Pattern Recognition Letters 53 (2015) 77–84

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

A grid-growing clustering algorithm for geo-spatial data[☆]

Qinpei Zhao[a], Yang Shi[a], Qin Liu[a,*], Pasi Fränti[b]

[a] *School of Software Engineering, Tongji University, Shanghai, 201804, China*
[b] *School of Computing, University of Eastern Finland, Joensuu, 80101, Finland*

ARTICLE INFO

Article history:
Received 15 March 2014
Available online 22 October 2014

Keywords:
Grid-based clustering
Grid-growing
Geo-spatial data
GPS devices
Regions of interest

ABSTRACT

Geo-spatial data with geographical information explodes as the development of GPS-devices. The data contains certain patterns of users. To dig out the patterns behind the data efficiently, a grid-growing clustering algorithm is introduced. The proposed algorithm takes use of a grid structure, and a novel clustering operation is presented, which considers a grid growing method on the grid structure. The grid structure brings the benefit of efficiency. For large geo-spatial data, the algorithm has competitive strength on the running time. The total time complexity of the algorithm is $O(N \log N)$, where the time complexity mainly comes from the seed selection step. The grid-growing clustering algorithm is useful when the number of clusters is unknown since the algorithm requires no parameter on the number of clusters. The clusters detected could have arbitrary shapes. Furthermore, sparse areas are treated as outliers/noises in the algorithm. An empirical study on several data sets indicates that the proposed algorithm works much more efficiently than other popular clustering algorithms.

1. Introduction

Due to the emergence of GPS-devices, variants of location-based applications are developed. Geo-tagged multimedia, such as trajectory, photo, video and text, are collected by users through the applications. The wealth of geo-spatial data provides an opportunity for performing further research topics.

Because of the convenience of collecting geo-spatial data, the size of the data usually explodes. Therefore, there comes up problems related to storage, visualization and detection of meaningful patterns. Enormous amounts of GPS trajectories, which record users' spatial and temporal information bring heavy burdens for both network transmission and data storage. A compression algorithm in [1] optimizes both the trajectory simplification and the coding procedure using the quantized data. The way of visualization [2] on the geo-tagged data needs to be well designed in order to avoid problems such as clutter problem.

GPS trajectories and other geo-spatial data often contain large information and unknown pattern. For example, a bunch of geo-spatial data can be collected from photos and trajectories uploaded by one user. From the data, it would be interesting for the user to know, manage and share his/her activity area. For other users, the activity area can be a suggested place to visit. In [3], research on extracting associative points-of-interest patterns from geo-tagged photos in Queensland, Australia is introduced. A system to provide both location and activity recommendations is introduced in [4], which is based on the location information of GPS data and some available user comments. Based on the concept of semantic trajectories, trajectories are observed as a set of stops and moves [4,5]. The start and end points of a trajectory could be interesting places with very high probability [6].

There is an increasing need for methods to extract knowledge from the data [7]. Clustering algorithms are applied on users' GPS data to discover spatio or temporal patterns. Typical clustering algorithms include model-based clustering (e.g., EM algorithm [8]), partition-based clustering (e.g., k-means and its variants [9,10]), graph-based clustering (e.g., spectral clustering [11]), density-based clustering (e.g., DBSCAN) and grid-based clustering. Different types of clustering algorithms have their advantages and disadvantages. Two aspects are usually considered when the algorithms are applied in geo-spatial data. One is the efficiency because the geo-spatial data is usually large. The other one is the adjustment on the algorithm to adapt in real applications.

A variant of k-means algorithm is proposed in [12] to cluster places where GPS signal is lost from the satellites into significant locations. The clusters are initially centered at K chosen points with a given radius, and iteratively move to a denser area. However, k-means related methods need to determine the parameter K beforehand. A hybrid clustering algorithm [13] that combines hierarchical method and grid-based method is presented to discover frequent spatial patterns

[☆] This paper has been recommended for acceptance by D. Dembele.
[*] Corresponding author. Tel.: +86 (0)21 69589976.
E-mail addresses: zhao@cs.joensuu.fi (Q. Zhao), qin.liu@tongji.edu.cn (Q. Liu).

http://dx.doi.org/10.1016/j.patrec.2014.09.017
0167-8655/© 2014 Elsevier B.V. All rights reserved.

among trips. Algorithm *SMoT* [14] and its alternative variant [5] are proposed to compute stops and moves from trajectory sample points. A clustering algorithm based on changing parts of the concepts in *DBSCAN* [15] is proposed in [5] to find low speed regions from trajectories. Because the DBSCAN has features of detection on clusters of arbitrary shape and noise detection, the algorithm is also commonly used for processing geo-spatial data. A *DBCLASD* [16] is designed with the advantages of discovering clusters of arbitrary shape and requiring no parameters. A new clustering algorithm based on DBSCAN [17] is presented specially for the problem of analysis of places and events using large collections of geo-tagged photos.

It is demonstrated in [18] that the grid-based technique obtains better results than the density-based one. The grid-based clustering algorithm [19] partitions the data space into a certain number of cells and performs clustering operations on the cells. Therefore, the algorithm is efficient when the number of cells (n) is much less than the size of the original data (N). The grid-based algorithms require at least one scan of all individual objects (points), which makes the time complexity of the algorithms at least O(N). A statistical information grid-based method (*STING*) was introduced in [20], which reduced the time complexity of O(N) to O(n) for each query. Since the method constructed a hierarchical structure by going through the whole data, the overall complexity is still linearly proportional to the size of data with a small constant factor. A grid-based hierarchical clustering algorithm was proposed in [13] for large-scale and event-based telematics data sets. A merge operation among neighborhood clusters is employed.

