DIDES: a fast and effective sampling for clustering algorithm

F. Ros, S. Guillaume

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

DIDES: a fast and effective sampling for clustering algorithm

Frédéric Ros*and Serge Guillaume†
(frederic.ros@univ-orleans.fr, serge.guillaume@irstea.fr)

February 26, 2016

Abstract

As clustering algorithms become more and more sophisticated to cope with current needs, large data sets of increasing complexity, sampling is likely to provide an interesting alternative. The proposal is a distance-based algorithm: the idea is to iteratively include in the sample the furthest item from all the already selected ones. Density is managed within a post-processing step, either low or high density areas are considered. The algorithm has some nice properties: insensitive to initialization, data size and noise, it is accurate according to the Rand index and avoids many distance calculations thanks to internal optimization. Moreover it is driven by only one, meaningful, parameter, called granularity, which impacts the sample size. Compared with concurrent approaches, it proved to be as powerful as the best known methods, with the lowest CPU cost.

Keywords: Space coverage, distance, density, Rand index

1 Introduction

Cluster analysis is the task of grouping objects. The goal is to design a partition of the input space in which the objects in a given cluster are more similar than those in the other groups. This unsupervised task is very important to organize, summarize and finally understand the data. It is widely used in various fields such as data mining, statistical analysis, bioinformatics or pattern recognition.

Many algorithms[3] have been proposed in recent decades to achieve this task but some important issues remain unsolved: for instance, the optimal number of clusters[4] or the most suitable distance function. To cope with data complexity, algorithms are becoming increasingly sophisticated. The goal is to make them able to manage data with clusters of various shapes and densities. However, this leads to an increased computational cost which limits their practical use.

*Laboratoire PRISME, Université d'Orléans, France
†Irstea, UMR ITAP, 34196 Montpellier, France

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

While it is still possible to optimize and speed up existing techniques, sampling appears as an interesting alternative to manage large data sets. It can be seen as a pre-processing step for clustering algorithms. The main sampling methods are based upon the distance and density concepts. A survey shows that the corresponding methods have reached a high level of maturity [39, 16, 50, 58, 20].

The goal of this paper is to introduce a new algorithm, DIDES, which stands for *DIstance and DEnsity based Sampling*, that combines the best of the available techniques in such a way that tractability is actually improved with a user friendly parameter setting. The sample selection is not done at random: the idea is to include in the sample, at each iteration, the furthest item from all the already selected ones. The sample size is not a priori defined, it depends on the data structure: the algorithm dynamically computes a threshold, on the maximum distance between a selected representative and its attached patterns, that is used as a stopping criterion. Density is specifically managed using mechanisms dedicated to handle either low or high density areas.

The main objective of the sampling is to select a part that behaves like the whole. To assess the sample representativeness, the partitions built from the sample sets are compared to the ones designed from the whole sets using the same clustering algorithm. The Rand index is used for partition comparison. Two representative clustering algorithms are tested, the popular *k-means* and a hierarchical one. The resulting sample size as well as the computational cost are carefully studied as they have a strong impact on the practical use of DIDES.

The state of the art of distance and density based methods is analyzed in Section 2. The *DIDES* algorithm is detailed in Section 3. Section 4 introduces the validation protocol used for studying the properties of the proposal (Section 5) and for comparing the results with concurrent approaches (Section 6). Finally Section 7 summarizes the main conclusions and opens up some perspectives.

2 Related work

The task of clustering is of prime concern in various fields such as data mining or pattern recognition. It aims to group items in such a way that those which belong to the same group, or cluster, are similar or close to one another, according to a given metric, and different from items grouped in other clusters. Several clustering algorithms have been proposed in the literature [29, 51]. They can be roughly classified in two categories: partitioning or hierarchical algorithms.

However, the data to be processed are now more numerous and more complex [57, 15] and the algorithms tend to mix several techniques [3] like Clarans, Birch, O-Cluster, EM or Dbscan extensions, among others. This often leads to an increased computational cost, limiting their use for large databases. Sampling [31, 63] is an interesting way to ensure tractability. The objective is to select a subset of the original data that behaves like the whole.

The first method to appear was random sampling, subject of many studies. The results are interesting from a theoretical point of view [11, 27, 12], but

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

they tend to overestimate the sample size in non worst-case situations. A lot of work has been done to improve this basic algorithm. In our case, sampling is a pre-processing step for clustering and clustering is assessed according to compactness (or cluster homogeneity) and group separability. This calls for two basic notions: density and distance. Clusters can be defined as dense input areas separated by low density transition zones.

Sampling algorithms are based upon these two notions, one driving the process while the other is more or less induced. Strategies have also been developed to improve and speed up the sampling process. Several approaches benefit from a kd-tree implementation [30].

2.1 Density biased sampling

The main idea of density methods [9, 10, 45] is to add a bias according to space density, giving a higher probability for patterns located in less dense regions to be selected so as to ensure the representation of small clusters. These methods can be grouped in two main families: space partition and kernel estimation.

The simplest space partition based method is a grid with non overlapping cells. Palmer and Faloutsos [47] introduced the problem of non uniform sampling for clusters corresponding to skew size distributions. The space is divided into equally sized bins. The bins with a small number of patterns are considered with a higher probability. This method is easy to implement and has a reasonable time complexity which allows its use with large data sets. However, the results are obviously sensitive to cell definition and bias level: noise can be selected when the bias is too high. Some work has been done to adapt the grid to the data [28]. Finding new boundaries is not a trivial task and leads to a substantial increase in the computational cost. Trees can be seen as an extension of grids, where the cells have not the same size but are specific to a node. Nanopoulos et al. presented one of the pioneering studies for clustering and sampling using R-tree [46]. Points belonging to the same node are considered to have an identical probability of being selected. An approximate local density is thus given by the ratio of the node cardinality to the corresponding volume (hyperrectangle). The sampling is done according to the biased local densities. The results are highly sensitive to the splitting strategy as well as to some key parameters such as the stopping criterion. Algorithms like the Minimum Spanning Tree [64] have been proposed to improve both cluster relevance and tractability. Kd-trees proved efficient for outlier detection [26]. The drawbacks induced by the binary split are well known: the partitions lack smoothness as similar data may be found on both sides of a given boundary.

The local density in each data point of the population can also be estimated using non parametric kernel or neighboring approaches.

The multi-dimensional kernel density estimator is defined as:

$$\hat{f}(x) = \frac{1}{nh_1 \dots h_d} \sum_{i=1}^{n} \left[\prod_{j=1}^{d} K\left(\frac{x_j - x_j^i}{h_j} \right) \right]$$

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

where d is the space dimension, n the set size, $K(\cdot)$ the kernel and h the bandwidth.

Under weak conditions ($h_1, \ldots, h_d$ decrease when n increases) the estimate converges in probability to the true density. A lot of work has been dedicated to kernel shape and its impact on results [41, 6, 42] but the most influential parameter is the bandwidth. It controls the smoothness of a density estimate. If improperly defined, it can lead to rough estimations.

The general expression for neighboring density estimation is the following:

$$\hat{f}(x) = \frac{k}{nV}$$

where V is the volume surrounding x and k the number of items located in V. The estimate can be reached using two methods: either setting V and computing k or finding the k nearest neighbors of x and then deducing the corresponding volume.

Once the local densities have been estimated, they are used to bias the sampling process. In the work by Kollios et al. [35], the kernel is that of Epanechnikov [19] and the bias is as follows:

$$\frac{s}{\sum \hat{f}(x)} \hat{f}(x)^b$$

where s is the desired sample size and b the bias parameter.

If $b = 0$, the process reduces to a random sampling $\left(\frac{s}{n}\right)$. Otherwise, if $b > 0$ (respectively $b < 0$) high density regions are sampled at a higher (lower) rate.

Although extensively studied, these approaches are rather difficult to parameterize. With an inappropriate setting, these methods are either likely to sample noise or to miss some regions of interest. Moreover, clusters' shapes are not taken into account, nothing ensures that they are preserved in the sample set. This is an intrinsic limitation of these methods. Moreover, they usually require significant storage capacity and have a high computational cost.

2.2 Distance-based sampling

Density is an important cluster feature. Distance is the other notion involved in cluster definition, as it is used to measure similarity and proximity between patterns. For this reason, it is widely used in clustering and sampling algorithms.

