2,\ldots,K-1 \text{ and } kk = k+1,\ldots,K.$$

This term is used to measure the separation of the clusters.

3.1.3 S_Dbw validity index [49]

$$S\_Dbw = \operatorname{scat}(K) + Dens\_bw(K). \tag{12}$$

The first term is the average scattering of the clusters which a measure of compactness of the clusters. It is defined as

$$\operatorname{scat}(K) = \frac{1}{K}\sum_{k=1}^{K}\frac{\|\sigma(\boldsymbol{C}_k)\|}{\|\sigma(\boldsymbol{Z})\|},$$

where $\sigma(\boldsymbol{C}_k)$ is the variance of cluster $\boldsymbol{C}_k$ and $\sigma(\boldsymbol{Z})$ is the variance of the data set $\boldsymbol{Z}$. The term $||z||$ is defined as $||z|| = (z^{\mathrm{T}}z)^{1/2}$, where z is a vector.

The second term in (12) evaluates the density of the area between the two clusters in relation to the density of the two clusters which is a measure of the separation of the clusters. It is defined as

$$Dens\_bw(K) = \frac{1}{K(K-1)} \times \sum_{k=1}^{K} \left[\sum_{\substack{kk=1 \\ k \neq kk}}^{K} \frac{\text{density}(\boldsymbol{b}_{k,kk})}{\max\{\text{density}(\boldsymbol{C}_k), \text{density}(\boldsymbol{C}_{kk})\}} \right],$$

where $\boldsymbol{b}_{k,kk}$ is the middle point of the line segment defined by $\boldsymbol{m}_k$ and $\boldsymbol{m}_{kk}$. The term density($\boldsymbol{b}$) is defined as,

$$\text{density}(\boldsymbol{b}) = \sum_{ll=1}^{n_{k,kk}} (z_{ll}, \boldsymbol{b}),$$

where $n_{k,kk}$ is the number of data vectors in clusters $\boldsymbol{C}_k$ and $\boldsymbol{C}_{kk}$. The function $f(z,\boldsymbol{b})$ is defined as

$$f(z,\boldsymbol{b}) = \begin{cases} 0 & \text{if } d(z,\boldsymbol{b}) > \sigma \\ 1 & \text{otherwise} \end{cases},$$

where

$$\sigma = \frac{1}{K}\sqrt{\sum_{k=1}^{K} ||\sigma(\boldsymbol{C}_k)||}$$

3.2 Time complexity

The time complexity of DCPSO is based on the complexity of four processes, namely, the partitioning of Z, calculating the quality of the partition, applying binary PSO, and applying K-means. Assuming T_1 is the number of iterations taken by the PSO to converge (step 6 of the algorithm), and T_2 is the number of iterations taken by the DCPSO to converge (step 10 of the algorithm). Then the complexity of partitioning Z is $O(s\ T_1\ T_2\ N_{\mathrm{c}}\ N_{\mathrm{p}}\ N_{\mathrm{d}})$, while the complexity of calculating the quality of a partition will depend on the time complexity of the validity index which is some constant, q, multiplied by N_{p} for all the indices used in this paper except the Dunn index. So, the complexity of this step will be $O(q\ T_1\ T_2\ N_{\mathrm{p}})$. Finally, the complexity of K-means is $O(N_{\mathrm{p}})$. The parameters T_1,T_2, N_{c}, s, and ξ can be fixed in advance. Typically, T_1, T_2, N_{c}, s, ξ, $N_{\mathrm{d}} \ll N_{\mathrm{p}}$. Let ς be the multiplication of s, T_1, T_2, N_{c}, and N_{d} (i.e., $\varsigma = sT_1T_2N_{\mathrm{c}}N_{\mathrm{d}}$). If $\varsigma \ll N_{\mathrm{p}}$ then the time complexity of DCPSO will be $O(N_{\mathrm{p}})$. However, if $\varsigma \approx N_{\mathrm{p}}$ then the time complexity of DCPSO will be $O(N_{\mathrm{p}}^2)$.

