Data Swarm Clustering

Christian Veenhuis and Mario Köppen

Fraunhofer IPK
Department Pattern Recognition
Pascalstr. 8-9, 10587 Berlin, Germany
{christian.veenhuis, mario.koeppen}@ipk.fhg.de

Summary. Data clustering is concerned with the division of a set of objects into groups of similar objects. In social insects there are many examples of clustering processes. Brood sorting observed in ant colonies can be considered as clustering according to the developmental state of the larvae. Also nest cleaning by forming piles of corpse or items is another example. These observed sorting and cluster capabilities of ant colonies have already been the inspiration of an ant-based clustering algorithm.

Another kind of clustering mechanism can be observed in flocks of birds. In some rainforests mixed-species flocks of birds can be observed. From time to time different species of birds are merging to become a multi-species swarm. The separation of this multi-species swarm into its single species can be considered as a kind of species clustering.

This chapter introduces a data clustering algorithm based on species clustering. It combines methods of Particle Swarm Optimization and Flock Algorithms. A given set of data is interpreted as a multi-species swarm which wants to separate into single-species swarms, i.e., clusters. The data to be clustered are assigned to datoids which form a swarm on a two-dimensional plane. A datoid can be imagined as a bird carrying a piece of data on its back. While swarming, this swarm divides into sub swarms moving over the plane and consisting of datoids carrying similar data. After swarming, these sub swarms of datoids can be grouped together as clusters.

1 Introduction

In nature a swarm is an aggregation of animals as, e.g., flocks of birds, herds of land animals or schools of fishes. The formation of swarms seems to be advantageous to protect against predators and increase the efficiency of foraging. To maintain the structure of the swarm, each swarm-mate behaves according to some rules as, e.g., keep close to your neighbors or avoid collisions. Even mixed-species flocks (i.e., multi-species swarms) of birds can be observed in nature [10].

Data clustering is concerned with the division of data into groups (i.e., clusters) of similar data. If a single species is considered as a type of data, then a multi-species swarm can be considered as a set of mixed data. The rules of the presented Data Swarm Clustering (DSC) algorithm divide this multi-species swarm of data into

several single-species swarms consisting of similar data. This way a data clustering is performed. Each single-species swarm represents a cluster of similar data.

Also in social insects clustering processes can be observed [2]. For instance, brood sorting in ant colonies can be considered as clustering according to the developmental state of the larvae. Nest cleaning by forming piles of corpse or items is another example.

Deneubourg et al. introduced in [3] two models of larval sorting and corpse clustering. These models have been the inspiration of an ant-based clustering algorithm. Ant-based clustering [4] simulates the clustering and sorting behavior of ants. The data items to be clustered are assigned to items on a two-dimensional grid. These items on the grid are piled up by the simulated ants building this way the data clusters. The ants walk randomly over the grid and every time they find an isolated or wrong placed item they pick it up and put it down close to the first randomly found similar item somewhere else.

Monmarche et al. hybridized the Ant-based clustering approach with the *k*-means algorithm as introduced in [9]. They applied the Ant-based clustering algorithm for a fixed number of iterations to create an initial partitioning of the data. Afterwards, they refined this initial partition by using the *k*-means algorithm. In [6] a very similar approach to [9] is used. But instead of *k*-means the fuzzy *c*-means algorithms is used to refine the initial partition created by the ants.

Another type of clustering was introduced by Omran et al. [12]. They used Particle Swarm Optimization [7] to determine a given number of cluster centroids (i.e., center of clusters). For this, they used particles which contain all centroid vectors in a row. If $\mathbf{c}_{ij}$ denotes the j^{th} cluster centroid of particle x_i, then a particle encoding N clusters is defined as $x_i = (\mathbf{c}_{i1}, \cdots, \mathbf{c}_{ij}, \cdots, \mathbf{c}_{iN})$. This way a particle has the dimensionality of number of centroids N times dimension of centroids. The swarm consists of many possible centroid vector sets. In [15] van der Merwe and Engelbrecht hybridized this approach with the *k*-means algorithm. A single particle of the swarm is initialized with the result of the *k*-means algorithm. The rest of the swarm is initialized randomly.

In some rainforests mixed-species flocks of birds can be observed [10]. Different species of birds are merging to become a multi-species swarm. The separation of this multi-species swarm into its single species can be considered as a kind of species clustering.

The method presented in this chapter is based on a simulation of species clustering. The data items to be clustered are assigned to objects called datoids placed on a two-dimensional plane. Similar to ant-based clustering the data items aren't clustered in their attribute space, but in the space of the datoids. This is an advantage, because data items belonging to the same class don't need to be close in the attribute space. They get close in the space of the datoids, if they belong together. Instead of randomly moving, the datoids have an affinity to move to their similar neighbors. The similar neighbors are determined based on a similarity distance function. In contrast to ant-based clustering, the data items (datoids) move on the two-dimensional plane by themself and aren't moved by additional entities like, e.g., ants.

This chapter is organized as follows. Section 2 gives a brief overview of clustering. Flock Algorithms and Particle Swarm Optimization are described in sections 3 and 4. The Data Swarm Clustering algorithm is described in section 5. The used experiments and results are presented in sections 6 and 7. Finally, in section 8 some conclusions are drawn.

2 Data Clustering

The task of data clustering is to divide a set of data X into sub-sets $C_i \subseteq X$ containing similar data. In Figure 1 a set of data containing three different kinds of objects is presented.

Clustering produces a set $C = \{C_i\}$ of all sub-sets $C_i \subseteq X$ as shown in Figure 2. A sub-set C_i is called a cluster and all clusters together result in X (i.e., $X = \bigcup_i C_i$). That means that each element of X is assigned to a cluster. Usually clusters are pair-wise disjoint (i.e., $\bigcap_i C_i = \emptyset$). That means that each element of X is assigned to exactly one cluster. The data contained within one cluster C_i are similar in some way and dissimilar to the data contained in other clusters.

Often the data are points $d_i = (d_{i1}, \cdots, d_{in})^T \in A$ in an n-dimensional attribute space $A = A_1 \times \cdots \times A_n$. Each $d_{iv} \in A_v$ $(1 \leq v \leq n)$ represents a single variable or attribute.

For instance, if a set of points $\{(x_i, y_i)\}$ on a two-dimensional plane $\mathbb{R}^2$ shall be clustered, then $\mathbb{R}^2$ is the attribute space, the dimensions are the attributes and the coordinates $x_i, y_i \in \mathbb{R}$ are attribute values.