The most popular algorithm representative of this family is the *k-means* [23], and its robust version called *k-medoids* [32]. This simple and powerful algorithm can be used for sampling large data sets. It remains one of the most influential and studied clustering algorithms [39, 34, 69]. Some work has been done about its computational efficiency [13], but most studies deal with the quality of the initial partition. They address the well-known *k-means* shortcoming: its sensitivity to initialization [68, 5, 8]. Different approaches based on sampling or condensing techniques, including evolutionary algorithms [24, 44], have been investigated.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

The original version is limited as it only produces spherical clusters. It has been enriched to deal with more complex data and to yield overlapping clusters. The fuzzy extension [7] is widely used, while the possibilistic version [36] does not constrain the sum of the membership degrees to be 1.

k-means has been successfully used as a pre-processing sampling step for sophisticated and expensive techniques such as hierarchical approaches or Support Vector Machine algorithms (SVM) [59, 65]. It is run with $k = s$, the number of representatives, such as: $c \ll s \ll n$, c being the unknown number of clusters.

While the *k-means* is an iterative algorithm, whose convergence is guaranteed, some single data-scan distance-based algorithms have also been proposed, such as *leader* clustering [38]. Each pattern, x is assigned to a leader, l, if $d(x, l) \leq t$, t being a predefined threshold. If there is no leader in x neighborhood, x becomes a new leader. The results are sensitive to the threshold, and to the initial pattern order. This basic version has been improved [56, 61]. In the latter various thresholds are used to yield clusters of different sizes. First the *k-means* is run on a small random sample of the original data, with $k \gg c$ groups. The centers (means) are $m_1, \ldots, m_k$. The threshold distance for a new leader, x, is computed from its two nearest means, m_i and m_j, as follows:

$$t(x) = \lambda\ (a - b), \quad 0 < \lambda \leq 1, \text{ where } a = \frac{\|m_i - m_j\|}{2} \text{ and}$$
$$b = (x - m_i)\frac{m_i - m_j}{\|m_i - m_j\|}.$$

The *leader* method is a pure distance-based algorithm that does not account for density. The *mountain method* [66, 14] can be considered as an improved leader approach in which local density is also taken into account. It finds first the most representative pattern in order to be less dependent on the presentation order. Like the *k-means*, *leader* algorithms can be used for sampling with a threshold $t' \ll t$ to give a number of representatives $s \gg c$.

The pioneering versions of distance-based methods, such as the leader family approaches, are simple and fast but clearly limited. When improved, by taking density into account [52], they become more relevant but their overall performance depends on the way both concepts are associated and on the increased computational cost.

The *mountain method* proposed by Yager and its modified versions [67] are good representatives of hybrid methodologies as well as the recent work proposed by Feldman et al. [20]. Density is managed by removing from the original set items already represented in the sample.

2.3 Stratification strategies

The stratification concept has been proposed to speed up algorithms with quadratic or exponential time complexity. The idea is to divide the set into subsets, called strata, that are processed, or sampled in our case, independently. The final sampling is built from the union of the samples of each data set.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

The main drawback of stratification methods stems from the splitting: it is done randomly, with no prior knowledge about the data distribution. This does not ensure that the most informative patterns are selected in each subset. This highlights the importance of the final aggregation step. When each subset has been sampled, the sampling can be reiterated on the union of the selected patterns to keep only the most representative ones.

In the earliest versions the strata had the same size, collectively exhaustive and mutually exclusive. As an example, Bagged clustering [17, 37] consists in running a cluster method on each subset with the same number of clusters.

Extensions have been proposed to work with non disjoint partitions, using replication techniques [40]. These methods take density into consideration. Dense areas are obviously represented in the strata while regions corresponding to noisy data are likely to be diluted in the whole set of strata. This way, they have less opportunity to be represented in the final set.

Stratification can be combined with boosting strategies [22]. The former speeds up the process while the latter improves the sampling or clustering relevance.

Reservoir algorithms [62] are incremental and can be seen as a special case of stratification approaches. They have been proposed to deal with dynamic data sets, like the ones to be found in web processing applications.

Many variants exist. As examples, [1] is adapted to handle heterogeneous data distribution and [18] considers sampling with and without replacement.

Some studies have been done to adapt the size of the reservoir [2] and to propose appropriate aggregation strategies. Density is explicitly managed in [33] using the weighted *k-means*.

This short survey shows that sampling for clustering techniques have been well investigated. Both concepts, density and distance, and the methods have reached a good level of maturity. Some work deal with algorithm computational efficiency [55], but only a few papers study the sample size [48, 15]. As far as we know, the question of parameter tuning has not really been addressed. The new challenge is to take the best of the available techniques and combine them in order to propose a self-adaptive, data independent and tractable algorithm able to process various kinds of large data sets with a standard setting.

3 The DIDES algorithm

This section aims to present the proposed sampling algorithm. Its objective is to select a set of s items, S, from an initial one T, with $n > s$ items. The selected items are called representatives. As the goal is to use it as a preprocessing step of clustering, the algorithm should achieve two main tasks, namely ensure space coverage and initial set representativeness, at a low computational cost.

It is an iterative algorithm that add a new representative at each step in order to ensure space coverage. This very important step is distance-based. The new representative is the furthest from the existing ones. The process is repeated

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

until the spatial distribution of the representatives is homogeneous enough to fit cluster shapes. The adaptive stopping criterion is thus also distance-based. Density is managed through a post-processing step. The goal is to avoid outlier representatives and to enrich high density area representation. The volume is estimated by the maximum distance between an attached pattern and the representative. The algorithm is driven by a parameter called granularity, and labeled g_r. Data independent, it is combined with the data size, n, to define the minimum size in the whole data to be represented in the sample. Its value carries important consequences: the smaller the granularity, the better the representation.

In the following, the main steps of the algorithm are detailed. The first one is the selection of a new representative. Then, the two followings subsections, 3.2 and 3.3, aim to introduce the stopping criterion. Once all the representatives are selected, a post-processing step, subsection 3.4, allows for density management. Both low and high densities are considered. The whole sampling process is illustrated in subsection 3.5. Finally, some optimization operations are proposed in subsection 3.6 to speed up the process.

3.1 Distance-based representative selection

The iterative representative selection algorithm is shown in Algorithm 1.

Algorithm 1 Sampling algorithm: representative selection

Input: $T = x_i$, $i = 1 \ldots, n$
Output: $S = \{y_j\}$, $T(y_j)$, $j = 1, \ldots, s$
Select an initial pattern $x_{init} \in T$
$S = \{y_1 = x_{init}\}$, $s = 1$
repeat
  for all $x_l \in T \setminus S$ **do**
    Find $d_{near}(x_l) = \min_{y_k \in S} d(x_l, y_k)$
    $T(y_k) = T(y_k) \cup \{x_l\}$ {Set of patterns represented by y_k}
  end for
  for all $y_k \in S$ **do**
    Find $d_{max}(y_k) = \max_{x_m \in T(y_k)} d(x_m, y_k)$
    Store $d_{max}(y_k)$, $x_{max}(y_k)$ {`where` $d_{max}(y_k) = d(x_{max}(y_k), y_k)$},
  end for
  {Select the furthest pattern from the set of representatives}
  $y_w = \arg\max_{y_k \in S} d_{max}(y_k)$, $x_w = x_{max}(y_w)$, $MaxDmax = d_{max}(y_w)$
  $S = S \cup \{x_w\}$, $s = s + 1$
until Stopping condition is met
return S

The first pattern can be randomly chosen or it can be computed as the furthest, according to the selected distance, from the minimum value in each

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

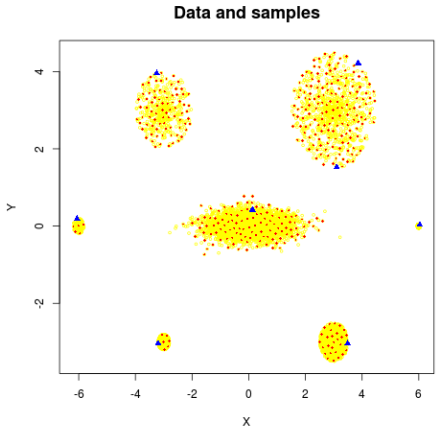

Figure 1: Sampling: first steps of the algorithm. Data (yellow) - Fisrt selected items (blue triangles) - Sample points (red)

input space dimension. After the initialization phase, the set S only counts this initial pattern, x_{init} (lines 3-4).

The main loop (lines 5-17) include two steps. First, each unselected pattern, $x \in T \setminus S$, is attached to the closest selected one in S (lines 6-9). At the first step, $T(y_1) = T \setminus \{x_{init}\}$. Then, for each set $T(y_k)$, the algorithm searches for the furthest attached pattern located at distance $d_{max}(y_k)$ (lines 10-13).

The next selected representative, x_w, is the furthest item attached to the already selected representative. It is chosen in the group, y_w, which corresponds to the maximum of the d_{max}, $MaxDmax$ (lines 15-16). So, boundary patterns are first chosen instead of inner ones. This way, the selected set spans the whole input space.