4 Experimental results

Experiments were conducted using both synthetic images and natural images. A tool developed by the authors, called SIGT [67], was used to generate 15 different synthetic images for which the actual number of clusters was known in advance. These images have different numbers of clusters with varying complexities; they consist of well-separated clusters, overlapping clusters, or a combination of both. Three of the synthetic images are shown in Table 1 along with their histograms.

The following well-known images were used as examples of natural images: *Lenna*, *mandrill*, *jet*, and *peppers*. Furthermore, one MRI and one satellite image of Lake Tahoe (shown in Table 2) have been used to show the wide applicability of the proposed approach.

The remainder of this section is organized as follows: in Sect. 4.1 the DCPSO was applied on the synthetic images using the three validity indices described in Sect. 3. These results were compared with the unsupervised fuzzy approach (UFA) proposed by Lorette et al. [43] and the SOM approach. In Sect. 4.2 the same experiments were conducted on the natural images. In Sect. 4.3 the GA and RS version of the proposed approach were applied on the natural images and compared with the PSO version. Finally, the use of different PSO models (namely, *lbest*, *gbest*, and *lbest-to-gbest*) was investigated.

The results reported in this section are averages and standard deviations over 20 simulations. In addition, we start with a *lbest* implementation of the PSO (with zero-radius neighborhood) and linearly increase the neighborhood radius until a *gbest* implementation of the PSO is reached. In this paper, this approach is referred to as *lbest-to-gbest*-PSO. This hybrid approach is used in order to initially avoid being trapped in local optima, by using a *lbest* approach [57]. The algorithm then attempts to converge into the best solution found by the initial phase by using a *gbest* approach. Furthermore, if the best solution has not been improved after a user-specified number of iterations (50 iterations was used for all the experiments conducted) then the algorithm will terminate (step 6 of the algorithm, Sect. 3). For the index proposed by Turi [38] the parameter, c, was set to 25 in all experiments as recommended by Turi [38]. The DCPSO parameters were empirically set as follows: $N_{\mathrm{c}} = 20$, $p_{\mathrm{ini}} = 0.75$ and $s = 100$ for all experiments conducted. In addition, the PSO parameters were empirically set as follows: $w = 0.72$, $c_1 = c_2 = 1.49$, and $V_{\max} = 255$. For UFA, the user-defined parameter, ε, was set equal to $1/N_{\mathrm{p}}$ as suggested by Lorette et al. [43]. For the SOM, all implementation issues were set as by Pandya and Macy [48] using a Kohonen network of 5×4 nodes.

Table 1 Samples of the synthetic images used along with the corresponding histograms

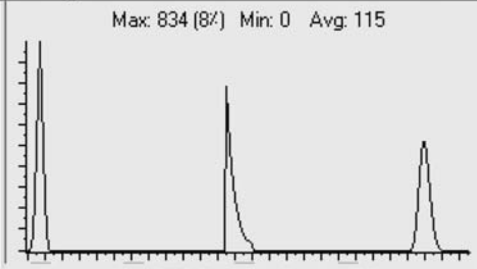
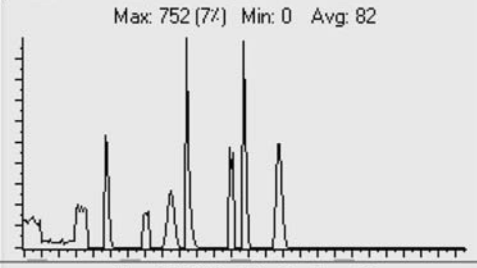
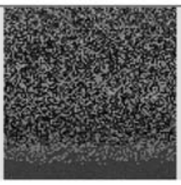
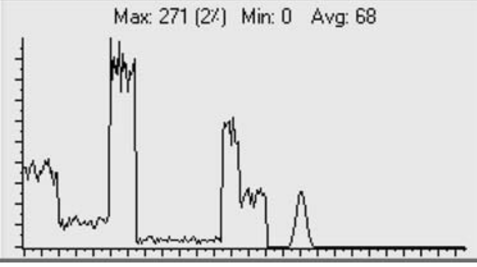
Synthetic image no.	Image	Histogram
2		
6		
9		