Often simply the distances between points in the attribute space are used as similarity measure to determine whether or not they are belonging to the same cluster (e.g., as in k-means).

Clustering is applied in a lot of fields as, e.g., character recognition, image segmentation, image processing, computer vision, data and web mining.

Clustering algorithms can be grouped among others into hierarchical clustering, partitioning relocation clustering, density-based partitioning and grid-based methods. Berkhin gives an extensive survey of clustering algorithms in [1].

3 Flock Algorithms

Flock Algorithm is a generic term for algorithms mimicking the natural aggregation and movement of bird flocks, herds of land animals or schools of fishes. Reynolds developed a distributed behavior model to compute the movement of flocks of birds within virtual environments [13] to get rid of modelling each single bird explicitly. A single entity of this model is called boid (bird-oid) and implements the following behavior:

- **Collision Avoidance**
  Each boid has to avoid collisions with nearby flock-mates.

$X =$

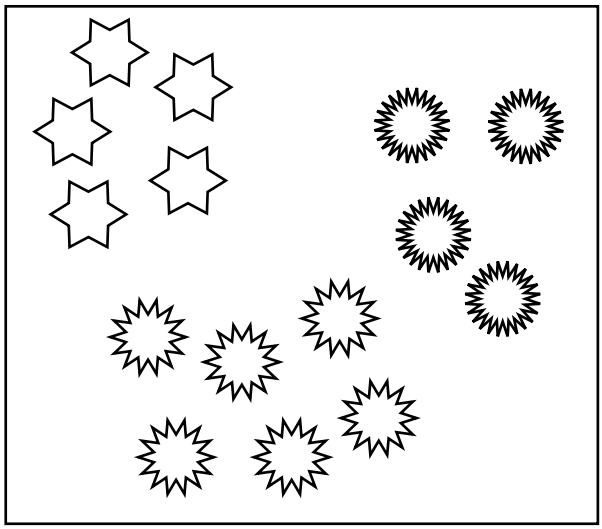

Fig. 1. A set of data containing three types of objects.

$C =$

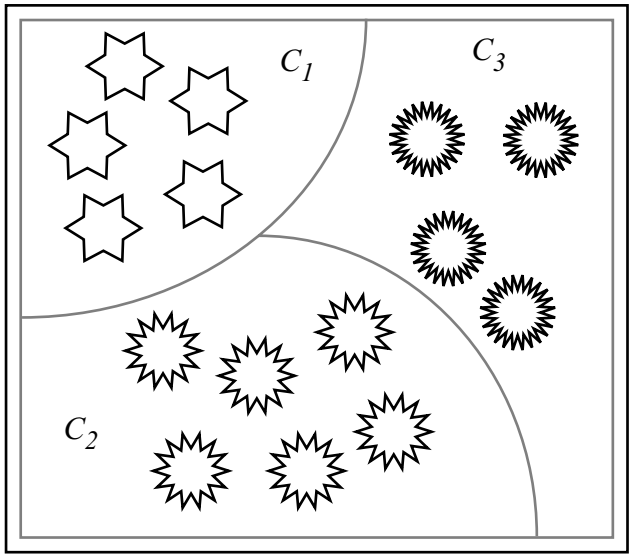

Fig. 2. A set of clusters $C = \{C_1, C_2, C_3\}$ containing three clusters ($|C| = 3$) of similar objects. Here, the similarity is based on the outer shape of the objects.

- **Velocity Matching**
  Each boid attempts to match the velocities of nearby flock-mates.
- **Flock Centering**
  Each boid tries to stay close to nearby flock-mates. For this, it steers to the center of the k nearest neighbors.
- **Obstacle Avoidance**
  Within virtual environments it is intended to avoid collision with obstacles (e.g., houses).
- **Following a Path**
  Usually the boids follow a specified path within the virtual environment.

Although this type of algorithm was thought for computer graphic purposes, it has been the inspiration for other algorithms as, e.g., the Particle Swarm Optimization.

4 Particle Swarm Optimization

Particle Swarm Optimization (PSO), as introduced by Kennedy and Eberhart [7] [8], is an optimization algorithm based on swarm theory. The main idea is to model the flocking of birds flying around a peak in a landscape.

In PSO the birds are substituted by particles and the peak in the landscape is the peak of a fitness function. The particles are flying through the search space forming flocks around peaks of fitness functions.

Let N_{dim} be the dimension of the problem (i.e., the dimension of the search space $\mathbb{R}^{N_{dim}}$), N_{part} the number of particles and P the set of particles $P = \{P_1, \cdots, P_{N_{part}}\}$. Each particle $P_i = (x_i, v_i, l_i)$ has a current position in the search space ($x_i \in \mathbb{R}^{N_{dim}}$), a velocity ($v_i \in \mathbb{R}^{N_{dim}}$) and the locally best found position in history, i.e., the own experience ($l_i \in \mathbb{R}^{N_{dim}}$) of this particle.

In PSO, the set of particles P is initialized at time step $t = 0$ with randomly created particles $P_i^{(0)}$. The initial l_i are set to the corresponding initial x_i. Then, for each time step t, the next position $x_i^{(t+1)}$ and velocity $v_i^{(t+1)}$ of each particle $P_i^{(t)}$ is computed as shown in Eqns. (1) and (2).

$$v_i^{(t+1)} = w_I^{(t)} v_i^{(t)} + w_L r_1 (l_i^{(t)} - x_i^{(t)}) + w_N r_2 (n_i^{(t)} - x_i^{(t)}) \tag{1}$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \tag{2}$$

Here, r_1 and r_2 are random numbers in $[0,1]$. $n_i^{(t)} \in \mathbb{R}^{N_{dim}}$ represents the best found local position of the best neighbor particle at time t. Because there are several possibilities to define the neighborhood of a particle [8], the best neighboring particle can be, e.g., the best particle of a pre-defined neighborhood; the best particle of the nearest neighbors according to the distance in search space; the globally best particle etc. The inertia weight $w_I^{(t)}$ determines the influence of the particle's own

velocity, i.e., it represents the confidence of the particle to its own position (typically $w_I \in [0.1, 1.0]$). To yield a better convergence, this weight is decreased over time [14] [8]. The typical way to decrease the inertia weight $w_I^{(t)}$ is to subtract a fixed amount (e.g., 0.001) each iteration. w_L is the influence of the locally best position found so far. The influence of the best particle of the neighborhood is denoted with w_N.

After computing, the next velocity $v_i^{(t+1)}$ is added to the position of the particle to get the new position. To avoid chaotic behavior, this new velocity $v_i^{(t+1)}$ is clamped to a pre-defined interval $[-V_{max}, +V_{max}]$.