This concept is not new. It has been used thirty years ago to initialize the *k-means* algorithm and is known as the furthest-first traversal (fft) algorithm [54, 25]. Sensitive to outliers, it has inspired Arthur and Vassilvitskii [5] to propose *kmeans++*: new seeds are randomly chosen with a probability proportional to their distance to already chosen ones. Effectiveness is improved but at an increased computational cost.

Figure 1 illustrates the first steps of our algorithm (blue triangles) with well-structured data including clusters of various shapes and densities, and then shows the input space is correctly spanned (red symbols).

This distance-based algorithm does not take space density into account. Specific processes are detailed in the following.

The stopping criterion (line 17) is not defined yet. Rather than the common number of samples we introduce step by step, in the following sections, an adaptive threshold on the $MaxDmax$ distance.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

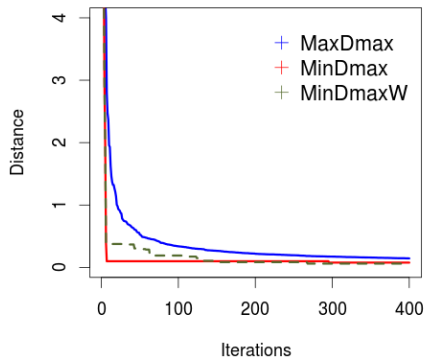

Figure 2: Distance evolution curves for the data in Fig. 1: $MaxDmax$ (blue), $MinDmax$ (red), $MinDmaxW$ (dashed green) is similar to $MinDmax$ when the mal cluster located at $(6, 0)$ has been removed

3.2 Estimating a lower bound of the threshold

The $MaxDmax$ criterion used for selecting a new representative is monotonically decreasing. Unfortunately, the evolution curve, shown in blue in Figure 2, does not indicate any break-point that would serve as a stopping criterion. A proportion in the decrease, with respect to the initial value, is not stable enough: the optimal threshold is really data dependent.

In the flat area, the number of samples is highly sensitive to the value of the $MaxDmax$ criterion. To ensure the representativeness, a cautious attitude would be to set a small threshold value, but this would lead to a number of selected samples higher than required to obtain the expected behavior. The challenge comes to estimate an acceptable trade-off.

Based on the common definition for a cluster, i.e., a dense area separated from another dense area by a sparsely populated zone, one needs to make sure that all the clusters are represented by the selected data. The narrowest cluster is correctly identified when the minimum of the d_{max} for all the representatives, hereafter called $MinDmax$ becomes equal to the smallest cluster dimension which is unknown. Let's make clear the item corresponding to the $MinDmax$ is never selected as a representative, only the one corresponding to the $MaxDmax$ is added to the sample set.

Once this value has been reached, the algorithm samples only the inner cluster structures. Defining the distance threshold according to the d_{min} evolution would ensure a perfect data coverage. However, in the case of singular clusters of small sizes, it would yield an over representation since the same distance needed for small clusters would also be used for the large ones. This is illustrated by the $MinDmax$ evolution shown in red in Figure 2.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

When the small cluster located at $(6, 0)$ in Figure 1 has been removed, this curve, $MinDmaxW$, is likely to provide valuable information, as shown with the dashed green line in Figure 2.

As no threshold can be identified, its estimation for a given data set is based on a $MaxDmax$ evolution model. But a new issue arises: how to characterize the evolution stage that allows for building an accurate model?

Let $MinCard$ be the minimum size, in the initial set T, for a cluster one wants to have representatives in S. y_k is labeled as a poor representative if $|T(y_k)| < MinCard$. This parameter can be set according to the initial data set size and the desired granularity, $MinCard = n\ g_r$.

Let P be the proportion of initial data represented by poor representatives:

$$P = \frac{1}{n} \sum_{k=1}^{s} \Big\{ |T(y_k)|,\ |T(y_k)| < MinCard \Big\}$$

P evolves from 0 to 1, when s is close to n. P is a function of s, it is nearly monotonically increasing. It is getting positive as soon as an outlier, part of an isolated small cluster or noise, has been selected as a representative. The idea is then to threshold P to model the $MaxDmax$ evolution curve. It is expected that a wide range of P value yield a similar threshold.

When P has reached its desired value, meaning that this proportion of T is represented in S by small sets, all the small clusters in T are represented in S according to the $MinCard$ parameter.

The $MaxDmax$ curve has reached the flat area, and its evolution can be modeled by a power function $f(x) = ax^p$. The representatives are now numerous enough to make a robust estimation of the two parameters thanks to a least squares minimization. The so-called asymptote value is used as a lower bound of the threshold. The asymptote is characterized by small variations in $f(x)$; it is reached for s_a such that:

$$\frac{f(s_a)}{f(s_a + 1)} \geq (1 - \epsilon) \tag{1}$$

where ϵ is a small positive value set to 0.001.

Taking the logarithm of Eq (1) yields:

$$\ln\left(\frac{x}{x+1}\right) \geq \frac{\ln(1-\epsilon)}{p} \text{ which gives: } s_a = -\frac{1}{1 - \mathrm{e}^{\frac{\ln(1-\epsilon)}{p}}}$$

Finally, $th_a = f(s_a) = a s_a^p$.

3.3 An adaptive stopping criterion

This lower bound of the distance threshold th_a has been computed with a generic configuration, defined by three hidden parameters: $P = 0.2$, $\epsilon = 0.001$ and $g_r = 0.01$. With this configuration, shapes in the original space are well represented in the selected sample, whatever the data distribution and the noise amount.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

To define the distance threshold that serves as a stopping criterion for the algorithm, the data as well as the granularity input parameter are now considered.

The selected sample is analyzed according to a twofold point of view: the $d_{max}(y_k)$ and the cardinality of the set attached to the sample, $|T(y_k)|$ for representative, y_k. Both distributions are used to define the threshold.

As the space is correctly covered, this is guaranteed by the P proportion for $g_r = 0.01$, outliers or noise representatives selected according to the distance criterion are now quite isolated, meaning both $d_{max}(y_k)$ and $|T(y_k)|$ are low. Let $S_t = \{y_k | \, |T(y_k)| < MinCard \,\&\, d_{max}(y_k) > th_a\}$ be a subset of representatives, th_a is the lower bound of the threshold and $MinCard$ is now computed with the user granularity. S_t includes small cluster representatives as well as outliers. The objective is to define a threshold that allows for keeping the former while removing the latter. The desired threshold, th, can be merely set as the average d_{max} over S_t:

$$th = \underset{y_k \in S_t}{mean} \; d_{max}(y_k) \tag{2}$$

Then, the algorithm iterates until this threshold is reached. If the user granularity is higher than 0.01, no more iteration may be needed.

3.4 Post-processing: density area management

As the selection is only done according to the distance from already selected items, density has to be taken into account with a twofold objective: remove noise points that might have been selected, especially during the first steps of the process and ensure dense area representativeness in the sample.

The first step is to remove noisy representatives. A threshold can be simply defined from the S_t set introduced in the previous section:

$$T_n = \underset{y_k \in S_t}{mean} \; |T(y_k)| \tag{3}$$

All the representatives with $|T(y_k)| < T_n$ are removed from S. The others are kept, meaning that even if they do not have $MinCard$ attached items they are not considered as noise.

Then density specific management is based on groups defined as connected areas. How to define a connected group of representatives? Representatives that belong to the same group are closer, one to each other, than representatives that belong to different groups. So, the group can be defined as the set of reachable points, from one another, within a given distance. The problem is now to set the threshold. It is based upon the distribution of distances between the representatives and their nearest neighbor, $d_{1nn}(y_k)$, and also on the $d_{max}(y_k)$ distribution. It is worth mentioning the $d_{1nn}(y_k)$ distances are updated at each iteration without any extra calculation.

The $d_{1nn}(y_k)$ distribution is filtered to only consider the potential connected pairs. As the regions of space containing data are homogeneously covered,

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

neighbors separated by a distance higher than twice the $MaxDmax$ cannot be part of the same group. So, these values, $d_{1nn}(y_k) > 2\ MaxDmax$, are not taken into account to compute the basics statistics of the distribution: the average, $\overline{d}$, and standard deviation, σ, of the d_{1nn} distribution, and let $MaxD1nn$ be the maximum of the distribution. The reachable threshold is defined as:

$$d_r = \min(\overline{d} + 2\sigma, MaxD1nn) \tag{4}$$

The density is considered at the group scale. A group, G, is characterized by the number of representatives part of the group, $G(s)$, and the number of patterns attached to the group: $G(t) = \sum_{r \in G} |T(r)|$.

Low density groups are removed. G is a small density group if $G(t) < MinCard$.

High density group representation is enhanced. Let $G(d) = \frac{G(t)}{G(s)}$ be the G group density, and let $\overline{d_g}$ the average density for all the groups. A group G is a high density group if $G(d) > \overline{d_g}$. The number of representatives in G is increased according to the density ratio: $s' = G(s)\left(1 - \frac{G(d)}{\overline{d_g}}\right)$. The s' new representatives in G are randomly chosen.