Table 2 MRI and Lake Tahoe images

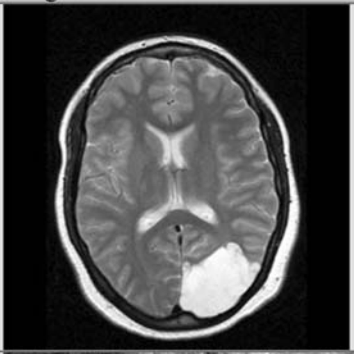
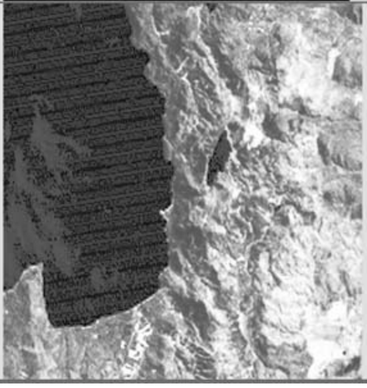
Image name	Image
MRI	
Tahoe	

Table 3 Mean and standard deviation (± SD) of the "optimal" number of clusters generated by the examined approached when applied to the synthetic images

Image	Actual number of clusters	DCPSO using D	DCPSO using V	DCPSO using S_Dbw	SOM	UFA
1	2	2 ± 0	2 ± 0	5.55 ± 5.22	2	20
2	3	3 ± 0	3 ± 0	3 ± 0	3	20
3	3	2 ± 0	2 ± 0	4.4 ± 4.852	6	20
4	3	2.7 ± 1.345	5.15 ± 0.357	10.9 ± 5.458	10	20
5	4	10.85 ± 1.878	5 ± 0	15.5 ± 1.323	7	20
6	10	9.55 ± 2.246	7.2 ± 0.872	9.3 ± 0.458	9	20
7	6	3.35 ± 1.526	7.9 ± 0.995	10.8 ± 2.925	9	20
8	4	8.8 ± 2.379075	5 ± 0	4 ± 0	4	20
9	7	4.25 ± 0.433	5 ± 0	14.1 ± 3.52	13	20
10	4	7.9 ± 1.729	7 ± 0	13.95 ± 1.77	9	20
11	10	10.0 ± 0.950	10 ± 0	10 ± 0	10	20
12	5	9.0 ± 2.168	7.2 ± 0.4	11.65 ± 1.06	6	20
13	5	12.1 ± 2.119	5 ± 0	9.6 ± 2.107	5	20
14	7	7.5 ± 1.204	5 ± 0	7.8 ± 2.088	7	20
15	5	5 ± 0	5 ± 0	5 ± 0	5	20

4.1 Synthetic images

Table 3 shows the results of DCPSO using the three validity indices described in Sect. 3, along with the UFA and SOM results. It appears that UFA tends to overfit the data since in all experiments it selects the maximum number of clusters as the correct one. The rationale behind this failure is the choice of ε which has a significant effect on the resulting number of clusters. DCPSO using S_Dbw also generally overfits the data. On the other hand, DCPSO using D, DCPSO using V and SOM have generally performed very well (especially DCPSO using V). Hence, it can be concluded that DCPSO using V is efficient with respect to the synthetic images.

