Pattern Recognition 139 (2023) 109454

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

K-sets and *k*-swaps algorithms for clustering sets

Mohammad Rezaei, Pasi Fränti*

School of Computing, University of Eastern Finland, Joensuu, Finland

ARTICLE INFO

Article history:
Received 16 March 2021
Revised 28 January 2023
Accepted 20 February 2023
Available online 24 February 2023

Keywords:
Clustering sets
Similarity of sets
K-means
K-medoids
Random swap
K-swaps
Customer segmentation
Clustering healthcare records

ABSTRACT

We present two new clustering algorithms called *k-sets* and *k-swaps* for data where each object is a set. First, we define the mean of the sets in a cluster, and the distance between a set and the mean. We then derive the *k-sets* algorithm from the principles of classical *k*-means so that it repeats the assignment and update steps until convergence. To the best of our knowledge, the proposed algorithm is the first *k*-means based algorithm for this kind of data. We adopt the idea also into random swap algorithm, which is a wrapper around the *k*-means that avoids local minima. This variant is called *k-swaps*. We show by experiments that this algorithm provides more accurate clustering results than *k-medoids* and other competitive methods.

1. Introduction

Clustering is unsupervised learning that aims to group a set of data objects so that similar objects are located in the same cluster and dissimilar objects in different clusters [1]. Most previous clustering algorithms focus on numerical data whose inherent geometric properties can be exploited naturally to define distance functions between data points [2]. However, other types of data such as *categorical* data exist in educational sciences, sociology, market studies, and bioinformatics [3]. Algorithms such as *k*-medoids [4,5], *k*-modes [6], *k*-histograms [2], *k*-entropies [3], and *k*-distributions [7] have been proposed for clustering categorical data.

Another type of data is *sets* data, where each data object consists of a subset of a large selection of possible items. This can be called *bag-of-items*. Typical applications include text documents [8], videos [9,10], services [11], and web pages [12,13], which can be represented by a set of predefined keywords called *tags*. Customer-product data is another popular application of such data, which is used for recommender systems. A user is modelled by the set of products that he purchases, likes, or even just browses on the web site. New products are then recommended to the user based on the similarity of the items, or based on what was purchased by similar users [14–16].

Another important application of the sets data can be found in electronic health systems where patient records are represented by a set of diagnosis. These are either the ICD-9 or ICD-10 codes. For example, the set {K02, E11, T12} represents a patient with three diseases: *dental caries, type 2 diabetes* and *unspecified fracture of lower limb*. This information have been used for modeling temporal disease progression [17], analysis of diagnosis progression [18], and inferring medical diagnoses from patient similarities [19]. A disease prediction system has been built in [20] using clustering and association analysis. Local prediction models are produced for subgroups of similar patients based on the patient disease history, to determine the set of possible future illnesses an individual could develop.

The above-mentioned sets data have been typically addressed by application-specific algorithms and heuristics. So far, *k*-means algorithm [21] has been adopted for clustering categorical data [5,6,22], mixed numerical and categorical data [23,24], sequential data [25,26], and for document clustering [27,28]. However, to the best of our knowledge, no general-purpose algorithm derived from *k*-means exists for the sets data.

In this paper, we propose two general-purpose algorithms to cluster sets data: *k-sets* and *k-swaps*. The first algorithm is a direct modification of the classical *k*-means algorithm [21] to work with sets. However, *k*-means has strong dependency to the initial solution [29]. If there are lots of well-separated clusters in the data, *k*-means is incapable to re-allocate the centroids and the result depends mainly on the initialization [30]. For this reason, we intro-

* Corresponding author.
E-mail addresses: rezaei@cs.uef.fi (M. Rezaei), franti@cs.uef.fi (P. Fränti).

https://doi.org/10.1016/j.patcog.2023.109454

duce the second algorithm called *k-swaps*. It is a direct modification of the existing *random swap* algorithm [31], which has been shown to reach the correct centroid allocation even with high dimensional data.

The main challenges in both algorithms are how to define suitable *distance* measure between data objects, and how to define *mean* of a set of sets. Distance function in *k*-means is needed to evaluate the cost of an object to belong to a cluster, which is done by measuring the distance between the object and the mean of the cluster. In case of sets, this reduces to a *set-matching problem*. The similarity of two sets can be measured by coefficients like *Jaccard, Dice*, and *Correlation*.

The second problem, how to calculate the mean, is a trickier problem. Existing clustering algorithms usually avoid this problem entirely. For example, *k*-medoids replaces the mean by medoid, which is straightforward to calculate but requires quadratic processing time with the cluster size. Medoid is defined as the object in the cluster which has minimal total distance to all other objects in the set [32].

Agglomerative merge-based clustering avoids the need for mean simply by evaluating only the cost of the entire clusters, before and after the merge. The merge-based clustering always selects the next pair of clusters as the ones whose merge increases the cost least. However, this leads to a slow $O(n^2)$-$O(n^3)$ algorithm or requires huge amount of memory even for moderate size datasets (n>10,000).

Graph-based clustering has also been used for sets data [15], where each set is considered as a node and a weighted graph is constructed. The weight on each edge is the similarity between two sets, which can be calculated by different measures such as Jaccard similarity. Several graph-based clustering algorithms can then be applied. For example, graph-partitioning algorithm builds the clusters by minimizing the number of intra-cluster edge cuts and maximizing the number of inter-cluster connections [15]. K-means have also been applied to graphs in [33].

The original *k*-means algorithm and its variants, including random swap, need the mean, which limits their applicability to numerical data, and thus, unsuitable for categorical and sets data as such. One solution is to map categorical data to numerical space by using a set of (artificial or real) reference objects. The new data object is the distance vector of the original objects to these reference objects [34]. However, such conversion usually leads to high dimensional and sparse data even for a moderate number of possible category values. For example, if we have attribute *profession* with 100 possible values: {*teacher, farmer, police, baker*, …} we would have 100-dimensional binary vector for that attribute alone. For this reason, we will work in the original data space and tune the methods to fit the data rather than trying to fit the data to the methods.

The rest of the paper is organized as follows. In Section 2, we introduce the new *k*-means based algorithm called *k-sets*, and the random swap variant called *k-swaps*. In Section 3, we introduce artificial datasets specifically generated for sets data. We create 15 datasets divided into 4 groups so that only one parameter varies within each group: number of clusters, cluster overlap, resolution, and cluster size imbalance. In Section 4, we provide experimental results for these datasets and a brief case study for patient diagnosis records obtained from *Siun Sote* in North Karelia, Finland.