The sampling algorithm is now completely defined. It should be highlighted that even if it requires some parameters, only one is left to the user: granularity. The others are either temporary, like the P proportion, associated to $g_r = 0.01$, which states the space is covered enough to model the $MaxDmax$ evolution, or quite natural, like thresholds defined as the average of a distribution.

This effort makes the algorithm really easy to use, as the only parameter is meaningful to the user.

3.5 Illustration of the whole process

The main steps of the algorithm are illustrated in Figure 3 with synthetic 2D-data, 40000 items, structured in clusters of various shapes and densities to which an important level of noise has been added.

The upper part of the figure shows the selected items at two evolution steps. When the proportion P becomes positive (left), the space is not properly covered, some important clusters are not represented, due to the noise. The right picture corresponds to $P > 0.2$. Space coverage is now homogeneous, and the $MaxDmax$ evolution can be accurately estimated. The fitting curve equation is: $y = 7.1\ x^{0.59}$, with $R^2 = 0.996$.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

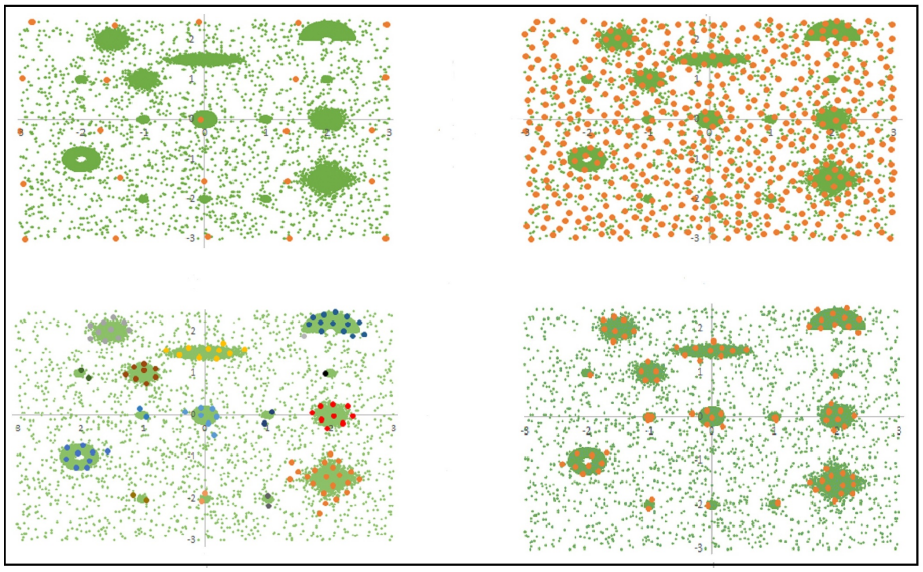

Figure 3: The main steps of the sampling: $P > 0$ (top left), $P > 0.2$ (top right), Output with $g_r = 0.001$ (bottom left) and $g_r = 0.01$ (bottom right)

The lower row of this figure shows the final results for two user granularity settings: $g_r = 0.001$ (left) and $g_r = 0.01$ (right). The granularity is used to compute the $MinCard$ internal parameter which impacts the distance and cardinality thresholds. In both cases, all the noisy representatives have been removed by the density management post-processing and all the clusters are represented. There are more representatives associated to the smaller granularity, even if both cardinalities are similar. In the bottom left graph the connected groups are displayed in distinct colors.

3.6 Time optimization

While many distance-based algorithms have a $O(n^2)$ complexity, this is not the case for the proposal, shown in Algorithm 2. Thanks to a reduced number of stored distances and algorithm optimization, the complexity is significantly decreased.

The time complexity of in Algorithm 1 is mainly due to the first two loops, lines 6-9 and lines 10-13. They are now combined in a single one, lines 9-24. This allows for only computing $(n - s)$ distances at each of the s iterations.

The complexity is then $O(ns)$, with $s \ll n$.

The number of distances to be computed is: $T = \sum_{l=s}^{n}(l-1) = \frac{n\ (n-1)}{2} - \frac{s\ (s-1)}{2}$.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

Algorithm 2 Sampling algorithm: final version

Input: $T = \{x_i\}$, $i = 1 \ldots, n$, g_r
Output: $S = \{y_j\}$, $\{T(y_j)\}$, $j = 1, \ldots, s$
Select an initial pattern $x_{init} \in T$
$d_{thresh} = 0$, $P_{thresh} = 0.2$
$S = \{y_1 = y_* = x_{init}\}$, $s = 1$
$d_{near}(x_i) = \infty$, $i = 1 \ldots, n$
repeat
  $MaxDmax$=0
  $F = \{T(y_j) | d(y_j, y_*) \geq 2\ d_{max}(y_j)\}$
  for all $x_l \in\ T \setminus \{S \cup F\}$ **do**
    if $d_{near}(x_l) > 0.5\ d(y(x_l), y_*)$ **then**
      Compute $d = d(x_l, y_*)$
      if $d < d_{near}(x_l)$ **then**
        $T(y_*) = T(y_*) \cup \{x_l\}$, $T(y(x_l)) = T(y(x_l)) \setminus \{x_l\}$
        $d_{near}(x_l) = d$, $y(x_l) = y_*$
      end if
      if $d > d_{max}(y_*)$ **then**
        $d_{max}(y_*) = d$, $x(y_*) = x_l$
        if $d > MaxDmax$ **then**
          $MaxDmax = d$, $y_n = x_l$
        end if
      end if
    end if
  end for
  $s = s + 1$,, $S = S \cup \{y_n\}$, $d_{max}(y_n) = 0$, $d_{1nn}(y_n) = y_*$, $y_* = y_n$
  if $(P > P_{thresh})$ and $(d_{thres} == 0)$ **then**
    Compute $d_{thres} = th$ {Eq. (2)}
  end if
until $MaxDmax > d_{thres}$
Post processing: Density management {Section 3.4}
return S, $\{T(y_j)\}$, $j = 1, \ldots, s$

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

Use of the triangle inequality. But this number can also be reduced thanks the inner structure of the algorithm. A given iteration only impacts the neighborhood of the new representative. So, when a new representative in S has been selected, y_*, the question is: should a given initial pattern, x_i, be attached to y_* instead of remaining in $T(y_j)$? The triangular inequality states: $d(y_j, y_*) \leq d(x_i, y_j) + d(x_i, y_*)$. And, $x_i \in T(y_*) \iff d(x_i, y_*) < d(x_i, y_j)$. So, if $d(y_j, y_*) \geq 2\ d(x_i, y_j)$, x_i remains in $T(y_j)$, no change needs to be made. Only two distances are needed to check the inequality, and discard any further calculations in the case of no change. In our algorithm, there is no need to check this inequality for all the initial patterns. Two optimization levels are implemented: representative and pattern levels. For each representative, y_k, $d_{max}(y_k)$ is stored. If $d(y_k, y_*) \geq 2\ d_{max}(y_k)$, meaning the furthest initial pattern from y_k remains attached to y_k, this also holds $\forall x_i \in T(y_k)$. Then, these representatives and their attached patterns are not concerned by the main loop of the algorithm (line 9-10). The number of patterns managed by this level increases with the the number of representatives, as the averaged induced volume decreases. When this is not the case, the same triangle inequality applied at the pattern level provides a useful threshold. All $x_i \in T(y_k)$ with $d_{near}(x_i) \leq 0.5\ d(y_j, y_*)$ remain attached to $T(y_k)$ (line 11).

To take advantage of the triangular inequality properties, the number of distances between representatives to be stored is $s(s-1)/2$.

Estimation of the number of computed distances. Even if the number of actually computed distances cannot be rigorously defined as it depends on the data, it can be roughly estimated under some weak assumptions. Let k be the number of neighbors to consider. The number of distances to be calculated is $(n-1)$ at the first step, then the number of representatives to take into account is $min(k, s)$ and the number of patterns for which the distance to the representatives has to be computed is only a proportion, δ, of the set of the attached ones as the others are managed by the triangular inequality properties. A value of $\delta = 0.5$ seems to be reasonable. The real number of computed distances can be estimated as follows.

$$C = (n-1) + \sum_{i=s}^{n-1} \sum_{l=1}^{min(k,s-i)} \delta * |T^i(y_l)| \tag{5}$$

where $|T^i(y_l)|$ is the number of patterns attached to representative l when i representatives are selected.