4.2 Natural images

Table 4 shows the results of DCPSO using the three validity indices described in Sect. 3. These results are compared with the results of UFA and SOM. In addition, the results of snob for the images of Lenna, mandrill, jet, and peppers are copied from Turi [38]. The optimal range for the number of clusters for the images of Lenna, mandrill, jet, and peppers is also taken from Turi [38] which was based on a visual analysis survey conducted by a group of ten people. Similarly, the optimal range for the MRI and Lake Tahoe images was estimated by the authors using a group of three people. It appears from the table that results of DCPSO using S_Dbw, UFA, SOM, and snob were poor. DCPSO using V always found a solution within the optimal range. Therefore, the remaining experiments will use V as the validity index. These results clearly show the efficiency of DCPSO. Table 5 shows samples of the resultant segmented images generated by DCPSO using V.

4.3 Comparison with GA and RS

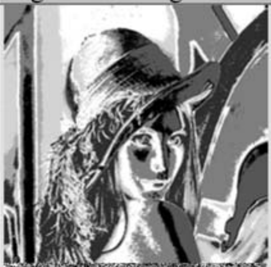
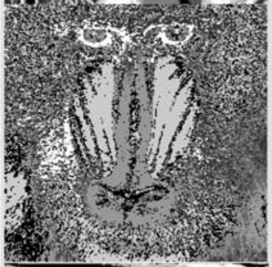
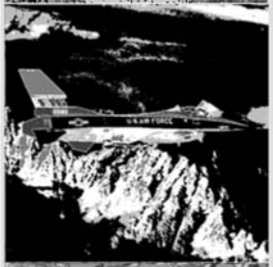
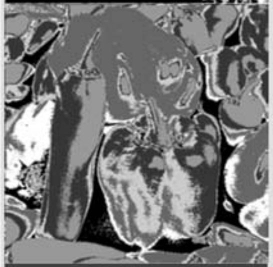
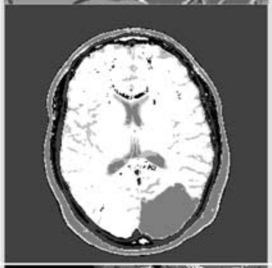
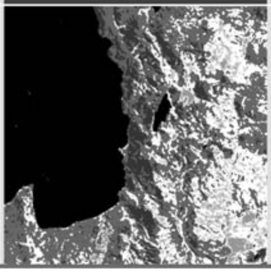
The previous experiments were conducted using a PSO version of the proposed approach (i.e., DCPSO). In this section, a GA and RS version of the proposed approach (called DCGA and DCRS, respectively) are examined and compared with DCPSO. Both DCGA and DCRS used 100 individuals. For DCGA, elitism was used, keeping the fittest chromosome for the next generation. In addition, random selection has been used along with uniform crossover. The crossover probability was set to 0.8 with a mixing ratio of 0.5; a mutation probability of $1/N_c$ was used. Table 6 shows the results of applying DCPSO, DCGA, and DCRS on the natural images. As expected, DCRS performed poorly due to its pure random search. DCGA performed comparably to DCPSO. As expected, using PSO and GA in the proposed approach is more efficient than using RS.

Reducing the swarm size (or population size in case of GA) from 100 to 20 particles (or GA chromosomes) did not generally affect the performance of both DCPSO

Table 4 Mean and standard deviation (± SD) of the "optimal" number of clusters generated by the examined approached when applied to the natural images

Image	Optimal range	DCPSO using D	DCPSO using V	DCPSO using S_Dbw	SOM	UFA	Snob
Lenna	5–10	10.35 ± 1.652	6.85 ± 0.477	19.3 ± 0.843	20	20	31
Mandrill	5–10	6.05 ± 1.658	6.25 ± 0.433	19.25 ± 0.766	20	20	42
Jet	5–7	3.35 ± 2.151	5.3 ± 0.459	18.05 ± 1.465	14	20	22
Peppers	6–10	10.55 ± 1.465	6 ± 0	18.8 ± 0.872	20	20	39
MRI	4–8	3 ± 0	5 ± 0	17.2 ± 1.4	19	20	–
Tahoe	3–7	3 ± 0	6.1 ± 0.539	14.3 ± 3.018	4	20	–