In this paper, we focus mainly on the efficiency of the algorithms. A grid-growing clustering algorithm is proposed for geo-spatial data specifically. To verify the validity of the algorithm, artificial data sets are tested firstly. Then, the efficiency is demonstrated by geo-spatial data which are sampled from trajectories. The experimental results demonstrate that the proposed algorithm is more effective and much faster than traditional clustering algorithms such as a k-means variant (*litekmeans*), Greedy EM, DBSCAN, a spectral clustering (*LSC*) and a pairwise random swap clustering (*PRS*).

The rest of the paper is organized as follows: Section 2 introduces the grid-growing clustering algorithm. An analysis on the time complexity of the algorithm is also included in the section. The experimental results are displayed in Section 3 and the conclusion is followed in Section 4.

2. The grid-growing clustering algorithm

The proposed grid-growing clustering algorithm (see Algorithm 1) is mainly designed for geo-spatial data. Let $D(x, y)$ be the location-based data with N points and P be the partitions as the result from the clustering algorithm.

Given the data D, the first step is to generate a grid structure $I(n, n)$, where n is the number of rows and columns in the grid structure. The number of n decides the size of the grids. For each data point, it is assigned to appropriate grids according to its locations.

The second step is to perform a region growing on the grid structure, which generates a certain number of groups. In this step, m seeds are firstly selected. With the selected seeds, regions are grown to adjacent points. K number of regions or clusters are formed after then.

Finally, the clustering partitions P are obtained from the K number of regions. We explain the details for each step in the following sections (Algorithm 2).

2.1. Grid construction

```
Input: $D(x, y), n, m$
Output: $P$
Step 1: Construction on grid structure $I$
$I(n, n) \leftarrow GridConstruct(D, n)$;
Step 2: Grid Growing on $I$,
$R \leftarrow GridGrowing(I > 0, seed)$ (see Algorithm 2);
Step 3: Get partitions $P \leftarrow GetPartition(R)$;
return $P$;
```

Algorithm 1: *Grid-growing* clustering algorithm

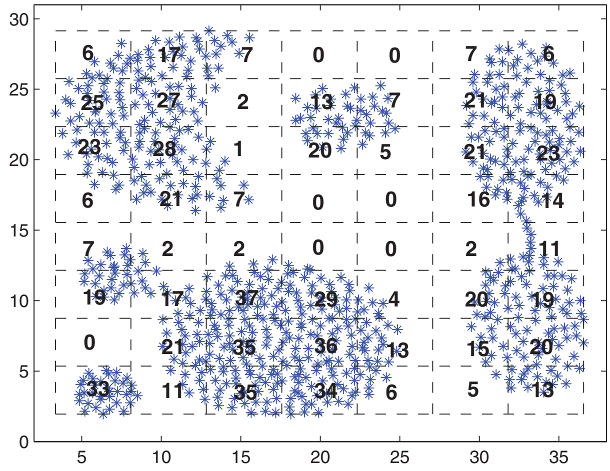


Fig. 1. An example of a grid structure on data *aggregation*. The numbers in the grids are the capacity of the grids.

For each point in data set $D(x, y)$, the row ($t \in 1, 2, \ldots, n$) and column ($s \in 1, 2, \ldots, n$) number of the grids that the point belongs are calculated as follows:

$$t = \left\lceil \frac{x_{\max} - x}{x_{\max} - x_{\min}} \cdot n_x \right\rceil$$

$$s = \left\lceil \frac{y_{\max} - y}{y_{\max} - y_{\min}} \cdot n_y \right\rceil \tag{1}$$

where, $x_{\max}$ and $y_{\max}$ are the maximum values among x and y coordinates, whereas, $x_{\min}$ and $y_{\min}$ are the minimum values. Each grid $I(t, s)$ is represented by t and s.

We define the capacity of grid $I(t, s)$ as n_{ts}, which represents the number of points that are located in the grid. For each point, n_{ts} is increased by one ($n_{ts}++$) after its assignment to the grid. After then, the grid structure is constructed.

An example of a grid structure is shown in Fig. 1. The original data distribution is also shown. The number of rows and columns in the grid structure are 8 and 7 respectively. The grid structure $I(n, n)$ is similar with a gray image structure, where each grid can be considered as a pixel and the intensity of the pixel is the capacity of the grid. The choice of n is important because the time complexity of the grid-based algorithm is linearly increased with n. However, a large n does not necessarily bring a good performance of the algorithm.

2.2. Grid growing

With the grid structure $I(n, n)$ constructed, a grid growing step is performed. The step begins with selecting m initial seed points on I and the initial regions begin with the exact location of these seeds. For a spatial data, high density locations indicate points of interest. Therefore, the seeds are selected based on the top m capacity values in I. There are also alternatives for selecting of seeds. A more straightforward and efficient way is to randomly select the seeds.

The regions start to grow from each seed by searching adjacent points. The search can be performed in 4-neighbors or 8-neighbors points, where the latter one brings more accurate result. If the

```
Input: $I$
Output: $R$
Initialize: $S \leftarrow SeedSelecting(I, m)$;
        $R \leftarrow \emptyset; L \leftarrow \emptyset; c = 0$;
FOR each seed in $S$ ;
    $R_c \leftarrow R_c \cup S_c; L \leftarrow L \cup S_c$;
    WHILE $L \neq \emptyset$;
        get 8-neighbor points $N_8$ of $L_c$;
        FOR each $N_8$;
            IF $N_8(i)$ is not checked &$I(i) > 0$;
                $L \leftarrow L \cup N_8(i)$;
                $R_c \leftarrow R_c \cup N_8(i)$;
            END IF;
        END FOR;
        $L \leftarrow L \backslash L_c$;
    END WHILE;
    $c \leftarrow c + 1$;
END FOR;
$R = \bigcup_{c=1}^{K} R_c$;
return $R$;
```

Doubt:
K = No. of seeds i.e. m ??
If it is so than the algo will always produce clusters equal to number of seeds???

