17th Int. Conf. on Scientific and Statistical Database Management (SSDBM'05), Santa Barbara, CA, 2005.

Clustering Moving Objects via Medoid Clusterings

Hans-Peter Kriegel, Martin Pfeifle
Institute for Computer Science
University of Munich, Germany
{*kriegel, pfeifle*}*@ifi.dbs.lmu.de*

Abstract

Modern geographic information systems do not only have to handle static information but also dynamically moving objects. Clustering algorithms for these moving objects provide new and helpful information, e.g. jam detection is possible by means of these algorithms. One of the main problems of these clustering algorithms is that only uncertain positional information of the moving objects is available. In this paper, we propose clustering approaches which take these uncertain positions into account. The uncertainty of the moving objects is modelled by spatial density functions which represent the likelihood that a certain object is located at a certain position. Based on sampling, we assign concrete positions to the objects. We then cluster such a sample set of objects by standard clustering algorithms. Repeating this procedure creates several sample clusterings. To each of these sample clusterings a ranking value is assigned which reflects its distance to the other sample clusterings. The clustering with the smallest ranking value is called the medoid clustering and can be regarded as the average clustering of all the sample clusterings. In a detailed experimental evaluation, we demonstrate the benefits of these medoid clusterings. We show that the medoid clustering is more suitable for clustering moving objects with fuzzy positions than arbitrary sample clusterings or clusterings based on the distance expectation values between the fuzzy positions of the moving objects.

1. Introduction

Clustering algorithms aim at grouping similar objects together, whereas dissimilar objects are assigned to different clusters. In the area of clustering moving objects, the similarity criterion is the distance between the objects. If we cluster objects moving on a spatial network [21], the distance between the objects on the network is used for clustering. If we aim at clustering objects which can freely move, the Euclidean distance between the objects can be used to measure the similarity, i.e. the closeness, between the objects [15].

Clustering moving objects has many different application ranges. For instance, clustering algorithms on a spatial network can be used for traffic jam detection and prediction. Clustering algorithms on freely moving objects can be used for weather forecasting [6], for detecting outliers, or for detecting animal migrations.

The problem of clustering moving objects is that often no accurate positional information is available. For instance, due to technical problems, the GPS system might not be able to pinpoint the exact positions of the moving objects. Another reason for uncertain positional information is that due to efficiency reasons it is not possible to update the exact position of the objects continuously. Clustering algorithms therefore have to deal with uncertain, outdated positional information.

In this paper, we propose an approach for clustering moving objects with uncertain positional information. We motivate a fuzzy modelling approach for describing moving objects and discuss several strategies which can be used for clustering these objects with standard clustering algorithms. After discussing the problems with the most straightforward approaches for clustering moving objects, we introduce an approach which uses the new concept of clustering rankings. Based on suitable distance functions between clusterings, we determine the medoid clustering from a set of sample clustgerings. The medoid clustering can be regarded as the clustering which represents all sample clusterings in the best possible way. Like ranking queries in databases, we can now return the sample clusterings according to their ranking values. The first returned clustering is the medoid clustering. In a give-me-more manner, the user can ask for more clusterings. Thus, the user gets a better picture of all clusterings which are possible when we cluster moving objects with uncertain positional information.

The remainder of this paper is organized as follows. In Section 2, we present the related work in the area of clustering moving objects. In Section 3, we introduce our fuzzy modelling approach which takes the uncertain positions of the moving objects into account. In Section 4, we present different approaches for clustering fuzzy moving objects. Our final approach relies on distance functions between clusterings. These distance functions are introduced in Section 5. In Section 6, we present our experimental evaluation, and conclude the paper in Section 7 with a short summary and a few remarks on future work.

2. Related Work

In this section, we will present the related work in the area of clustering moving objects. In Section 2.1, we first classify well-known clustering algorithms according to different categorization schemes. Then, in Section 2.2, we present the basic concepts of fuzzy clustering algorithms, and describe how the approach of this paper differs from the fuzzy clustering approaches presented in the literature. Finally, in Section 2.3, we present various approaches for clustering moving objects as presented in the literature.

2.1. Clustering Algorithms

Clustering algorithms can be classified along different, independent dimensions. One well-known dimension categorizes clustering methods according to the *result* they produce. Here, we can distinguish between *hierarchical* and *partitioning clustering* algorithms [12]. Partitioning algorithms construct a flat (single level) partition of a database D of n objects into a set of k clusters such that the objects in a cluster are more similar to each other than to objects in different clusters. Hierarchical algorithms decompose the database into several levels of nested partitionings (clusterings), represented for example by a dendrogram, i.e. a tree that iteratively splits D into smaller subsets until each subset consists of only one object. In such a hierarchy, each node of the tree represents a cluster of D.

Another dimension according to which we can classify clustering algorithms is from an *algorithmic* point of view. Here we can distinguish between *optimization based* or *distance based* algorithms and *density based* algorithms. Distance based methods use the distances between the objects directly in order to optimize a global criterion. In contrast, density based algorithms apply a local cluster criterion. Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density (noise).

The following representatives of the 4 categories are used throughout our experimental evaluation:

	distance based	*density based*
partitioning	*k*-means[16]	DBSCAN[7]
hierarchical	Single-Link[12]	OPTICS[1]

2.2 Fuzzy Clustering

In real applications there is very often no sharp boundary between clusters so that fuzzy clustering is often better suited for the data. Membership degrees between zero and one are used in fuzzy clustering instead of crisp assignments of the data to clusters. The most prominent fuzzy clustering algorithm is the fuzzy c-means algorithm, a fuzzification of the partitioning clustering algorithm *k*-means. For more details about fuzzy clustering algorithms, we refer the reader to [11].

In contrast to fuzzy clustering algorithms where objects are assigned to different clusters, we cluster in this paper fuzzy object representations. The fuzzy spatial objects are assigned to exactly one cluster.

2.3. Clustering Moving Objects

In this section, we present some recent approaches from the literature dealing with the problem of clustering moving objects.

Yiu and Mamoulis [21] tackled the complex problem of clustering moving objects based on a spatial network. Here, the distance between the objects is defined by their shortest path distance over the network. Based on this distance measure they proposed variants of well-known clustering algorithms.

In [15], Han et al. applied micro-clustering [23] to moving objects. They propose techniques to keep the spatial extension of the moving micro-clusters small. To detect crucial events, e.g. split events, they measured the compactness of the moving micro-clusters by means of their bounding rectangles. If the size of the bounding rectangle exceeds a certain threshold, the micro-cluster is split. Different clustering algorithms can then be carried out on the moving micro-clusters instead of the individual points. In contrast to the experimental approach presented in [15], Har-Peled presented a more theoretical approach which also sacrifices quality in order to gain efficiency [10].

