Article

An adaptive Sweep-circle Spatial Clustering Algorithm Based on Gestalt

Qingming Zhan[1], Shuguang Deng[1,2,*] and Zhihua Zheng[3]

[1] School of Urban Design, Wuhan University, 129 Luoyu Road, Wuhan 430079, China; qmzhan@whu.edu.cn;dsgsos@foxmail.com
[2] Department of civil and Surveying Engineering, Guilin university of Technology at Nanning,15 Anji Road,Nanning 532100,China;dsgsos@foxmail.com
[3] Land and Resources Information Center of Guangxi Province,2 Zhongxin Road,Nanning, 530028, China; zheng_zhihua@foxmail.com
* Correspondence: dsgsos@foxmail.com; Tel.: +86-134-0771-2608

Abstract: An adaptive spatial clustering (ASC) algorithm is proposed that employs sweep-circle techniques and a dynamic threshold setting based on Gestalt theory to detect spatial clusters. The proposed algorithm can automatically discover clusters in one pass, rather than through the modification of the initial model (for example, a minimal spanning tree, Delaunay triangulation, or Voronoi diagram). It can quickly identify arbitrarily shaped clusters while adapting efficiently to non-homogeneous density characteristics of spatial data, without the need of priori knowledge or parameters. The proposed algorithm is also ideal for use in data streaming technology with dynamic characteristics flowing in the form of spatial clustering large data sets.

Keywords: spatial clustering; sweep-circle; Gestalt theory; data stream

1. Introduction

Rapid advancements in geographic spatial information technology, generation, and collection have created exponential growth in spatial data, resulting in increasingly complex data structures. It is more than ever necessary to address the challenges involved in extracting useful information and knowledge from large-scale, highly complex, spatial data sets. Data mining from the spatial data set is a valuable way to obtain valuable information; spatial clustering has played an extremely indispensable role in spatial data mining research. Clustering is the process of grouping spatial data objects into a series of meaningful clusters so that objects within a particular cluster shares similarities, while being dissimilar to other clusters[1,2]. Spatial point clustering has been applied to a wide variety of fields, including urban planning, remote sensing, geographic information, bio-engineering, geology and minerals, as well as computer science [3-5]. Current spatial clusterings have been roughly classified into the following categories:

- Partitioning methods, e.g., K-Means[6] and K-Medoids[7].
- Hierarchical methods, e.g., CURE[8], BIRCH[9], and CHAMELEON[10].
- Density-based methods, e.g., DBSCAN[11], OPTICS[12], and DENCLUE[13].
- Graph-based methods, e.g., ZEMST[14], AUTOCLUST[15], and SMTIN[16].
- Grid-based methods STING[17] and WaveCluster[18].
- Model-based methods, e.g., EM[19], COBWEB[20] and SOM[21].
- Hybrid methods and large data set methods CLIQUE[22],NN-Density[23],and ACODF[24].

These traditional approaches have been successful in managing a number of specific applications across different domains, but significant limitations exists. Most traditional clustering methods rely on user-specified arguments or a priori knowledge, and cannot manage clusters of irregular shapes or of different sizes, and are not effective in sets with non-uniform inner density, outliers, or noise. In fact, no particular clustering method has been shown to be superior to its competitors among all the necessary aspects[25,26]. To date, the advantages and disadvantages of various algorithms have been extensively analyzed [26-31].Some analysis of the classical spatial clustering algorithms are summarized in Table 1.

Relation to geographical space and large amounts of data are common characters of data. The spatial object is highly complex requires high correlation. This demands that clustering algorithms be highly efficient. Efficient spatial clustering algorithms are valuable for many real-world, dynamic applications[32]. Large data sets are challenging for computational systems when processed with conventional algorithms, particularly as the amount of spatial data increases exponentially in the real world. Popular traditional clustering algorithms require repeated access to the data set, as well as multiple clustering operations, which decreases their efficiency as data set size increases[27] [5,33]. This paper proposes an adaptive spatial clustering algorithm (ASC) that employs both sweep-circle techniques and a dynamic threshold setting based on Gestalt theory to detect spatial clusters. Empirical results and comparison demonstrated that the proposed ASC can automatically discover clusters in one pass, rather than modifying the initial model. A minimal spanning tree, Delaunay triangulation, or Voronoi diagram, can be quickly identified even with arbitrarily shaped clusters. The proposed ASC can identify the non-homogeneous density characteristics of spatial data without the need for a prior knowledge or parameters. It is compatible with streaming dynamic, large-scale data found in spatial clustering.

The remainder of this paper is organized as follows: In Section 2, the relation of ASC to previous methods is described. In Section 3, the proposed algorithm is explained in detail. Section 4 describes the ASC-based streaming process as-applied to large data sets. Section 5 reports our analysis of the algorithm including its time complexity and comparison against other clustering methods. Section 6 provides an example of the proposed algorithm applied to a real-world data set, and Section 7 concludes with an outlook for further research.

2. Related work

2.1 Plane-sweep techniques

The plane-sweep is a popular acceleration technique used to solve 2D Euclidean space geometric problems[34]. This technique initially sorts the geometric elements, then it is imagined that a sweep-line glides over the plane and stops at geometric elements (typically called "event points")[35], where the corresponding data structure is then updated. The plane-sweep method cannot move backwards across the event points.

The sweep-plane technique was initially applied to computational geometry problems[36]. Shamos and Hoey later applied a unidirectional sweep-plane algorithm that used time O ($n\log n$) to determine whether or not a finite number of line segments are any two intersect in a plane[37].

Table 1. Comparisons of some classical spatial clustering algorithms

Based on category	Typical algorithm	Shape of suitable data set	Discovery of clusters with even ensity	Scalability	Requirement of prior knowledge	Sensitive to noise/out	For large-scale data	Complexity (times)
Partition	K-means	Convex	No	Middle	Yes	Highly	Yes	Low
	CLARANS	Convex	No	Middle	Yes	Little	Yes	High
Hierarchy	BIRCH	Convex	No	High	Yes	Little	Yes	Low
	CURE	Arbitrary	No	High	Yes	Little	Yes	Low
	CHAMELEON	Arbitrary	Yes	High	Yes	Little	No	High
Density	DBSCAN	Arbitrary	No	Middle	Yes	Little	Yes	Middle
	OPTICS	Arbitrary	Yes	Middle	Yes	Little	Yes	Middle
	DENCLUE	Arbitrary	No	Middle	Yes	Little	Yes	Middle
Graph theory	MST	Arbitrary	Yes	High	Yes	Highly	Yes	Middle
	AMEOBA	Arbitrary	Yes	High	No	Little	No	Middle
	AUTOCLUST	Arbitrary	Yes	High	No	Little	No	Middle
Grid	STING	Arbitrary	No	High	Yes	Little	Yes	Low
	CLIQUE	Arbitrary	No	High	Yes	Moderately	No	Low
	WaveCluster	Arbitrary	No	High	Yes	Little	Yes	Low
Model	EM	Convex	No	Middle	Yes	Highly	No	Low

Bentley and Ottmann extended this algorithm to determine the existence of intersecting line segments, but were also able report all k intersections of n line segments within time O ((n+k)log$n$), where k is the number of intersections[38]. The sweep-line algorithm was also used to construct a Voronoi diagram, i.e., dual Delaunay triangulation[39]. The Delaunay algorithm examined in this study is based on plane scattered point sets used by Žalik[35] and Zhou[40]. Žalik was the first to suggest the use of sweep-line techniques for spatial clustering[41].

