Data Swarm Clustering

Christian Veenhuis and Mario Köppen
Fraunhofer Institute for Production Systems and Design Technology
Department Pattern Recognition
Pascalstr. 8-9, 10587 Berlin, Germany
Email: veenhuis@ipk.fhg.de, koeppen@ipk.fhg.de

Abstract— **A data clustering algorithm based on swarm theory is presented. 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 so-called datoids which form a swarm on a two-dimensional plane. While iterating, this swarm divides into sub swarms moving over the plane and consisting of datoids carrying similar data. After the last iteration these sub swarms of datoids can be grouped together as clusters.**

I. 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.

Data clustering is concerned with the division of data into groups (i.e., clusters) of similar data. 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.

Up to now, there aren't much clustering algorithms based on swarms. Ant-based clustering [2] simulates the behavior of ants forming piles of corpse or items, e.g., to clean their nest. 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.

Another type of clustering was introduced by Omran et al. [7]. They use Particle Swarm Optimization to determine a given number of centroids (i.e., center of clusters). For this they use particles which contain all centroid vectors in a row. This way a particle has the dimensionality of number of centroids times dimension of centroids. The swarm consists of a lot of possible centroid vector sets.

The method presented in this paper assigns the data items to be clustered 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 paper is organized as follows. Section II gives a brief overview of clustering. Flock Algorithms and Particle Swarm Optimization are described in sections III and IV. The Data Swarm Clustering algorithm is described in section V. The used experiments and results are shown in sections VI and VII. Section VIII draws some conclusions and shows some future work possibilities.

II. Data Clustering

The task of data clustering is to divide a set of data X into sub-sets $C_i \subseteq X$. Often the data are points $d_i = (d_{i1}, \cdots, d_{in}) \in A$ 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. A sub-set $C_i \subseteq X$ is called a cluster and all clusters together result in X (i.e., $X = \bigcup_i C_i$). Usually clusters are pair-wise disjoint (i.e., $\bigcap_i C_i = \emptyset$).

The data contained within one cluster C_i are similar in some way and dissimilar to data contained in other clusters. Often simply distances between points 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 / 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].

III. 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 [8] 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.
- **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, they steer 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.

Althoug 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.

IV. Particle Swarm Optimization

Particle Swarm Optimization (PSO) as introduced by Kennedy and Eberhart [4] [5] 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 with 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 number of dimensions of the problem (i.e., the size of the search space $\mathbb{R}^{N_{dim}}$), N_{part} the number of particles and $\mathcal{P}$ the set of particles $\mathcal{P} = \{P(1),...,P(N_{part})\}$. Each particle $P(i) = (x_i, v_i, l_i)$ consists of 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 ($l_i \in \mathbb{R}^{N_{dim}}$) by this particle.

First, the set of particles is initialized with randomly created particles. The first l_i are set to the appropriate first x_i. Then, the next position of each particle $P(i)$ can be computed as shown in equations 1 and 2.

$$v_i^{(t+1)} = w_I^{(t)} v_i^{(t)} + w_L R_{0,1}(l_i^{(t)} - x_i^{(t)}) + w_N R_{0,1}(n_i^{(t)} - x_i^{(t)}) \quad (1)$$

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

$R_{0,1}$ means a random-number between 0 and 1. $n_i^{(t)} \in \mathbb{R}^{N_{dim}}$ represents the best found local position of the nearest neighbor particle at time t. Because there are several possibilities to define the neighborhood of a particle [5], the nearest neighbor particle can be, e.g., the best particle of a pre-defined neighborhood, the nearest neighbor according to the distance in search space etc. The inertia weight $w_I^{(t)}$ determines the influence of the own velocity, i.e., how strong is 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 [9] [5]. w_L is the influence of the best local position found so far. The influence of the nearest neighbor particle is denoted with w_N.