Clustering moving objects is not only interesting in its own, but can also beneficially be used for spatio-temporal selectivity estimation [22]. Zhang and Lin proposed a new clustering based spatio-temporal histogram, called CSTH, which allows to estimate the selectivity of predictive spatio-temporal queries accurately.

3. Modelling Fuzzy Moving Objects

In this section, we motivate the use of spatial density probability functions for describing the location of moving objects. This approach is quite similar to the approach presented by Behr and Güting [3] which use “degree or affinity” values to describe the probability that a certain point is included in a fuzzy spatial object.

Normally modern GPS systems can determine the exact position of moving objects very accurately. But, for instance, in the case of a war, this precision is reduced due to security aspects. Although, the system assigns a position $p(o, t) = (x, y)$ to each object o at a certain time t, we cannot be sure that the object o is located at the point (x, y) at time t. Nevertheless, it is very likely, that o is close to (x, y). This closeness can be modelled by assigning a 2-dimensional Gaussian density probability function o_{fuzzy} to the object (cf. Figure 1). The center of this probability function is at point (x, y) and the standard deviation σ is determined by the accuracy of the GPS system.

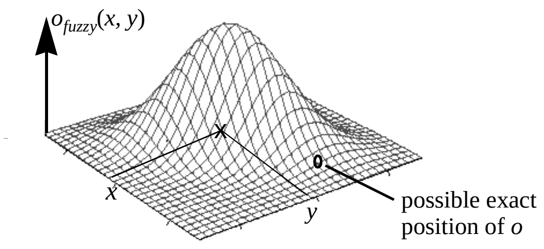


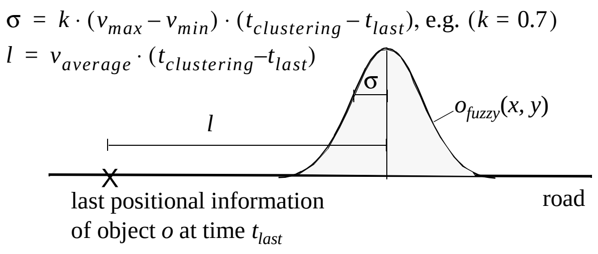
Figure 1. Fuzzy object representations for freely moving objects.

There exist other examples where it is beneficial to assign a 2-dimensional Gaussian distribution function o_{fuzzy} to an object o. For instance, animals or pedestrians which can freely move in the 2-dimensional space with a certain maximum velocity can be modelled by such a spatial density function. In order to cluster these objects effectively, it also seems reasonable to describe their positions by a 2-dimensional density probability function (cf. Figure 1). The center of this probability function is the last sent position of the object. The standard deviation σ depends on the maximum velocity of the object and the time passed since the object last sent its exact position.

If we assume that the object moves on a spatial network, e.g. cars moving on roads, we can assign a 1-dimensional Gaussian distribution function to the object (cf. Figure 2). The center of this probability function is a certain distance l away from the last sent position of the object. The value of l depends on the average velocity v_{avg} and the time t_{last} which passed since the object last sent its exact position. The standard deviation σ depends on the difference between the maximum and minimum assumed velocity, i.e. v_{max} - v_{min}, and on t_{last}.

Finally, if we, for instance, follow the approach presented in [15], we also cannot determine an exact position of an object o at clustering time. But as we know that the object is located within the bounding rectangle of the moving micro-cluster, we can assign to each object of the micro-cluster a density-probability function which assigns to each position a value $1/A_{box}$ where A_{box} denotes the area of the bounding rectangle. Note that we assign to each object of a micro-cluster the same density probability function (cf. Figure 3).

As shown in the above examples the position of a moving object cannot be described by only one single positional value. A better way, to describe a fuzzy moving object is to assign to each object a set of possible positions. To each of these positions, we assign a probability value which indicates the likelihood that this position is the exact one. Obviously, the sum of all these probability values is equal to 1.



$\sigma = k \cdot (v_{max} - v_{min}) \cdot (t_{clustering} - t_{last})$, e.g. $(k = 0.7)$
$l = v_{average} \cdot (t_{clustering} - t_{last})$
σ
$o_{fuzzy}(x, y)$
l
last positional information of object o at time t_{last}
road

Figure 2. Fuzzy object representations for objects moving on a spatial network.

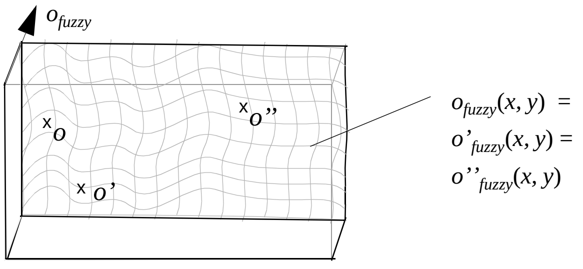


Figure 3. Fuzzy object representations for objects within a moving micro-cluster.

Definition 1 (*fuzzy moving object*)
Let $o \in DB$ be a moving object. To each moving object, we assign a fuzzy moving object function o_{fuzzy}: $IR^2 \rightarrow IR_0^+ \cup \infty$ for which the following condition holds:

$$\int_{IR^2}\int o_{fuzzy}(x, y)dxdy = 1$$

Figure 1, 2 and 3 show different fuzzy moving object functions o_{fuzzy} for two dimensional moving objects o. The functions o_{fuzzy} assigns a probability value $o_{fuzzy}(x, y) > 0$ to each possible position (x, y) of o. In the following, we use the term fuzzy moving object for both the object o and the corresponding function o_{fuzzy}.

4. Clustering Fuzzy Moving Objects

In this section, we present three different approaches which enable us to cluster fuzzy moving objects. All three approaches are based on sampling. In Section 4.1, we determine for each object o a concrete position based on the corresponding fuzzy moving object function. We use the resulting sample points as input parameters for the clustering algorithms. In Section 4.2, we carry out the clustering algorithms based on the distance expectation values between our fuzzy moving objects. The distance expectation values between our fuzzy moving objects are again computed by means of sampling. In Section 4.3, we determine a medoid clustering from a set of sample clusterings. The sample clusterings are computed as shown in Section 4.1. Then, we use suitable distance functions (cf. Section 5) between our sample clusterings to determine the corresponding medoid clustering.