2.2 Sweep-circle algorithm

The sweep-circle is another important sweep-line technique, where points are initially sorted according to their distances from a fixed pole O in the convex hull of S. It is assumed there is a circle C centered at O, with radius increasing from 0 to +∞, which stops at event points and updates the data structure. A part of the problem being swept (inside the circle) is already solved, while the remaining part (out of the circle) is unsolved.. Dehne and Klein[42] were the first to use a circle that emanates from a fixed point resulting in a Voronoi. Adam[43] and Biniaz and Dastghaibyfard (2012) suggested that the incremental sweep-circle algorithm is better suited to constructing Delaunay triangulations[42] (Figure 1).

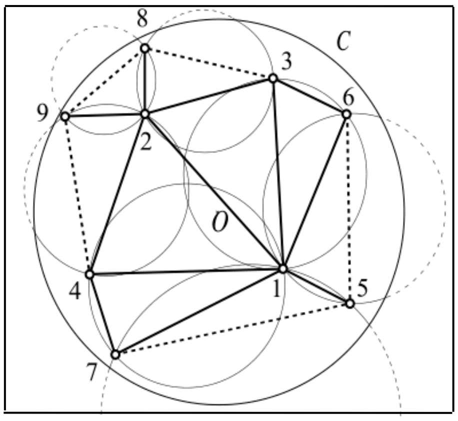

Figure 1. Incremental sweep circle algorithm constructs Delaunay triangulation

2.3 Data stream technique

The "data stream" is an unbounded orderly sequence of information, which can consecutively arrive in large quantities, however this technique can only process data sequentially with appropriate access. Data mining algorithms based on data streaming techniques are commonly used in satellite remote sensing, geographic information, network monitoring, and financial services. Traditional typical spatial clustering algorithms that repeatedly access the entire data sets repeatedly cannot be readily applied to data streaming, as their high complexity and computational cost renders them unable manage such a large amount of data. In fact, data stream clustering algorithms have become important from within data mining research and there have been many algorithms based on data stream technology proposed, including the commonly-used one-pass algorithm[44] [9] [45-48]. This algorithm divides the non-stream data sets into data blocks so as to fit requirements of the memory space and one-pass sweeping data objects. The traditional clustering algorithm can be applied to the data-streaming environment once the data blocks have arrived from the data stream[49]. For example, K-Means and K-Medians algorithms[44] can be used to process large data sets, and the Squeezer algorithm can allocate the data into similar globes before clustering using one-pass sweeping[46]. The BIRCH algorithm uses a clustering feature tree to minimize I/O requests prior to one-pass sweeping for clustering[9]. Guha et al. also conducted valuable research on the one-pass algorithm using similar data sets[44] [47].

2.4 Sweep-line clustering algorithm

Žalik (2009) proposed an innovative, agglomerative hierarchical clustering algorithm for spatial data using a sweep-line in O ($n$log$n$) time in the worst case. This algorithm does not rely on domain knowledge or modification of the initial model, and can determine clusters of arbitrary shapes while completing spatial clustering of large data sets. In this algorithm, there are horizontal sweep-lines S_1 and S_2, where the distance from S_1 to the front of S_2 is d. It is assumed that S_1 sweeps the p_{i-1} set points in accordance to the proximity parameter d to form part of the clusters, and the points of front line (AF) are sorted in accordance to the x coordinates. When S_1 encounters point p_i, p_i is projected to the frontier toward AF, and a cluster is found by comparing the distance from p_i to p_l and p_i to p_r using proximity parameter d (Figure 2). When S_1 moves to the next point, S_2 follows it at distance d. The points that have been swept by S_2 are removed from the AF. If the projection misses the AF (that can also be empty), the corresponding end-point of the AF is tested to determine it is close enough to point p_i to discover a new cluster[41]. It is difficult to determine the global parameter d that accounts for uneven distribution of data sets. If the parameters are set without a priori knowledge (or experimental measurement result) it is difficult to find true clusters accurately.

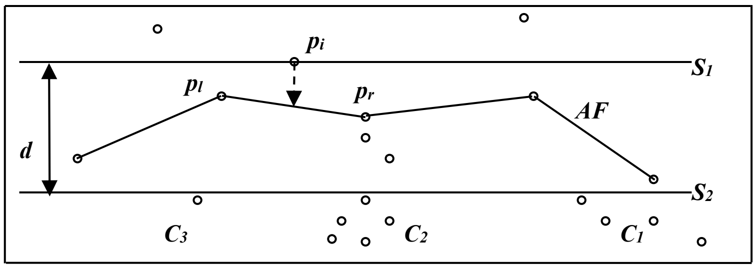

Figure 2. Sweeping the points by the sweep- lines

3. ASC algorithm

In the polar coordinate system where the ASC algorithm is applied, $\boldsymbol{p}_i$ is swept from the initial frontier outward in accordance with the increasing distance from pole O, i.e., the sweep-circle center. $\boldsymbol{p}_i$ is projected onto the segment of frontier edge ($\boldsymbol{p}_l\boldsymbol{p}_r$) along the circle in O direction (Figure 3). According to Tobler, the first law of geography is that "Everything is related to everything else, but near things are more related than distant things"[50]. The points are considered to be similar if the points are within a specific distance of each other, such as points $\boldsymbol{p}_i$ and $\boldsymbol{p}_l$ or $\boldsymbol{p}_i$ and $\boldsymbol{p}_r$,as shown in Figure 3. These values fall under a threshold value used to determine the formation of clusters.

The algorithm proposed in this paper utilizes Gestalt theory and the associated definition of the dynamic adaptive threshold. It can efficiently locate the adaptive clusters of arbitrary shapes and can acclimate to the uneven density characteristics of spatial data to avoid necessitating preset global parameters such those necessary for DBSCAN, DENCLUE, and other algorithms[41]. ASC works in a four-phase process: basic conceptualization, initialization, clustering, and cluster merging.

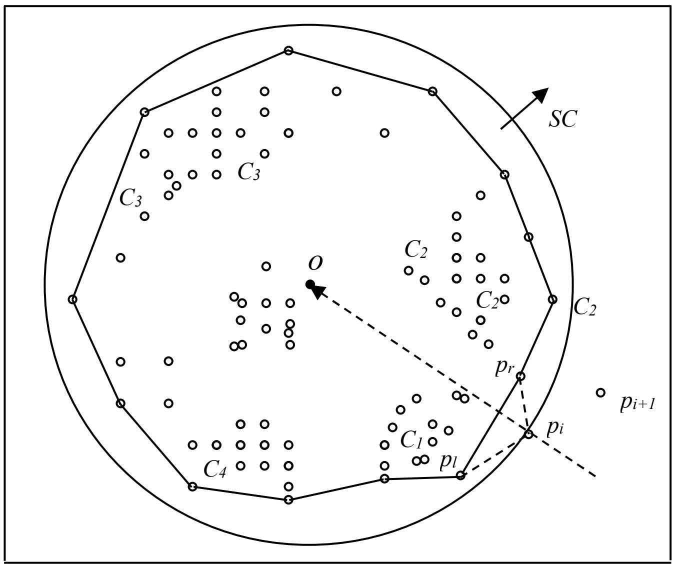

Figure 3. Sweeping the data set to obtain clusters

3.1 Basic concepts and initialization

Cluster definitions. Given *n* collection of discrete points S=｛p_1,p_2,p_3,···,p_n｝ on 2D set ($\boldsymbol{R}^2$), using the degree of similarity between data points, where data set then divides S into k clusters C=｛C_1,C_2, …,C_k｝ $C_k \subseteq S$ to define the cluster. Where: $\bigcup_{i=1}^{k} C = S$, $C_i \cap C_j = \varnothing$ $(i \neq j)$, With a cluster of objects with high similarity, different clusters of objects with high dissimilarity.