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

If the new position $x_i^{(t+1)}$ has a better fitness than the best solution found so far of particle $P(i)$, it's stored in memory as shown in eq. 3. The fitness is determined by a goodness function $G : \mathbb{R}^{N_{dim}} \rightarrow \mathbb{R}$.

$$l_i^{(t+1)} = \begin{cases} x_i^{(t+1)} & , \quad G(x_i^{(t+1)}) < G(l_i^{(t)}) \\ l_i^{(t)} & , \quad otherwise \end{cases} \quad (3)$$

The best solution of the run is found at particle $P(b)$ with the best lokal solution l_b. Best solution l_b is always element of $\{l_i\}, \forall i \in \{1, \cdots, N_{part}\}$. The best fitness value is $G(l_b) = \min_{i \in \{1,\cdots,N_{part}\}}\{G(l_i)\}$.

V. 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 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 quite similar to the first three rules of Flock Algorithms as described in section III.

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 $\mathcal{D}$ the set of

datoids $\mathcal{D} = \{D(1), ..., D(N_{dat})\}$. 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). Each datoid is placed on a quadratic two-dimensional plane with a site length of X_{max}. That is, X_{max} restricts the two-dimensional plane in both dimensions. The maximal possible distance between two datoids is denoted as $d_{max} = \sqrt{X_{max}^2 + X_{max}^2}$.

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

1) Initialize $\mathcal{D}$ with randomly created datoids (see section V-A).
2) For n = 1 to N_{iter}: iterate swarm (see section V-B).
3) Retrieve the clusters (see section V-C).

A. Initialization

The initialization phase creates the set $\mathcal{D}$ 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 $D(i) \in \mathcal{D}, \forall i \in \{1, \cdots, N_{dat}\}$ the following initialization is performed. First, the data object o_i bound to the datoid $D(i)$ is set to the appropriate data $d_i \in X$.

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 IV).

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

B. Iteration

While iterating, a similarity function is needed to provide a similarity measure between two datoids. This similarity function as shown in eq. 4 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.

$$S : X \times X \rightarrow [0, 1] \tag{4}$$

The distance between two datoids on the two-dimensional plane is determined by Euclidian distance $d : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}$ with $d(a, b) = \sqrt{\sum_{e=1}^{2} (a_e - b_e)^2}$.

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

First, the nearest similar neighbor $D(n_{i,similar})$ of $D(i)$ is determined. For this, the similarity distance function $SD(i,j) = S(o_i, o_j) d(x_i, x_j) + (1 - S(o_i, o_j)) d_{max}$ is used to compute the distance between two datoids. 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 $n_{i,similar} = n$, $SD(i,n) = \min\{SD(i,j) \mid \forall j \in \{1, \cdots, N_{dat}\}, i \neq j\}$. The nearest similar neighbor is needed for rule 3: *Velocity Matching of Similar Datoids*.

Then, the nearest dissimilar neighbor $D(n_{i,dissimilar})$ of $D(i)$ is determined. For this, the dissimilarity distance function $DD(i,j) = (1 - S(o_i, o_j)) d(x_i, x_j) + S(o_i, o_j) d_{max}$ is used to compute the distance between two datoids. 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 $n_{i,dissimilar} = n$, $DD(i,n) = \min\{DD(i,j) \mid \forall j \in \{1, \cdots, N_{dat}\}, i \neq j\}$. 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. If $n = (n_1, \cdots, n_k)$ represents the sequence of indices of the k nearest similar neighbors (according to SD), then the center of those neighbors is computed as $c_{i,similar} = (\frac{1}{k} \sum_{j=1}^{k} x_{n_j 1}, \frac{1}{k} \sum_{j=1}^{k} x_{n_j 2})^T$.