2. *K*-sets and *k*-swaps

The input for the clustering is a dataset of N objects, where each object X_i consists of l_i items:

$$X_i = \left\{x_i^1, x_i^2, \ldots, x_i^{l_i}\right\} \quad (1)$$

$$X_1 = \{A, E, B, C, F\}$$
$$X_2 = \{D, G, A, B\}$$
$$X_3 = \{A, H, C, B\}$$
$$X_4 = \{I, A, C, J, B\}$$
$$X_5 = \{B, D, C, A\}$$

Histogram

A	B	C	D	E	F	G	H	I	J
5	5	4	2	1	1	1	1	1	1

Normalized histogram

A	B	C	D	E	F	G	H	I	J
1	1	0.8	0.4	0.2	0.2	0.2	0.2	0.2	0.2

Fig. 1. Example of representing a cluster of m=5 objects by histogram (above) and by normalized histogram (below).

The items are labels from a predefined set of L possible items. The number of items can be different for every data object. We consider two well-known measures including *Jaccard* (J_{ij}) and *Cosine coefficients* (C_{ij}) to calculate the similarity between two objects X_i and X_j. Other set similarity measures such as *Overlap* or *Dice* could be used as well. Cosine similarity between two sets is also known as *Otsuka–Ochiai* coefficient.

$$J_{ij} = \frac{\left|X_i \cap X_j\right|}{\left|X_i X_j\right|} \quad (2)$$

$$C_{ij} = \frac{\left|X_i \cap X_j\right|}{\sqrt{\left|X_j\right| \times \left|X_j\right|}} \quad (3)$$

The size of the set L defines the *resolution* of the data. If L is large, it may happen that the intersections of two sets becomes very small or empty and the similarity can become meaningless. In such case, the cardinality of the sets (intersection and union) could be replaced by a *soft cardinality* [35]. In other words, the intersection of X_i and X_j would not be based on exact matches of the items but using some application-specific similarity measure to produce soft counts for the cardinality. If the items are natural words, we can use semantic relatedness [36] or syntactic similarity [37].

Another possible approach is to use so-called *pivot tree* [38] where each item is represented by a path in a product tree. The idea is to represent the items via hierarchical categorization so that every item has multiple labels from high level (generic label) to a low level (very specific). This allows better similarity measurement in case of large item set. In the rest of this paper, we use the original hard cardinality, i.e., the mere counts of the matched items as the set size. This keeps the method simple and independent of the application with wider generalization potential.

2.1. Cluster representative

From every cluster we form a histogram by counting how often each item appears in the cluster, see Fig. 1. Histograms have also been used for *categorical data* and *document clustering*. K-means variants for clustering categorical include *k-histograms* [2], *k-representatives* [39] and *k-entropies* [3]. The difference to our data is that categorical attribute can have only one possible item (category) whereas sets can have any number of items selected. The histograms are also calculated for each attribute separately, whereas the set data has a common item set. Categorical data can be represented as sets data where all sets have the same length.

Therefore, our proposed algorithms can be applied to categorical data as well but not vice versa.

In document clustering, histograms are constructed from the words in all documents. This approach has been referred to as *bag-of-words, bag-of-items* or *bag-of-X*. A document is then represented using vector-space model where each document is considered to be a vector. The most popular approach for setting the values in the vector is the combination of *term frequencies* (TF) and *inverse document frequency* (IDF) leading to TF-IDF. Numerical clustering methods are then employed. The difference to sets is that documents have (relative) frequencies for every item whereas our data item can take only binary value {0,1}. We apply histogram only for the cluster representatives, not for the data objects.

Medoid is another possible representative, which can have been used for numerical, categorical and also for sets data. It is defined as the object in the cluster, which has maximum average similarity (or minimum average distance) to all the objects in the cluster [32]. Other types of representatives include modes [6], distributions [7], and entropies [3] which have all been used for categorical data.

The histogram is constructed as follows. Suppose that a cluster includes m distinct items $y_i \ldots y_m$. The cluster representative has length m, and is formulated as follows:

$$\left[\begin{array}{c|c|c|c} y_1 & y_2 & \cdots & y_m \\ n(y_1) & n(y_2) & \cdots & n(y_m) \end{array}\right] \tag{4}$$

where $n(y_i)$ is the frequency of the item y_i. Other cluster representatives can be defined similarly. For example, the values in the representative can be normalized to the number of sets in the cluster to provide a representative with normalized values, see Fig. 1.

To avoid sparsity of the histogram, we limit its dimension to $m \leq 20$. If there are more items in the cluster, we simply select 20 most frequent items. Larger histograms would also work but the sparsity has slight negative effect to the clustering performance.

2.2. Distance calculation

For comparing histograms, classical approach is to use some metric. Hamming and Euclidean distance have been used but Cosine distance has become more popular due to its superior performance over Euclidean distance despite it is not metric. As observed in [40], after normalizing it becomes equal to Euclidean distance. Different divergence measures [41] have also been used, of which the most popular is Kullback-Leibler.

However, in our case the data objects are simple item sets and only the cluster representative is histogram. We therefore tailor our distance measures suited for this case. The distance of a set X of size l to the cluster representative h in (4) is defined corresponding to the Jaccard (d_J) and Cosine coefficient (d_C) in (2) and (3) as follows:

$$d_J(X,\ h) = 1 - \frac{\sum_{i=1}^{m} n(y_i) \times \delta_1(y_i)}{\sum_{i=1}^{m} n(y_i) + \sum_{i=1}^{l} \delta_2(x_i)} \tag{5}$$

$$d_C(X,\ h) = 1 - \frac{\sum_{i=1}^{m} n(y_i) \times \delta_1(y_i)}{\sqrt{\sum_{i=1}^{m} (n(y_i))^2}\sqrt{l}} \tag{6}$$

where $\delta_1(y_i)$=1 if the item y_i in the cluster is in the set X, and $\delta_1(y)$=0 otherwise. If the item x_i in the set X does not exist in the cluster representative, then $\delta_2(x_i)$=1, and $\delta_2(x_i)$=0 otherwise. The distance measure (5) can be considered as weighted Jaccard. If a cluster includes only one set, the distance between a set and the cluster representative reduces to the distance between two sets. In specific, (5) reduces to the standard Jaccard distance. Suppose that we have two clusters and their representatives as follows:

$X_1 = \{A, E, B, C, F\}$; $X_2 = \{D, G, A, B\}$; $X_3 = \{A, H, C, B\}$; $X_4 = \{I, A, C, J, B\}$; $X_5 = \{B, D, C, A\}$; $X_6 = \{A, E, J, I, F\}$; $X_7 = \{F, G, H, I\}$; $X_8 = \{I, H, C, B\}$; $X_9 = \{I, H, F, J, G\}$; $X_{10} = \{J, D, I, H\}$;

Histogram 1

A	B	C	D	E	F	G	H	I	J
5	5	4	2	1	1	1	1	1	1

Histogram 2

A	B	C	D	E	F	G	H	I	J	K
1	1	1	1	1	3	2	4	5	3	1

Consider the object $X_{11} = \{A, B, C, D, K, L, M\}$ for which we need to find its nearest cluster. If we just considered the number of shared items (4 and 5) as in the original weighted Jaccard distance, X_{11} would belong to cluster 2. However, the first cluster have higher frequencies for A, B, C and D, and its distance to X_{11} should be smaller. For this reason, our modified Jaccard has $n(y_i) \times \delta_1(y_i)$ in the numerator in (5) instead of just $\delta_1(y_i)$.