Determining the center of the sweep-circle. S corresponds to the coordinate set｛p_1 (x_1,y_1), p_2 (x_2,y_2), p_3 (x_3,y_3),···, p_n (x_n,y_n)｝, where the origin of the polar coordinate O (p_x,p_y) is the center of the sweep-circle. Select O (p_x,p_y) as the average of the largest (x_{max}, y_{max}) and smallest (x_{max}, y_{max}) values of input S.

Calculating the polar coordinates of input points and sorting. The polar coordinates of input points are calculated and sorted by increasing distance from O as follows:

$$r_i = \sqrt{(x_i - p_x)^2 + (y_i + p_y)^2} \tag{1}$$

$$\theta = \begin{cases} \arccos(\dfrac{x_i - p_x}{r_i}) & if\,(y_i - p_y) > 0 \\ \pi + \arccos(\dfrac{x_i - p_{x)}}{r_i}) & if\,(y_i - p_y) < 0 \end{cases} \tag{2}$$

Each point p_i (x_i,y_i) in the Cartesian coordinates can be transformed to p_i (r_i,θ_i), where the points are sorted according to their r-coordinate found in the polar coordinates. If two points have the same r-coordinate, they are sorted by the secondary criterion θ . In a special case where the first point coincides with the origin O (i.e., its r-coordinate is zero), the point is removed from the list.

Constructing the initial frontier and clusters. The three points located nearest to center O are used to form a triangle where it is assumed that the three points are nonlinear. The three edges of the triangle form a polyline referred to as the frontier. Any spatial clustering algorithm should work based on various distances such as the Euclidean distance, the Manhattan distance, or the Minkowski distance. This algorithm uses the Euclidian distance between data points to measure the distance needed for spatial clustering. Figure 4 shows an example of the nearest three points to the center O that forms the initial cluster.

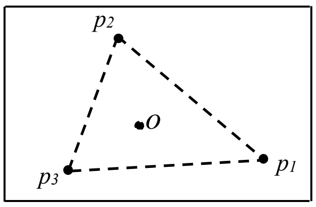

Figure 4. Initial frontier

Clustering the threshold. The threshold setting $\mathcal{E}$ serves is the distance measurement of $\boldsymbol{d}$ ($\boldsymbol{p}_i$,$\boldsymbol{p}_j$) ,which determines if two points are grouped into the same cluster. If the distance between the two points is less than or equal to this value, they belong to the same cluster; otherwise, they do not. This is calculated as follows:

$$Cp = \cup \{ p_i, p_j \} \mid \boldsymbol{d}(p_i, p_j) \le \mathcal{E},\ (j \ne i, p_i, p_j \in S) \qquad (3)$$

The clustering process for global threshold setting $\mathcal{E}$ is sensitive to density changes, particularly internal density changes within the clusters. To manage the gradual changes of the local density, the fact that the spatial data mining process obeys not only the objective law of the geographical entity itself but also relates to the concept of recognition in cognitive psychology, specifically, the Gestalt theory was taken into account.

The Gestalt theory summarizes the cognitive law of human vision, and the pattern organization discipline generated by the gestalt principle have been applied to pattern recognition and spatial clustering [14].The main principle of the Gestalt perception model is "the whole is greater than the sum of its parts," suggesting that people tend to perceptually recognize structural integrity and can initially observe the visual object as a whole, and then brake the object down into partials[14]. Gestalt is interpreted through principles of visual recognition, such as proximity, similarity, closure, continuity, orientation, and common fate[51]. We combined a subset of Gestalt principles operating simultaneously to build the dynamic adaptive threshold model $\mathcal{E}$ ($\boldsymbol{p}_i$). In this model, each new event point $\boldsymbol{p}_i$ is processed to correspond to the threshold according to the following three Gestalt principles:

- Proximity, where objects placed close together tend to be perceived as a group.
- Continuity, where spatial objects arranged in a logical order are easily perceived as a group or a continuous graph.
- Closure, where the observer tends to prioritize closeness and "perfection" of objects, so gaps between objects may be perceived as being filled to create a unified whole.

Spatial clustering should be consistent with visual psychology principles and spatial cognition from simple to complex. In agreement with Tobler's First Law of Geography, proximity is important within spatial clustering (and is also the basis of continuity and closure). The easier it is to form continuity and closure among spatial data, the greater the similarity. Accordingly, when event point $\boldsymbol{p}_i$ is projected onto the frontier toward O, the triangle including the frontier is identified and we can define the mean of the triangle's perimeter L_i as adaptive dynamic threshold $\mathcal{E}$ ($\boldsymbol{p}_i$):

$$\mathcal{E}(p_i) = 1/3\, \alpha\, L_i \quad (i>3) \qquad (4)$$

where α is a constant factor. We can enlarge or reduce $\mathcal{E}(p_i)$ by manipulating the value of α, although this may affect the quality of clusters and reflect the hierarchical relation. The value of α is usually set to 1 within ASC.

The three Gestalt principles operate simultaneously within ASC to build the dynamic adaptive threshold model $\mathcal{E}(p_i)$ as shown in Figure 5. Input point p_4 is an event point projected on the edge (p_2,p_3) of triangle $\triangle p_1p_2p_3$ toward O, where p_4 is closest to p_2 and p_3 that results in the formation of a cluster due to the rule of proximity. Under the closure rule, p_2 and p_3 combine with p_4 to form a simple triangle $\triangle p_4p_2p_3$ adjacent to $\triangle p_1p_2p_3$ with a common edge (p_2p_3). Both proximity and continuity Gestalt clusters occur at triangle $\triangle p_4p_2p_3$ and $\triangle p_1p_2p_3$, which maintain closely related spatial properties. Therefore, p_4's distance from p_2,p_3 is used to form a cluster, where the mean of perimeter (L_4) of triangle $\triangle p_1p_2p_3$ forms the adaptive dynamic threshold $\mathcal{E}(p_4)$. When a new event point(p_5) is obtained, the mean of perimeter (L_5) of triangle $\triangle p_1p_2p_3$ serves as the adaptive dynamic threshold $\mathcal{E}(p_5)$.

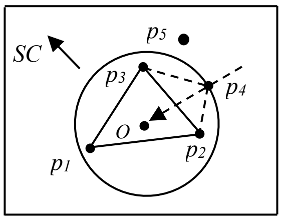

Figure 5. Adaptive dynamic threshold example

Thresholds like these are often set in similar real-world applications for use in situation-specific guidelines for users. For example, during urban planning, the threshold value is set according to the minimum radius of the public service area being covered. The threshold can vary, still allowing for the analysis of the distribution of buildings in residential, commercial, or industrial areas, as well as reflecting the hierarchical structure of the relationships among different structures.

3.2 *Clustering*

In a system where the sweep-circle SC has already passed the first three points and assigns them to one cluster, an algorithm surrounds the points by single-closure bordering polylines (i.e., the frontier) as shown in Figure 6a. When SC increases and sweeps to the new point p_i, the projection of p_i hits the edge (p_l, p_r) of the frontier toward O. This manner of projection will typically hit the frontier, since the O lies inside the frontier and new points lie outside of it. By connecting p_i and p_l, and p_i and p_r, the distances $dist(p_i,p_l)$ and $dist(p_i,p_r)$ are calculated, where the threshold $\mathcal{E}(p_i)$ can be set accordingly. According to Eq. (3) there are four possibilities when moving forward:

- $dist(p_i,p_l) > \mathcal{E}(p_i)$ and $dist(p_i,p_r) > \mathcal{E}(p_i)$, where p_i is the first element of a new cluster.
- $dist(p_i,p_l) > \mathcal{E}(p_i)$ and $dist(p_i,p_r) \le \mathcal{E}(p_i)$, where the right side of p_i is assigned to a cluster (C_r) (Figure 6b).
- $dist(p_i,p_l) \le \mathcal{E}(p_i)$ and $dist(p_i,p_r) > \mathcal{E}(p_i)$, where the left side of p_i is assigned to a cluster (C_l) (Figure 6b).
- $dist(p_i,p_l) \le \mathcal{E}(p_i)$ and $dist(p_i,p_r) \le \mathcal{E}(p_i)$, where if p_l and p_r are members of the same cluster, then p_i is placed into the same cluster; otherwise, p_i is a merging point between left and right clusters[41].