The fitness of a particle is determined by a fitness function F which maps a position vector to a real number (Eq. (3)).

$$F : \mathbb{R}^{N_{dim}} \rightarrow \mathbb{R} \tag{3}$$

If the new position $x_i^{(t+1)}$ has a better fitness than the best solution found so far for particle P_i, it is stored in memory as shown in Eq. (4) (in case of minimization).

$$l_i^{(t+1)} = \begin{cases} x_i^{(t+1)} & , F(x_i^{(t+1)}) < F(l_i^{(t)}) \\ l_i^{(t)} & , otherwise \end{cases} \tag{4}$$

The best solution of the run is found at particle P_b with the best local solution l_b. Best solution l_b is always element of the set of all best local solutions $\{l_i\}, \forall i \in \{1, \cdots, N_{part}\}$. Equation (5) determines the best fitness value $F(l_b)$ simply as the minimal fitness value of all local solutions.

$$F(l_b) = \min_{\forall i \in \{1, \cdots, N_{part}\}} \{F(l_i)\} \tag{5}$$

5 Data Swarm Clustering

The Data Swarm Clustering (DSC) algorithm mimicks a sort of separation of different species forming one big multi-species swarm into sub-swarms consisting only of individuals of the same species (i.e., single-species swarms). The multi-species swarm can be considered as a set of mixed data X and a sub-swarm as cluster $C_i \subseteq X$.

To realize this, a PSO method with two-dimensional particles is used. The PSO particle is modified to hold one data object of X. Because a data object can be of every possible type or structure, an entity of DSC is called datoid (data-oid) instead of particle.

The separation (i.e., clustering) of a swarm of datoids can be described with three rules:

1. **Swarm Centering of Similar Datoids**
   Each datoid tries to stay close to similar nearby swarm-mates. For this, it steers to the center of the k nearest similar neighbors.
   $\Rightarrow$ This is the moving power to build the raw sub-swarms of similar datoids.

2. **Approach Avoidance to Dissimilar Datoids**
   Each datoid avoids to approach to the nearest dissimilar swarm-mate.
   $\Rightarrow$ This helps to prevent dissimilar sub-swarms to merge.
3. **Velocity Matching of Similar Datoids**
   Each datoid attempts to match the velocity of the nearest similar neighbor.
   $\Rightarrow$ This advantages the emergence of synchronized sub-swarms which build a better unity.

These rules are modified versions of the rules of Reynolds Flock Algorithm as described in section 3. The modifications include:

- The differentiation of similar and dissimilar neighbors.
- Collision avoidance is replaced by an approach avoidance to dissimilar datoids.
- Velocity is matched only to the nearest similar neighbor instead to all nearby flock-mates.

Data Swarm Clustering consists of three phases: (1) initialization, (2) multiple iterations and (3) retrieval of the formed clusters.

A swarm of datoids can be described quite similar as in PSO. Let N_{dat} be the number of datoids and D the set of datoids $D = \{D(1), \cdots, D(N_{dat})\}$. Each datoid $D(i) = (x_i, v_i, o_i)$ consists of a current position on a two-dimensional plane ($x_i \in \mathbb{R}^2$), a velocity ($v_i \in \mathbb{R}^2$) and the data object bound to this datoid ($o_i \in X$). Each datoid is placed on a quadratic two-dimensional plane with a side length of X_{max}. That is, X_{max} restricts the two-dimensional plane in both dimensions. The maximal possible distance d_{max} between two datoids is simply the diagonal of the two-dimensional plane as computed in Eq. (6).

$$d_{max} = \sqrt{X_{max}^2 + X_{max}^2} \tag{6}$$

If N_{iter} denotes the number of iterations, the main algorithm can be described as follows:

1. Initialize D with randomly created datoids (see section 5.1).
2. For n = 1 to N_{iter}: iterate swarm (see section 5.2).
3. Retrieve the clusters (see section 5.3).

5.1 Initialization

The initialization phase creates the set D with randomly created datoids. Because all data $d_i \in X$ have to be clustered, the number of datoids N_{dat} is set to the number of data in X (i.e., $N_{dat} = |X|$).

For each datoid $D(i) \in D, \forall i \in \{1, \cdots, N_{dat}\}$ the following initialization is performed. First, the data object o_i bound to the datoid $D(i)$ is set to the corresponding data item $d_i \in X$.

Afterwards, the position of the datoid on the two-dimensional plane is set randomly. For this, each element of the position vector x_i is set to a random number between 0 and X_{max}.

At last, each element of the velocity vector v_i of datoid $D(i)$ is set to a random number between $-V_{max}$ and $+V_{max}$. V_{max} restricts the velocity as in PSO (see section 4).

The whole algorithm is shown in the following ($R_{a,b}$ means a random-number between a and b):

$N_{dat} \leftarrow |X|$
for $i = 1$ **to** N_{dat}
  $o_i \leftarrow d_i \in X$
  $x_i \leftarrow (R_{0,X_{max}}, R_{0,X_{max}})^T$
  $v_i \leftarrow (R_{-V_{max},+V_{max}}, R_{-V_{max},+V_{max}})^T$
end for i

5.2 Iteration

While iterating, a similarity function is needed to provide a similarity measure between two datoids. This similarity function as shown in Eq. (7) gets the data objects of two datoids and returns a value in $[0,1]$, whereby 1 means that both datoids are equal and 0 that both are maximal dissimilar.

$$S : X \times X \rightarrow [0,1] \tag{7}$$

This similarity function is problem-specific. For the datasets used in the experiments later on, the Euclidian distance as defined in Eq. (25) (section 7) is used.

The distance between two datoids on the two-dimensional plane is also determined by Euclidian distance as shown in Eq. (8).

$$d : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R} \ , \ d(a,b) = \sqrt{\sum_{c=1}^{2} (a_c - b_c)^2} \tag{8}$$

An iteration step performs the following computation to each datoid $D(i) \in D, \forall i \in \{1, \cdots, N_{dat}\}$.

First, the nearest similar neighbor $D(n_{i,similar})$ of $D(i)$ as depicted in Figure 3 is determined. For this, the similarity distance function $SD(i,j)$ as defined in Eq. (9) is used to compute the distance between two datoids.

$$SD(i,j) = S(o_i,o_j)d(x_i,x_j) + (1 - S(o_i,o_j))d_{max} \tag{9}$$

The more similar two datoids are, the more the real distance is used as similarity distance. The more dissimilar they are, the more the maximal possible distance on the plane is used. This punish dissimilar neighbors by increasing their similarity distance. The index number of the nearest similar neighbor can be computed as in Eq. (10).

$$n_{i,similar} = n \quad \text{s.t.} \quad SD(i,n) = \min_{\substack{\forall j \in \{1,\cdots,N_{dat}\} \\ i \neq j}} \{SD(i,j)\} \tag{10}$$

The nearest similar neighbor is needed for rule 3: *Velocity Matching of Similar Datoids*.