To approximate C, one can consider that on average the representatives have a similar weight $\forall y, |T^i(y_l)| \approx n/i$. When the two cases, $i \leq k$ and $i > k$, are specifically handled, the approximation becomes:

$$C = (n-1) + \delta \left(\sum_{i=2}^{k+1} (i-1)\ \frac{n}{i} + \sum_{i=s}^{n-k-2} k\ \frac{n}{i} \right)$$

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

As $\sum_{i=2}^{k+1}(i-1)\ (n/i) \leq \sum_{i=2}^{k+1}(i)\ (n/i)$ and $\sum_{i=s}^{n-k-2} k\ \frac{n}{i} \leq \sum_{i=s}^{n-k-2} k\ \frac{n}{s}$, an upper bound of C can be defined as follows:

$$C \leq (n-1) + \delta\left(n(k-1) + k\frac{n}{s}(n-k-2-s)\right)$$

As an illustration, using $n = 20000$, $s = 250$, $k = 10$ and $\delta = 0.6$ the decrease ratio is:

$$D = \frac{C}{T} \leq 5\%$$

This estimation is clearly confirmed by the experiments.

The spatial complexity for this time optimization can be considered as reasonable: the final version stores $n + 2 * s$ distances. n $d_{near}(x)$, s $d_{max}(y)$ and $d_{1nn}(y)$ as well as the corresponding elements, y for $d_{near}(x)$, and x for $d_{max}(y)$.

The algorithm behavior is now studied according to the sampling objectives and the sensitivity to the various parameters that have been defined.

4 Validation protocol

The algorithm is evaluated according to three criteria in order to study its properties and to compare its performance with concurrent approaches.

1. Quality of the representation
2. Sample set size
3. Algorithm CPU cost

The last two are easy to assess and implement. To assess the first objective, the resulting partitions of the same clustering algorithm applied to the whole set and the selected sample are compared using the Rand Index. The CPU cost is the sum of the time needed by the sampling algorithm and the time needed to run the clustering algorithm on the sample set.

4.1 Rand Index

This index [49] is a measure of similarity between two partitions of the same data set yielded, for instance, by two clustering algorithms. There is no assumption on the number of groups in the two partitions, P and Q.

The index is based on the comparison of pairs of patterns. The total number of pairs $\binom{n}{2}$ can be decomposed as follows:

- a: number of pairs that are in the same group in P and in Q
- b: number of pairs that are in different groups in P and in Q
- c: number of pairs that are in the same group in P and in different groups in Q

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

- d: number of pairs that are in different groups in P and in the same group in Q

The Rand index is defined by the proportion of agreements between P and Q:

$$R = \frac{a+b}{a+b+c+d}$$

The closer R is to 1, the more similar the two partitions are.

4.2 Two clustering algorithms

The algorithm behavior and performance were tested using two kinds of clustering algorithms: the *k-means* and *the hierarchical clustering* [43]. Both are useful and illustrate different clustering processes with different complexity, $O(nk)$ for *k-means* and $O(n^2)$ for hierarchical clustering.

In the partition computed from the sample S, all the patterns in T are assigned the cluster of their representative in S: $\forall x \in T(y_k), C(x) = C(y_k)$.

To take into account the random initialization, the *k-means* was run 100 times for $k = 2, \ldots, 20$ number of groups. To ensure a fair comparison, each run with the sample S was initialized with the same values randomly chosen in the corresponding run with the whole set T.

Various partitions can be generated from a dendrogram given by the hierarchical clustering algorithm. The studied partitions of T are selected thanks to the Elbow method[1] [60] applied to the percentage of explained variance according to the number of clusters. The chosen partitions correspond to the number of clusters located in the elbow of the plot. Beyond this quite imprecise area, when the number of clusters increases, the gain in the explained variance gets smaller.

The Rand Index is computed for the same number of clusters in S and T with *k-means*. With *the hierarchical* clustering, a partition, with a given number of clusters, is generated in S. The corresponding explained variance serves as the stopping criterion to process the partition in T. The results are averaged for all the selected configurations.

5 Study of the algorithm properties

To investigate the properties of the proposed algorithm, twelve synthetic 2-D data sets were considered. They were designed with various natural clusters of different shape, size, density and overlap. They are shown in Figure 4 and numbered from $S1$, top left, to $S12$, bottom right.

It is impossible to summarize all the results of the extensive tests. Only some illustrations, strongly supported by the experiments, are given to highlight the algorithm properties.

[1] https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

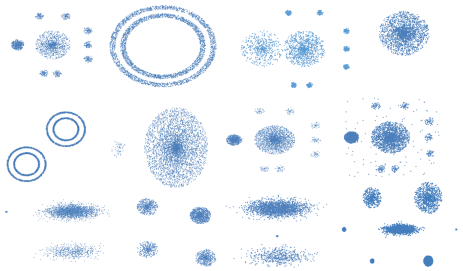

Figure 4: Twelve synthetic data sets ($S1, \ldots, S12$)

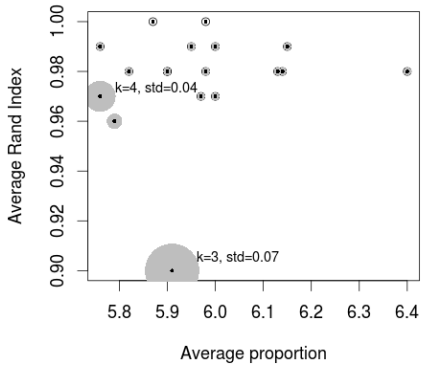

Figure 5: Sensitivity to initialization with data set $S1$: the Rand Index vs the sampling proportion, $|S|/|T|$, for $k = 2, \ldots, 20$ clusters

5.1 Little sensitivity to initialization

The sampling algorithm can be initialized with a virtual extreme pattern computed as the minimum (or the maximum) of all the patterns in each dimension. In this case the algorithm would be deterministic but this approach adds an extra iteration. The alternative is to randomly select the initial representative. The protocol has been applied with the nominal parameters.

The tests confirm that the initialization has no critical influence on the results. This is expected because the algorithm aims to ensure space coverage, and the selected representatives are all located at the boundaries of the input space.

Figure 5 illustrates the results with data set $S1$. The *k-means* algorithm was run with both the initial and the sample sets. Each point corresponds to a given number of clusters, $k \in \{2, \ldots, 20\}$. The abscissa is the ratio of the number of representatives to the whole size, $\frac{|S|}{|T|}$, the ordinate is the Rand Index value. The

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

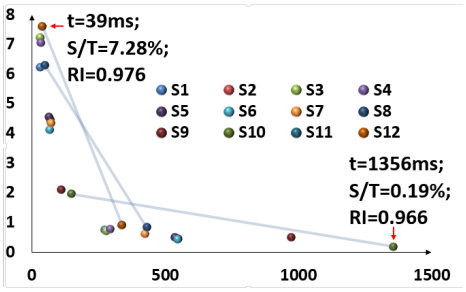

Figure 6: Data set size influence: the sampling proportion, $|S|/|T|$, vs the CPU time (ms), for each of the 12 synthetic data sets and a 10 times enriched version. The sampling proportion tends to zero for the enriched sets.

size of the sphere is proportional to the Rand index standard deviation over the 100 experiments.

The worst result corresponds to $k = 3$ clusters. In this case, the average proportion of representatives is 5.9%, the average Rand index is 0.90 and its standard deviation is 0.065.

5.2 Sample size independent on data size

For each data set, the test consists in comparing the algorithm performances on the initial set and an enriched similar one. The latter results from the aggregation of new patterns without modifying the data structure: each data point was replicated 10 times with an additional uniform random noise in the range $[-0.1, +0.1]\sigma_j$ where σ_j is the standard deviation of the feature j.

The results are illustrated on Figure 6 with $g_r = 0.005$. The same symbol and color are used for a given data set and its enriched version. The abscissa is the sampling CPU time (ms) and the ordinate the proportion of representatives, $\frac{|S|}{|T|}$. It tends to zero for the enriched sets, meaning that the number of representatives mainly depends on the data structure, and not on the data set size. Some examples are labeled with the (x, y) coordinates and the Rand Index. The ratio depends on the database, and especially on the different densities.

5.3 Robust to noise

For each data set, the test consists in adding an increasing amount of uniform random noise (from 1% to 9%). The new values are computed independently in each dimension, according to the whole range of the given feature: $noise_j = \min_j + U[0, 1] * (\max_j - \min_j)$.