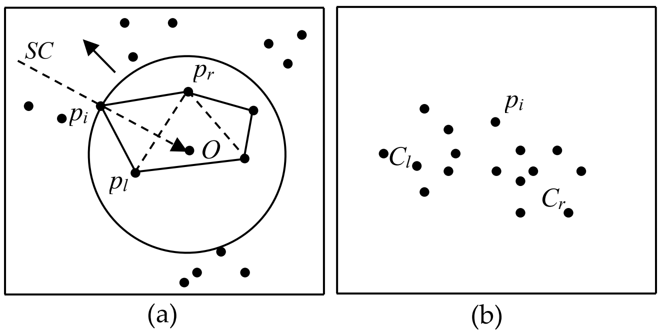

Figure 6. ASC algorithm cluster basics:(a)sweep the points;(b)2 clusters are obtained

The frontier plays an important role in the process of discovery of clusters.In order to effectively implement of frontier, a heap or balanced binary search trees (e.g., AVL tree, B-tree, Read-Black tree) can be often selected. In our case, a simple hash-table on a circular double-linked list is used to implement the algorithms efficiently and to ensure that large data sets were manipulated correctly (Figure. 7).Each record of the frontier stores the key, vertex index P_i, the index of the triangle T_i sharing its edge with the frontier (generating an adaptive threshold), and the generated initial clustered index C_i.Fortunately, ASC will not appear projection missed frontier relative to literature[41].

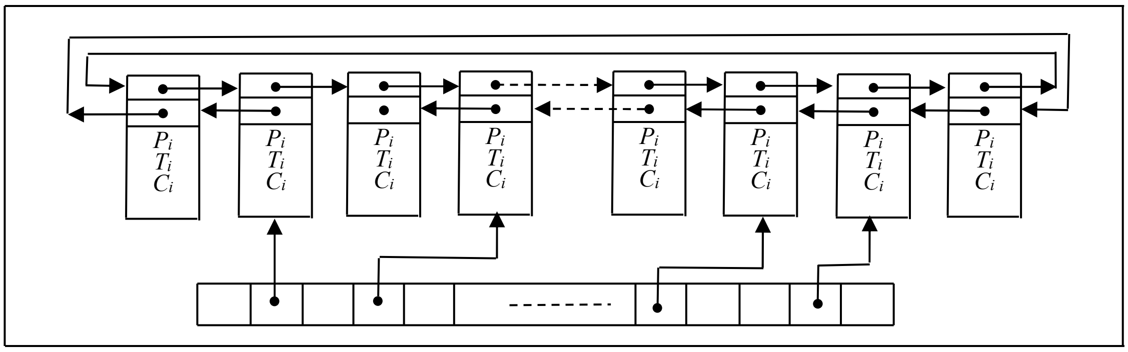

Figure 7. Hash-table on a circular double-linked list for sweep-circle clustering

3.3 Merging clusters

The indices of clusters must be merged, i.e., the initial clusters must be adjusted during the final phase in accordance with the merged points. We used a previously applied method [41] to merge the indices of the clusters, see Figure 6b for an example. In this example, the clusters C_l and C_r are merged via $\boldsymbol{p}_i$ and the smallest index value is preserved. In each list, any point that does not belong to any cluster is treated as an outlier/noise.

3.4 Point collinearity

In the ASC algorithm, the "point collinearity" occurs when more than one spatial point is located on the same θ of the polar coordinates. This is a special case that must be treated accordingly in terms of setting the adaptive threshold $\mathcal{E}(\boldsymbol{p}_i)$ to ensure the stability of the algorithm. The mean of the triangle perimeter (including previously even points) is defined as the adaptive dynamic threshold that occurs when the sweep-circle located a new even point. When the projection of the next point $\boldsymbol{p}_5$ hits the vertex $\boldsymbol{p}_4$ of the triangle $\Delta \boldsymbol{p}_2\boldsymbol{p}_4\boldsymbol{p}_3$, the mean of the perimeter can be calculated as the threshold, which determines if the points $\boldsymbol{p}_4$ and $\boldsymbol{p}_5$ are grouped into the same cluster as seen in Figure 8. The threshold of $\boldsymbol{p}_6$ is obtained according to the triangle $\Delta \boldsymbol{p}_2\boldsymbol{p}_5\boldsymbol{p}_3$.

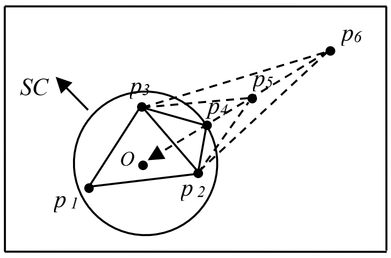

Figure 8. Vertexes located on the same line

4. ASC-based stream clustering

Bezdekg and Hathaway categorize any data set containing 10^8 objects as a "large data set"[52]. ASC extends the streaming clustering technique to large spatial data sets repeating a small number of sequential passes over objects (ideally, single passes) and clustering the objects using the average memory space, where the size of is a fraction of the stream length. The ASC-based stream clusters use a two-stage online and offline approach, as found in most streaming algorithms. In the online stage, the data set is split into blocks that are divided until they fit into the computer's main memory bank as the data points are swept with in an increasingly large circle. ASC is applied until all spatial data objects in the blocks are processed. In the current experiment we, implemented cluster indexing that stores the data in units of clusters grouped by ASC within storage systems. In the offline stage, the user sets the threshold $\mathcal{E}$ and the corresponding clustering number $\boldsymbol{K}$ is identified. Atom clusters in the online stage are repeatedly computed via ASC until the process is complete and the results are outputted.

Online stage

- The large data set S is divided into a sequence of data blocks $S=\{X_1,X_2,\ldots,X_i\}$ according to the memory size. A load monitor[53] ensures that the loading of spatial data fits the main memory.
- ASC is applied to each data block X_i to form atom clusters $C_i=\{C_1,C_2,\ldots,C_l\}$.

Offline stage

- It is assumed that the user provides a suitable threshold value $\mathcal{E}$ and the clustering number $\boldsymbol{K}$ is set in advance for the obtained atom clusters. ASC is repeatedly implemented until forming a final (macro) space cluster by processing retrieval queries from the cluster indexes into the adjacent data blocks.

The above algorithm can manage static data, as well as extend to the processing of dynamic data.

5. Results and Discussion

5.1 Time complexity analysis

All space points n are transformed to polar coordinates in O (n) and sorted according to their r-coordinates by Quicksort in O ($n$log$n$). The total time complexity of the initialization phase is:

$$T_{\text{inti}}=\mathrm{O}(n)+\mathrm{O}(n\log n)=\mathrm{O}(n\log n) \qquad (5)$$

The sweep-circle status is represented by the frontier, where points must be located to identify the hit the projected edge. This point location is found in the hash table. A previously reported formula was used[35] to determine the number of entries into the hash-table in ASC:

$$h=1+\lfloor n/k \rfloor \quad (6)$$

where h is the size of the hash-table, n is the number of table entries, and k is the constant factor.