4.1. Sampling

The most straightforward approach for clustering fuzzy moving objects is to assign to each moving object o an exact position according to its spatial density-probability function o_{fuzzy}. Figure 4 shows two possible positions p' and p'' of our fuzzy moving object o. Although position p' is much more

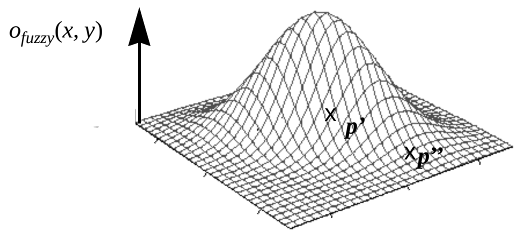


Figure 4. Two possible positions *p'* and *p'* of a moving object *o*.

likely, it is also possible that o is at position p''. For each fuzzy object o_{fuzzy}, we assume a position $p \in IR^2$. We can then apply any given clustering algorithm (cf. Section 2.1) to our fuzzy moving objects. The similarity between two fuzzy moving objects o_{fuzzy} and o'_{fuzzy} is then determined by an application dependent distance function, e.g. the Euclidean distance or the network distance, between the assumed positions p and p'. Based on this simple similarity measure between two fuzzy objects, we can apply any standard clustering algorithm.

Note that the thereby created clustering heavily depends on what positions we assumed for our fuzzy moving objects. Figure 5, for instance, shows a density-based clustering [7] based on sample positions. The resulting sample clustering does not reflect the intuitive clustering. If we look at the figure, we would rather derive a clustering $Cl = \{\{o_1, o_2\}, \{o_3, o_4\}\}$ which groups o_1 and o_2 together and o_3 and o_4. On the other hand, the sample clustering groups o_2 and o_3 together and assigns the objects o_1 and o_4 to noise.

4.2. Distance Expectation Values

In this section, we introduce the distance expectation value between fuzzy moving objects. This similarity measure between fuzzy moving objects is based on distance functions which do not express the similarity between two fuzzy moving objects by a single numerical value. Instead, we propose to use *fuzzy distance functions*, where the similarity between two objects is expressed by means of a probability function which assigns a probability value to each possible distance value. Then, we carry out the clustering algorithms based on the expectation values of the fuzzy distance functions (cf. Figure 6).

Definition 2 (*fuzzy distance function*)
Let d: $O \times O \rightarrow IR_0^+$ be a distance function, and let $P(a \le d(o, o') \le b)$ denote the probability that $d(o, o')$ is between a and b. Then, a probability density function p_d: $O \times O \rightarrow (IR_0^+ \rightarrow IR_0^+ \cup \infty)$ is called a *fuzzy distance function* if the following condition holds:

$$P(a \le d(o, o') \le b) = \int_a^b p_d(o, o')(x)dx$$

If the distance $\tau = d(o,o')$ between two objects can exactly be determined, the fuzzy distance function p_d is equal to the dirac-delta-function δ, i.e. $p_d(o, o')(x) = \delta(x\text{-}\tau)$ [2]. Thus the traditional approach can be regarded as a special case of Definition 2.

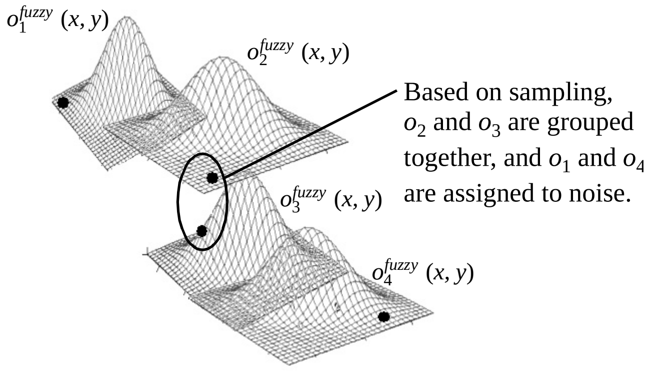

$o_1^{fuzzy}(x, y)$
$o_2^{fuzzy}(x, y)$
Based on sampling, o_2 and o_3 are grouped together, and o_1 and o_4 are assigned to noise.
$o_3^{fuzzy}(x, y)$
$o_4^{fuzzy}(x, y)$

Figure 5. Clustering moving objects based on sampling.

As traditional algorithms can only handle distance functions which yield a unique distance value, we propose to extract the distance expectation value from these fuzzy distance functions. The distance expectation value E_d: $O \times O \rightarrow IR_0^+$ represents the fuzzy distance function in the best possible way by one single value $E_d(o, o') = \int_{-\infty}^{\infty} x \cdot p_d(o, o')(x)dx$ (cf. Figure 6b).

Although, this distance expectation value expresses the distance between two fuzzy moving objects in the best possible way, clustering based on these expectation values might be misleading. Figure 7, for instance, shows the computation of the core object condition for a fuzzy moving object o. Density based clustering algorithms like DBSCAN [7], for instance, decide for each object o whether *MinPts* objects are located within an ε–range of o. If this is the case, we call o a core object. Although, the object o in Figure 7a does not seem to be located in a very dense area, it is a core object according to the distance expectation approach. This holds as the distance expectation value between o and *MinPts*=4 other objects is smaller than ε. On the other hand, it is very unlikely that all *MinPts* objects are indeed located in $N_\varepsilon(o)$. Therefore, the probability that o is a core object is very small. In Figure 7b the reverse situation is sketched. Object o is located in a very dense area but there do not exist *MinPts* objects o' for which $E_d(o, o') \le \varepsilon$ holds. Therefore, o is no core object according to the distance expectation approach, although it is very likely that there exist *MinPts* elements o' for which $d(o, o') \le \varepsilon$ holds. To sum up, clustering based on the distance expectation values might be misleading.

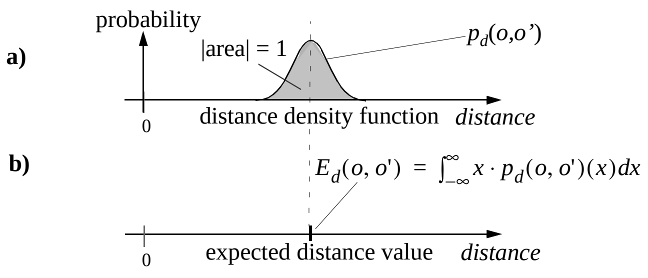
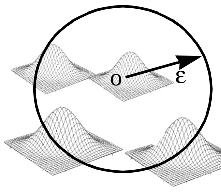

a)
probability
|area| = 1
$p_d(o,o')$
0
distance density function
distance
b)
$E_d(o, o') = \int_{-\infty}^{\infty} x \cdot p_d(o, o')(x)dx$
0
expected distance value
distance

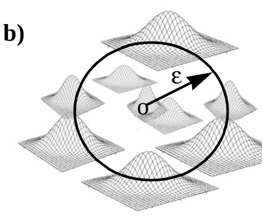
Figure 6. Fuzzy distance functions.

a)



core object according to the approach based on the distance expectation values E_d.

b)



no core object according to the approach based on the distance expectation values E_d.

Figure 7. Determination of the core object property based on the distance expectation value (*MinPts*=4).