Table 5 Samples of segmented images resulting from DCPSO using V along with the corresponding number of clusters

Image	Segmented image	No. of clusters
Lenna		7
Mandrill		6
Jet		5
Peppers		6
MRI		5
Tahoe		6

and DCGA as shown in Table 7. These results verify the observation of Shi and Eberhart [56] stating that the performance of the PSO is not sensitive to the swarm size. Hence, the computational requirements of DCPSO and DCGA can be reduced without affecting the performance of DCPSO and DCGA.

Table 6 Comparison of PSO-, GA-, and RS-versions of the proposed approach when applied to the natural images

Image	Optimal range	DCPSO using V	DCGA using V	DCRS using V
Lenna	5–10	6.85 ± 0.477	6.45 ± 0.74	9.8 ± 1.661
Mandrill	5–10	6.25 ± 0.433	6.05 ± 0.589	8.75 ± 2.095
Jet	5–7	5.3 ± 0.459	5.3 ± 0.557	11.05 ± 1.627
Peppers	6–10	6 ± 0	6.05 ± 0.218	10.55 ± 1.532
MRI	4–8	5 ± 0	5.5 ± 0.742	8.1 ± 1.179
Tahoe	3–7	6.1 ± 0.539	6.1 ± 0.831	9.25 ± 1.479

Table 7 Comparison of PSO- and GA-versions of the proposed approach using a swarm size $s=20$ when applied to the natural images

Image	Optimal range	DCPSO using V	DCGA using V
Lenna	5–10	6.5 ± 0.806	6.4 ± 0.8
Mandrill	5–10	6.15 ± 0.357	5.85 ± 0.476
Jet	5–7	5.3 ± 0.458	5.35 ± 0.477
Peppers	6–10	6.05 ± 0.218	6 ± 0
MRI	4–8	5.2 ± 0.4	5.15 ± 0.357
Tahoe	3–7	6.05 ± 0.384	6.2 ± 0.4

4.4 Comparison of *lbest*-, *gbest*-, and *lbest*-*to*-*gbest*-PSO

In this section, the effect of different models of PSO is investigated. A comparison is made between *gbest*-, *lbest*-, and *lbest*-*to*-*gbest*-PSO (which has been used in the above experiments). For *lbest*-PSO a neighborhood size of $l=2$ was used. Table 8 shows the result of the comparison. Studying the results, it appears that *gbest*-PSO prematurely converges in two cases (i.e., MRI and Tahoe). The rationale for the premature convergence is the fast flow of information in *gbest*-PSO. On the other hand, *lbest*-PSO performed comparably to *lbest*-*to*-*gbest*-PSO.

5 Conclusions

This paper presented DCPSO, a new dynamic clustering algorithm based on PSO with application to image segmentation. DCPSO clusters a data set without requiring the user to specify the number of clusters in advance. This is an important feature since knowing the number of clusters in advance is often not easy. DCPSO uses a validity index to measure the quality of the resultant clustering. One of the advantages of this approach is that DCPSO can work with any validity index. In addition, the proposed approach can be used with GA and/or RS. DCPSO has been applied on synthetic (where the number of clusters was known in advance) as well as natural images (including MRI and satellite images), and has been compared with other unsupervised clustering techniques. From these experiments it can be concluded that DCPSO using the validity index proposed by Turi [38] has outperformed other approaches. In general, DCPSO successfully found the optimum number of clusters on the tested images. DCPSO was then compared to both DCGA and DCRS, with DCPSO and DCGA outperforming DCRS. The use of different PSO models (namely, *lbest*, *gbest*, and *lbest*-*to*-*gbest*) was investigated.