According to a previous analysis[35], the relation between CPU time spent and the number of entries into the hash-table h correspond to k. If k is too small or too large, the run time will be significantly impacted. The k was set to 100 within these experiments, in accordance to previous literature[35]. During the sweeping phase, each point was projected onto the frontier, where it reached the frontier in a time period calculated as follows

$$T_{\text{locate}}=1+\lfloor n/k \rfloor= \text{O}\ (n/100)= \text{O}\ (n) \quad (7)$$

The frontier that corresponds to threshold $\varepsilon\ (\boldsymbol{p}_i)$ is computed in constant time O(n). The frontier projections and their corresponding distances under the adaptive threshold are used to determine if the clusters require O($n$log$n$). In the final phase the merged clusters that are adjusted indices require O ($n$log$n$). The total time complexity of clustering is as follows:

$$T_{\text{clus}}=\text{O}\ (n)+\text{O}\ (n\log n)+\text{O}\ (n)=\text{O}\ (n\log n) \quad (8)$$

where the total expected time complexity of the proposed ASC algorithm is:

$$T_{\text{total}}=T_{\text{inti}}+T_{\text{locate}}+T_{\text{clus}}=\text{O}\ (n\log n) \quad (9)$$

5.2 Comparison and analysis of experimental results

In order determine if the ASC clustering method could handle data with complex distribution, we utilized three 2D simulated spatial testing data sets:D1-D3, as well as a real-world spatial database. D1 and D2 are benchmark CHAMELEON data sets that satisfy the similarity test requirements in terms of spatial proximity, thematic attributes, spatial distribution, and hierarchy.

Data set D3 is very challenging for most clustering algorithms,as there are not only clusters with arbitrary shapes, different densities, and noise, but also distinctly uneven internal densities. Traditional clustering algorithms were also tested for comparison sake including K-Means (the most commonly used method), DBSCAN (which can determine arbitrary shape clusters), CURE (which can identify clusters of more complex shapes and wide variances in size, while preferentially filtering the isolated points), and AMOEDA (which can adapt to the clusters that are arbitrarily shaped or with different density without any a priori parameters using the Delaunay triangulation). The proposed ASC sweep-circle algorithm was also compared with Žalik's sweep-line algorithm[41]. A real-world GIS data set was used in order to imitate the proposed algorithm's ability to deal with actual spatial data.

D1includes 8000 points with eight arbitrary shape clusters of different densities as well as random noise as seen in Figure 9a. The DBSCAN, CURE, Žalik's sweep-line algorithm (from here on referred to simply as "Žalik") were all compared with the ASC. The DBSCAN had, MinPts set to 4, Eps was fixed to 5.4, the shrink factor of CURE was set to 0.3, and the number of representative points was set to 12. Parameter d was set to 12 for the Žalik algorithm. The ASC algorithm automatically discovered arbitrary shapes, clusters of different densities, and nested clusters (Figure 9b). It not only effectively detected all eight clusters but also correctly identified the noise in D1. The CURE algorithm was unable identify spatial clusters with complex shape and incorrectly defined less dense spatial data as noise. The DBSCAN algorithm could not readily adapt to the density variations among clusters, and the Žalik algorithm used global parameters that prevented it from identifying clusters with varying densities.

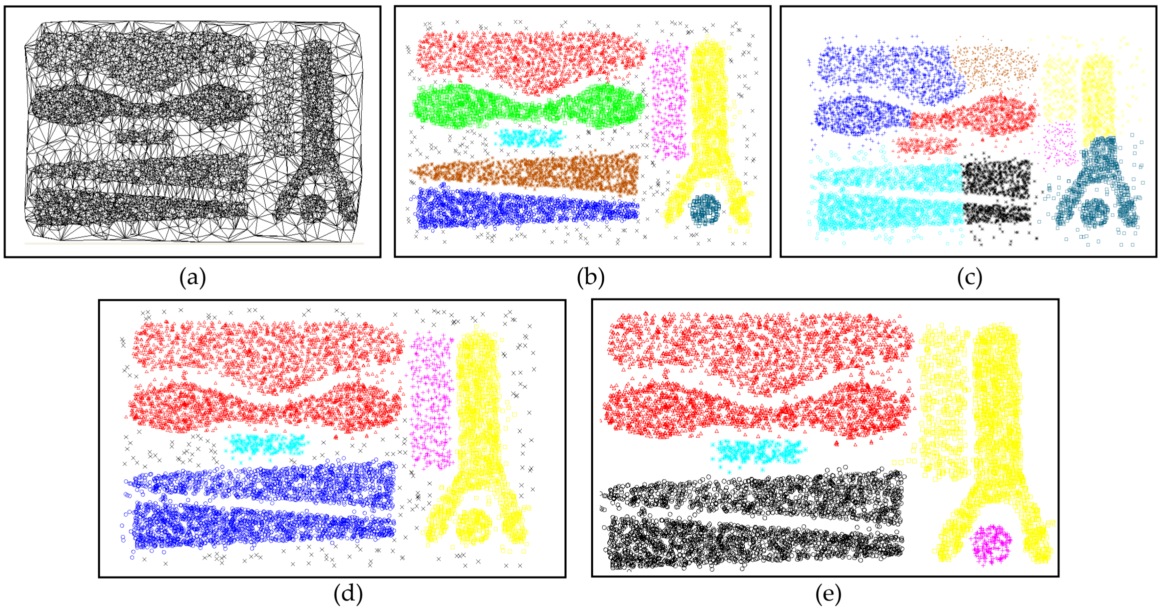
(a) (b) (c) (d) (e)

Figure 9. Testing data set D1 of ASC. (a) Graph built by triangulation of D1; (b) clustering result by ASC; (c) clustering result by DBSCAN; (d) clustering result by CURE; (e) clustering result by Žalik.

There were 10000 points in data set D2, see Figure 10. We varied the scaling factor α, which causes changes in the threshold in order to form clusters at different hierarchies. The results demonstrated that if the threshold was small when α was small, resulting in a large number of clusters. Conversely, the threshold was large if α was large, resulting in a smaller number of clusters forming (i.e., relatively shallow hierarchy). When the value of α nears 1, clusters are easily and accurately distinguished from noise. In fact, the two effects create a favorable balance. This implicit hierarchical relationships with different thresholds related to α are often used as the basis of analysis in practice[16].

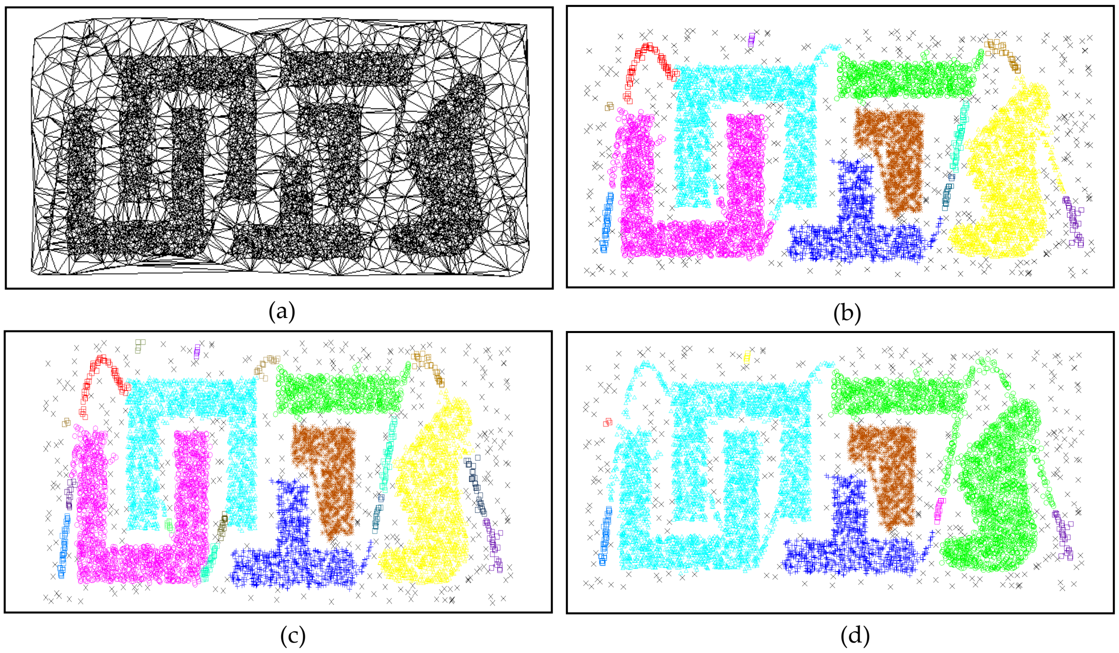
(a) (b) (c) (d)

Figure 10. Results largely dependent on parameters. (a) Graph built by triangulation of D2; (b) 20 clusters obtained when $\alpha = 1$; (c) 30 clusters obtained when $\alpha = 0.8$; (d) 8 clusters obtained when $\alpha = 1.2$.