4.3. Medoid Clustering

In this section, we propose a third approach which is based on the computation of sample clusterings. As shown in Section 4.1, we can compute a clustering of our moving objects based on sampling. Obviously, we can compute several of these sample clusterings. The question at issue is which is the most suitable of these sample clusterings. The idea of this paper is that we propose to compute the medoid clustering from these sample clusterings. In order to determine the average clustering, we need suitable distance functions between the sample clusterings (cf. Section 5). If we assume that we have functions which express the similarity between two clusterings, we can assign to each clustering Cl a clustering ranking value (cf. Definition 3) which sums up all the distances to all the other clusterings. The clustering with the smallest ranking value is called the medoid clustering (cf. Definition 4).

Definition 3 (*clustering ranking value*)
Let DB be a database of fuzzy objects, and let $Cl_1, ..., Cl_s$ be s sample clusterings of DB. Furthermore, let d be a distance function between clusterings. Then, we assign to each clustering Cl_i a clustering ranking value R_i:

$$R_i = \sum_{\substack{j=1 \\ j \neq i}}^{s} d(Cl_i, Cl_j)$$

Obviously, the clustering having the smallest ranking value represents the set of clusterings in the best possible way. It is called the medoid clustering.

Definition 4 (*medoid clustering*)
Let DB be a set of fuzzy objects, and let $Cl_1, ..., Cl_s$ be s sample clusterings of DB. Furthermore, let d be a distance function between clusterings. Then, Cl_i is called the medoid clustering if $\forall j \in 1 \ldots s : R_i \leq R_j$ holds.

Note, that in the example of Figure 5 it is very unlikely that the clustering $Cl = \{o_2, o_3\}$ is the medoid clustering, although it might be one sample clustering. If we compute, for instance, $s = 5$ clusterings, we might once get the above clustering, once we would assign all objects to noise and three times the sample clusterings are identical to the intuitive clustering $Cl = \{\{o_1, o_2\}, \{o_3, o_4\}\}$. Suitable metric distance functions between clusterings (cf. Section 5) would detect that the medoid clustering corresponds to the intuitive clustering $Cl = \{\{o_1, o_2\}, \{o_3, o_4\}\}$.

Similarly, if we look at the example presented in Figure 7, our medoid clustering approach seems to be more suitable than the approach based on the distance expectation values. Although in Figure 7a it might be possible that one sample clustering would decide that o is a core object, the majority of the samplings would decide that o is no core object. Therefore, it is very likely that the resulting medoid clustering would classify o correctly, i.e. assign it to noise. Similar, in Figure 7b it is very likely that our medoid clustering approach would decide that o is a core object, and, again, would classify the object correctly.

In the following, we will present an approach which helps us to compute the medoid clustering efficiently, if we assume that several slave computers are available.

4.3.1. Parallelization. If we assume that L different slave computers are available, we can easily parallelize the computation of the s sample clusterings. Obviously, each slave has at most $\lceil s/L \rceil$ clusterings to compute. Each slave can independently compute its clusterings and send the final results to all the other slaves. So all slaves have the final s clusterings before the computation of the medoid clustering based on the s sample clusterings starts.

As the computation of the distance measures between the clusterings can be very time consuming, we propose an approach which parallelizes the execution of the $s \cdot (s-1)/2$ distance computations between our s sample clusterings.

The idea is that a master triggers the computation of the clustering distances which are then carried out by the available slaves. Thus, one of the primary goals is that all slaves have an equal workload. To achieve that, the master keeps an $s \times s$ matrix M which indicates which distance computations between clusterings have already taken place. Furthermore, the master maintains an ordered list of the clusterings. The clusterings are ordered ascendingly according to their current clustering ranking values. Initially, all ranking values are set to zero. If a slave has computed a distance between two clusterings Cl_i and Cl_j, the master updates the corresponding ranking values R_i and R_j of these two clusterings and reorganizes the sorted list of clusterings. Furthermore, the master indicates in the matrix that the distance between Cl_i and Cl_j has been computed.

After initializing the matrix and the sorted list of clusterings, the master continuously checks whether there exist a slave S which is out of work. If this is the case, the master takes the first clustering Cl_{first} from the sorted list and checks by means of the matrix M whether all distances between Cl_{first} and the other s-1 clusterings have already been computed. If there is still one distance computation missing, the master asks the slave S to carry out this distance computation. If we assume that the distance d between the clusterings is a metric, i.e. $\forall Cl, Cl' : (d(Cl, Cl') \geq 0)$ holds, the algorithm terminates if all s-1 distance computations of Cl_{first} have already been computed. Then, the master knows that Cl_{first} is the searched medoid clustering without any further distance computations. Note that the ranking value of all the other clusterings can only increase but never decrease if we carry out further distance computations. Obviously, if the user is not only interested in the clustering having the smallest ranking value, the master continues with the above described ranking process.

The approach presented in this section is applicable to arbitrary distance functions between clusterings. In the following section, we introduce concrete distance functions between clusterings which are used throughout our experimental evaluation.

5. Similarity Measures between Clusterings

In the literature there exist some approaches for comparing partitioning [5, 17] and hierarchical [9] clusterings to each other. All of these approaches do not take noise objects into consideration which naturally occur when using density-based clustering algorithms such as DBSCAN [7] or OPTICS [1]. In [13] similarity measures are introduced which are suitable for generally measuring the similarity between partitioning and hierarchical clusterings even if noise is considered. In this section, we adapt these measures to our needs. We introduce distance functions between clusterings which can be used for computing medoid clusterings from sample clusterings. Based on the similarity measures for clusterings, we introduce quality measures which allow us to compare fuzzy clustering approaches to reference clusterings. In our experimental evaluation, we use these quality measures to compare the approaches presented in Section 4 to a reference clustering which is computed based on the exact positions of the moving objects[1].

Let us first introduce some basic terms necessary for describing clusterings. The following definitions are rather generic and can be applied to both reference clusterings and approximated fuzzy clusterings.

Definition 5 (*cluster*)
A *cluster* C is a non-empty subset of objects from a database DB, i.e. $C \subseteq DB$ and $C \neq \emptyset$.

Definition 6 (*partitioning clustering*)
Let DB be a database of arbitrary objects. Furthermore, let C_1, ..., C_n be pairwise disjoint clusters of DB, i.e. $\forall\, i, j \in 1, ..., n$: $i \neq j \Rightarrow C_i \cap C_j = \emptyset$. Then, we call $CL_p = \{C_1, ..., C_n\}$ a *partitioning clustering* of DB.

Note that due to the handling of noise, we do not demand from a partitioning clustering $CL_p = \{C_1, ..., C_n\}$ that $C_1 \cup ... \cup C_n = DB$ holds. Each hierarchical clustering can be represented by a tree. Even the density-based hierarchical clustering algorithm OPTICS which computes a hierarchical clustering order can be transformed into a tree structure by means of suitable cluster recognition algorithms [1, 4, 20].