For future research, the application of the DCPSO algorithm to general data needs to be investigated. Furthermore, the effect of high dimensionality on the performance of the DCPSO should be investigated. The DCPSO uses the K-means clustering algorithm to refine the cluster centroids. Future research can investigate the use of other more efficient clustering algorithms such as FCM and KHM. In addition, incorporating spatial information into the DCPSO algorithm needs to be investigated.

Table 8 Comparison of *gbest*-, *lbest*-, and *lbest*-*to*-*gbest*- PSO versions of DCPSO using V ($s=20$) when applied to the natural images

Image	Optimal range	*gbest*- PSO	*lbest* PSO ($l=2$)	*lbest*-*to*-*gbest*- PSO
Lenna	5–10	6.6 ± 0.735	6.55 ± 0.669	6.5 ± 0.806
Mandrill	5–10	6.1 ± 0.3	6.1 ± 0.539	6.15 ± 0.357
Jet	5–7	5.5 ± 0.5	5.25 ± 0.433	5.3 ± 0.458
Peppers	6–10	6.15 ± 0.726	6.15 ± 0.654	6.05 ± 0.218
MRI	4–8	2.3 ± 2.551	5.3 ± 0.458	5.2 ± 0.4
Tahoe	3–7	3.0 ± 3.0	6.1 ± 0.3	6.05 ± 0.384

6 About the authors

Mahamed G. H. Omran received his B.Sc. and M.Sc. degrees with Honors in Computer Engineering from Kuwait University, State of Kuwait, in 1998 and 2000, respectively. He completed his Ph.D. in the Department of Computer Science at University of Pretoria, South Africa, in April 2005.

His current research interest is in the area of computational intelligence with special emphasis on Particle Swarm Optimization (PSO) and Differential Evolution (DE). In addition, Dr. Omran is interested in the area of data clustering, unsupervised image classification and color image quantization.

Ayed Salman received his MS and PhD degrees from Syracuse University, NY, USA, in 1996 and 1999, respectively. He is an Assistant Professor at the Department of Computer Engineering, Kuwait University, Kuwait.

His research interests include Application of evolutionary computations in data mining, web mining and multi-objective optimization; E-Commerce. Neural Network, Machine Learning.

Andries Engelbrecht received the M.Sc and PhD degrees from the University of Stellenbosch, Stellenbosch, South Africa, in 1994 and 1999, respectively. He is a Full Professor at the Department of Computer Science, University of Pretoria, Pretoria, South Africa, leading a research group of three staff members and 45 Masters and PhD students.

His research interests include aspects of swarm intelligence, evolutionary computation, artificial immune systems, data mining, swarm robotics, and neural networks, with several publications in those fields.