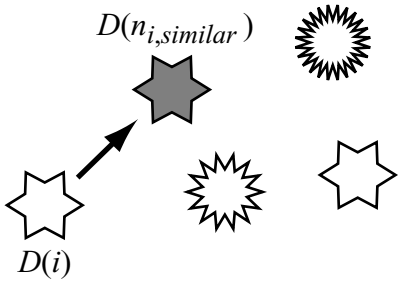

Fig. 3. The nearest similar neighbor $D(n_{i,similar})$. The gray-colored object is the nearest similar neighbor to $D(i)$. It is the one with the best similarity of the nearest neighbors.

Then, the nearest dissimilar neighbor $D(n_{i,dissimilar})$ of $D(i)$ as illustrated in Figure 4 is determined. For this, the dissimilarity distance function $DD(i,j)$ as defined in Eq. (11) is used to compute the distance between two datoids.

$$DD(i,j) = (1 - S(o_i,o_j))d(x_i,x_j) + S(o_i,o_j)d_{max} \tag{11}$$

The more dissimilar two datoids are, the more the real distance is used as dissimilarity distance. The more similar they are, the more the maximal possible distance on the plane is used. This punish similar neighbors by increasing their dissimilarity distance. The index number of the nearest dissimilar neighbor can be computed as in Eq. (12).

$$n_{i,dissimilar} = n \quad \text{s.t.} \quad DD(i,n) = \min_{\substack{\forall j \in \{1,\cdots,N_{dat}\} \\ i \neq j}} \{DD(i,j)\} \tag{12}$$

The nearest dissimilar neighbor is needed for rule 2: *Approach Avoidance to Dissimilar Datoids*.

For rule 1: *Swarm Centering of Similar Datoids* the center position of the k nearest similar neighbors $c_{i,similar}$ of $D(i)$ on the two-dimensional plane is needed. This is illustrated in Figure 5.

If $n = (n_1,\cdots,n_k)$ represents the sequence of indices of the k nearest similar neighbors (according to the similarity distance function SD), then Eq. (13) computes the needed center of those neighbors.

$$c_{i,similar} = \begin{pmatrix} \frac{1}{k}\sum_{j=1}^{k} x_{n_j 1} \\ \\ \frac{1}{k}\sum_{j=1}^{k} x_{n_j 2} \end{pmatrix} \in \mathbb{R}^2 \tag{13}$$

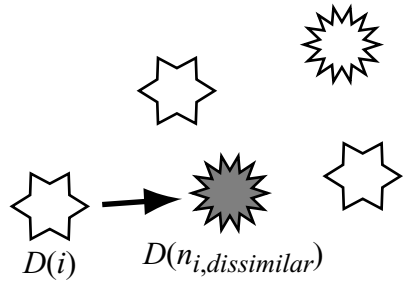

Fig. 4. The nearest dissimilar neighbor $D(n_{i,dissimilar})$. The gray-colored object is the nearest dissimilar neighbor to $D(i)$. It is the one with the best dissimilarity of the nearest neighbors.

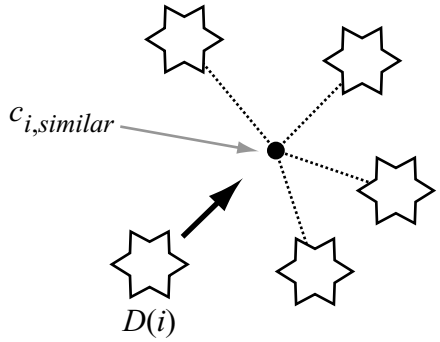

Fig. 5. The center position $c_{i,similar}$ of the k nearest similar neighbors to $D(i)$.

In Flock Algorithms a boid tries to match its velocity to the velocity of its neighbors. This mechanism is used in DSC to synchronize the sub-swarms to build a better unity. Instead of several neighbors, in DSC the velocity is matched only to the nearest similar neighbor as in Eq. (14). In Figure 6 this is presented by a line between the gray-colored datoid and its nearest similar neighbor.

The gray-colored datoid matches its velocity to its neighbor as defined in Eq. (14). The degree of matching is weighted with w_V.

$$\Delta_{velo} = w_V \, (v^{(t)}_{n_{i,similar}} - v^{(t)}_i) \tag{14}$$

The swarm-mates in PSO and Flock Algorithms try to stay close to nearby swarm-mates. This behavior produces sub-swarms or one big compact swarm, depending on the size of neighborhood. In PSO there is often a pre-defined and pre-assigned neighborhood to which a particle tries to stay close. This reduces the computation time. But as shown in Figure 7 in DSC a datoid steers to the center of the k nearest similar neighbors as in Flock Algorithms.

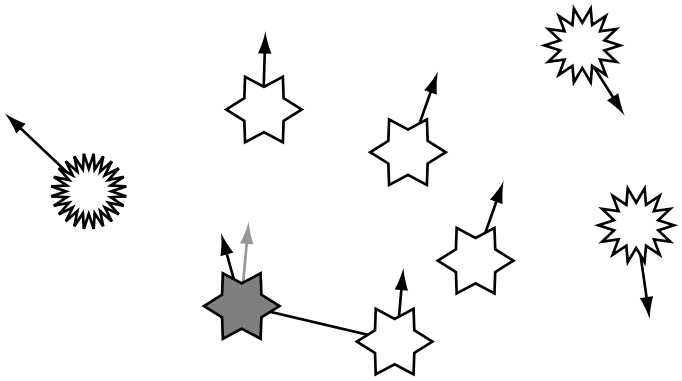

Fig. 6. Datoids match their velocity to their nearest similar neighbor. The new velocity is drawn as gray-colored arrow.