Algorithm 2: Grid growing step

adjacent points are not visited and the point is larger than zero, the point is included into the current region. The points that have been visited will be marked and not be visited again. During the growing, candidates for growing are stored in a list L. When the current region finds no adjacent points, a new region starts. The growing step stops when all the points in I are visited. The growing generates a number of regions $R = \{R_1, R_2, \ldots, R_K\}$.

The regions obtained from the growing step are transformed to partitions $P = \{P_1, P_2, P_i \ldots, P_N\}$ based on Eq. (1), where P_i indicates the class label that the ith point belongs to.

2.3. Analysis on efficiency and effectiveness

The size of geo-spatial data obtained from a huge number of users by geo-tagged devices has been growing tremendously. The efficiency, therefore, is a crucial challenge for the methods.

K-means is known for its efficiency among the clustering algorithms, the time complexity of which is O(cKN), where K is the number of clusters, N the data size and c the number of iterations. As a typical partition-based clustering algorithm, k-means is restricted to its cluster type, i.e., Gaussian-distributed clusters. Another issue of k-means is the problem of determining the number of clusters K. The selection of K affects the final result directly. Although there have been research on how to determine the number of clusters by cluster validity indexes, it is still an open problem for users. The litekmeans [10] is an efficient MATLAB implementation on k-means, which is accelerated by matrix operations in MATLAB.

The landmark spectral clustering (LSC) [11] needs to construct a graph, which takes O(KN) and O($K^3 + K^2N$) to compute the eigenvectors. The litekmeans is used to select the landmarks in the algorithm, which takes O(KN). The pairwise random swap algorithm (PRS) [9] takes use of the similarity between two sets of centroids, and fine-tune the clustering results by a swap function and k-means. The time complexity of the algorithm is O($N + K^2$), which is affected by the data size and the number of clusters.

Greedy EM algorithm (GEM) [8] is an improved version on conventional EM (Expectation-Maximization) algorithm. It detects the clusters by estimating the parameters of *Gaussian Mixture Models* (GMM). It increases the number of components (K) by one at each iteration. The algorithm divides the data into disjoint sets, one for each component. The time complexity of the GEM is O(K^2N) or O(cKN) if $K < c$, where c is the number of candidates.

As a representative algorithm in density-based clustering, DBSCAN [15] has been employed in many areas. The basic idea of it is that every point in a data set should contain a minimum number of *MinPts* points in its neighborhood of radius ϵ, where two parameters are needed therefore. The number of *MinPts* affects the number of clusters generated by the algorithm. Although K is not so sensitive to the setting of *MinPts*, users have to decide the value of *MinPts*. The setting of ϵ is negligible, which makes the algorithm easier on the setting of parameters. The time complexity of the algorithm is O(N^2). One way to improve the time complexity is to build an index over the data set like a R^*-tree spatial index.

The time complexity on the construction of a grid structure in the Algorithm 1 is O(N). For the grid growing step, the time complexity is based on the size of I, which is n^2. As shown in Fig. 5, the size of n is better to be large enough to detect all the clusters, however, it is not necessary to be too large. For the time complexity of the growing step, it depends on the way of selecting seeds. Selecting the seeds randomly takes constant time. Other ways such as selecting the seeds from top m capacity values take O($n^2 \log n^2$). Therefore, the total time complexity of the growing step could be O($cn^2 \log n^2$), where c represents a constant number coming from the seed number and 8-neighbor search. The time complexity of the entire algorithm is O($N + cn^2 \log n^2$). Considering the size of the grid structure is controlled in a relatively small size, i.e., $n \propto \sqrt{N}$, the time complexity of the grid-growing algorithm is O($N \log N$) in worse case and O(N) in best case.