1. In order to follow the main idea of this paper, you do not have to understand all details presented in this section. Thus, you might continue reading with Section 6.

Definition 7 (*hierarchical clustering*)
Let DB be a database of arbitrary objects. A *hierarchical clustering* is a tree t_{root} where each subtree t represents a cluster C_t, i.e. $t = (C_t, (t_1, ..., t_n))$, and the n subtrees t_i of t represent non-overlapping subsets C_{t_i}, i.e. $\forall i, j \in 1, ..., n$: $i \neq j \Rightarrow C_{t_i} \cap C_{t_j} = \emptyset \wedge C_{t_1} \cup ... \cup C_{t_n} \subseteq C_t$. Furthermore, the root node t_{root} represents the complete database, i.e. $C_{t_{root}} = DB$.

Again, we do not demand from the n subtrees t_i of $t = (C_t, (t_1, ..., t_n))$ that $C_{t_1} \cup ... \cup C_{t_n} = C_t$ holds.

5.1. Similarity Measure for Clusters

As outlined in the last section, both partitioning and hierarchical clusterings consist of flat clusters. In order to compare flat clusters to each other we need a suitable distance measure between sets of objects. The similarity of two clusters depends on the number of identical objects contained in both clusters which is reflected by the *symmetric set difference*.

Definition 8 (*symmetric set difference*)
Let C_1 and C_2 be two clusters of a database DB. Then the *symmetric set difference* d_Δ: $2^{DB} \times 2^{DB} \rightarrow [0..1]$ and the *normalized symmetric set difference* d_Δ^{norm}: $2^{DB} \times 2^{DB} \rightarrow [0..1]$ are defined as follows:

$$d_\Delta(C_1, C_2) = |C_1 \cup C_2| - |C_1 \cap C_2| \quad and$$

$$d_\Delta^{norm}(C_1, C_2) = \frac{|C_1 \cup C_2| - |C_1 \cap C_2|}{|C_1 \cup C_2|}$$

Note that $(2^{DB}, d_\Delta)$ and $(2^{DB}, d_\Delta^{norm})$ are metric spaces.

5.2. Similarity Measure for Partitioning Clusterings

In this section, we will introduce a suitable distance measure between sets of clusters. Several approaches for comparing two sets S and T to each other exist in the literature. In [8] the authors survey the following distance functions: the *Hausdorff distance*, the *sum of minimal distances*, the *(fair-)surjection distance* and the *link distance*. All of these approaches rely on the possibility to match several elements in one set to just one element in the compared set which is questionable when comparing clusterings to each other.

A distance measure on sets of clusters that demonstrates to be suitable for defining similarity between two partitioning clusterings is based on the *minimal weight perfect matching* of sets. This well known graph problem can be applied here by constructing a complete bipartite graph $G = (Cl, Cl', E)$ between two clusterings Cl and Cl'. The weight of each edge $(C_i, C'_j) \in Cl \times Cl'$ in this graph G is defined by the distance $d_\Delta(C_i, C'_j)$ introduced in the last section between the two clusters $C_i \in Cl$ and $C'_j \in Cl'$. A perfect matching is a subset $M \subseteq Cl \times Cl'$ that connects each cluster $C_i \in Cl$ to exactly one cluster $C'_j \in Cl'$ and vice versa. A minimal weight perfect matching is a matching with maximum cardinality and a minimum sum of weights of its edges. Since a perfect matching can only be found for sets of equal cardinality, it is necessary to introduce weights for unmatched clusters when defining a distance measure between clusterings. We propose to penalize each unmatched cluster by its cardinality. Thereby, large clusters which cannot be matched are penalized more than small clusters which is a desired property for an intuitive similarity measure between partitioning clusterings.

Definition 9 (*partitioning clustering distance* $d^{partitioning}_{clustering}$)
Let DB be a database. Let $Cl = \{C_1, ..., C_{|Cl|}\}$ and $Cl' = \{C'_1, ..., C'_{|Cl'|}\}$ be two clusterings. We assume w.l.o.g. $|Cl| \le |Cl'|$. Let π be a mapping that assigns to all $C' \in Cl'$ a unique number $i \in \{1, ..., |Cl'|\}$, denoted by $\pi(Cl') = (C'_1, ..., C'_{|Cl'|})$. The family of all possible permutations of Cl' is called $\Pi(Cl')$. Then the *partitioning clustering distance* $d^{partitioning}_{clustering}: 2^{2^{DB}} \times 2^{2^{DB}} \to IR$ is defined as follows:

$$d^{partitioning}_{clustering}(Cl, Cl') = \min_{\pi \in \Pi(Cl')} \left(\sum_{i=1}^{|Cl|} d_\Delta(C_i, C'_{\pi(i)}) + \sum_{i=|Cl|+1}^{|Cl'|} \left| C'_{\pi(i)} \right| \right)$$

Let us note that the *partitioning clustering distance* is a specialization of the metric *netflow distance* [19]. The partitioning clustering distance $d^{partitioning}_{clustering}(Cl, Cl')$ can be computed in $O(max(|Cl|,|Cl'|)^3)$ time using the algorithm proposed in [18].

Based on Definition 9, we can define our final quality criterion which helps to assess the quality of partitioning fuzzy clusterings to reference clusterings. We compare the costs for transforming the fuzzy clustering Cl^{fuzzy} into a reference clustering Cl^{ref}, to the costs piling up when transforming Cl^{fuzzy} first into $\emptyset$, i.e. a clustering consisting of no clusters, and then transforming $\emptyset$ into Cl^{ref}.

Definition 10 (*fuzzy partitioning clustering quality* Q_{FPC})
Let Cl^{fuzzy} be a fuzzy partitioning clustering and Cl^{ref} be the corresponding reference clustering. Then, the fuzzy partitioning clustering quality $Q_{FPC}(Cl^{fuzzy}, Cl^{ref})$ is equal to 1 if $Cl^{ref} = Cl^{fuzzy}$, else it is defined as

$$1 - \frac{d^{partitioning}_{clustering}(Cl^{fuzzy}, Cl^{ref})}{d^{partitioning}_{clustering}(Cl^{fuzzy}, \emptyset) + d^{partitioning}_{clustering}(\emptyset, Cl^{ref})}$$

Note that our quality measure Q_{FPC} is between 0 and 1. If Cl^{fuzzy} and Cl^{ref} are identical, $Q_{FPC}(Cl^{fuzzy}, Cl^{ref}) = 1$ holds. On the other hand, if the clusterings are not identical and the clusters from Cl^{fuzzy} and Cl^{ref} have no objects in common, i.e. $\forall C_j^{ref} \in Cl^{ref}, \forall C_i^{fuzzy} \in Cl^{fuzzy} : C_j^{ref} \cap C_i^{fuzzy} = \emptyset$ holds, then $Q_{FPC}(Cl^{fuzzy}, Cl^{ref})$ is equal to 0.