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. 5. The degree of matching is weighted with w_V.

$$\Delta_{velo} = w_V \, (v_{n_{i,similar}}^{(t)} - v_i^{(t)}) \tag{5}$$

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 in DSC a datoid steers to the center of the k nearest similar neighbors as in Flock Algorithms. This way the neighborhood is variable which 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.

$$\Delta_{neighbors} = S(o_i, o_{n_{i,similar}}) \, w_N \, R_{0,1} \, (c_{i,similar}^{(t)} - x_i^{(t)}) \tag{6}$$

The influence of the neighbors $\Delta_{neighbors}$ as shown in eq. 6 is computed quite similar to PSO. The difference of the current position $x_i^{(t)}$ and the center of the similar neighbors $c_{i,similar}^{(t)}$ is weighted by w_N. 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., $d(x_i, x_{n_{i,dissimilar}}) < \tau_d$). 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. 7).

$$a = \begin{cases} x_i^{(t)} - x_{n_{i,dissimilar}}^{(t)} & , \; d(x_i, x_{n_{i,dissimilar}}) < \tau_d \\ 0 & , \; otherwise \end{cases}$$

$$\Delta_{avoidance} = (1 - S(o_i, o_{n_{i,similar}}))\, w_A\, a \qquad (7)$$

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. 8 can be computed.

$$v_i^{(t+1)} = w_I^{(t)}\, v_i^{(t)} + \Delta_{velo} + \Delta_{neighbors} + \Delta_{avoidance} \qquad (8)$$

The inertia weight $w_I^{(t)}$ determines the influence of the previous velocity. As in PSO this weight is decreased over time (see section IV). 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 $[-V_{max}, +V_{max}]$, i.e., $v_{ie}{}^{(t+1)} = \max[-V_{max}, \min(+V_{max}, v_{ie}{}^{(t+1)})], \forall e \in \{1,2\}$. 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
$n_{i,dissimilar} \leftarrow$ index of nearest dissimilar neighbor
$c_{i,similar} \leftarrow$ center of k nearest similar neighbors
Compute new velocity $v_i^{(t+1)}$ by eq. 8
Compute new position $x_i^{(t+1)}$ by eq. 2

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

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

C. 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 $D(i), D(j) \in \mathcal{D}$ $(\forall i, j \in \{1, \cdots, N_{dat}\}, i \neq j)$ whose Euclidean distance $d(x_i, x_j) < \tau_c$ is lower than a given threshold τ_c are grouped together as a cluster.

VI. Experimental Setup

To evaluate the cluster capabilities of DSC it was tested on four datasets: two synthetically generated and two real life datasets.

The synthetical datasets as shown in figure 1 are (1) *Corners* containing 4 randomly created clusters located at the 4 corners of the grid and (2) *Nested* containing 2 randomly created clusters, whereby one is located in the center area of the grid and the other surrounds it.

The real life datasets are (1) *Iris* containing 3 clusters in 150 records with 4 numerical attributes and (2) Wisconsin *Breast* Cancer containing 2 clusters in 683 records with 10 numerical attributes. Both datasets are taken from the UCI Repository Of

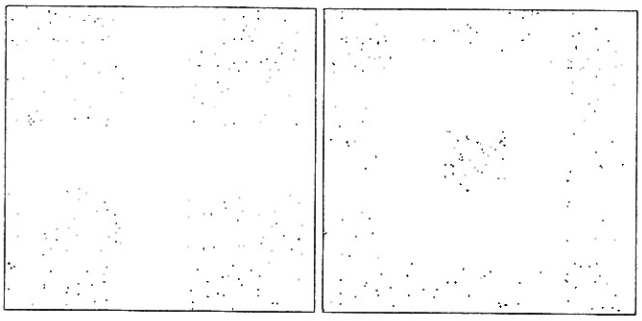

Fig. 1. Synthetical datasets *Corners* (left) and *Nested* (right).

Machine Learning Databases [6] and the attribute values were normalized.