Then, the sampling is applied with the standard parameters and the results of the clustering are summarized in Figure 7. The Rand index is plotted against the noise level for each data set.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

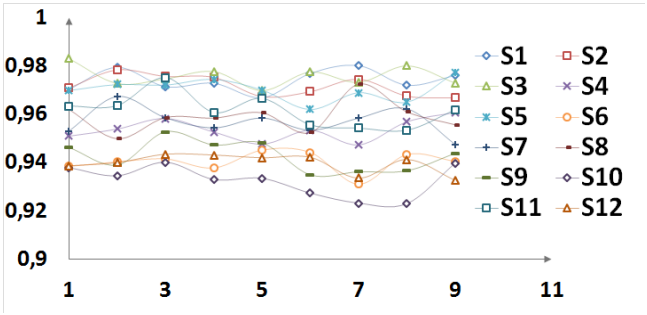

Figure 7: Noise influence: the Rand index vs the noise level (%) for the 12 synthetic data sets

The noise obviously impacts the selection of representatives, but the figure clearly shows that the influence on the Rand index is negligible. Thanks to its internal noise management, the sampling algorithm is still able to identify and represent the data structure even with a high level of noise.

5.4 User parameter sensitive

The *DIDES* algorithm includes some internal, hidden, parameters but only one user parameter, g_r. It is important to know how sensitive the algorithm is to the granularity. The expected behavior is: the smaller the granularity, the better the representation.

An illustration with data set $S7$, which counts 7200 items is provided in Table 1.

Table 1: User parameter sensitivity for $S7$ data

g_r	.01	.0091	.0084	.0077	.0069	.0062	.0054	.0046	.0039	.0032
$\|S\|$	127	134	139	153	167	182	216	244	263	285
RI	0.95	0.955	0.955	0.96	0.96	0.97	0.97	0.975	0.975	0.975

g_r varies from 0.003 to 0.01 and the number of representatives, $|S|$, is monotonically decreasing with granularity. There is no linear relationship between g_r and the sample size. This underlines a certain level of adaptability to the data structure for the algorithm.

Considering the Rand Index, there is a clear trend, even if the relationship is not always monotone: decreasing g_r tends to improve the index. This is in agreement with the increasing number of selected representatives.

These experiments clearly show there is no need to decrease g_r below a given threshold as the increased cost only slightly improves the accuracy.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

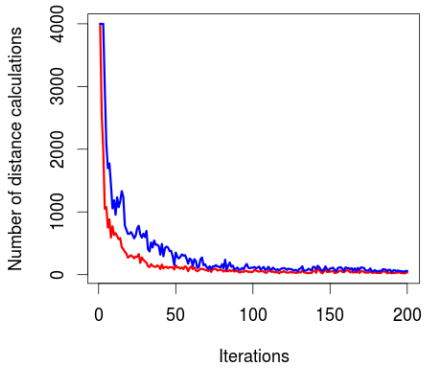

Figure 8: Optimization: distance calculations at each sampling iteration for data set $S4$ - First step only, Algo 2, line 9 (blue) - Two steps, lines 11 and 13, (red)

5.5 Computationally effective

Tractability is another desirable characteristic for this kind of algorithm. Several optimization steps are included in the algorithm and the number of distance calculations is estimated to be reduced to about 5% of the total. The objective is to evaluate how the different optimization steps contribute to the final result.

In this test, the number of calculations was recorded for all the data sets. The first optimization (lines 9 in Algorithm 2) accounts for 85 to 95 % of the reduction while the second one (line 11,13) allows to save 2 to 10 % of the calculus.

An illustration is given, for data set $S4$, in Figure 8.

200 representatives were selected from the 4000 patterns. The upper blue curve shows the real number of distance computations when the first optimization step is active. The lower red one corresponds to the same information when both the first and the second steps are used. Finally, in this case, 97.7% of the computations are avoided, 94% thanks to the first step.

As expected, the first iterations of the algorithm are responsible for the main part of the running time. The number of real computations swiftly drops to less than 100 after 30 iterations.

Extensive tests were carried out on twelve synthetic data sets, using two clustering algorithms, with a large range of number of clusters. This diversity, which is likely to include unstable situations, highlights some interesting properties of the DIDES algorithm. It ensures an accurate representation, according to the Rand index, it is insensitive to the initialization and to the data set size, and it is able to cope with noise. Only one parameter is left to the user and it is really fast. It is now compared to concurrent approaches with real world data

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

Table 2: The seven real world databases

R	Size	Dim	Name
1	434874	4	3D Road Network
2	45781	4	Eb.arff
3	5404	5	Phoneme
4	1025010	10	Poker Hand
5	58000	9	Shuttle
6	245057	4	Skin Segmentation
7	19020	10	Telescope

sets.

6 Dealing with real world data

In order to compare *DIDES* with alternative approaches, 7 databases from the UCI machine learning repository[2] were selected. They are of various sizes and space dimensions and with unknown data distribution. Their main characteristics are summarized in Table 2. All the variables are centered and normalized.

These data were first processed using *DIDES*, then the results were compared with concurrent approaches. The protocol remains the one described in Section 4, with two clustering algorithms, *k-means* and the *hierarchical* one. All data sets were randomly sampled, 3000 items, to be processed by the *hierarchical* algorithm.

6.1 DIDES analysis

The algorithm was run with different values for the granularity parameter from 0.001 to 0.1. The results are illustrated in Table 3, with the *k-means*, for two values in the middle range.

Table 3: *DIDES*: two neighboring *granularity* for the 7 data sets

	$g_r = 0.03$			$g_r = 0.05$		
	$100\|S\|/\|T\|$	Time	RI	$100\|S\|/\|T\|$	Time	RI
R1	0.36	126	0.96	0.25	120	0.95
R2	0.57	166	0.91	0.37	160	0.90
R3	2.05	53	0.95	1.98	43	0.95
R4	1.28	1448	0.90	0.80	1274	0.87
R5	0.45	211	0.95	0.26	190	0.94
R6	0.38	131	0.95	0.26	121	0.95
R7	2.37	696	0.90	1.86	178	0.89

[2] https://archive.ics.uci.edu/ml/index.html

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

Table 4: The twelve concurrent approaches

	Name	Param(s)	Range
A1	Leader (pioneer) [38]	$t = \frac{d_m}{\lambda^*}$	$[2, 10]$
A2	Leader (improved) [61]	$\|S\| = \frac{\|T\|}{\lambda^*}, c = \frac{\|T\|}{\mu^*}$	$[10, 500], [2, 5]$
A3	k-means sampling[65]	$\|S\| = \frac{\|T\|}{\lambda^*}$	$[10, 500]$
A4	Kernel sampling[35]	b,$\|S\| = \frac{\|T\|}{\lambda^*}$	$[10, 500]$
A5	Grid sampling[47]	$b, N^*_{cut}(axis)$	$[2, 10]$
A6	k-nearest-neighbors [21]	$b, k = \lambda^* \sqrt{\|T\|}$	$[0.2, 0.5]$
A7	Tree sampling[53]	$b, min_{size} = \frac{\|T\|}{\lambda^*}, N^*_{cut}$	$[50, 200], [1, 4]$
A8	Bagged sampling [17]	$N^*_{strata}, N_r = \frac{\|T\|}{\lambda^*}$	$[4, 20], [2, 100]$
A9	furthest-first traversal [54]	$\|S\| = \frac{\|T\|}{\lambda^*}$	$[10, 500]$
A10	k-means++ [5]	$\|S\| = \frac{\|T\|}{\lambda^*}$	$[10, 500]$
A11	Hybrid [20]	β^*, $\|S\| = \frac{\|T\|}{\lambda^*}$	$[10, 100], [10, 500]$

The table highlights the monotonic behavior of the algorithm even for two neighboring values of granularity. The relationship is not strictly monotone, the sample size, the Rand Index and the time (ms) do not decrease, and generally increase, with a smaller granularity. The Rand Index is higher than 0.85 in all cases. Only ambiguous items fall into different partitions.

With the hierarchical algorithm the results are likely to depend on the aggregation criterion. The RI is $(\mu, \sigma) = (0.91, 0.03)$ with the $Ward$ criterion and $(0.93, 0.02)$ with the *single link* one. The average and standard deviation were computed for the 7 databases and 10 partitions (from 5 to 14 clusters in S). The corresponding time ratio is obviously smaller than for the *k-means*: 0.06% in average for 3000 patterns.

6.2 Comparison with known algorithms

The 11 sampling approaches[3], representative of the literature, used for comparison with the proposal are shown in Table 4 with their main influential parameters and the corresponding variation range. The bias level b ranges in $[-1, +1]$.

To get comparable results, the other methods have been parameterized in order to yield the same number of representatives (an approximate one for the leader approach) than *DIDES* with $g_r = 0.03$. Only the mainly influen-

[3] A7 is similar to [53] except that the bins are not fuzzy, and are ordered via their weights until reaching a lower bound.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

tial parameter has been adjusted. The other ones were set at nominal values. These values were selected from previous tests as the ones that lead to a good trade-off between accuracy and tractability in average. They are $b = \{-0.2, -0, 15, -0.2, -0.1\}$ for $\{A4, A5, A6, A7\}$, $N_{cut} = 2$ for $A5$ and $A7$, $k = 0.5$ for $A7$, $N_{strata} = 15$ for $A8$ and $\beta = 20$ for $A11$.