The following example demonstrates how the distance between a sample set (X_1 or X_2) and the cluster representative in Fig. 1 is calculated based on (5).

$$X_1 = \{A,\ E,\ B,\ C,\ F\}$$

$$X_2 = \{A,\ E,\ B,\ K,\ L\}$$

$$d_J(X_1,\ h) = 1 - \frac{5+1+5+4+1}{22} = 0.27$$

$$d_J(X_2,\ h) = 1 - \frac{5+1+5+0+0}{22+2} = 0.46$$

Distance calculation in case of normalized histogram is performed by applying a slight modification to (4): $\delta_2(x_i)$ is normalized to the cluster size similar to normalizing histogram. Overall, it is like the numerator and denominator are divided by the cluster size, and therefore, the result is the same regardless of whether we use the original or the normalized histogram. Distance calculation for X_2 in case of normalized histogram is shown below. For simplicity, we use the original histogram without any normalization.

$$d_J(X_2,\ h)1 - \frac{1+1+0.2+0+0}{4.4+2/5} = 0.46$$

Both the modified Jaccard and Cosine distance can be used within the following algorithms. In the experiments, we will compare the performance of the two measures and will find out that there is only minor difference in their performance.

2.3. K-sets algorithm

Similar to k-means, the k-sets algorithm includes two steps: *assignment* and *update*. The algorithm starts by randomly selecting k data objects as the initial representatives. At this stage, the representatives are simply the histogram of the selected data objects. In the assignment step, each data object is assigned to its nearest histogram according to (5). The *sum of distances to histograms* (SDH) is used as the objective function that we want to minimize:

$$SDH = \sum_{j=1}^{k} \sum_{i=1,\ x_i \in h_j}^{N} d\left(x_i,\ h_j\right) \tag{5}$$

where h_j is the histogram of cluster j, j=1..k, and distance d is calculated from (5). In the update step, the histogram of each cluster is calculated. The assignment and update steps are repeated until no further improvement in SDH is achieved, or until the change becomes less than a predefined threshold. We use TH=10^{-8} in this

***K*-sets(*X, k, TH, h*) → *label, h*, SDH**

If *h* is empty
  X' ← Select *k* random objects from *X*
  For *j* = 1 To ***k***
    h(*j*) ← construct histogram of *X'*(*j*)
SDH-new = *MAX_VALUE*
Repeat
  SDH ← *SDH-new*
  // Assign each object to its nearest histogram
  For *i* = 1 To ***N***
    label(*i*) = $argmin_j(d(X(i), h(j))$
  // Sum of distances to histograms
  SDH-new = 0
  For *i* = 1 To ***N***
    SDH-new = *SDH-new* + *d*(*X*(*i*), *h*(*label*(*i*))
  For *j* = 1 To ***k***
    h(*j*) ← Calculate histogram from *label*

Until |*SDH-new* – *SDH*| < *TH*

Fig. 2. Pseudo code of the *k*-sets algorithm

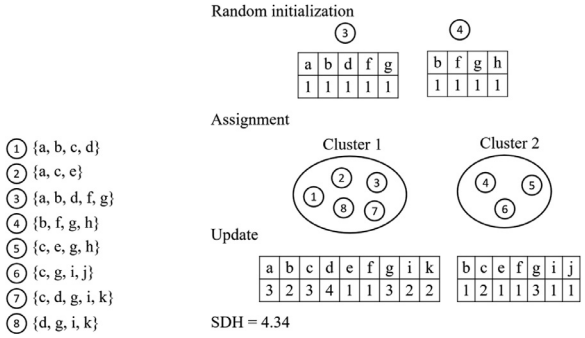

Fig. 3. Illustration of the *k*-sets algorithm for a toy dataset

paper. The pseudo code of the algorithm is shown in Fig. 2, and its operation illustrated in Fig. 3.

The time complexities of the assignment and update steps are O(*kNL*) and O(*NL*), respectively, where *L* is the average length of objects. The bottleneck is the calculation of the distance between the objects and the histograms in the assignment step. The time complexity of the algorithm is therefore O(*kNL*) in each iteration. This can be compared to *K*-medoids and agglomerative clustering algorithm which need all pairwise distance calculations, which makes their time complexity O(N^2), at minimum, and therefore, impractical for large data sets.

2.4. The k-swaps *algorithm*

Similar to *k*-means and *k*-medoids, the proposed *k*-sets algorithm also converges to a local optimum, which can be significantly worse than the global optimum. The result of the algorithm also depends on the choice of the initial histograms, which leads to different local optima [42]. We therefore adopt the idea into a more robust algorithm called *random swap* [42], which has been shown to be superior to *k*-means with the basic clustering benchmark with 11 numerical datasets [30].

***K*-swaps(*X, k, TH, Iter*) → *label, h*, SDH**

label, h, SDH ← *K*-sets (*X, k, TH*)
For *i* = 1 **To** *Iter*
  m ← random value from 1 to *k*
  j ← random value from 1 to *N*
  h(*m*) ← Calculate histogram of *X*(*j*)
  labels-new, h-new, SDH-new ← *K*-sets(*X, k, TH, h*)

  If *SDH-new* < *SDH*
    SDH ← *SDH-new*
    labels ← *labels-new*
    h ← *h-new*

Fig. 4. Pseudo code of *k-swap* algorithm.

Random swap can be considered as an evolutionary algorithm, which starts with a random solution and iteratively improves it for a fixed number of iterations. In each iteration, one centroid is randomly removed and replaced to a location given by a randomly chosen data point. Two iterations of *k*-means is applied to fine tune the solution. The new solution is used in the next iteration if it achieved a better objective function (lower *mean square error*). Otherwise, the previous solution is restored.

We call the random swap modification as *k-swaps* algorithm. It works as follows. Initial solution is obtained by applying the *k*-sets algorithm. The swap is then performed by replacing the histogram of a randomly selected cluster by a random data object. *K*-sets algorithm is applied again, and the objective function SDH is calculated. If the new SDH is smaller than the previous one, the new solution is accepted and used in the next iteration. The process is repeated for a fixed number of iterations, which is the only parameter given by the user. It can be set as high as time can be allocated. Originally, 5000 iterations was recommended (to be safe) but we have selected here *Iter*=300 as it seemed to be enough with our data. The pseudo code of this algorithm is shown in Fig. 4 and its operationm demonstrated in Fig. 5.