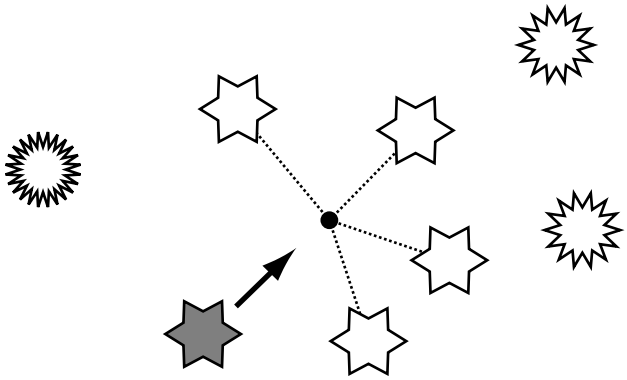

Fig. 7. Datoids steer to the center of the nearest similar neighbors.

This way the neighborhood is variable what is neccessary, because a pre-defined neighborhood would mean a pre-defined sub-swarm (i.e., cluster) of arbitrary datoids. This would work contrary to the clustering process.

$$\Delta_{neighbors} = S(o_i, o_{n_{i,similar}})\, w_N\, R_{0,1}\, (c^{(t)}_{i,similar} - x^{(t)}_i) \tag{15}$$

The influence of the neighbors $\Delta_{neighbors}$ as shown in Eq. (15) is computed quite similar to PSO. The difference of the current position $x^{(t)}_i$ and the center of the similar neighbors $c^{(t)}_{i,similar}$ is weighted by w_N. Here, $R_{0,1}$ is a random number in $[0,1]$. Additionally, the influence of the neighbors is weighted by the degree of similarity between them and the current datoid $D(i)$. As representant of the k nearest similar neighbors, the nearest similar neighbor $D(n_{i,similar})$ is used. The similarity between the data objects bound to the current datoid and the nearest similar neighbor $S(o_i, o_{n_{i,similar}})$ weights the influence of the neighbors.

To provide the separation of dissimilar datoids and to avoid the merging of dissimilar sub-swarms, the current datoid tries to move away from the nearest dissimilar neighbor, if the distance between them becomes too close (i.e., $d(x_i, x_{n_{i,dissimilar}}) < \tau_d$). This is illustrated in Figure 8.

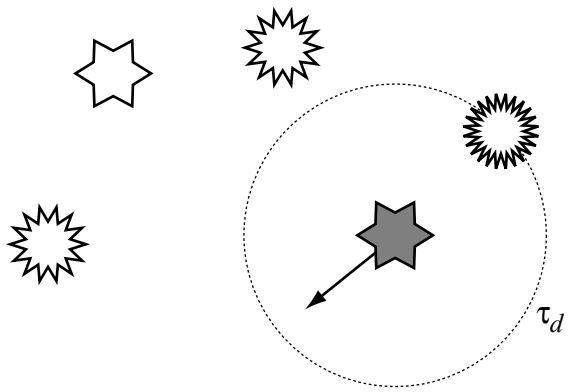

Fig. 8. Datoids try to move away from their nearest dissimilar neighbor.

For this, the difference between the current position $x_i^{(t)}$ and the nearest dissimilar neighbor $x_{n_{i,dissimilar}}^{(t)}$ is computed in a way that it points away from the nearest dissimilar neighbor (see Eq. (16)).

$$a = \begin{cases} x_i^{(t)} - x_{n_{i,dissimilar}}^{(t)} & , d(x_i, x_{n_{i,dissimilar}}) < \tau_d \\ 0 & , otherwise \end{cases}$$
$$\Delta_{avoidance} = (1 - S(o_i, o_{n_{i,similar}}))\, w_A\, a \tag{16}$$

The avoidance vector a is weighted with w_A and set to a dependency to the influence of similar neighbors. That is, if a datoid has very similar neighbors, the avoidance to the dissimilar neighbor is reduced by $(1 - S(o_i, o_{n_{i,similar}}))$. On the other hand, if a datoid has few similar neighbors, the effect of avoidance is stronger.

After determining the velocity matching Δ_{velo}, the influence of the k nearest similar neigbors $\Delta_{neighbors}$ and the avoidance $\Delta_{avoidance}$ to dissimilar neighbors, the new velocity $v_i^{(t+1)}$ as in Eq. (17) can be computed.

$$v_i^{(t+1)} = w_I^{(t)}\, v_i^{(t)} + \Delta_{velo} + \Delta_{neighbors} + \Delta_{avoidance} \tag{17}$$

The inertia weight $w_I^{(t)}$ determines the influence of the previous velocity. As in PSO this weight is decreased over time (see section 4). This decreasing procedure starts at iteration number s_I ($1 \leq s_I \leq N_{iter}$). Every iteration the inertia weight is decreased by an amount of a_I. Decreasing w_I results in more compact and dense sub-swarms which is an advantage in the latter cluster retrieval.

After computing the new velocity, the new position $x_i^{(t+1)}$ is computed as in Eq. (2).

Afterwards, the new velocity $v_i^{(t+1)}$, is clamped to $[-V_{max}, +V_{max}]$ by Eq. (18).

$$\forall c \in \{1,2\} : {v_{ic}}^{(t+1)} = \max[-V_{max}, \min(+V_{max}, {v_{ic}}^{(t+1)})] \tag{18}$$

If the new position $x_i^{(t+1)}$ of the datoid lies outside the restricted plane, the datoid bounces against the border.

In the following the pseudo-code of the DSC iteration is shown:

---

for i = 1 **to** N_{dat}

// Move datoid $D(i)$
$n_{i,similar} \leftarrow$ index of nearest similar neighbor by Eq. (10)
$n_{i,dissimilar} \leftarrow$ index of nearest dissimilar neighbor by Eq. (12)
$c_{i,similar} \leftarrow$ center of k nearest similar neighbors by Eq. (13)

Compute new velocity $v_i^{(t+1)}$ by Eq. (17)
Compute new position $x_i^{(t+1)}$ by Eq. (2)

// Ensure validity of datoid
for c = 1 **to** 2
// Clamp velocity
${v_{ic}}^{(t+1)} \leftarrow \max[-V_{max}, \min(+V_{max}, {v_{ic}}^{(t+1)})]$

// Bounce against the borders
if ${x_{ic}}^{(t+1)} < 0$
${x_{ic}}^{(t+1)} \leftarrow {x_{ic}}^{(t+1)} + 2(-{x_{ic}}^{(t+1)})$
${v_{ic}}^{(t+1)} \leftarrow -{v_{ic}}^{(t+1)}$
else if ${x_{ic}}^{(t+1)} > X_{max}$
${x_{ic}}^{(t+1)} \leftarrow {x_{ic}}^{(t+1)} - 2({x_{ic}}^{(t+1)} - X_{max})$
${v_{ic}}^{(t+1)} \leftarrow -{v_{ic}}^{(t+1)}$
end if
end for c
end for i

// Update parameters
if numIterations $\geq s_I \wedge w_I > 0.1$
$w_I \leftarrow w_I - a_I$
end if
numIterations $\leftarrow$ numIterations + 1

The variable `numIterations` represents the number of iterations performed so far and is initialized to 0 before calling the first iteration.