5.3 Similarity Measure for Hierarchical Clusterings

In this section, we first present a similarity measure between hierarchical clusterings. Based on these distance functions, we then introduce a quality criterion suitable for measuring the quality of fuzzy hierarchical clusterings. As already outlined, a hierarchical clustering can be represented by a tree (cf. Definition 7). In order to define a meaningful quality measure for fuzzy hierarchical clusterings, we need a suitable distance measure for describing the similarity between two trees t and t'. Note that each node of the trees reflects a flat cluster, and the complete trees represent the entire hierarchical clusterings.

A common and successfully applied approach to measure the similarity between two trees is the degree-2 edit distance [24]. It minimizes the number of edit operations necessary to transform one tree into the other using three basic operations, namely the insertion and deletion of a tree node and the change of a node label.

Definition 11 (*cost of an edit sequence*)
An edit operation e is the insertion, deletion or relabeling of a node in a tree t. Each edit operation e is assigned a non-negative cost $c(e)$. The cost $c(S)$ of a sequence of edit operations $S = \langle e_1, \ldots, e_m \rangle$ is defined as the sum of the cost of each edit operation, i.e. $c(S) = c(e_1) + \ldots + c(e_m)$.

Definition 12 (*degree-2 edit distance*)
The degree-2 edit distance is based on degree-2 edit sequences which consist only of insertions and deletions of nodes n with $degree(n) \le 2$, and of relabelings. Then, the degree-2 edit distance between two trees t and t', $ED_2(t, t')$, is the minimum cost of all degree-2 edit sequences that transform t into t' or vice versa: $ED_2(t, t') = min\{c(S) \mid S$ *is a degree-2 edit sequence transforming t into t'*$\}$.

Our final distance measure between two hierachical clusterings is based on the degree-2 edit distance.

Definition 13 (*hierarchical clustering distance* $d^{hierarchical}_{clustering}$)
Let DB be a database. Let Cl and Cl' be two hierarchical clusterings represented by the trees t and t'. Then, the hierarchical clustering distance $d^{hierarchical}_{clustering}$ is defined by:

$$d^{hierarchical}_{clustering}(Cl, Cl') = ED_2(t, t')$$

It is important to note that the degree-2 edit distance is well defined. Two trees can always be transformed into each other using only degree-2 edit operations. This is true because it is possible to construct any tree using only degree-2 edit operations. As the same is true for the deletion of an entire tree, it is always possible to delete t completely and then build t' from scratch resulting in a distance value for this pair of trees. In [24] Zhang, Wang, and Shasha presented an algorithm which computes the degree-2 edit distance in $O(|t| \cdot |t'| \cdot D)$ time, where D denotes the maximum fanout of the trees, and $|t|$ and $|t'|$ denote the number of tree nodes.

We propose to set the cost $c(e)$ for each insert and delete operation e to 1. Furthermore, we propose to use the *normal-*

ized symmetric set difference d_{Δ}^{norm} as introduced in Definition 8 to weight the relabeling cost. Using the normalized version allows us to define a well-balanced trade-off between the relabeling cost and the other edit operations, i.e. the insert and delete operations.

Based on the described similarity measure between hierarchical clusterings, we can define a quality measure for evaluating fuzzy hierarchical clustering algorithms. We compare the costs for transforming a fuzzy hierarchical clustering Cl^{fuzzy} modelled by a tree t^{fuzzy} into a reference clustering Cl^{ref} modelled by a tree t^{ref}, to the costs piling up when transforming t^{fuzzy} first into an “empty” tree t^{nil}, which does not represent any hierarchical clustering, and then transforming t^{nil} into t^{ref}.

Definition 14 (*fuzzy hierarchical clustering quality* Q_{FHC})
Let t^{ref} be a tree representing a hierarchical reference clustering Cl^{ref}, and t^{nil} a tree consisting of no nodes at all, representing an empty clustering. Furthermore, let t^{fuzzy} be a tree representing a fuzzy hierarchical clustering Cl^{fuzzy}. Then, the fuzzy hierarchical clustering quality Q_{FHC} (Cl^{fuzzy}, Cl^{ref}) is equal to 1 if $Cl^{ref} = Cl^{fuzzy}$, else it is defined as:

$$1 - \frac{d_{clustering}^{hierarchical}(C^{fuzzy}, C^{ref})}{d_{clustering}^{hierarchical}(C^{fuzzy}, \varnothing) + d_{clustering}^{hierarchical}(\varnothing, C^{ref})}$$

As the *hierarchical clustering distance* $d_{clustering}^{hierarchical}$ is a metric [24], the fuzzy hierarchical clustering quality Q_{FHC} is between 0 and 1.

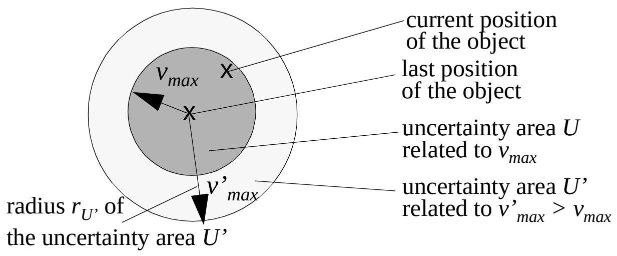
6. Evaluation

In this section, we present a detailed experimental evaluation which demonstrates the characteristics and benefits of our new approach.

6.1. Settings

As test data sets for the effectiveness evaluation we used 1.000 2-dimensional points arbitrarily distributed in a data space [0..1] x [0..1]. For the efficiency evaluation, we used 10.000 of these points. The points moved at each timetick with an arbitrary velocity $v \in [0..v_{max}]$ in an arbitrary direction. Figure 8 shows that the higher the value of v_{max} is, the more uncertain is the position of the object after one timetick. Each position within the circular uncertainty area of the object is equally likely. As parameter for the experiments we used the radius r_U of the uncertainty area U.

In order to evaluate the quality of the various algorithms, we arbitrarily distributed the points in the data space. The reference clustering, was created by letting the points move as described above. A sample clustering was created by choosing one point arbitrarily from the uncertainty area of the object. From the resulting s sample clusterings we computed the medoid clustering by using the distance function of Definition 9 and 13 between clusterings. For the fuzzy clustering based on the distance expectation values, we used also the



current position of the object
last position of the object
uncertainty area U related to v_{max}
uncertainty area U' related to $v'_{max} > v_{max}$
v_{max}
v'_{max}
radius $r_{U'}$ of the uncertainty area U'

Figure 8. Test data set.

sample positions in the uncertainty areas. The distance between two moving objects is then equal to the average distance between their sample points.