3. Generating artificial sets data

We generated 15 synthetic datasets by varying the following parameters: number of clusters, resolution, cluster overlap, and relative cluster density. The program for generating the datasets were implemented by Python. The sizes of all datasets are fixed to *N*=1200, and the number of items in each set is varied between 4 and 20. The range of each parameter is selected as follows:

- Number of clusters: *k*=4, 8, 16, 32 (default=16)
- Resolution: *L*=100, 200, 400, 800 (default=200)
- Cluster overlap: *o*=0%, 5%, 10%, 20%, 40% (default=5%)
- Imbalanced clusters: five types of cluster sizes: (default=Type 1)

In order to examine the effect of each factor, we change one parameter at a time while keeping the others fixed. The default setting of the parameters is: *k*=16, *L*=200, *o*=5%, and type 1 of imbalanced clusters which indicates equal size clusters.

To generate a dataset with *k* clusters, we first created *k* representatives. A representative is formed by randomly selecting a few items from all items (e.g. *L*=200). Although the size of the sets varies between 4 and 20, the size of the representative is restricted in the mid-range between 6 and 10. By knowing the representative and the size of cluster, we then create data objects for each cluster by randomly selecting items from the item set. The resulting set is accepted if its distance to the representative is less than

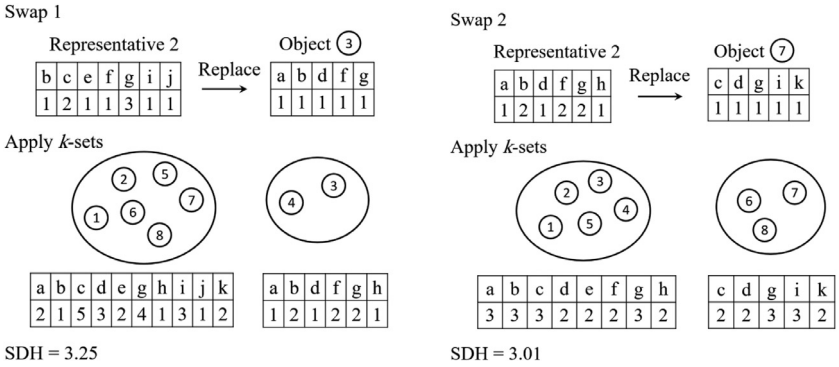

Fig. 5. Illustration of *k-swaps* algorithm for the same toy dataset in Figure 3.

Table 1
Five types of cluster sizes including equal size clusters and four types in which there are 4 big and 12 small clusters.

Type	4 Big	12 Small	Ratio
1	75	75	1
2	120	60	2
3	150	50	3
4	187-188	37-38	5
5	210	30	7

a threshold. This threshold controls the overlap between clusters, which is calculate by counting the number of objects which are closer to the representative of another cluster than the representative of their own cluster [30].

The first four datasets (k4, k8, k16, k32) include the number of clusters from 4 to 32 in order to examine the performance of clustering methods for low to high number of clusters. The other parameters have the default values.

The next four datasets (L100, L200, L400, L800) have varying resolution from 100 to 800. We define the resolution of a dataset as the total number of different elements that exist in the set objects. For example, for L=200, we create an object by selecting 4-20 items from item set of size 200. Higher number of items results in higher sparsity among data objects.

The third series of datasets (o0, o5, o10, o20, o40) is generated by varying the overlap threshold. We found the suitable thresholds for each dataset by trial and error manner. A small threshold would guarantee that the overlap between clusters is zero. However, datasets with compact clusters far away from each other would be too easy for many clustering algorithms to solve. We therefore use the largest threshold that satisfies the overlap condition. Overall, five datasets are generated with overlap values of 0%, 5%, 10%, 20% to 40%. The corresponding thresholds are 0.600, 0.790, 0.805, 0.820, and 0.843.

The fourth series of datasets are generated by considering five types of imbalance. As shown in Table 1, Type 1 contains 16 clusters with equal size. Type 2 to Type 5 include 4 big and 12 small clusters where the ratio between the larger and smaller cluster sizes is 2, 3, 5, and 7, respectively. The datasets are publicly available on the web[1].

4. Experiments

4.1. Experimental setup

We perform the following algorithms for all the 15 datasets:

- *k-medoids* [4,5]
- Single-link agglomerative clustering [43]
- Complete-link agglomerative clustering [43]
- *k-sets* [proposed]
- *k-swaps* [proposed]

We use cosine distance as the default measure in *k-sets* and *k-swaps*. The effect of this choice will be studied in Section 4.2.5. The k-medoids algorithm which is introduced in [5] is the most well-known algorithm for *k*-medoids clustering and is also based on *k*-means principle. Medoids are first selected randomly from the data objects, and then the assignment and update steps are repeatedly performed until convergence. In the update step, the medoid of each cluster is found, but this is much more time consuming than calculating the mean of cluster in *k*-means. Single-link and complete-link are agglomerative clustering algorithms in which each object is initially located in its own cluster. Two closest clusters are iteratively merged until the desired number of clusters is reached. The difference of the two algorithms is how to select the next two clusters to be merged.

The algorithms listed above have different cost functions. For example, *k*-sets and *k*-swaps minimize SDH, while *k*-medoids aims at minimizing sum of distances to medoids (SDM). For evaluating the performance of the clustering algorithms, we use *sum of pairwise distances* (SPD) in each cluster as the common objective function for all algorithms, and *adjusted Rand index* (ARI) [44] in respect to the ground truth partition.

K-medoids and *k*-sets produce different results every time because of the random initialization. To make the resulting qualities statistically valid, we therefore repeat these algorithms 100 times, and report the average values of SPD and ARI. We applied *k*-swaps algorithm with 300 iterations.

The programs for clustering algorithms and calculating cost functions were implemented by Python, and the program for calculating ARI was implemented by MATLAB. They are all publicly available on the web[2]. We performed all experiments on Windows 10 with Intel(R) Core(TM) i5-6600, 3.30 GHz processor and 8 GB RAM.

4.2. Results

4.2.1. Number of clusters

Results in Table 2 show that *k*-swaps gives the best result regardless of the number of clusters. Based on the ARI values, single-link (0.00) and complete-link (0.03) algorithms do not work at all as they basically produce random partitions. They end up creating one big cluster and several tiny clusters. For example, single-link produces one cluster of size 1197 and three clusters of size 1 with

[1] https://cs.uef.fi/sipu/datasets/

[2] https://cs.uef.fi/ml/software/

Table 2
Clustering quality with different number of clusters. Results with high accuracy (ARI ≥ 0.95) are emphasized.