Table 5 summarizes these extensive experiments for the *k-means* algorithm and a reduced number of clusters, from 5 to 14. The reported results are the best RI, on average, over the 7 data sets, of all the tested configurations and five trials for each of them. The time is given in ms.

Table 5: Known algorithms: best configuration Rand Index and time for the 7 data sets

	R1	R2	R3	R4	R5	R6	R7	$\overline{RI}$	$\overline{Time}$
A1	0.91	0.90	0.90	0.86	0.86	0.87	0.87	0.88	2989
A2	0.93	0.92	0.93	0.88	0.91	0.94	0.89	0.91	17444
A3	0.96	0.94	0.94	0.89	0.94	0.90	0.87	0.92	18228
A4	0.94	0.92	0.92	0.89	0.92	0.91	0.85	0.91	51983
A5	0.94	0.93	0.94	0.86	0.94	0.92	0.91	0.92	354
A6	0.93	0.92	0.90	0.84	0.92	0.91	0.88	0.90	138928
A7	0.95	0.94	0.95	0.89	0.91	0.94	0.90	0.93	11033
A8	0.94	0.92	0.94	0.86	0.93	0.91	0.90	0.91	413
A9	0.91	0.88	0.94	0.87	0.94	0.91	**0.91**	0.91	2109
A10	0.93	**0.95**	0.91	0.86	**0.97**	0.94	0.86	0.92	1276
A11	0.92	0.91	0.95	0.86	0.93	0.91	0.90	0.91	1418
DIDES	**0.96**	0.91	**0.95**	**0.90**	0.95	**0.95**	0.90	**0.93**	284
$100\|S\|/\|T\|$	0.35	0.57	2.04	1.28	0.45	0.38	2.37		

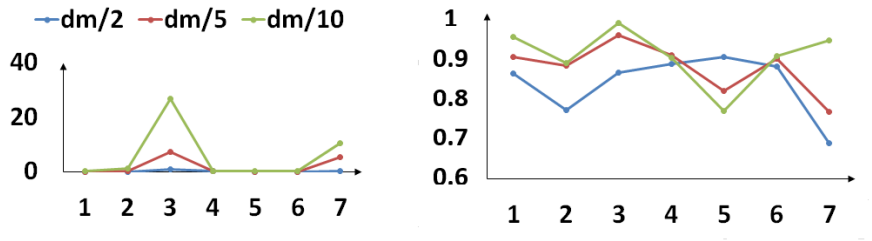

Figure 9: Leader behavior according to λ^*: sampling proportion (left) and corresponding Rand Index (right) for the 7 real world data sets

Some algorithms are not at all competitive according to the time criterion: $A2$, $A3$, $A4$, $A6$ and $A7$. The leader version $A1$ has to be discarded too as it is difficult to find a generic setting, and its behavior is not monotonic. The d_m parameter was estimated on a 10% random sample set and computed as:

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

$d_m = max(d(\mu, x_i)$, μ is the average of the set of vectors x_i. The results for different values of λ^* are shown in Figure 9. The left part shows the sample size (% of the whole), the right part illustrates the RI variation according to λ^*: no monotonicity can be deduced; the best value depends on the data.

The maximum of the RI for each data set is highlighted in bold font. When several algorithms have reached the same accuracy, and *DIDES* is among this group, the bold font is used in the *DIDES* row.

The reported RI are generally higher than 0.85. The chosen concurrent methods are rather accurate when using the same number of representatives than *DIDES*.

The grid sampling ($A5$)may lead to good results but remains difficult to tune. The *bagged* ($A8$), *fft* ($A9$), *kmeans++* ($A10$) and *Hybrid* ($A11$) algorithms seem to represent the best alternatives among the known methods

6.3 Complementary analysis

The most competitive algorithms, according to the three criteria, are now tested with the data shown in Figure 3, which includes an important amount of noise. As the RI gets better with a growing number of clusters, Table 6 shows the results for three sample sizes, and for a small (4) number of clusters.

Table 6: Results with data shown in Fig 3

	$100\|S\|/\|T\|$	Time (ms)	RI
A5	0.15	102	0.82
	0.5	112	0.92
	1.0	112	0.95
A8	0.15	102	0.91
	0.5	120	0.93
	1.0	275	0.96
A9	0.15	890	0.78
	0.5	1090	0.81
	1.0	2689	0.89
A10	0.15	387	0.95
	0.5	544	0.99
	1.0	1704	0.99
A11	0.15	1078	0.95
	0.5	1288	0.97
	1.0	1297	0.98
DIDES	0.15	88	0.98
	0.23	100	0.99
	0.48	108	0.99

When the sample size is high enough, all these methods prove to be accurate. When the sample size is constrained to 0.15% of the whole, only $A10$, $A11$ and

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

DIDES appear to be robust in presence of noise and with a small number of clusters.

This is not so surprising as they are based upon similar concepts. The main difference between $A10$ and *DIDES* stems from the noise management, which is achieved in a random way in $A10$. In the $A11$ approach, dense areas are first covered by a uniform random sampling, then the initial patterns represented in the sample are no more considered in the next iterations. The corresponding sample sizes are also comparable, even if *DIDES* samples are always smaller. As *DIDES* is not driven by the sample size, the reported results correspond to the granularity values that gives a similar size or RI.

Five of the concurrent approaches yield comparable results with *DIDES* when tested with real world data sets. The complementary tests on a synthetic database with clusters of various densities, shapes and with some noise provide interesting additional information. *DIDES* capacity to manage this type of data is empirically proven. In comparison, this test highlights some weaknesses of the other approaches. Only $A10$ and $A11$ remain competitive, but *DIDES* is still faster and parsimonious. It should also be underlined that *DIDES* highly improves the *fft* algorithm, both methods share the maximum distance concept.

7 Conclusion

A new sampling for clustering algorithm has been proposed in this paper. It is a distance-based one which also manages density. Even if the basic concepts of distance and density are known, their specific use produces a really new algorithm, *DIDES*.

While the results published in the literature highly depend on the tuning of the algorithm, only one parameter, called granularity, is left to the user with *DIDES* and its meaning is very clear: the smaller the granularity, the higher the number of representatives and, as a result, the better the representation. All the other parameters are hidden and inferred from the data itself. *DIDES* does not require any user expertise, as the internal mechanisms are smart enough to achieve the sampling task with data of various shape or density.

DIDES has some nice properties: it is insensitive to initialization, to the initial size and to noise and it is accurate according to the Rand index for the large diversity of data sets studied, either synthetic or real world.

It is at least as accurate as concurrent approaches. Internal optimization, which leads to a reduced number of distance computations, makes *DIDES* tractable where alternative algorithms fail. The CPU cost proved to be very low.

Two main perspectives remain open. It has been highlighted that the first iterations of the algorithm account for a large part of the computational cost. Speeding up the process at the beginning would allow the management of huge data sets. Is it possible to assess the adequate level of granularity while running the algorithm? An adaptive granularity would make *DIDES* a parameter free algorithm. Future work will be dedicated to both directions of improvement.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

References

[1] M. Al-Kateb and B.S. Lee. Adaptive stratified reservoir sampling over heterogeneous data streams. *Information Systems*, 39(1):199–216, 2014.

[2] Mohammed Al-Kateb, Byung Suk Lee, and Xiaoyang Sean Wang. Adaptive-size reservoir sampling over data streams. In *Scientific and Statistical Database Management, 2007. SSBDM'07. 19th International Conference on*, pages 22–22. IEEE, 2007.

[3] Bill Andreopoulos, Aijun An, Xiaogang Wang, and Michael Schroeder. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics*, 10(3):297–314, 2009.

[4] Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús M Pérez, and Iñigo Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 2013.

[5] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[6] A. Azzalini and N. Torelli. Clustering via nonparametric density estimation. *Statistics and Computing*, 17(1):71–80, 2007.

[7] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Functions Algorithms*. Plenum Press, New York, 1981.

[8] M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

[9] S. Chaudhuri, G. Das, and V. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Transactions on Database Systems*, 32(2), 2007.

[10] M.H. Chehreghani, H. Abolhassani, and M.H. Chehreghani. Improving density-based methods for hierarchical clustering of web pages. *Data and Knowledge Engineering*, 67(1):30–50, 2008.

[11] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[12] H. Chernoff. A note on an inequality involving the normal distribution. *Annals of Probability*, 9:533–535, 1981.

[13] M.-C. Chiang, C.-W. Tsai, and C.-S. Yang. A time-efficient pattern reduction algorithm for k-means clustering. *Information Sciences*, 181(4):716–731, 2011.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

[14] Stephen L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2:267–278, 1994.