References

1. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
2. Jain AK, Duin R, Mao J (2000) Statistical pattern recognition: a review. IEEE Trans Pattern Anal Mach Intell 22(1):4–37
3. Judd D, Mckinley P, Jain AK (1998) Large-scale parallel data clustering. IEEE Trans Pattern Anal Mach Intell 20(8):871–876
4. Abbas HM, Fahmy MM (1994) Neural networks for maximum likelihood clustering. Signal Process 36(1):111–126
5. Coleman GB, Andrews HC (1979) Image segmentation by clustering. Proc IEEE 67:773–785
6. Jain AK, Dubes RC (1988) Algorithms for clustering data. Prentice Hall, New Jersey
7. Ray S, Turi RH (1999) Determination of number of clusters in K-means clustering and application in colour image segmentation. In: Proceedings of the 4th international conference on advances in pattern recognition and digital techniques (ICAPRDT'99), Calcutta, India, pp 137–143
8. Carpineto C, Romano G (1996) A lattice conceptual clustering system and its application to browsing retrieval. Mach Learn 24(2):95–122
9. Lee CY, Antonsson EK (2000) Dynamic partitional clustering using evolution strategies. In: The third Asia-Pacific conference on simulated evolution and learning
10. Hamerly G, Elkan C (2003) Learning the K in K-means. In: 7th annual conference on neural information processing systems
11. Frigui H, Krishnapuram R (1999) A robust competitive clustering algorithm with applications in computer vision. IEEE Trans Pattern Anal Mach Intell 21(5):450–465
12. Leung Y, Zhang J, Xu Z (2000) Clustering by space-space filtering. IEEE Trans Pattern Anal Mach Intell 22(12):1396–1410
13. Halkidi M, Batistakis Y, Vazirgiannis M (2001) On clustering validation techniques. Intell Inform Syst J 17(2–3):107–145
14. Theodoridis S, Koutroubas K (1999) Pattern recognition. Academic, New York
15. Rosenberger C, Chehdi K (2000) Unsupervised clustering method with optimal estimation of the number of clusters: application to image segmentation. In: International conference on pattern recognition (ICPR'00) 1:1656–1659
16. Kuncheva L, Bezdek J (1998) Nearest prototype classification: clustering, genetic algorithms, or random search? IEEE Trans Syst Man Cybernet C: Appl Rev 28(1):160–164
17. Forgy E (1965) Cluster analysis of multivariate data: efficiency versus interpretability of classification. Biometrics 21:768–769
18. Davies E (1997) Machine vision: theory, algorithms, practicalities, 2nd edn. Academic, New York
19. Bezdek J (1980) A convergence theorem for the fuzzy ISODATA clustering algorithms. IEEE Trans Pattern Anal Mach Intell 2:1–8
20. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Plenum, New York
21. Bishop C (1995) Neural networks for pattern recognition. Clarendon, Oxford
22. McLachlan G, Krishnan T (1997) The EM algorithm and extensions. Wiley, New York
23. Rendner R, Walker H (1984) Mixture densities, maximum likelihood and the EM algorithm. SIAM Rev 26(2)
24. Hamerly G (2003) Learning structure and concepts in data using data clustering, PhD thesis, University of California, San Diego
25. Alldrin N, Smith A, Turnbull D (2003) Clustering with EM and K-means (November 15 2003), Unpublished Manuscript; http://louis.ucsd.edu/~nalldrin/research/cse253_wi03.pdf.
26. Zhang Y, Brady M, Smith S (2001) Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. IEEE Trans Med Imag 20(1):45–57
27. Zhang B, Hsu M, Dayal U (1999) K-harmonic means–a data clustering algorithm. Technical report HPL-1999–124), Hewlett-Packard Labs
28. Zhang B (2000) Generalized K-harmonic means–boosting in unsupervised learning. Technical report HPL-2000–137), Hewlett-Packard Labs
29. Omran M, Engelbrecht A, Salman A (2005) Particle swarm optimization method for image clustering. Int J Pattern Recogn Artif Intell 19(3):297–322
30. Frigui H, Krishnapuram R (1999) A robust competitive clustering algorithm with applications in computer vision. IEEE Trans Pattern Anal Mach Intell 21(5):450–465
31. Ball G, Hall D (1967) A clustering technique for summarizing multivariate data. Behav Sci 12:153–155
32. Huang K (2002) A synergistic automatic clustering technique (Syneract) for multispectral image analysis. Photogrammetric Eng Remote Sens 1(1):33–40