3. Experiments

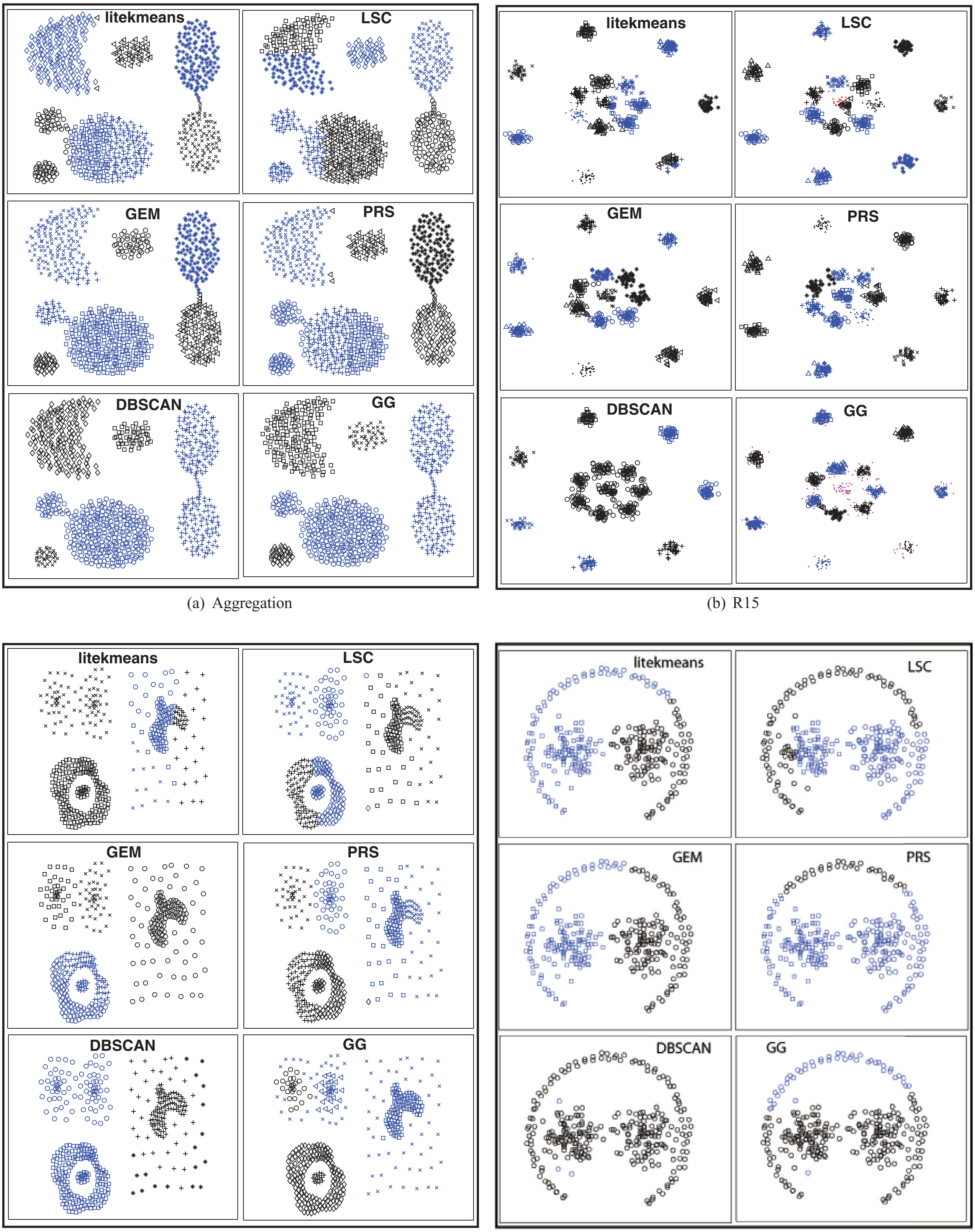
The proposed grid growing clustering algorithm is designed specifically for large data sets with GPS points or trajectories to detect places-of-interest. The efficiency of the algorithm brings competitive strength on large data sets. We consider different types of clustering algorithms: the fastest MATLAB implementation of k-means (litekmeans), Greedy EM, DBSCAN, spectral clustering (LSC) and pairwise random swap (PRS) in the experiment. Synthetic data sets and real GPS points are tested with the algorithms. The methods in the experiments are all implemented in MATLAB. The implementations of the algorithms and the tested data sets can be found here.[1]

The synthetic data sets (see Fig. 2) are 2-dimensional, which makes the validation easier from just visualization. Based on human judgment by visualization, we manually generate reference labels for the synthetic data sets to get a numeric evaluation. *Aggregation* [21] consists of seven perceptually distinct groups of points, where there are non-Gaussian clusters. Data *R15* [22] is generated as 15 similar 2-dimensional Gaussian distributions that are positioned in rings. *Compound* [23] is composed of six different structures of clusters, where there are connected clusters, noise and embedded-cluster. Data *path-based* [24] consists of a circular cluster with an opening near the bottom and two Gaussian distributed clusters inside. Each cluster contains 100 data points.

The real data sets *MOPSI-fi* and *MOPSI-joensuu* are generated through mobile devices with GPS in a project called *MOPSI*.[2] The data MOPSI-fi contains 13,467 GPS locations, which are obtained from user's trajectories by taking the start and end points and photo collections in Finland. The start and end points are point-of-interest in high probability and taking only the start and end points can reduce the sample size. The data MOPSI-joensuu has a size of 4509 GPS locations in city Joensuu, Finland. We also test the proposed algorithm on a taxi data, which comes from Wireless and Sensor Network Lab, Shanghai, Jiaotong University. Each taxi with GPS devices in Shanghai sends its GPS information every three minutes. We extract the data within a

[1] http://sse.tongji.edu.cn/zhaoqinpei/Software/.

[2] http://cs.uef.fi/mopsi.



(a) Aggregation

(b) R15

(c) Compound

(d) pathbased

Fig. 2. Clustering results (partitions) from different clustering algorithms.

time period of 5:30 pm to 6:00 pm, which is the rush hour in Shanghai, China. The data contains 29,492 GPS points from 2074 taxies.

We firstly show the results of the algorithms on the synthetic data sets to validate the proposed algorithm. In Fig. 2, we demonstrate one among 100 clustering partition from six algorithms on four synthetic data. Different clusters are represented in different symbols with colors. For data Aggregation, the DBSCAN and GG have exactly same partitions with only five clusters detected since there are two narrow "bridges" between clusters. The litekmeans, LSC and PRS produces similar results generally. Only the GEM generates seven

Table 1
The running time of the algorithms in million seconds on different data sets. The algorithm with the least running time for each data set is listed in bold.

	litekmeans	GEM	DBSCAN	LSC	PRS	GG
Aggregation	4.6	39.1	32.5	35.0	43.6	**0.5**
Compound	2.4	46.6	11.5	23.8	30.4	**0.6**
pathbased	5.3	20.3	7.0	23.4	6.6	**0.3**
R15	3.0	67.6	31.4	52.8	119.7	**0.7**
MOPSI-fi	233.1	1323.8	15551.7	1843.3	13316.3	**5.3**
MOPSI-joensuu	23.6	411.4	2062.7	1201.0	2481.7	**2.8**
Taxi	4609.9	63999.5	49150.4	NA	75621.0	**793.2**

Table 2
Numerical evaluation by external indexes on the algorithms. The highest mean values of the indexes for each data set are noted in bold. Those stable results (std = 0) are noted in italic.