Algorithm	k=4		k=8		k=16		k=32		Average ARI
	SPD	ARI	SPD	ARI	SPD	ARI	SPD	ARI	
Single link	519.97	0.00	533.26	0.00	540.47	0.00	533.79	0.00	0.00
Complete link	517.28	0.00	511.69	0.03	508.95	0.04	473.76	0.07	0.03
K-medoids	402.35	1.00	401.71	0.81	396.78	0.91	389.26	0.90	0.90
K-sets	402.35	1.00	394.30	1.00	396.42	0.92	390.79	0.88	0.95
K-swaps	402.35	1.00	390.54	1.00	386.39	1.00	377.56	0.99	1.00

Table 3
Clustering quality with different resolutions. Results with high accuracy (ARI ≥ 0.95) are emphasized.

Algorithm	L=100		L=200		L=400		L=800		Average ARI
	SPD	ARI	SPD	ARI	SPD	ARI	SPD	ARI	
Single link	517.91	0.00	540.47	0.00	555.62	0.00	567.00	0.00	0.00
Complete link	443.69	0.11	508.95	0.04	550.40	0.00	564.73	0.00	0.04
K-medoids	336.24	0.90	396.78	0.91	449.73	0.89	483.92	0.90	0.90
K-sets	334.80	0.91	396.42	0.92	449.04	0.92	484.31	0.93	0.92
K-swaps	323.17	0.99	386.39	1.00	441.21	0.99	477.69	0.99	0.99

Table 4
Clustering quality with different levels of overlap between clusters. Results with high accuracy (ARI ≥ 0.95) are emphasized.

Algorithm	o=0%		o=5%		o=10%		o=20%		o=40%		Average ARI
	SPD	ARI	SPD	ARI	SPD	ARI	SPD	ARI	SPD	ARI	
Single link	273.57	0.93	540.47	0.00	537.59	0.00	535.62	0.00	529.30	0.00	0.19
Complete link	333.83	0.59	508.95	0.04	517.65	0.02	524.82	0.00	520.07	0.00	0.13
K-medoids	290.98	0.85	396.78	0.91	406.12	0.91	413.65	0.90	415.92	0.99	0.91
K-sets	294.55	0.86	396.42	0.92	409.42	0.86	414.94	0.91	423.82	0.92	0.89
K-swaps	260.54	1.00	386.39	1.00	397.47	0.99	405.86	0.99	415.85	0.99	0.99

k=4. This is a known weakness of the algorithms in another context [45].

The performance of k-medoids (0.90) is slightly better than k-sets (0.88) at k=32. However, k-sets (0.95) and k-swaps (1.00) perform better than k-medoids, on average. The number of clusters has similar effect as was observed with numerical data in [30]: the more clusters, the more difficult the data becomes for k-means (and also k-medoids and k-sets here). We note that sometimes k-sets and k-swaps algorithms may over optimize SDH at the cost of less accurate clustering. In Euclidean vector space, this corresponds to optimizing for sum-of-squared errors (SSE) which equals to the scaled version of sum of pairwise distances (SPD). However, the same relationship does not hold between SDH and SPD. For this reason, it may happen that optimizing for SDH may not provide the best clustering result.

4.2.2. Resolution

K-sets and k-medoids have similar performance with different resolutions, see Table 3, while k-swaps performs much better with ARI=0.99, on average. Single-link and complete-link again fail to provide reasonable clustering result. Performance of the k-means variants (k-medoids, k-sets) is mostly invariant to the resolution. This corresponds to the results in [30] showing that the performance of k-means is not directly affected by the dimensionality.

4.2.3. Overlap

The results in Table 4 show that single-link and complete-link provide reasonably good results only when there is no overlap between clusters (o=0%) but they start to fail even with slight clusters overlap. The results for k-sets and k-medoids are almost similar but k-swaps provides better results with ARI=0.99, on average. The effect of the overlap on k-means variants is not clearly visible. With numerical data it was observed that k-means requires certain amount of overlap (around 4%) to be able to move the centroids between clusters [30]. This phenomenon is also observed here as k-medoids and k-sets provide worse results when the clusters have no overlap.

4.2.4. Imbalanced clusters

In this experiment, we examine the performance of clustering algorithms when there are imbalance in the cluster sizes. The imbalance increases from type 1 to type 5. The SPD and ARI results in Table 5 show that k-swaps provide the best results as always. The results of k-medoids and k-sets degrade when imbalanced clusters exist, however, we see that k-swaps performs well in all conditions. K-medoids (ARI=0.90), on average, performs better than k-sets (ARI=0.86). Unlike k-medoids, k-sets does not need all pairwise distances between data points. Calculating centroids in k-sets is therefore much faster that finding medoids in k-medoid algorithm. K-sets is therefore significantly faster than k-medoids.

4.2.5. Distance metrics

We next compare the effect of the distance measure. We consider the proposed modified Jaccard in Eq. (5), and Cosine distance in Eq. (6) within the best performing *k-swaps* algorithm. In addition, we also consider vector-space representation (TF-IDF) using Euclidean distance as this algorithm would be the closest to those used for document clustering. The original *random swap* algorithm is used instead of *k-swaps* because the data is in numerical space. We set the number of iterations $Iter = 1000$ for *random swap*, and $Iter = 300$ for k-swaps.

The results for the selected (most challenging) datasets are summarized in Table 6. The results show that *k-swaps* both with Cosine and modified Jaccard distance leads to the highest accuracy. *Euclidean* distance is worse than those two but still slightly better than *k-medoids* and *k-sets*. There are two contributors to the good accuracy: the histogram-based approach tailored for sets data, and the swapping technique. Both are needed to reach the high accuracy.

Table 5
Clustering quality with different relative density of clusters. Results with high accuracy (ARI ≥ 0.95) are emphasized.

Algorithm	*Type* 1		*Type* 2		*Type* 3		*Type* 4		*Type* 5		*Average ARI*
	SPD	ARI	SPD	ARI	SPD	ARI	SPD	ARI	SPD	ARI	
Single link	540.47	0.00	538.67	0.00	535.68	0.00	530.39	0.00	525.30	0.02	0.00
Complete link	508.95	0.04	511.46	0.02	494.75	0.07	503.65	0.06	501.56	0.07	0.05
K-medoids	396.78	0.91	393.62	0.92	393.23	0.95	396.20	0.82	403.61	0.88	0.90
K-sets	396.42	0.92	398.04	0.86	396.38	0.80	393.61	0.85	397.66	0.85	0.86
K-swaps	386.39	1.00	386.20	0.99	383.28	0.99	383.27	1.00	388.45	0.99	0.99

Table 6
Clustering quality (ARI) of *k-swaps* with different distance measures. Jaccard and Cosine are used for sets data and Euclidean for binary data. Results from *k-medoids* and *k-sets* have been added for reference. Results with high accuracy (ARI ≥ 0.95) are emphasized.