33. Pelleg D, Moore A (2000) X-means: extending K-means with efficient estimation of the number of clusters. In: Proceedings of the 17th international conference on machine learning, Morgan Kaufmann, San Francisco, CA, pp 727–734
34. Kass R, Wasserman L (1995) A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion. J Am Stat Assoc 90(431):928–934
35. Hamerly G (2003) Learning structure and concepts in data using data clustering. PhD thesis, University of California, San Diego
36. Wallace CS, Dowe DL (1994) Intrinsic classification by MML—the snob program. In: Proceedings 7th Australian joint conference on artificial intelligence, UNE, Armidale, NSW, Australia, pp 37–44
37. Wallace CS (1984) An improved program for classification. Technical report No. 47, Department of Computer Science, Monash University, Australia
38. Turi RH (2001) Clustering-based colour image segmentation. PhD Thesis, Monash University, Australia
39. Wallace CS, Boulton DM (1968) An information measure for classification. Comput J 11:185–194
40. Oliver JJ, Hand D (1994) Introduction to minimum encoding inference. Technical report No. 94/205, Department of Computer Science, Monash University, Australia
41. Bischof H, Leonardis A, Selb A (1999) MDL principle for robust vector quantization. Pattern Anal Appl 2:59–72
42. Gath I, Geva A (1989) Unsupervised optimal fuzzy clustering. IEEE Trans Pattern Anal Mach Intell 11(7):773–781
43. Lorette A, Descombes X, Zerubia J (2000) Fully unsupervised fuzzy clustering with entropy criterion. In: International conference on pattern recognition (ICPR'00) 3:3998-4001
44. Boujemaa N (2000) On competitive unsupervised clustering. In: International conference on pattern recognition (ICPR'00) 1:1631–1634
45. Frigui H, Krishnapuram R (1997) Clustering by competitive agglomeration. Pattern Recogn Lett 30(7):1109–1119
46. Kohonen T (1995) Self-organizing maps. Springer, Berlin Heidelberg New York
47. Mehrotra K, Mohan C, Rakka (1997) Elements of artificial neural networks. MIT, Cambridge
48. Pandya A, Macy R (1996) Pattern recognition with neural networks in C++. CRC, Boca Raton
49. Halkidi M, Vazirgiannis M (2001) Clustering validity assessment: finding the optimal partitioning of a data set. In: Proceedings of ICDM conference, CA, USA
50. Dunn JC (1974) Well separated clusters and optimal fuzzy partitions. J Cybern 4:95–104
51. Davies, Bouldin (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 1(2)
52. Halkidi M, Vazirgiannis M (2002) Clustering validity assessment using multi representative. In: Proceedings of SETN conference, Thessaloniki, Greece
53. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, Perth, Australia 4:1942–1948
54. Kennedy J, Eberhart R (2001) Swarm intelligence. Morgan Kaufmann, San Francisco
55. Engelbrecht A (2002) Computational intelligence: an introduction. Wiley, New York
56. Shi Y, Eberhart R (1998) Parameter selection in particle swarm optimization. Evolutionary Programming VII: Proceedings of EP 98:591–600
57. Suganthan P (1999) Particle Swarm Optimizer with Neighborhood Optimizer. In: Proceedings of the congress on evolutionary computation, pp 1958–1962
58. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: Proceedings of the IEEE international conference on evolutionary computation, Piscataway, NJ, pp 69–73
59. Kennedy J, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: Proceedings of the congress on evolutionary computation, pp 1931–1938
60. Kennedy J, Mendes R (2002) Population structure and particle performance. In: Proceedings of the IEEE congress on evolutionary computation, Honolulu, Hawaii
61. Van den Bergh F (2002) An analysis of particle swarm optimizers. PhD thesis, Department of Computer Science, University of Pretoria
62. van den Bergh F, Engelbrecht AP (2002) A new locally convergent particle swarm optimizer. In: Proceedings of the IEEE conference on systems, man, and cybernetics, Hammamet, Tunisia
63. Kennedy J, Eberhart R (1997) A discrete binary version of the particle swarm algorithm. In: Proceedings of the conference on systems, man, and cybernetics, pp 4104–4109
64. Pal NR, Pal SK (1993) A review on image segmentation techniques. Pattern Recogn 26:1277–1294
65. Fu KS, Mui JK (1981) A survey on image segmentation. Pattern Recogn 13:3–16
66. Goldberg D (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading
67. Salman A, Omran M, Engelbrecht A (2005) SIGT: synthetic image generation tool for clustering algorithms. ICGST Int J Graph Vision Image Process (GVIP) 2:33–44