Data	Index	litekmeans		GEM		DBSCAN		LSC		PRS		GG	
		Mean	std	Mean	std	Mean	std	Mean	std	Mean	std	Mean	std
Aggregation	RI	0.91	0.01	0.90	0.02	**0.95**	0.04	0.93	*0.00*	0.93	*0.00*	0.93	*0.00*
	ARI	0.72	0.05	0.69	0.07	**0.86**	0.12	0.76	0.01	0.81	*0.00*	0.81	*0.00*
	JAC	0.64	0.05	0.60	0.08	**0.81**	0.15	0.67	0.02	0.75	*0.00*	0.75	*0.00*
	FM	0.78	0.04	0.75	0.06	**0.89**	0.09	0.81	0.01	0.87	*0.00*	0.87	*0.00*
R15	RI	0.97	0.02	0.98	0.01	0.97	0.02	**0.99**	0.01	0.75	*0.00*	0.91	0.02
	ARI	0.78	0.10	0.87	0.06	0.79	0.10	**0.95**	0.07	0.26	*0.00*	0.49	0.06
	JAC	0.68	0.13	0.79	0.09	0.69	0.12	**0.92**	0.11	0.21	*0.00*	0.36	0.05
	FM	0.81	0.09	0.88	0.05	0.82	0.08	**0.96**	0.06	0.46	*0.00*	0.55	0.04
Compound	RI	0.84	0.04	0.83	0.01	0.86	0.03	0.84	0.01	0.90	*0.00*	**0.97**	*0.00*
	ARI	0.55	0.11	0.51	0.04	0.60	0.10	0.54	0.02	0.76	*0.00*	**0.92**	*0.00*
	JAC	0.49	0.11	0.45	0.03	0.54	0.10	0.47	0.02	0.71	*0.00*	**0.89**	*0.00*
	FM	0.66	0.09	0.62	0.03	0.70	0.08	0.64	0.02	0.84	*0.00*	**0.94**	*0.00*
Pathbased	RI	0.76	*0.00*	**0.83**	0.04	0.74	*0.00*	0.76	*0.00*	0.34	*0.00*	0.46	*0.00*
	ARI	0.49	*0.00*	**0.63**	0.08	0.46	*0.00*	0.49	*0.00*	0.00	*0.00*	0.08	*0.00*
	JAC	0.51	*0.00*	**0.61**	0.06	0.49	*0.00*	0.51	*0.00*	0.33	*0.00*	0.33	*0.00*
	FM	0.68	*0.00*	**0.76**	0.05	0.67	*0.00*	0.68	*0.00*	0.57	*0.00*	0.54	*0.00*

clusters. For data Compound, the GG, LSC and PRS generates six clusters, however, the DBSCAN has a different partitioning, which is composed of four clusters. The litekmeans and GEM generate five clusters. Only the DBSCAN generates eight clusters for R15, the rest of the algorithms can detect 15 clusters. All of the algorithms fail to detect the clusters for data pathbased.

Unlike partition-based (k-means and its variants) and model-based clusterings (EM), the grid-based clustering (GG) can detect non-shaped clusters. Also, the grid-based algorithm includes automatic outlier detection. Those data points far from the main cluster body are regarded as outliers. Compared to density-based clustering (DBSCAN), the GG is capable of producing more sensible result.

The algorithms are performed on the data sets 100 times to study the average performance of them. To compare the algorithms numerically, the clustering labels of the four synthetic data sets from the algorithms are compared to the reference labels by external validity indexes. External validity indexes measure how well the results of a clustering match the ground truth (if available), a reference or another clustering [25,26]. We compare the performance of the algorithms by external indexes such as rand index (*RI*), adjusted rand index (*ARI*), Jaccard coefficient (*JAC*) and Fowlkes and Mallows index (*FM*) in Table 2. The index values are ranged in $[0, 1]$, where one indicates a completely match. The average values (*mean*) and standard deviation values (*std*) of 100 runs are listed in the table. As it is shown, there is no algorithm that is best for all the four data sets. The proposed algorithm has a best performance on the data Compound. According to the std values, the DBSCAN and GG produce same results in each run for most of the data. In other words, the two algorithms are more stable.

For the efficiency, as shown in Table 1, the proposed algorithm is a lot faster than the other algorithms. The GEM, DBSCAN, LSC and PRS are almost at the same scale level. Restricted by the memory issue in MATLAB, the LSC algorithm cannot generate any result on the taxi data. For real data set MOPSI-fi and MOPSI-joensuu, the DBSCAN takes much more time than GEM. The results indicate that the time complexity of DBSCAN is affected by the data size heavily, which makes the DBSCAN not proper for real applications with large size. The GEM has better performance when the data gets larger. However, it is still not fast enough for real data. The litekmeans is claimed as the fastest implementation of k-means, which is proved to be quite efficient. However, it is still one scale level more than the proposed algorithm.

We get 49 clusters for the taxi data by the GG algorithm. Each cluster reflects the places with high population and traffic. As shown on Fig. 3, the clusters are mainly the CBDs in city Shanghai. The airports and railway stations are also the places with taxies. The clustering result reflects the city's traffic information.

The original MOPSI data contains not only trajectories, it includes comments in text also. To detect the places-of-interest, the geo-spatial data with only location information needs to be clustered. The comments in text thereby can be employed to generate a tag for the places. To demonstrate the clustering result in a clear way, we extract one user's data in city Joensuu, Finland, which is shown in Fig. 4. We display the generated clusters into green ellipses with a google marker as the centroid. The high efficiency of the proposed algorithm makes it applicable in a real application. Since the tag generation based on texts is out of scope in this paper, we tag the clusters manually with labels “home”, “work place” and “sports”. In our future work, the tagging step will be performed automatically.