Method	$L = 200$	$L = 800$	$O = 0\%$	$O = 40\%$	*Type 4*	*Type 5*	K=32	**Average**
k-swaps / random swap*								
Cosine	1.00	0.99	1.00	0.99	1.00	0.99	0.99	**0.99**
Modified Jaccard	1.00	0.99	1.00	0.99	0.99	1.00	0.99	**0.99**
*Euclidean**	1.00	0.86	1.00	0.91	0.98	0.87	0.90	**0.93**
k-means variants								
K-medoids	0.91	0.90	0.85	0.99	0.82	0.88	0.90	**0.89**
K-sets	0.92	0.93	0.86	0.92	0.85	0.85	0.88	**0.89**

Table 7
Clustering quality of the 38,451 *Siun Sote* patients (k=16).

Algorithm	SPD	SDH	Processing time
K-medoids	16635	-	4237 s
K-sets	16434	28195	49 s
K-swaps	16500	26824	4918 s

The problem of the Euclidean distance is that conversion to numerical space results in sparse high-dimensional data. The swapping technique compensates this, but the use of the data-specific representation is still important, regardless of the distance function.

4.3. Application to health data

In this experiment, we used *Siun Sote* patient records of 51,909 patients collected from electronic health care records in North Karelia region of Finland. It is a subset of a larger nationwide dataset consisting of 58 M patient visits between 2015 and 2018 [46]. Each patient record includes a set of diagnoses represented by ICD-10 codes along with other information. We used only ICD-10 codes for the clustering. There are very similar diseases such as K00, K02 and K04 in the data. WHO provides a grouping of these diseases as one[3]. We used the grouped version of the codes because, according to our experiments, clustering without grouping provided less interesting (too obvious) clusters.

After the grouping, we eliminated rare codes that appear less than 50 times in the data. We also removed patients with less than four diagnoses as they can be considered as outliers that might mislead the clustering result and hide more important general trends. After the above preprocessing steps, the number of patients is reduced to 38,451. The number of different items (diagnoses) in the processed data is 174. The quality of the clustering results is reported in Table 7.

Qualitative analysis of the clusters generated by *k*-swaps algorithm is presented next to show that the method generates meaningful clusters for the application. Distribution of the diagnoses in selected clusters are summarized in Fig. 6 and summarized in Table 8. To extract the most dominant diagnoses, we calculated the frequency of each diagnosis in the cluster relative to its total frequency in the dataset. Any number above 1 indicates that the diagnosis is over-represented in the cluster compared to that of the overall population. It is therefore representative to characterize the patients in the cluster.

Table 8
Summary of the most dominant diagnoses in the selected clusters.

Cluster 2
J40: Chronic lower respiratory diseases
J95: Other diseases of the respiratory system
I10: Hypertensive diseases
R00: Symptoms and signs involving the circulatory and respiratory systems
Cluster 3
I20: Ischaemic heart diseases
I30: Other forms of heart disease
Cluster 4
O94: Other obstetric conditions, not elsewhere classified
O30: Maternal care related to the fetus and amniotic cavity and possible delivery problems
O20: Other maternal disorders predominantly related to pregnancy
O00: Pregnancy with abortive outcome
F60: Disorders of adult personality and behavior
A50: Infections with a predominantly sexual mode of transmission
B15: Viral hepatitis
Cluster 10
R10: Symptoms and signs involving the digestive system and abdomen
I10: Hypertensive diseases
N25: Other disorders of kidney and ureter
O94: Other obstetric conditions, not elsewhere classified
Cluster 11
F10: Mental and behavioral disorders due to psychoactive substance use
B15: Viral hepatitis
T36: Poisoning by drugs, medicaments and biological substances
F90: Behavioral and emotional disorders
O00: Pregnancy with abortive outcome
Cluster 12
M20: Other joint disorders
A30: Other bacterial diseases

From this simple analysis, we can observe that the clustering method has created meaningful clusters. While patients can have multiple different diseases, clusters 2, 3, and 10 have collected lots of certain type of problems. Cluster 2 includes respiratory diseases. Cluster 3 is characterized by heart diseases (I20 and I30) and Cluster 10 by digestive related diagnoses (R10, N25, O94). Cluster 4 is the largest cluster consisting of 9137 patients. Compare this to the average size of all clusters of 2403 patients. The cluster is charac-

[3] https://icd.who.int/browse10/2019/en#/XVIII

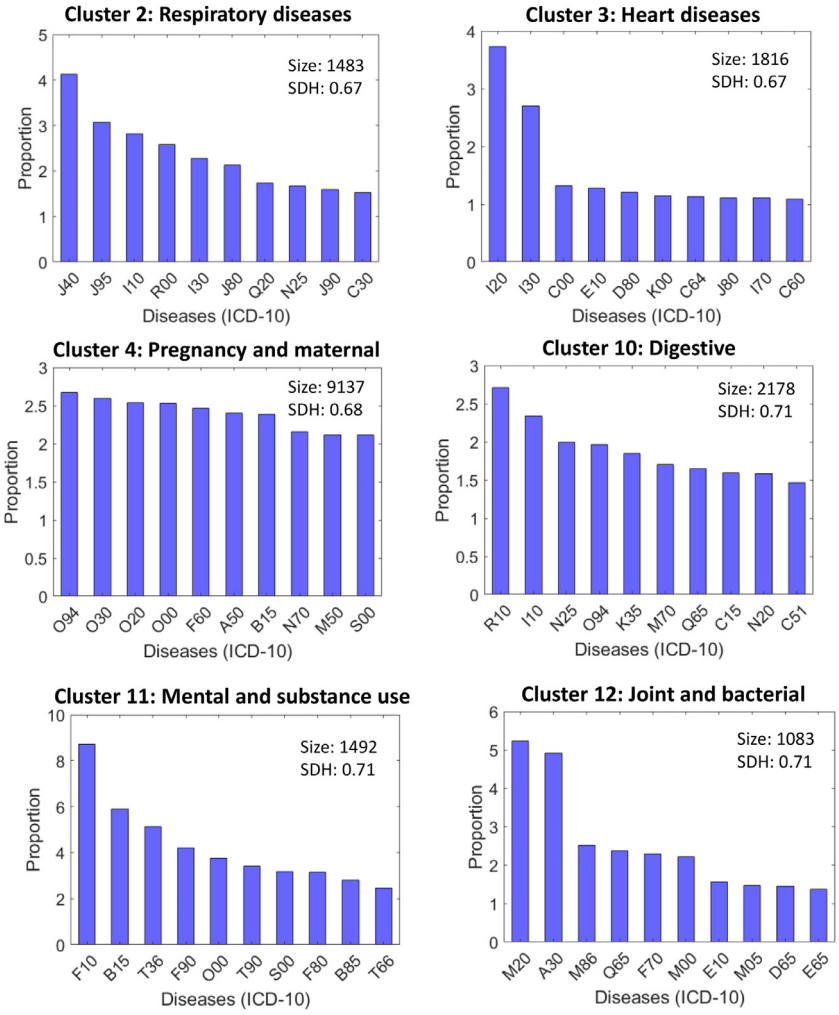

Fig. 6. Selected clusters generated by the *k*-swaps algorithm. Clusters are sorted from most compact (smallest SDH) to least compact. Each cluster is represented by a histogram of its ten most dominant diseases based on relative risk. The size and compactness (SDH) of the clusters are also shown.

terized by pregnancy related diagnoses (O00, O20, O30, O94), but also codes related to mental (F60) and sexual transmitted diseases (A50).