Data set D3 containing 264 test points was used to test the recognition effectiveness of the ASC algorithm in clusters with uneven internal densities and non-uniform data distribution. The clustering results of D3 by K-Means, DBSCAN (Minpts=4, Eps=0.78), AMOEBA, and Žalik (*d*=0.0074) and the ASC algorithm are seen in Figure 11, where the ASC algorithm was best able to discover clusters of uneven internal density. The clusters were simply divided into several parts by K-Means, where DBSCAN detected noise accurately, however failed to separate nearby clusters. Both the AMOEBA and the Žalik failed to identify clusters of uneven internal density.

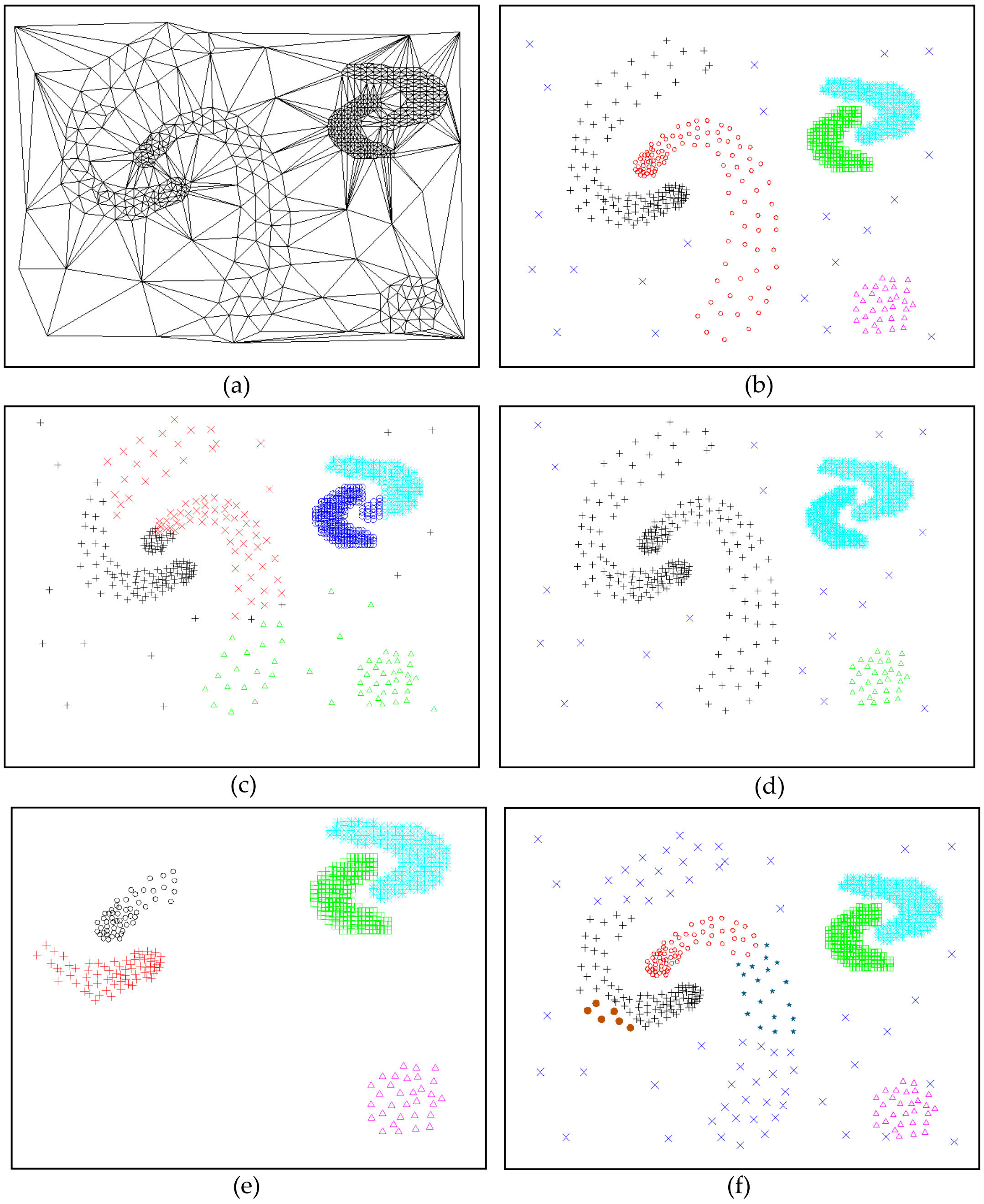

Figure 11. Clustering results of data set D3 by comparison: (a) Graph built by triangulation of D3; (b) clustering result by ASC; (c) clustering result by K-Means; (d) clustering result by DBSCAN; (e) clustering result by Žalik. ; (f) clustering result by AMOEBA.

To illustrate the practical adaptability of ASC, we applied it to a real-world GIS data set collected obtained from the DCW (Digital Chart of the World) that focused on 15067 position data points taken from Chinese cities, towns, and villages pertaining to populations in 2002. The results from the ASC showed that the algorithm adapted well and is effective for this manner of practical application (See Figure 12).

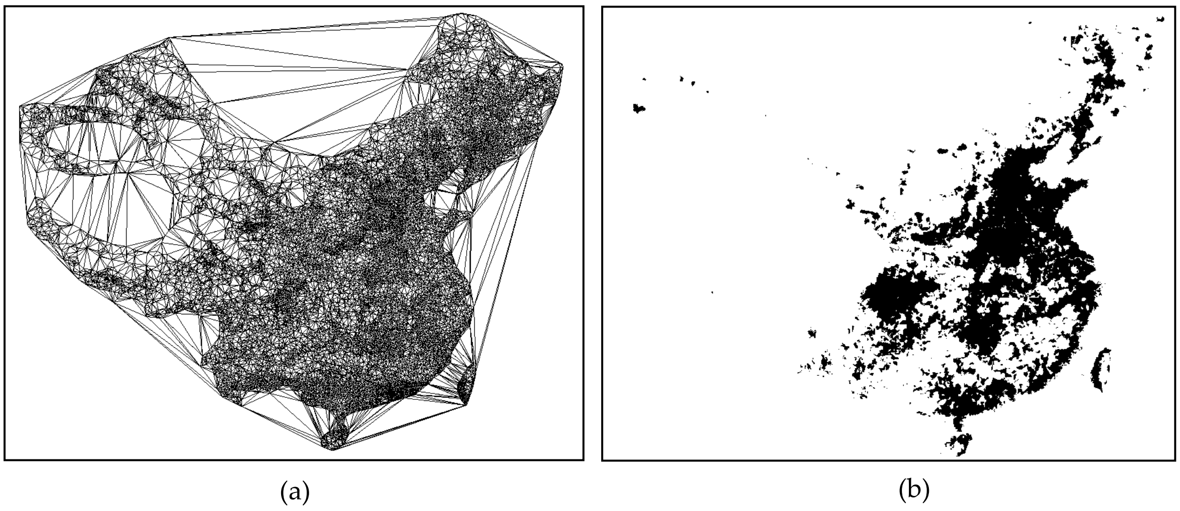

Figure 12. ASC in large spatial datasets discovered clusters.(a)is the graph built by triangulation of GIS datasets;(b)is the clustering result of GIS large spatial datasets generated by ASC.