The qualities of the fuzzy clusterings w.r.t. the exact clusterings were measured by the quality criterions introduced in Section 5. For DBSCAN [7] and for *k*-means [16], we used the one introduced in Definition 10, and for OPTICS [1] and for Single-Link [12], we used the one introduced in Definition 14.

If not otherwise stated, we used a sample rate s=10 throughout our experiments. For all clustering algorithms, we used a parameter setting which created a clustering according to intuition. For DBSCAN, for instance, we used a parameter setting so that we approximately detected 20 clusters containing 80% of all objects.

All clustering algorithms, the used quality measures, and the heuristic to accelerate the computation of the reference clustering were implemented in Java 1.4. The experiments were run on a Windows laptop with a 730 MHz processor and 512 MB main memory.

6.2. Experiments

6.2.1. Sample-Clusterings. In a first set of experiments, we investigated the maximum and minimum quality resulting from sampling w.r.t. the reference clustering. We compared these quality values to the quality achieved by the medoid clustering. Figure 9 shows clearly, that for all clustering algorithms the quality decreases with an increasing uncertainty area. Furthermore, we can see that there exist quite noticeable quality differences between the best and the worst sample clustering. This is especially true for interesting uncertainty values U which are neither too small nor too large. If the uncertainty area is too large, the quality is around zero for all sample clusterings, which means that the sample clusterings and the reference clustering are quite different from each other. On the other hand, if the uncertainty area is very small, all sample clusterings are almost identical to the reference clustering resulting in high quality values. Furthermore, the figure shows that the quality of the medoid clustering is somewhere in between the best and the worst sample clustering, and often quite close to the best sample clustering. Obviously, using the medoid clustering instead of an arbitrary sample

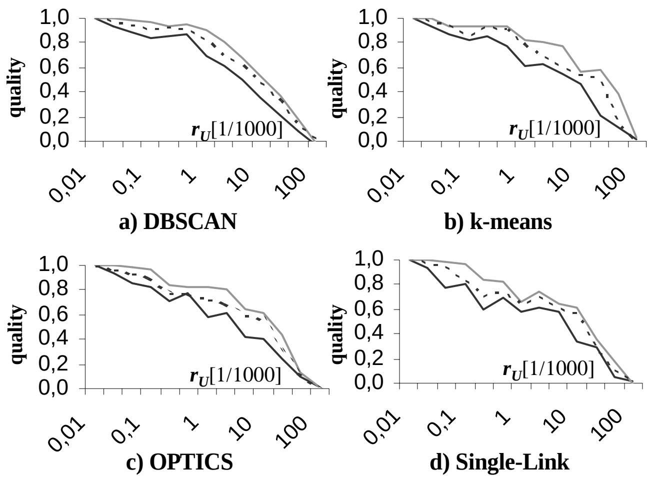


Figure 9. Sample Clusterings.
——— worst sample clustering ——— best sample clustering
- - - - medoid clustering

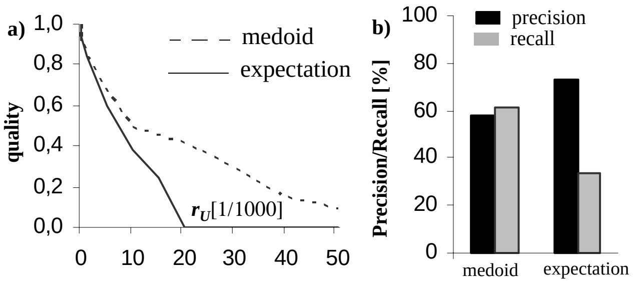


Figure 11. Distance Expectation Value (DBSCAN).
a) quality **b)** core-object classification (r_U=0.01)

clustering reduces the probability that the determined clustering is very dissimilar to the reference clustering. Furthermore, let us note that Figure 9 also indirectly demonstrates the suitability of the distance functions and quality measures presented in Section 5.

As the partitioning density based clustering paradigm seems to be the most important and adequate clustering approach for moving objects [21], we concentrate in the following on the flat density-based clustering algorithm DBSCAN.

Figure 10 shows that the quality of the medoid clustering increases with increasing sampling rate *s*. This holds especially for small values of *s*. For values of *s* higher than 10 the increase of the quality is only marginal indicating that rather high values of *s* are not necessary to produce good clustering results. Furthermore, we can see that the quality of the worst sample clustering decreases with increasing sample rate *s*. Likewise, the quality of the best sample clustering increases. Obviously, the higher the sample rate is, the more likely it is that we generate a clustering which has a very small or a very high distance to the reference clustering. For the other clustering algorithms we made basically the same observations.

6.2.2. Distance Expectation Values. In Figure 11, the results of the clustering approaches based on the distance expectation value and the medoid clustering are compared to each other. Figure 11a shows clearly, that for high uncertainty values the quality achieved by the medoid clustering approach is much higher than the quality achieved by a DBSCAN run based on the distance expectation values. It is noteworthy, that in this case often the worst sample clustering achieved higher quality values than the distance expectation approach. The explanation for the bad performance of the distance expectation approach can be found in Figure 11b. Although the precision of the detected core objects is very high, the recall is very low, i.e. the approach fails to detect many core objects. Thus we have very often the situation depicted in Figure 7b. Let us note that for small uncertainty values the difference between the two approaches is less significant.

6.2.3. Other Comparison Partners. In [14] a density-based approach for clustering multi-represented objects was proposed which is based on DBSCAN. The authors propose for sparse data sets, the union-method which assumes that an object is a core object if *MinPts* objects are found within the union of all ε-neighborhoods of all representations. Furthermore, the intersection method was introduced where an object is a core-object, if it is a core object in each representation. We used these two approaches as comparison partners where a representative corresponds to a sample value. Figure 12 shows again that our medoid clustering approach outperforms the union and intersection method by far.

6.2.4. Efficiency. In a last set of experiments, we investigated the efficiency of our approaches. In all tests we did not use any index structure and all data was kept in main memory. Figure 13 shows clearly that if only one slave is available the single sampling approach is by far the most efficient approach. Obviously, the distance expectation approach is much slower due to the much more expensive distance computation between two objects. Note that the runtimes of the union/intersection approach are similar to the ones of the expectation approach. When using only one slave, the medoid

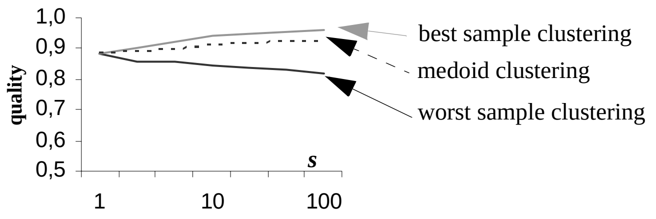


Figure 10. Medoid Clustering (DBSCAN) (r_U = 0.001).