5.3 Cluster Retrieval

After a certain number of iterations, sub-swarms of similar datoids have formed. These sub-swarms represent the clusters. Therefore, the datoids of a given sub-swarm need to be grouped together as a cluster.

To realize this, a sort of an agglomerative clustering algorithm applied to the positions of the datoids on the two-dimensional plane is used.

All datoids $D(i), D(j) \in D$ ($\forall i, j \in \{1, \cdots, N_{dat}\}, i \neq j$) whose Euclidean distance $d(x_i, x_j) \leq \tau_c$ is lower than a given threshold τ_c belong to the same cluster.

6 Experimental Setup

To evaluate the cluster capabilities of DSC it was tested on four datasets: two synthetical and two real life datasets. These datasets as well as the used parameterization of DSC are described in the following sections.

6.1 Synthetical Datasets

The synthetically generated datasets used are:

Corners
The dataset *Corners* contains 4 randomly created clusters in 200 records located at the 4 corners of a quadratic grid as presented in Figure 9. All clusters are separable by lines on the grid, i.e., in the attribute space.

The $N_{dat} = 200$ records are divided by four to create four corners of similar size. If $Int(x)$ denotes the integer part of x then the number n of records per class is computed as $n = Int(0.25 \cdot N_{dat})$.

Let X_{max} be the length of a side of the quadratic grid. Then, the side length of a single quadratic corner is computed as $s_{corner} = 0.4 \cdot X_{max}$.

The four corners can now be defined as relations:

Top Left is $TL = \{0, \cdots, s_{corner}\} \times \{0, \cdots, s_{corner}\}$
Top Right is $TR = \{X_{max} - s_{corner}, \cdots, X_{max} - 1\} \times \{0, \cdots, s_{corner}\}$
Bottom Left is $BL = \{0, \cdots, s_{corner}\} \times \{X_{max} - s_{corner}, \cdots, X_{max} - 1\}$
Bottom Right is $BR = \{X_{max} - s_{corner}, \cdots, X_{max} - 1\} \times \{X_{max} - s_{corner}, \cdots, X_{max} - 1\}$

The four clusters are created as follows:

Cluster 0 (top left): Randomly create n points $(x(i), y(i)) \in TL$.
Cluster 1 (top right): Randomly create n points $(x(i), y(i)) \in TR$.
Cluster 2 (bottom left): Randomly create n points $(x(i), y(i)) \in BL$.
Cluster 3 (bottom right): Randomly create $N_{dat} - 3n$ points $(x(i), y(i)) \in BR$.

Nested
As shown in Figure 10 the dataset *Nested* contains 2 randomly created clusters in 200 records, whereby one cluster is located at the center area of a quadratic grid and the other surrounds it. The clusters are not separable by lines on the grid, i.e., in the attribute space.

The $N_{dat} = 200$ records are divided into five sets for the four border areas as well as the center area. The number n_{border} of records per border area is computed as $n_{border} = Int(0.2 \cdot N_{dat})$. The number n_{center} of records for the center area is computed as $n_{center} = N_{dat} - 4 \cdot n_{border}$.

Again, let X_{max} be the side length of the quadratic grid. Then, the margin m of the center area is computed as $m = 0.4 \cdot X_{max}$.

The five sets for the four border areas and the center area can be defined as relations:

Border Top is $BT = \{0,\cdots,X_{max}-1\} \times \{0,\cdots,0.5 \cdot m\}$
Border Bottom is $BB = \{0,\cdots,X_{max}-1\} \times \{X_{max}-0.5 \cdot m,\cdots,X_{max}-1\}$
Border Left is $BL = \{0,\cdots,0.5 \cdot m\} \times \{0,\cdots,X_{max}-1\}$
Border Right is $BR = \{X_{max}-0.5 \cdot m,\cdots,X_{max}-1\} \times \{0,\cdots,X_{max}-1\}$
Center Area is $CA = \{m,\cdots,X_{max}-m\} \times \{m,\cdots,X_{max}-m\}$

The two clusters are created as follows:

Cluster 0 (borders): Randomly create n_{border} points $(x(i),y(i)) \in BT$, n_{border} points $(x(i),y(i)) \in BB$, n_{border} points $(x(i),y(i)) \in BL$ and n_{border} points $(x(i),y(i)) \in BR$.
Cluster 1 (center): Randomly create n_{center} points $(x(i),y(i)) \in CA$.

6.2 Real Life Datasets

The following real life datasets are used:

Iris: The dataset *Iris* contains 3 clusters in 150 records with 4 numerical attributes (sepal length, sepal width, petal length, petal width). Each of the 3 classes (Setosa, Versicolour, Virginica) contains 50 records.

WBC: The dataset Wisconsin Breast Cancer (*WBC*) contains 2 clusters of 2 classes (Benign, Malignant) in 683 records with 10 numerical attributes (Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, Mitoses).

Both datasets are taken from the UCI Repository Of Machine Learning Databases [11] and the attribute values were normalized.

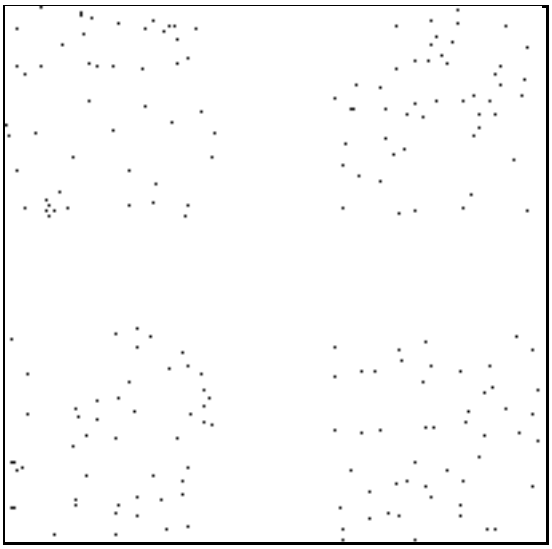

Fig. 9. Synthetical dataset *Corners*.