In the proposed algorithm, there is a step of selecting seeds for growing. Two aspects need to discuss in the experiment. One is the method of selecting seeds. We introduce that the selection method is based on the top m capacity values. An intuitive way is to select the seeds randomly. The two methods are compared in the experiment and we use *random* to represent the random method and *max* for the top m capacity. The other aspect is the strategy for seed growing. We compare the 8-neighbor (N_8) and 4-neighbor (N_4) growing strategies. As shown in Table 3, the *random* way of selecting seeds

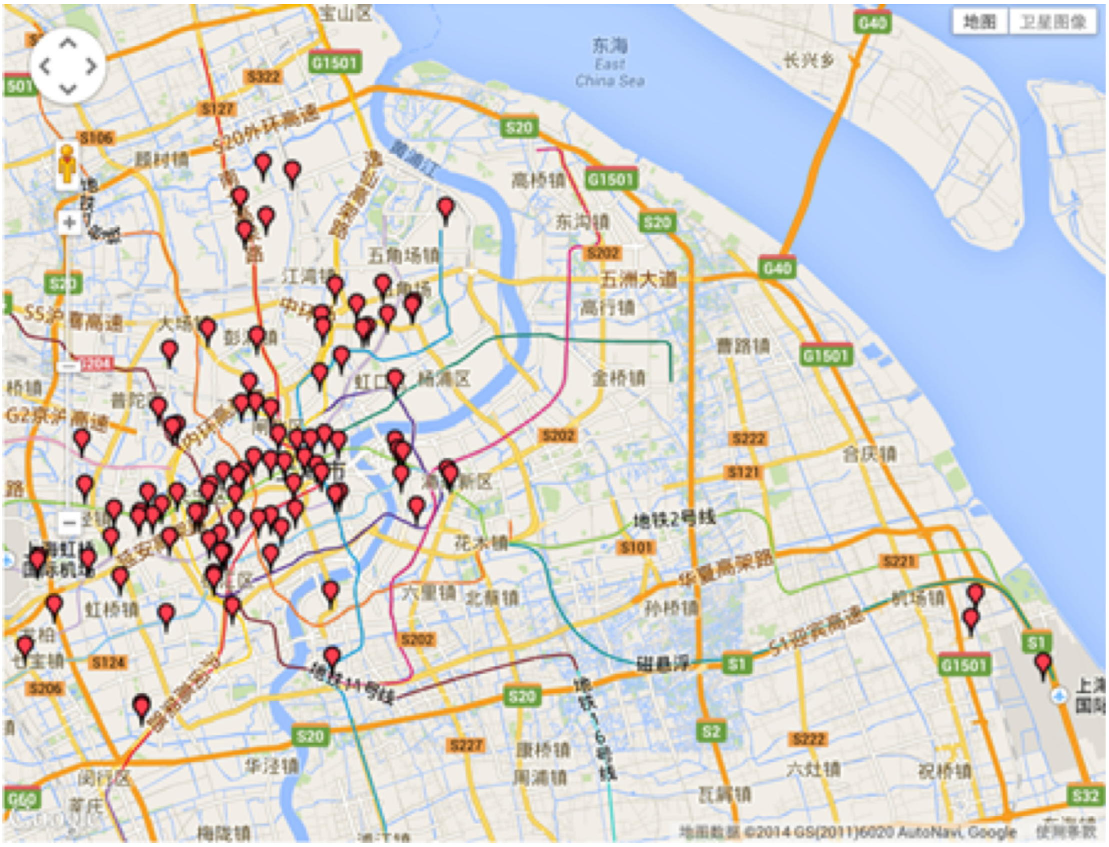


Fig. 3. The clusters obtained from the proposed algorithm on data Shanghai taxi.

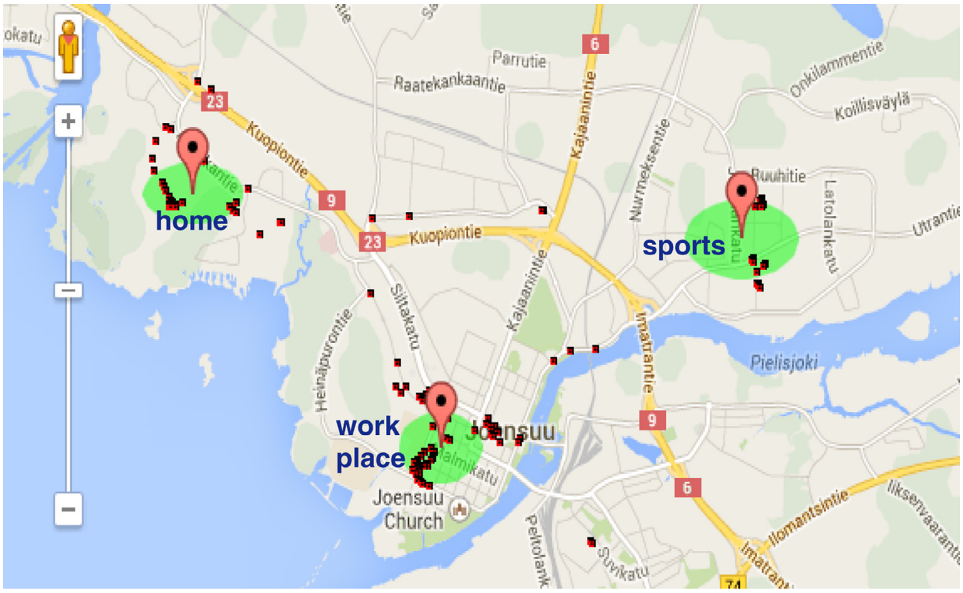


Fig. 4. Three clusters obtained from the clustering algorithm on one user's data.

is faster. However, from the observation on the clustering result, the random way generates much less clusters than the *max* way. A large number of seeds might help on the situation. However, it brings the difficulty to control the number of seeds that should be selected. When the 4-neighbor searching is employed, it is also true that the efficiency has been improved because less points have been checked. However, the algorithm sometimes generates more clusters when it is 4-neighbor searching. The algorithm employing 4-neighbor searching labels more points not connecting to others as outliers, which increases the number of clusters. In general, the seed selection method should be adapted by applications. Since the 4-neighbor growing brings little improvement on efficiency, we suggest to use the 8-neighbor growing. In this paper, the combination of the *max* selection and 8-neighbor growing is the best choice.