Clusters 11 and 12 are somewhat more surprising as they combine different types of diseases. Cluster 11 combines various diagnoses such as mental disorders and substance abuse, but also abortion (O00). Cluster 12 is dominated by joint disorders (M20) and bacterial diseases (A30). The significance of these findings would require medical expertise and is out of scope of this work. Nevertheless, the clustering seems to provide potential new and medically relevant information.

5. Conclusions

By introducing the *k*-sets algorithm, we have shown that standard *k*-means can be applied to cluster data where the objects are sets from a fixed item set. Replacing the centroid by histogram of the items, we defined distance function between the objects (set) to the cluster (histogram). The significance of the proposed algorithm is that there is no need for very complex algorithms to cluster such data. The algorithms are general as they can be applied to any applications where the objects are described as subsets of a common item set. For example, it can be used to cluster customers using customer-product connections, and patients based on patient-disease connections. The proposed algorithm can also be applied to categorical data since categorical data can be converted to sets data.

The *k*-set algorithm inherits the algorithmic weakness of *k*-means but there are algorithmic modifications to overcome this weakness. We selected the random swap algorithm which is a wrapper around the *k*-means and modified it to work with the sets data. The proposed modification, called *k*-swaps, provides highest clustering accuracy among all tested variants. This shows that there is no need for more complex algorithms to the problem as such simple modifications can already provide high quality clustering results.

We also introduced a benchmark for the set data by varying the number of clusters, resolution of the item set, overlap of the clusters, and cluster size imbalance. The results show that the *k*-sets algorithm is invariant to the resolution, but the performance decreases when the number of clusters increases.

While the time complexities of the proposed algorithms are lower than their alternatives, k-medoids and agglomerative clustering, the processing time can still be a bottleneck especially for big data. Existing parallel solutions for k-means [47] and random swap [48] are expected to be applicable also for k-sets and k-swaps algorithms with reasonable efforts. Some algorithmic speed-up techniques like Ball k-means [49], however, rely on vector data in Eu-

clidean space and cannot apply to sets as such. Some ideas like the activity detection of cluster centroids [50] (also used in [49]) are based on more general properties and should be applicable for the sets data as such.

The proposed algorithms also leave the number of clusters to the user to decide, similarly as standard k-means. However, general frameworks such as [51] can be adopted to k-sets and k-swaps by integrating existing cluster validity like silhouette coefficient as a constraint in the clustering objective. Stability-based approach [52,53] would also be applicable for the sets data directly for this purpose.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All data and program codes are publicly available

Acknowledgments

The work was funded by the Strategic Research Council (SRC) at the Academy of Finland (IMPRO project); Grant No. 336330 and 336325.