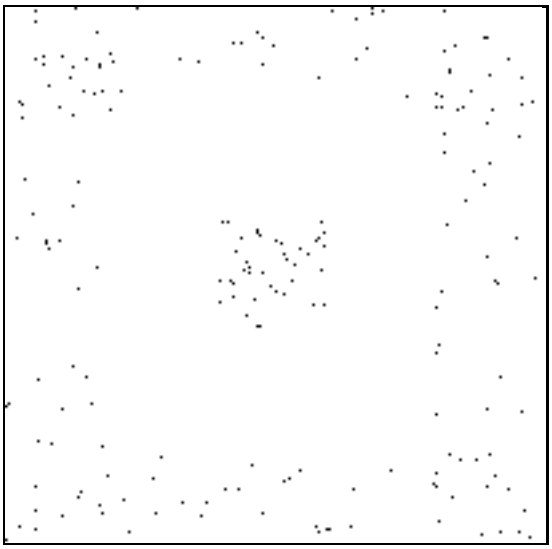

Fig. 10. Synthetical dataset *Nested*.

6.3 Parameters

In Table 1 the used parameter settings for DSC are shown. These parameters are determined by experimentation and have the following meaning:

N_{dat}	:= Number of datoids.
N_{iter}	:= Number of iterations.
X_{max}	:= Size of 2D plane.
V_{max}	:= Range of velocity.
k	:= Number of considered neighbors, i.e., size of neighborhood.
w_I	:= Start value of inertia weight.
s_I	:= Iteration number to start decreasing of inertia weight.
a_I	:= Amount of decreasing w_I.
w_V	:= Weight of velocity matching.
w_N	:= Weight of neighbors.
w_A	:= Weight of avoidance.
τ_d	:= Distance threshold to dissimilar datoids.
τ_c	:= Threshold for cluster retrieval.

	Corners	*Nested*	*Iris*	*WBC*
N_{dat}	200	200	150	683
N_{iter}	200	300	1500	100
X_{max}	400	400	400	400
V_{max}	10	10	10	10
k	20	20	15	68
w_I	1.0	1.0	1.0	1.0
s_I	0	0	500	0
a_I	0.001	0.001	0.001	0.01
w_V	0.5	0.5	0.5	0.5
w_N	0.5	2.0	0.5	0.5
w_A	0.5	0.5	0.5	0.5
τ_d	10	10	10	10
τ_c	5	10	5	10

Table 1. Used parameters for DSC.

The experiments showed that the parameters aren't too sensitive. In spite of little changes the clustering process works. Merely the numbers of correct clustered data items and of formed clusters change. Therefore, one can work towards a working parameter set for a given clustering problem.

7 Results

In section 6 the used datasets and their parameter settings are described. To evaluate the cluster capabilities, the DSC algorithm was applied to the datasets 50 times, i.e., 50 independent runs for each dataset. The results as shown in Table 2 are the averaged results over these 50 runs. The used measures are described in the following.

First, all correctly clustered data items are counted. For this it is neccessary to determine the cluster type of a cluster which means, to which of the real known classes of the dataset belongs the cluster. If C is the set of computed clusters $C_i \subseteq X$ and T the set of labels t of the real known classes of a dataset, then the class of cluster C_i is computed as shown in Eq. (19),

$$Class(C_i) = c \quad \text{s.t.} \quad N_{ci} = \max_{t \in T}\{N_{ti}\}, \;\; c \in T \tag{19}$$

where N_{ti} is the number of data items of class t within cluster C_i. The class of a cluster is the class of the biggest part of data items belonging to the same class. With this assumption the proportion of correctly clustered data items is just the number of data items which represent the class of the cluster summed over all clusters C_i as shown in Eq. (20).

$$Correct(C) = \frac{1}{|X|} \sum_{C_i \in C} \max_{t \in T}\{N_{ti}\} \tag{20}$$

$Correct(C)$ is to maximize. A second measure is just the number of found clusters $|C|$. This is important, because in DSC the number of clusters is not given by the user.

Another measure used is the entropy within a cluster as in Eq. (21),

$$Entropy(C_i) = -\frac{1}{\log(|X|)} \sum_{t \in T} \frac{N_{ti}}{N_i} \log(\frac{N_{ti}}{N_i}) \tag{21}$$

where N_i is the size of cluster C_i, i.e., $N_i = |C_i|$. The entropy measures the relative degree of randomness of cluster C_i. That is, it is 0 if the cluster contains data only from one class and 1 if the cluster is uniformly filled with data of all classes. The overall entropy $Entr(C)$ as given in Eq. (22) is the average over all clusters and is to minimize.

$$Entr(C) = \frac{1}{|C|} \sum_{C_i \in C} Entropy(C_i) \tag{22}$$

The last measure is the F-measure known from information retrieval as shown in Eq. (23). It uses the purity of the considered cluster C_i with $Prec(t,C_i) = \frac{N_{ti}}{N_i}$, i.e., how strong belongs cluster C_i completely to class t. Furthermore, it considers, how much of the data of class t are contained within cluster C_i with $Rec(t,C_i) = \frac{N_{ti}}{N_t}$ and N_t being the number of data in class t.

$$FMeasure(t,C_i) = \frac{2 \cdot Prec(t,C_i) \cdot Rec(t,C_i)}{Prec(t,C_i) + Rec(t,C_i)} \tag{23}$$

The best situation is to have each cluster consisting completely of data of the same class t ($Prec(t,C_i) = 1$) and for each class t having all data placed in just one cluster ($Rec(t,C_i) = 1$). This measure is limited to $[0,1]$ and to be maximized. The overall F-measure value is determined as in Eq. (24).

$$FMeas(C) = \sum_{t \in T} \frac{N_t}{|X|} \max_{C_i \in C} \{FMeasure(t,C_i)\} \tag{24}$$

All described measures are computed for each dataset and presented in Table 2. For each dataset, simply the Euclidian distance normalized by the maximal possible dissimilarity between two data items is used as similarity function as defined in Eq. (25).

$$S(d_i,d_j) = 1 - \frac{\sqrt{\sum_{c=1}^{n}(d_{ic} - d_{j_c})^2}}{\sqrt{\sum_{v=1}^{n} \max(A_v)^2}} \tag{25}$$

The synthetic dataset *Corners* is a very simple one having four separable classes. This works very well as expectable.

The synthetic dataset *Nested* is not so simple but can be solved very good by DSC. The reason is that the clustering doesn't occur in the attribute space, but on a plane where the data items are carried by datoids. The datoids interact according to their similarity and not only to their positions on the plane.