Since there are two parameters e.g., the size of the grid structure (n) and seed number for growing (m), involved in the proposed algorithm, an experiment is performed in Fig. 5. The GG algorithm is run on data Compound with different settings on the number of seeds m

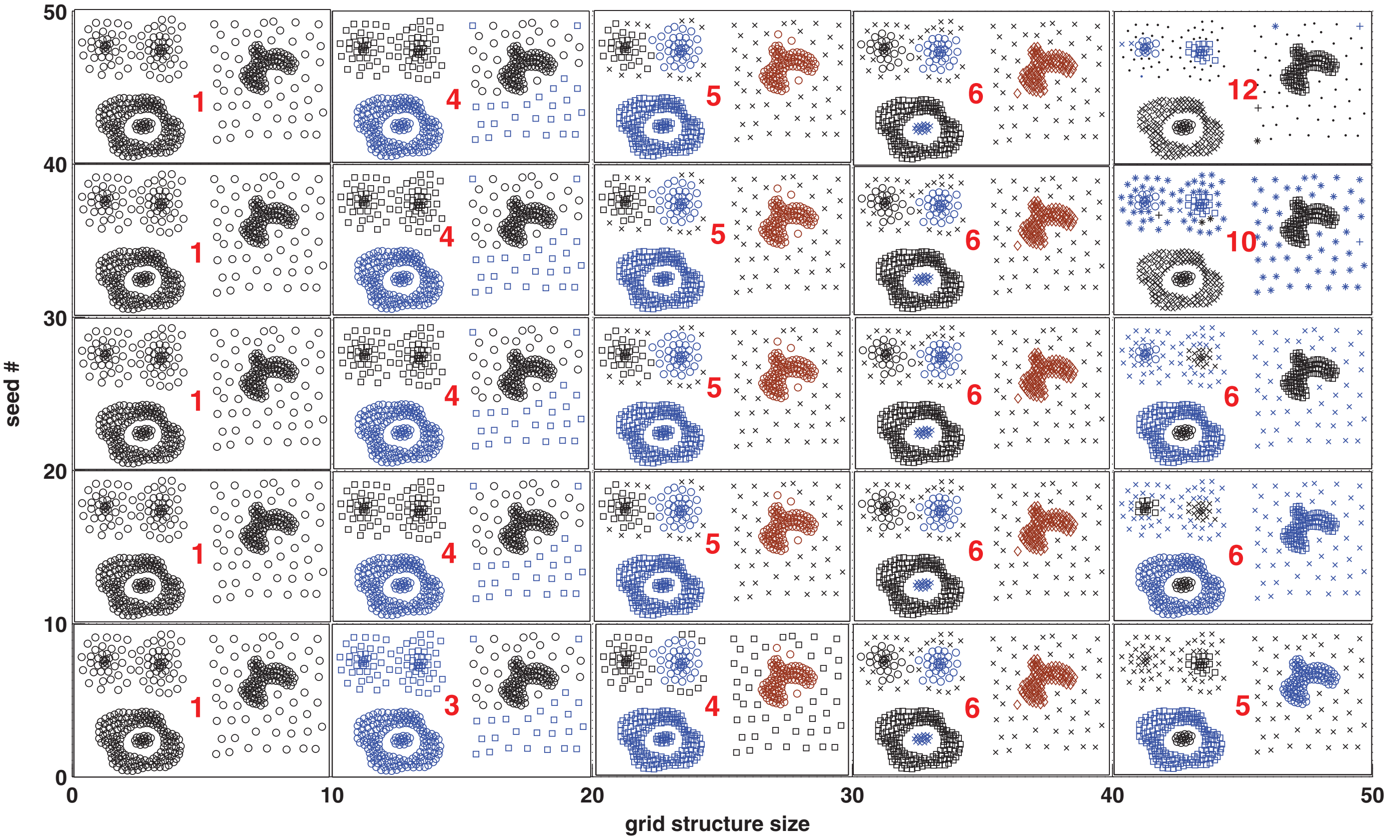


Fig. 5. How the parameters setting in grid-growing clustering algorithm affects the final clustering? The size of the grid structure n and number of seeds m are studied.

Table 3
How the seed selection method and neighborhood searching strategy affect the final result?

	Random, N_8	Max, N_8	Max, N_4
Aggregation	0.5 ($K = 5$)	0.6 ($K = 5$)	0.5 ($K = 6$)
Compound	0.4 ($K = 5$)	0.5 ($K = 6$)	0.5 ($K = 6$)
pathbased	0.4 ($K = 2$)	0.4 ($K = 2$)	0.4 ($K = 2$)
R15	0.4 ($K = 3$)	0.8 ($K = 15$)	0.7 ($K = 24$)
MOPSI-fi	1.9 ($K = 9$)	5.3 ($K = 19$)	5.1 ($K = 22$)
MOPSI-joensuu	1.0 ($K = 9$)	2.9 ($K = 14$)	2.8 ($K = 16$)

and the size of grid structure n. The values of m and n are set as 10, 20, 30, 40 and 50 pairwisely. The partitions obtained from the proposed algorithm with different m and n are shown and a number on each clustering result indicates the number of clusters obtained finally. It seems that the parameter m in the growing step affects less than n on the final clustering result. With certain n, for example $n = 20$, the clustering result has no change when $m > 20$. When n changes, the clustering result also changes even with fixed m. It can be concluded that the larger n brings more sensible result, however, it is not necessary to set the n as large as possible. For example, when $n = 40$ and $n = 50$, the clustering results reach quite similarly with certain setting of m. Therefore, it is the first priority to set up n and then is m. Since the efficiency of the algorithm is controlled by the setting of n, the knowledge that setting high values on n hel ps little on the result will cheer users up.