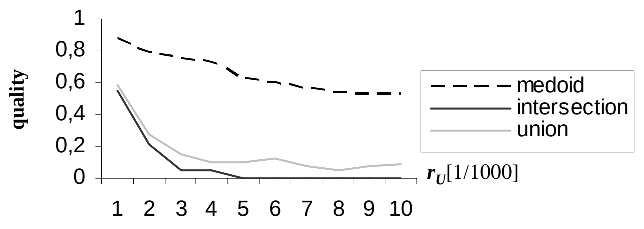


Figure 12. Union / Intersection Approach (DBSCAN).

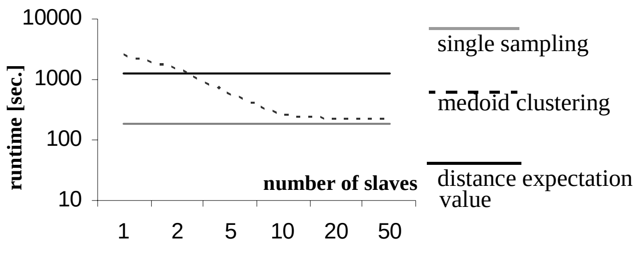


Figure 13. Efficiency Evaluation (DBSCAN). sampling rate s = 10, uncertainty radius $r_U = 0.001$

approach is even slower than the distance expectation approach because we have to determine the medoid clustering from the sample clusterings. The more slave computers are available, the more benefits our medoid approach. If *s* (=sample rate) slave computers are available, we can carry out a sample clustering on each slave. Therefore, we have an almost linear speed-up until *s* slaves are used. For a higher number of slaves, we can only parallelize the computation of the medoid clusterings from the sample clusterings, but not the generation of the sample clusterings. Therefore, we suggest to use *s* slaves for the computation of the medoid clustering.

In all our tests, we noticed that the heuristic introduced in Section 4.3.1 saves on average 12% of all distance computations between two clusterings. The ratio between the runtimes needed for the determination of the sample clusterings and the runtimes needed for the determination of the medoid clustering from these sample clusterings depends on the ratio of objects to be clustered and on the detected number of clusters. If we detect only a small number of clusters, the computation of the distances between two clusterings can be done efficiently when using the distance measures introduced in Section 5. On the other hand, distance computations between clusterings containing many clusters are rather expensive.

To sum up, the medoid approach is the method of choice for clustering fuzzy moving objects, especially if several slaves are available.

7. Conclusions

In this paper, we tackeled the complex problem of clustering moving object with uncertain positions. In order to do this effectively, we introduced the concept of *medoid clusterings*. We showed that these medoid clusterings are more suitable to cluster fuzzy moving objects than other approaches which are purely based on sampling or which are based on the distance expectation values between the fuzzy objects.

In our future work, we will show that density probability functions describing the positions of fuzzy moving objects can also beneficially be used in the context of location-based services.

References

[1] Ankerst M., Breunig M., Kriegel H.-P., Sander J.: *OPTICS: Ordering Points To Identify the Clustering Structure*. SIGMOD'99, pp. 49-60.

[2] Bracewell, R. *The Impulse Symbol.* Ch. 5 in The Fourier Transform and Its Applications, 3rd ed. : McGraw-Hill, 1999.

[3] Behr T., Güting R.H.: *Fuzzy Spatial Objects: An Algebra Implementation in SECONDO.* To appear at ICDE 2005.

[4] Brecheisen S., Kriegel H.-P., Kröger P., Pfeifle M.: *Visually Mining Through Cluster Hierarchies*. Proc. SIAM Int. Conf. on Data Mining (SDM'04), 2004, pp. 400-412.

[5] Banerjee A., Langford J.: *An Objective Evaluation Criterion for Clustering*. Proc. 10th ACM SIGKDD, 2004, pp. 515-520.

[6] Chudova D., Gaffney S., Mjolsness E., Smyth P.: *Translation-invariant mixture models for curve clustering*. Proc. 9th ACM SIGKDD, 2003.

[7] Ester M., Kriegel H.-P., Sander J., Xu X.: *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* KDD'96, pp. 226-231.

[8] Eiter, T., Mannila, H.: *Distance Measures for Point Sets and Their Computation*. Acta Informatica 34 (1997), pp. 103–133.

[9] Fowlkes E., Mallows C.: *A method for comparing two hierarchical clusterings*. Journal of American Statistical Association, 78, 1983, pp.553-569.

[10] Har-Peled S.: *Clustering Motion*. Discrete and Computational Geometry, 31(4):545-565, 2003.

[11] Höppner F., Klawonn F., Kruse R., Runkler T: *Fuzzy Cluster Analysis*. Wiley (1999).

[12] Jain A. K., Murty M. N., Flynn P. J.: *Data Clustering: A Review*. ACM Computing Surveys, Vol. 31, No. 3, Sep. 1999, pp. 265-323.

[13] Kriegel H.-P., Pfeifle M.: *Measuring the Quality of Approximated Clusterings*. BTW 2005.

[14] Kriegel H.-P., Kailing K., Pryakin A., Schubert M.: *Clustering Multi-Represented Objects with Noise*. Proc. 8th PAKDD 20004, pp. 394-403.

[15] Li Y., Han J., Yang J.: *Clustering Moving Objects*. Proc. 10th ACM SIGKDD, 2004.

[16] McQueen J.: *Some Methods for Classification and Analysis of Multivariate Observation.* Proc. 5th Berkeley Symp. on Math. Statist. and Prob., Vol. 1, 1965.

[17] Meila M.: *Comparing Clusterings by the Variation of Information*. Proc. 16th Annual Conference on Computational Learning Theory (COLT'03), pp. 173-187.

[18] Munkres, J.: *Algorithms for the assignment and transportation problems*. Journal of the SIAM 6 (1957) 32–38.

[19] Ramon J., Bruynooghe M.: *A polynomial time computable metric between point sets*. Acta Informatica 37 (2001), pp. 765–780.

[20] Sander J., Qin X., Lu Z., Niu N., Kovarsky A.: *Automatic Extraction of Clusters from Hierarchical Clustering Representations*. Proc. 7th PAKDD, 2003, pp 75-87.

[21] Yiu M. L., N. Mamoulis N.: *Clustering Objects on a Spatial Network*. Proc. 23th ACM SIGMOD, 2004, pp. 443-454.

[22] Zhang Q., Lin X.: *Clustering Moving Objects for Spatio-Temporal Selectivity Estimation.* Proc 15th Australasian Database Conference (ADC), 2004 pp. 123-130.

[23] Zhang T., Ramakrishnan R., Livny M.: *BIRCH: An efficient data clustering method for very large databases.* Proc. 15th ACM SIGMOD, 1996, pp. 104-114.

[24] Zhang K., Wang J., Shasha D.: *On the editing distance between undirected acyclic graphs*. International Journal of Foundations of Computer Science, 7(1):43–57, 1996.