5.3 CPU time

Actual CPU run-time for data processing is a greater concern within real-world applications, so we tested the proposed algorithm and compared it to several other methods accordingly. All algorithms were executed with the same development language, development environment, operating system, and hardware (Intel R-core i3-3220 CPU@3.30GHz 3.29 GHz and 2GB memory, Seagate SV35 7200 rpm, access time 14.7 MS).

CPU time spent for clustering

The CPU time spent while clustering data sets from Figure 9-12 is compared in Table 2. The actual efficiency of CPU time is correlated to the number of clusters and test points generated – more CPU time was spent on larger numbers of clusters than when the number of test points were the same. When the same number of clusters were obtained, the run time decreased when there were fewer test points. Additional time was spent when clustering or merging many small clusters rather than one larger cluster.

Table 2.CPU time (s) spent for clustering

Data set	Points	CPU time (s)
Figure 12	264	0.014
Figure 9	8000	0.074
Figure 10b	10,000	0.116
Figure 10c	10,000	0.124
Figure 10d	10,000	0.243
Figure 11	15,067	0.457

Most current adaptive algorithms (AMOEB, AUTOCLUST, and ANDC) were developed based on the Delaunay triangulation (i.e., high spatial proximity). See Table 3 for a comparison of the traditional adaptive method AUTOCLUST against the proposed algorithm. AUTOCLUST is a relatively new algorithm developed to manage complex data sets (such as those with clusters of varying density and arbitrarily shapes). This algorithm is similar to the proposed algorithm in that it can adaptively discover spatial clusters without the need to set parameters in advance. The implementation class of AUTOCLUST can be obtained from the WEB[29].

There are several methods available for constructing Delaunay triangulations. We used the fastest SL algorithm among them according to the literature[35]. The experimental results suggested that the ASC is more efficient than AUTOCLUST see Table 3. AUTOCLUST spent more CPU time on both the Delaunay triangulation phase of the algorithm and when dealing with repeated global and local uninteresting edges during the clustering process when the number of edges exceeded the number of points.

Table 3.CPU time (s) spent by ASC and AUTOCLUST for clustering

Dataset	ASC	AUTOCLUST		
	Clustering time	DT time	Clustering time	Total (s)
100,00	0.249	0.026	0.314	0.340
20,000	0.422	0.082	2.450	2.532
50,000	0.941	0.287	5.125	5.412
100,000	2.653	0.607	14.234	14.841
200,000	5.324	2.290	61.124	63.414

The CPU time spent between Žalik and ASC was compared (Table 4) using the same dataset, but with the different corresponding phases of the ASC algorithm. Initialization involved sorting input points by Quicksort in Žalik, which accounted for 22% of the total time spent. Initialization involved calculating the polar coordinate with Quicksort in ASC, which accounted for 30% of the total time. ASC employed the hash-table to speed up the efficiency during the clustering and merging phases when searching the event queue and locations on the borders. It was not necessary to consider projections that surpassed the border, however, this portion of the run time in Žalik was spent both on missed projections and deleting the useless AF. In ASC, 8-9% of the total time was spent calculating polar coordinates of the input points and adaptive thresholds of the event points. The CPU spent less time on the whole, and it is possible that the ASC algorithm could be truncated even further if the polar coordinates of the input points were obtained in advance.

Table 4.CPU time (s) spent by ASC and Žalik for different algorithm phases

Data set	100,000		200,000	
Algorithm	ASC	Žalik	ASC	Žalik
Initialization	0.646	0.717	1.597	2.700
Clustering	1.026	1.944	2.715	7.242
Merging	0.481	0.624	1.012	2.332
Total	2.153	3.285	5.324	12.274

CPU time spent for ASC-based stream clustering

The ASC algorithm was also tested based on stream techniques. Input spatial data sets were generated with high-resolution images obtained from Bing Maps (http://www.bing.com/maps/) containing 1.3G bytes of data points. As shown in Figure 13, varying thresholds for $\mathcal{E}$ (e.g., 100-600 m) and clusters K (e.g., 100-500) were provided to test the CPU time spent running the algorithm. The obtained clusters decreased as $\mathcal{E}$ increased, the running time of the algorithm was lower in the offline phase, and the $\mathcal{E}$ value ranged between 100-300 resulted in relatively accurate clusters as shown in Figure 13. It took longer to generate a larger number of adaptive sub-clusters during the online phase.

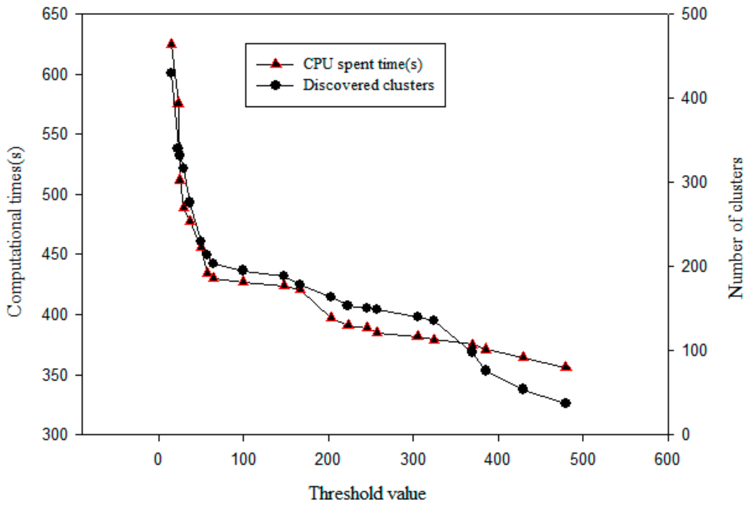

Figure 13. Stream clustering speed comparison

6. Practical application of ASC

In order to explore the feasibility of the ASC algorithm in real world scenarios, it was used to forecast geological disasters and quake magnitude based on geography. The real-world spatial data set containing 1264 geological disaster spots in Congzuo was collected from the geologic hazard database at the land department of Guangxi Zhuang Autonomous Region, China. The clustering results are shown in Figure 14.

ASC successfully detected nine clusters with varying densities within the disaster spots, which were divisible into a preliminary distribution range of disaster-prone geographical areas (Figure 14c). It is possible to set a specific threshold (e.g., $\mathcal{E}$=1000 m) in order to find the areas most prone to major disasters, which could be beneficial for early warning purposes (Figure 14d).