References

[1] M.S. Yang, K.L. Wu, A similarity-based robust clustering method, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004) 434–448.
[2] Z. He, X. Xu, S. Deng, and B. Dong, "K-histograms: an efficient clustering algorithm for categorical dataset," arXiv preprint cs/0509033, 2005.
[3] V. Hautamäki, A. Pöllänen, T. Kinnunen, K.A. Lee, H. Li, P. Fränti, in: A comparison of categorical attribute data clustering methods," in Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer Berlin Heidelberg, 2014, pp. 53–62.
[4] L. Kaufman and P.J. Rousseeuw, "Clustering by means of medoids," *Proceedings of the Statistical Data Analysis Based on the L1–Norm and Related Methods*, Vol.31, 1987.
[5] H.S. Park, C.H. Jun, A simple and fast algorithm for K-medoids clustering, Expert Syst. Appl. 36 (2009) 3336–3341.
[6] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Min. Knowl. Discov. 2 (1998) 283–304.
[7] Z. Cai, D. Wang, L. Jiang, K-distributions: a new algorithm for clustering categorical data, in: Proceedings of the International Conference on Intelligent Computing, 2007, pp. 436–443.
[8] S.S. Kang, Keyword-based document clustering, in: Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages-Volume 11, 2003, pp. 132–137.
[9] J. Magalhães, F. Ciravegna, S. Rüger, Exploring multimedia in a keyword space, in: Proceedings of the 16th ACM International Conference on Multimedia, 2008, pp. 101–110.
[10] A. Balagopalan, L.L. Balasubramanian, V. Balasubramanian, N. Chandrasekharan, A. Damodar, Automatic keyphrase extraction and segmentation of video lectures, in: Proceedings of the IEEE International Conference on Technology Enhanced Education (ICTEE), 2012, pp. 1–10.
[11] M. Rezaei, P. Fränti, Matching similarity for keyword-based clustering, in: Proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), 2014, pp. 193–202.
[12] W.T. Yih, J. Goodman, V.R. Carvalho, Finding advertising keywords on web pages, in: Proceedings of the 15th International Conference on World Wide Web, 2006, pp. 213–222.
[13] M. Rezaei, N. Gali, P. Fränti, ClRank: a method for keyword extraction from web pages using clustering and distribution of nouns, in: Proceedings of the Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE/WIC/ACM, 2015, pp. 79–84.
[14] P. Melville, V. Sindhwani, Recommender systems, Encyclopedia of Machine Learning and Data Mining, Encyclopedia of machine learning 1 (2010) 829–838.
[15] G.K. Gupta, J. Ghosh, Value-balanced agglomerative connectivity clustering, in: Proceedings of the Data Mining and Knowledge Discovery: Theory, Tools, and Technology III, 2001, pp. 6–15.
[16] H. Yan, Y. Tang, Collaborative filtering based on gaussian mixture model and improved jaccard similarity, IEEE Access 7 (2019) 118690–118701.
[17] A.B. Jensen, P.L. Moseley, T.I. Oprea, S.G. Ellesøe, R. Eriksson, H. Schmock, et al., Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients, Nat. Commun. 5 (2014) 4022.
[18] E. Jeong, K. Ko, S. Oh, H.W. Han, Network-based analysis of diagnosis progression patterns using claims data, Sci. Rep. 7 (2017) 15561.
[19] A. Gottlieb, G.Y. Stein, E. Ruppin, R.B. Altman, R. Sharan, A method for inferring medical diagnoses from patient similarities, BMC Med. 11 (2013) 194.
[20] F. Folino, C. Pizzuti, A comorbidity-based recommendation engine for disease prediction, in: Proceedings of the IEEE 23rd International Symposium on Computer-Based Medical Systems (CBMS), 2010, pp. 6–12.
[21] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.
[22] T.H.T. Nguyen, V.N. Huynh, A k-means-like algorithm for clustering categorical data using an information theoretic-based dissimilarity measure, Springer International Publishing, 2016.
[23] Z. Huang, Clustering large data sets with mixed numeric and categorical values, in: Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining,(PAKDD), 1997, pp. 21–34.
[24] D.K. Roy, L.K. Sharma, Genetic k-means clustering algorithm for mixed numeric and categorical data sets, Int. J. Artif. Intell. Appl. 1 (2010) 23–28.
[25] V. Guralnik, G. Karypis, A scalable algorithm for clustering sequential data, in: *international conference on data mining*, 2001, pp. 179–186.
[26] S. Soheily-Khah, A. Douzal-Chouakria, E. Gaussier, Generalized k-means-based clustering for temporal data under weighted and kernel time warp, Pattern Recognit. Lett. 75 (2016) 63–69.
[27] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," 2000.Technical Report; 00-034; https://conservancy.umn.edu/handle/11299/215421
[28] M. Mahdavi, H. Abolhassani, Harmony K-means algorithm for document clustering, Data Min. Knowl. Discov. 18 (2009) 370–391.
[29] P. Fränti, S. Sieranoja, How much can k-means be improved by using better initialization and repeats? Pattern Recognit. 93 (2019) 95–112.
[30] P. Fränti, S. Sieranoja, K-means properties on six clustering benchmark datasets, Appl. Intell. 48 (12) (December 2018) 4743–4759, doi:10.1007/s10489-018-1238-7.
[31] P. Fränti, Efficiency of random swap clustering, J. Big Data 5 (2018) 13.
[32] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, 344, John Wiley & Sons, 2009.
[33] S. Sieranoja, P. Fränti, Adapting k-means for graph clustering, Knowl. Inf. Syst. 64 (2022) 115–142.
[34] H. Ralambondrainy, A conceptual version of the K-means algorithm, Pattern Recognit. Lett. 16 (1995) 1147–1157.
[35] S. Jimenez, F. Gonzalez, A. Gelbukh, Text comparison using soft cardinality, in: Proceedings of the International Symposium on String Processing and Information Retrieval, 2010, pp. 297–302.
[36] Q. Zhao, M. Rezaei, H. Chen, P. Fränti, Keyword clustering for automatic categorization, in: Proceedings of the Pattern Recognition (ICPR), 21st International Conference on Pattern Recognition, 2012, pp. 2845–2848.
[37] N. Gali, R. Mariescu-Istodor, D. Hostettler, P. Fränti, Framework for syntactic string similarity measures, Expert Syst. Appl. 129 (2019) 169–185.
[38] X. Chen, J. Xu, R. Zhou, P. Zhao, C. Liu, J. Fang, et al., S 2 R-tree: a pivot-based indexing structure for semantic-aware spatial keyword search, Geoinformatica 24 (2020) 3–25.
[39] O.M. San, V.N. Huynh, Y. Nakamori, An alternative extension of the k-means algorithm for clustering categorical data, Int. J. Appl. Math. Comput. Sci. 14 (2004) 241–247.
[40] S. Zhong, Efficient online spherical k-means clustering, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 2005, pp. 3180–3185.
[41] F. Nielsen, R. Nock, S.I. Amari, On clustering histograms with k-means by using mixed α-divergences, Entropy 16 (2014) 3273–3301.
[42] P. Fränti, J. Kivijärvi, Randomised local search algorithm for the clustering problem, Pattern Anal. Appl. 3 (2000) 358–369.
[43] P. Fränti, O. Virmajoki, V. Hautamaki, Fast agglomerative clustering using a k-nearest neighbor graph, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 1875–1881.
[44] M. Rezaei, P. Fränti, Set matching measures for external cluster validity, IEEE Trans. Knowl. Data Eng. 28 (2016) 2173–2186.
[45] M. Rezaei, "Clustering validation," PhD Thesis, School of Computing, University of Eastern Finland, 2016.https://erepo.uef.fi/bitstream/handle/123456789/16786/urn_isbn_978-952-61-2145-1.pdf?sequence=1
[46] P. Fränti, S. Sieranoja, K. Wikström, T. Laatikainen, Clustering diagnoses from 58M patient visits in Finland between 2015 and 2018, JMIR Med. Inform. 10 (5) (2022) e35422.
[47] R. Mussabayev, N. Mladenovic, B. Jarboui, R. Mussabayev, How to use K-means for big data clustering? Pattern Recognit. 137 (2023) 109269.
[48] L. Nigro, F. Cicirelli, P. Fränti, Parallel random swap: an efficient and reliable clustering algorithm in Java, Simul. Model. Pract. Theory 124 (2023) 102712 1-17, April.
[49] S. Xia, D. Peng, D. Meng, C. Zhang, G. Wang, E. Giem, … Z. Chen, Ball k-means: fast adaptive clustering with no bounds, IEEE Trans. Pattern Anal. Mach. Intell. 44 (01) (2022) 87–99.
[50] T. Kaukoranta, P. Fränti, O. Nevalainen, A fast exact GLA based on code vector activity detection, IEEE Trans. Image Process. 9 (8) (2000) 1337–1342.

[51] A.M. Bagirov, R.M. Aliguliyev, N. Sultanova, Finding compact and well-separated clusters: clustering using silhouette coefficients, Pattern Recognit. 135 (2023) 109144.
[52] M. Rezaei, P. Fränti, Can the number of clusters be determined by external indices? IEEE Access 8 (2020) 89239–89257.
[53] J. Saha, J. Mukherjee, CNAK: cluster number assisted K-means, Pattern Recognit. 110 (2021) 107625.

Mohammad Rezaei received the BSc degree in electronic engineering and the MSc degree in biomedical engineering from the Amirkabir University of Technology, Tehran, Iran, in 1996 and 2003, respectively. He received his Ph.D. in Computer Science from the University of Eastern Finland. His research interests include data clustering, multimedia processing, classification, and retrieval.

Pasi Fränti received his Master and PhD degrees from University of Turku, Finland in 1991 and 1994 respectively. He has been tenure Professor in the University of Eastern Finland since 2000 where he is currently the leader of the machine learning group. He has published over 100 journal and 179 peer review conference papers. His current research interests include clustering algorithms, location-based services, machine learning, web and text mining, and optimization of health care services. He has supervised 32 Ph.D. graduates and is currently supervising seven more. He is editor-in-chief of AIMS Press journal Applied Computing and Intelligence.