4. Conclusion

A grid-growing clustering algorithm is proposed in this paper, which possesses the advantages of both k-means and DBSCAN. The algorithm needs no settings on the number of clusters. Similar to DBSCAN, it is not restricted to the cluster shape and it can detect the outliers at the same time. The most important point is that its time complexity is $O(N \log N)$. The algorithm is much more efficient than other clustering algorithms, litekmeans, LSC, PRS, Greedy EM and DBSCAN, which are shown to be restricted in real applications because of their running time on large data sets. Considering the effectiveness and efficiency obtained from the proposed algorithm, it is more suitable in real applications. Therefore, it has been employed on geo-spatial data for detecting regions of interest. Manual tagging on the regions of interest in the experiment has been done. We will work on conceptualization from short texts and tag the regions of interest automatically in our future work.

Acknowledgments

The authors in this work are sponsored by the Fundamental Research Funds for the Central Universities, the National Natural Science Foundation Committee of China under contract no. 61202382 and Science and Technology Commission of Shanghai Municipality, P.R. China (no. 12510706200) respectively.

References

[1] M. Chen, M. Xu, P. Fränti, Compression of GPS trajectories, Data Compression Conf. (2012) 62–71.
[2] N. Elmqvist, J. Fekete, Hierarchical aggregation for information visualization: overview, techniques and design guidelines, IEEE Trans. Vis. Comput. Graph. 16 (3) (2010) 439–454.
[3] I. Lee, G. Cai, K. Lee, Exploration of geo-tagged photos through data mining approaches, Expert Syst. Appl. 41 (2014) 397–405.
[4] V. Zheng, Y. Zheng, X. Xie, Q. Yang, Collaborative location and activity recommendations with GPS history data, Int. Conf. World Wide Web (2010) 1029–1038.
[5] A. Tietbohl, V. Bogorny, B. Kuijpers, L. Alvares, A clustering-based approach for discovering interesting places in trajectories, in: Proceedings of the 2008 ACM Symposium on Applied Computing, 2008, pp. 863–868.
[6] Y. Zheng, X. Xie, W. Ma, Understanding mobility based on GPS data, in: Proceedings of the 10th International Conference on Ubiquitous Computing, 2008, pp. 312–321.

[7] M. Lin, W. Hsu, Mining GPS data for mobility patterns: a survey, Pervasive Mobile Comput. (2013).

[8] J. Verbeek, N. Vlassis, B. Krose, Efficient greedy learning of Gaussian mixture models, Neural Comput. 15 (12) (2003) 469–485.

[9] Q. Zhao, P. Fränti, Centroid ratio for a pairwise random swap clustering algorithm, IEEE Trans. Knowl. Data Eng. 26 (5) (2014) 1090–1101.

[10] D. Cai, Litekmeans: the fastest matlab implementation of kmeans, Available at: http://www.zjucadcg.cn/dengcai/Data/Clustering.html, 2011 (accessed 08.14).

[11] X. Chen, D. Cai, Large scale spectral clustering with landmark-based representation, AAAI (2011) 1090–1101.

[12] D. Ashbrook, T. Starner, Using GPS to learn significant locations and predict movement across multiple users, Personal Ubiquitous Comput. (2003) 275–286.

[13] Q. Cao, B. Bouqata, P. Mackenzie, D. Messier, J. Salvo, A grid-based clustering method for mining frequent trips from large-scale, event-based telematics datasets, Int. Conf. Syst. Man Cybernet. (2009) 2996–3001.

[14] L. Alvares, V. Bogorny, B. Kuijpers, J. de Macedo, B. Moelans, A. Vaisman, A model for enriching trajectories with semantic geographical information, Annu. ACM Int. Symp. Adv. Geogr. Inf. Syst. (2007) 1–8.

[15] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, Int. Conf. Knowl. Discov. Data Mining (1996) 226–231.

[16] X. Xu, M. Ester, H. Kriegel, J. Sander, A distribution-based clustering algorithm for mining in large spatial databases, Int. Conf. Data Eng. (1998) 324–331.

[17] S. Kisilevich, F. Mansmann, D. Keim, P-dbscan: a density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos, Int. Conf. Exhibit. Comput. Geospatial Res. Appl. (2010) 1–4.

[18] R. Montoliu, D. Gatica-Perez, Discovering human places of interest from multimodal mobile phone data, Int. Conf. Mobile Ubiquitous Multimedia (2010) 1–10.

[19] N. Lin, C. Chang, H. Chuen, H. Chen, W. Hao, A deflected grid-based algorithm for clustering analysis, WSEAS Trans. Comput. 7 (4) (2008) 125–132.

[20] W. Wang, J. Yang, R. Muntz, STING: a statistical information grid approach to spatial data mining, Int. Conf. Very Large Data Bases (1997) 186–195.

[21] A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, ACM Trans. Knowl. Discov. Data 1 (1) (2007) 1–30.

[22] C. Veenman, M. Reinders, E. Backer, A maximum variance cluster algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 24 (9) (2002) 1273–1280.

[23] C. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Trans. Comput. 100 (1) (1971) 68–86.

[24] H. Chang, D. Yeung, Robust path-based spectral clustering, Pattern Recognit. 41 (1) (2008) 191–203.

[25] Q. Zhao, M. Xu, P. Fränti, Expanding external validity measures for determining the number of clusters, Intl. Conf. Intell. Syst. Des. Appl. (2012) 931–936.

[26] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. Perez, I. Perona, An extensive comparative study of cluster validity indices, Pattern Recognit. 46 (1) (2013) 243–256.