The real dataset *Iris* is not easy, because two attributes of two classes are strongly correlated. But the results of DSC are comparable with other clustering methods as can be seen in Table 3. There, DSC is compared to Ant-based clustering [4] and the well-known *k*-means algorithm. The comparative values are taken from [5]. On the one hand the determined number of clusters in average is a bit more stable in Ant-based clustering compared to DSC. In *k*-means the number of expected clusters is given *a priori*. Thus, this point is not comparable. On the other hand, the F-measure value of DSC is better than those of Ant-based clustering and *k*-means. That means, the computed clusters have a better purity.

The real dataset *WBC* is not solved well by DSC. An acceptable high number of similar data items are correctly clustered, but DSC produces too much clusters at all. This results in a bad F-measure value as revealed in Table 4, because the data items of a class are spread over several clusters. Ant-based clustering and *k*-means are better in clustering *WBC*.

The number of clusters and the F-measure values of the real datasets reveal one weak point of DSC. DSC sometimes produces several sub-swarms (i.e., clusters) which belong to the same class t while having a purity of 1 ($Prec(t,C_i) = Prec(t,C_j) = 1$, $C_i \neq C_j$). That is the data items of a class can be split up to several clusters and those clusters consist of data items of just this class.

A positive property of DSC is the transformation of the data items to the plane of the datoids. This is an advantage in problems like *Nested*.

The entropy of all datasets shows that each determined cluster is good dominated by data items of the same class. The clusters aren't mixed strongly. It seems that DSC produces clusters with a good purity.

	Corners	*Nested*	*Iris*	*WBC*
$Correct(C) \cdot 100$	100.0%	100.0%	94.786667%	94.585652%
standard deviation	0.0%	0.0%	3.496513%	5.754342%
$\|C\|$ (real number)	4 (4)	2 (2)	4 (3)	6 (2)
standard deviation	0	0	1.363818	1.462874
$Entr(C)$	0	0	0.026367	0.02042
standard deviation	0	0	0.006275	0.006602
$FMeas(C)$	1	1	0.830683	0.685732
standard deviation	0	0	0.073264	0.062420

Table 2. Averaged results over 50 independent runs for all datasets.

Iris	DSC	Ant-based Clustering	*k*-means
$\|C\|$	4	3.02	3 (given *a priori*)
standard deviation	1.36	0.14	0 (given *a priori*)
$FMeas(C)$	0.830683	0.816812	0.824521
standard deviation	0.073264	0.014846	0.084866

Table 3. Results for the dataset *Iris*. The comparative values are taken from [5].

WBC	DSC	Ant-based Clustering	*k*-means
$\|C\|$	6	2	2 (given *a priori*)
standard deviation	1.46	0	0 (given *a priori*)
$FMeas(C)$	0.685732	0.967604	0.965825
standard deviation	0.062420	0.001447	0

Table 4. Results for the dataset *WBC*. The comparative values are taken from [5].

8 Conclusion

The experimentations show that it is possible to cluster data by using swarm techniques. The power of swarm clustering is due to the local interaction between similar datoids. The data items to be clustered aren't clustered in their attribute space, but in the space of the datoids. Therefore, data items belonging to the same class don't need to be close in the attribute space. Datoids move in their space and have an affinity to their nearest similar neighbors. This allows the datoids to perform good on problems like the *Nested* dataset. The data items in the top region and bottom region of the *Nested* dataset aren't close in their space. But the datoids group together with their similar neighbors. The next near similar neighbors of the data items in the bottom region are the ones on both sides. And the next near similar neighbors of the side regions are the ones in the top region. Because of this behavior based on local interaction between similar datoids the data items of the four sides can be separated from the nested data items.

DSC uses a similarity function S to determine the similarity between two datoids. Thus, it can work with each data structure or attribute type, because DSC only gives

the data objects carried by datoids to this similarity function. Therefore, a lot of properties of DSC depend on the used similarity function. One disadvantage of DSC is the great number of needed parameters. On the other hand, you don't need to specify the number of clusters *a priori*.

References

1. Berkhin P (2002) Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, California
2. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York, NY
3. Deneubourg JL, Goss S, Franks N, Sendova-Franks A, Detrain C, Chretien L (1991) The Dynamics of Collective Sorting: Robot-like Ants and Ant-like Robots. In: Proc. First International Conference on Simulation of Adaptive Behaviour: From Animals to Animats, pp. 356-363, MIT Press, Cambridge, MA
4. Handl J, Knowles J, Dorigo M (2003) Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1D-som. Technical Report TR/IRIDIA/2003-24. IRIDIA, Universite Libre de Bruxelles, Belgium
5. Handl J, Knowles J, Dorigo M (2003) On the performance of ant-based clustering. In: Proc. 3nd International Conference on Hybrid Intelligent Systems, pp. 204-213, IOS Press, Amsterdam, The Netherlands
6. Kanade PM, Hall LO (2003) Fuzzy Ants as a Clustering Concept. In: Proc. 22nd International Conference of the North American Fuzzy Information Processing Society, pp. 227-232, Chicago, Piscataway, NJ: IEEE Service Center
7. Kennedy J, Eberhart RC (1995) Particle Swarm Optimization. In: Proc. IEEE International Conference on Neural Networks, pp. 1942-1948, Perth, Australia, IEEE Service Center, Piscataway, NJ
8. Kennedy J, Eberhart RC, Shi Y (2001) Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco, ISBN: 1-55860-595-9
9. Monmarche N, Slimane M, Venturini G (1999) AntClass: discovery of clusters in numeric data by an hybridization of an ant colony with the kmeans algorithm. Internal Report No. 213, E3i, Laboratoire d'Informatique, Universite de Tours
10. Morse DH (1970) Ecological aspects of some mixed-species foraging flocks of birds. Ecological Monographs: Vol. 40, No. 1, pp. 119-168
11. Murphy PM, Aha DW (1994) UCI Repository of machine learning databases. [http://www.ics.uci.edu/∼mlearn/MLRepository.html], Irvine, CA: University of California, Department of Information and Computer Science
12. Omran M, Salman A, Engelbrecht AP (2002) Image Classification using Particle Swarm Optimization. In: Proc. 4th Asia-Pacific Conference on Simulated Evolution and Learning, pp. 370-374, Singapore
13. Reynolds CW (1987) Flocks, herds and schools: a disctributed behavioral model. Computer Graphics 21, pp. 25-33
14. Shi YH, Eberhart RC (1998) A Modified Particle Swarm Optimizer. In: Proc. IEEE International Conference on Evolutionary Computation, pp. 69-73, IEEE Press, Piscataway, NJ
15. van der Merwe DW, Engelbrecht AP (2003) Data clustering using particle swarm optimization. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, pp. 215-220, Piscataway, NJ: IEEE Service Center