In table I 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
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*	*Breast*
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 I
USED PARAMETERS FOR DSC

VII. RESULTS

In section VI 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 II 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. 9,

$$Class(C_i) = c,\ N_{ci} = \max_{t \in T}\{N_{ti}\},\ c \in T \tag{9}$$

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. 10.

$$Correct(C) = \frac{1}{|X|}\sum_{C_i \in C}\max_{t \in T}\{N_{ti}\} \tag{10}$$

$Correct(C)$ 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. 11,

$$Entropy(C_i) = -\frac{1}{\log(|X|)}\sum_{t \in T}\frac{N_{ti}}{N_i}\log(\frac{N_{ti}}{N_i}) \tag{11}$$

where N_i is the size of cluster 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 $Entr(C) = \frac{1}{|C|}\sum_{C_i \in C} Entropy(C_i)$ is the average over all clusters and is to minimize.

The last measure is the F-measure known from information retrieval as shown in eq. 12. 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.

$$FMeasure(t, C_i) = \frac{2 \cdot Prec(t, C_i) \cdot Rec(t, C_i)}{Prec(t, C_i) + Rec(t, C_i)} \tag{12}$$

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. 13.

$$FMeas(C) = \sum_{t \in T}\frac{N_t}{|X|}\max_{C_i \in C}\{FMeasure(t, C_i)\} \tag{13}$$

All described measures are computed for each dataset and presented in table II. For each dataset, simply the Euclidian distance normalized by the maximal possible dissimilarity between two data items is used as similarity function as depicted in eq. 14.

$$S(d_i, d_j) = 1 - \frac{\sqrt{\sum_{e=1}^{n}(d_{ie} - d_{je})^2}}{\sqrt{\sum_{v=1}^{n} \max(A_v)^2}} \quad (14)$$

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 (compare to [3]).

The real dataset *Breast* is not solved well by DSC. A 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.

	Corners	*Nested*	*Iris*	*Breast*
$Correct(C) \cdot 100$	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 II
RESULTS OF THE RUNS

The number of clusters and the F-measure 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 the datasets shows that each determined cluster is good dominated by data items of the same class. The clusters aren't mixed strongly.

VIII. Conclusions and Future Work

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*.

The DSC algorithm as presented in this paper is a first step. There is room for a lot of future work as, e.g.:

- Improving the cluster retrieval algorithm to merge clusters belonging to the same class. This will improve the F-measure value and correct the number of determined clusters.
- Analysing the dependencies of the parameters to derive functions or rules of thumb to determine them for a given clustering problem.
- Doing a comparative study on other clustering methods and testing more datasets with different similarity functions.
- Trying modified rules to improve the behavior, e.g., of avoiding the approaching of dissimilar sub-swarms or to advantage the merging of similar ones.
- To get rid of the cluster retrieval step, it should be examined whether it's possible to connect very similar datoids while iterating.

References

[1] P. Berkhin, *Survey of clustering data mining techniques*, Technical report, Accrue Software, San Jose, California, 2002

[2] J. Handl, J. Knowles and M. Dorigo, *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, 2003

[3] Julia Handl, Joshua Knowles and Marco Dorigo, *On the performance of ant-based clustering*, Proc. 3nd International Conference on Hybrid Intelligent Systems, IOS Press, Amsterdam, The Netherlands, 2003

[4] J. Kennedy and R.C. Eberhart, *Particle Swarm Optimization*, IEEE International Conference on Neural Networks, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995

[5] J. Kennedy and R.C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann Publishers, ISBN: 1-55860-595-9, 2001

[6] P.M. Murphy, D.W. Aha, *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, 1994

[7] M. Omran, A. Salman, AP. Engelbrecht, *Image Classification using Particle Swarm Optimization*, 4th Asia-Pacific Conference on Simulated Evolution and Learning, 2002

[8] C.W. Reynolds, *Flocks, herds and schools:a disctributed behavioral model*, Computer Graphics 21:25-33, 1987

[9] Y.H. Shi, R.C. Eberhart, *A Modified Particle Swarm Optimizer*, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, 1998