[15] Swee Chuan Tan, Kai Ming Ting, and Shyh Wei Teng. A general stochastic clustering method for automatic cluster discovery. *Pattern Recognition*, 44(10):2786–2799, 2011.

[16] Frank De Morsier, Devis Tuia, Maurice Borgeaud, Volker Gass, and Jean-Philippe Thiran. Cluster validity measure and merging system for hierarchical clustering considering outliers. *Pattern Recognition*, 48(4):1478–1489, 2015.

[17] S. Dolnicar and F. Leisch. Segmenting markets by bagged clustering. *Australasian Marketing Journal*, 12(1):51–65, 2004.

[18] Pavlos S Efraimidis and Paul G Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185, 2006.

[19] Vassiliy A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969.

[20] Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. In *Advances in Neural Information Processing Systems*, pages 2142–2150, 2011.

[21] Hector Franco-Lopez, Alan R Ek, and Marvin E Bauer. Estimation and mapping of forest stand density, volume, and cover type using the k-nearest neighbors method. *Remote sensing of environment*, 77(3):251–274, 2001.

[22] Bernd Gutmann and Kristian Kersting. Stratified gradient boosting for fast training of conditional random fields. In *Proceedings of the 6th International Workshop on Multi-Relational Data Mining*, pages 56–68, 2007.

[23] John A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[24] Abdolreza Hatamlou, Salwani Abdullah, and Hossein Nezamabadi-pour. A combined approach for clustering based on k-means and gravitational search algorithms. *Swarm and Evolutionary Computation*, 6:47–52, 2012.

[25] Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Mathematics of operations research*, 10(2):180–184, 1985.

[26] Victoria J. Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[27] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

[28] M. R. Ilango and V Mohan. A survey of grid based clustering algorithms. *International Journal of Engineering Science and Technology*, 2(8):3441–3446, 2010.

[29] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[30] Mon-Fong Jiang, Shian-Shyong Tseng, and Chih-Ming Su. Two-phase clustering process for outliers detection. *Pattern recognition letters*, 22(6):691–700, 2001.

[31] Steven k. Thompson. *Sampling*. Wiley, 3rd edition, 2012.

[32] Leonard Kaufman and Peter Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the L1-Norm and Related Methods*, pages North–Holland, 1987.

[33] Kittisak Kerdprasop, Nittaya Kerdprasop, and Pairote Sattayatham. Density-biased clustering based on reservoir sampling. In *Database and Expert Systems Applications, 2005. Proceedings. Sixteenth International Workshop on*, pages 1122–1126. IEEE, 2005.

[34] Shehroz S Khan and Amir Ahmad. Cluster center initialization algorithm for k-modes clustering. *Expert Systems with Applications*, 40(18):7444–7456, 2013.

[35] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold. Efficient biased sampling for approximate clustering and outlier detection in large data sets. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1170–1187, 2003.

[36] Raghu Krishnapuram and J. M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, (1):98–110, May 1993.

[37] F. Leisch and S. Dolnicar. Winter tourist segments in austria: Identifying stable vacation styles using bagged clustering techniques. *Journal of Travel Research*, 41(3):281–292, 2003.

[38] Robert F Ling. Cluster analysis algorithms for data reduction and classification of objects. *Technometrics*, 23(4):417–418, 1981.

[39] Yinghua Lv, Tinghuai Ma, Meili Tang, Jie Cao, Yuan Tian, Abdullah Al-Dhelaan, and Mznah Al-Rodhaan. An efficient and scalable density-based clustering algorithm for datasets with complex structures. *Neurocomputing*, 2015.

[40] Kristína Machová, Miroslav Puszta, František Barčák, and Peter Bednár. A comparison of the bagging and the boosting methods using the decision trees classifiers. *Computer Science and Information Systems*, 3(2):57–72, 2006.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

[41] G. Menardi and A. Azzalini. An advancement in clustering via nonparametric density estimation. *Statistics and Computing*, 24(5):753–767, 2014.

[42] P. Mitra, C.A. Murthy, and S.K. Pal. Density-based multiscale data condensation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):734–747, 2002.

[43] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[44] MC Naldi and RJGB Campello. Comparison of distributed evolutionary k-means clustering algorithms. *Neurocomputing*, 163:78–93, 2015.

[45] A. Nanopoulos, Y. Theodoridis, and Y. Manolopoulos. Indexed-based density biased sampling for clustering applications. *Data and Knowledge Engineering*, 57(1):37–63, 2006.

[46] Alexandros Nanopoulos, Yannis Manolopoulos, and Yannis Theodoridis. An efficient and effective algorithm for density biased sampling. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 398–404, 2002.

[47] Christopher R. Palmer and Christos Faloutsos. Density biased sampling: An improved method for data mining and clustering. In *ACM SIGMOD Intl. Conference on Management of Data*, pages 82–92, Dallas, 2000.

[48] Md Anisur Rahman and Md Zahidul Islam. A hybrid clustering technique combining a novel genetic algorithm with k-means. *Knowledge-Based Systems*, 71:345–365, 2014.

[49] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[50] Muhammad Zia-ur Rehman, Tianrui Li, Yan Yang, and Hongjun Wang. Hyper-ellipsoidal clustering technique for evolving data stream. *Knowledge-Based Systems*, 70:3–14, 2014.

[51] Gerhard X. Ritter, José-A. Nieves-Vázquez, and Gonzalo Urcid. A simple statistics-based nearest neighbor cluster detection algorithm. *Pattern Recognition*, 48(3):918 – 932, 2015.

[52] F. Ros, M. Pintore, A. Deman, and J.R. Chrétien. Automatical initialization of rbf neural networks. *Chemometrics and Intelligent Laboratory Systems*, 87(1):26–32, 2007.

[53] F Ros, O Taboureau, M Pintore, and JR Chretien. Development of predictive models by adaptive fuzzy partitioning. application to compounds active on the central nervous system. *Chemometrics and intelligent laboratory systems*, 67(1):29–50, 2003.

Author-produced version of the article published in Knowledge and Information Systems, 2017, N°50(2), p.543-568.
The original publication is available at https://link.springer.com
http://dx.doi.org/10.1007/s10115-016-0946-8

[54] Daniel J Rosenkrantz, Richard E Stearns, and Philip M Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6(3):563–581, 1977.

[55] T Hitendra Sarma, P Viswanath, and B Eswara Reddy. Speeding-up the kernel k-means clustering method: A prototype based hybrid approach. *Pattern Recognition Letters*, 34(5):564–573, 2013.

[56] T.H. Sarma, P. Viswanath, and B.E. Reddy. Speeding-up the kernel k-means clustering method: A prototype based hybrid approach. *Pattern Recognition Letters*, 34(5):564–573, 2013.

[57] MGR Sause, A Gribov, AR Unwin, and S Horn. Pattern recognition approach to identify natural clusters of acoustic emission signals. *Pattern Recognition Letters*, 33(1):17–23, 2012.

[58] Vladimir Shenmaier. Complexity and approximation of the smallest k-enclosing ball problem. *European Journal of Combinatorics*, 48:81–87, 2015.

[59] D.M.J. Tax and R.P.W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[60] Robert L. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.

[61] P. Viswanath, T.H Sarma, and B.E. Reddy. A hybrid approach to speed-up the k-means clustering method. *International Journal of Machine Learning and Cybernetics*, 4(2):107–117, 2013.

[62] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

[63] Liang Wang, Christopher Leckie, Ramamohanarao Kotagiri, and James Bezdek. Approximate pairwise clustering for large data sets via sampling plus extension. *Pattern Recognition*, 44(2):222–235, 2011.

[64] Xiaochun Wang, Xiali Wang, and D Mitch Wilkes. A divide-and-conquer approach for minimum spanning tree-based clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 21(7):945–958, 2009.

[65] Y. Xiao, B. Liu, Z. Hao, and L. Cao. A k-farthest-neighbor-based approach for support vector data description. *Applied Intelligence*, 41(1):196–211, 2014.

[66] Ronald R. Yager and D. P. Filev. Generation of fuzzy rules by mountain clustering. *Journal of Intelligent and Fuzzy Systems*, 2:209–219, 1994.

[67] Miin-Shen Yang and Kuo-Lung Wu. A modified mountain clustering algorithm. *Pattern analysis and applications*, 8(1-2):125–138, 2005.

[68] Sobia Zahra, Mustansar Ali Ghazanfar, Asra Khalid, Muhammad Awais Azam, Usman Naeem, and Adam Prugel-Bennett. Novel centroid selection approaches for kmeans-clustering based recommender systems. *Information Sciences*, 2015.

[69] Caiming Zhong, Mikko Malinen, Duoqian Miao, and Pasi Fränti. A fast minimum spanning tree algorithm based on k-means. *Information Sciences*, 295:1–17, 2015.