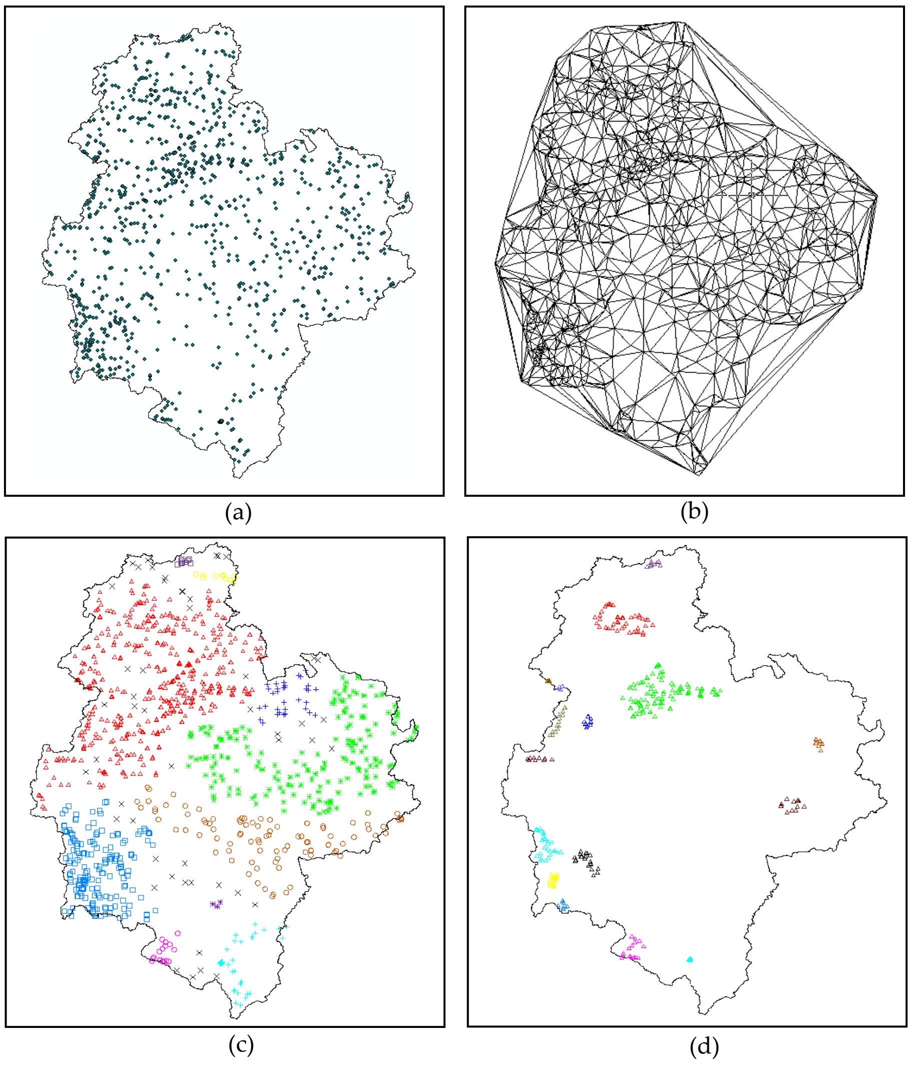

Figure 14. Spatial clustering results on disaster database by ASC:(a) Distribution of disaster data points; (b) description of spatial neighborhood relations via Delaunay triangulation; (c) clustering result of ASC; (d) clustering result of user-defined threshold setting ($\mathcal{E}$=1000m).

7. Conclusion

The most notable conclusions of this study can be summarized as follows:

- The Gestalt theory was successfully applied to enhance the adaptability of the spatial clustering algorithm. Both the sweep-circle technique and the dynamic threshold setting were employed to detect spatial clusters.
- The ASC algorithm can automatically locate clusters in a single pass, rather than through modifying the initial model (i.e., via minimal spanning tree, Delaunay triangulation, or Voronoi diagram). The algorithm could quickly adapt to identify arbitrarily shaped clusters, and could locate the non-homogeneous density characteristics of spatial data without necessitating a priori knowledge or parameters. The time complexity of the ASC algorithm was approx. O ($n\log n$), where n is the size of the spatial database.
- Scalability in ASC was not limited to the size of the data set, demonstrating that the algorithm is suitable for data streaming technology to cluster large, dynamic spatial data

sets.

- The proposed algorithm was efficient, feasible, easily understood, and easily implemented.

The vast amount of information contained in spatial data sets and their relative complexity represent challenges yet to be solved. In the future, we believe we may benefit from further exploiting the characteristics of human vision; humans can easily clusters connected by chains and/or necks, for example, and those touching Gaussian clusters[14] – ASC is unable to discover these special clusters. Additionally, in ASC, points which do not belong to any cluster are treated as outliers/noise, where multiple outliers or noise points could be processed as new, independent clusters. Finally, the algorithm can potentially be extended to clustering spatial data with higher dimensionality than those discussed in the present study.

Acknowledgments: This work has been supported by the National Science and Technology Support Program (Grant No. 2014BAL05B07) funded by the Ministry of Science and Technology of China.

Author Contributions: Qingming Zhan and Shuguang Deng conceived and designed the experiments; Shuguang Deng performed the experiments; All the authors analyzed the data; Qingming Zhan and Shuguang Deng wrote the paper.All authors contributed to revising the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Han, J.; Kamber, M.; Pei, J. 10 - cluster analysis: Basic concepts and methods. In *Data mining (third edition)*, Morgan Kaufmann: Boston, 2012; pp 443-495.
2. Chen, J.; Lin, X.; Zheng, H.; Bao, X. A novel cluster center fast determination clustering algorithm. *Applied Soft Computing* **2017**, *57*, 539-555.
3. Deng, M.; Liu, Q.; Cheng, T.; Shi, Y. An adaptive spatial clustering algorithm based on delaunay triangulation. *Computers, Environment and Urban Systems* **2011**, *35*, 320-332.
4. Liu, Q.; Deng, M.; Shi, Y. Adaptive spatial clustering in the presence of obstacles and facilitators. *Computers & Geosciences* **2013**, *56*, 104-118.
5. Bouguettaya, A.; Yu, Q.; Liu, X.; Zhou, X.; Song, A. Efficient agglomerative hierarchical clustering. *Expert Systems with Applications* **2015**, *42*, 2785-2797.
6. MacQueen, J. In *Some methods for classification and analysis of multivariate observations*, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, Berkeley, Calif., 1967, 1967; University of California Press: Berkeley, Calif., pp 281-297.
7. Ng, R.T.; Han, J. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers Inc.: 1994; pp 144-155.
8. Guha, S.; Rastogi, R.; Shim, K. Cure: An efficient clustering algorithm for large databases ☆. *Information Systems* **1998**, *26*, 35-58.
9. Zhang, T. Birch: An efficient data clustering method for very large databases. *Acm Sigmod Record* **1999**, *25*, 103-114.
10. Karypis, G.; Han, E.-H.; Kumar, V. Chameleon: Hierarchical clustering using dynamic modeling. *Computer* **1999**, *32*, 68-75.

11. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. In *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proc. of 2nd International Conference on Knowledge Discovery and, 1996; pp 226-231.

12. Ankerst, M.; Breunig, M.M.; Kriegel, H.-P.; #246; Sander, r. Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, ACM: Philadelphia, Pennsylvania, USA, 1999; pp 49-60.

13. Hinneburg, A.; Keim, D.A. In *An efficient approach to clustering in large multimedia databases with noise*, KDD, 1998; Agrawal, R.; Stolorz, P.E.; Piatetsky-Shapiro, G., Eds. AAAI Press: pp 58-65.

14. Zahn, C.T. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers* **1971**, *C-20*, 68-86.

15. Estivill-Castro, V.; Lee, I. In *Autoclust: Automatic clustering via boundary extraction for mining massive point-data sets*, In Proceedings of the 5th International Conference on Geocomputation, 2000.

16. Kang, I.-S.; Kim, T.-w.; Li, K.-J. A spatial data mining method by delaunay triangulation. In *Proceedings of the 5th ACM international workshop on Advances in geographic information systems*, ACM: Las Vegas, Nevada, USA, 1997; pp 35-39.

17. Wang, W.; Yang, J.; Muntz, R.R. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers Inc.: 1997; pp 186-195.

18. Sheikholeslami, G.; Chatterjee, S.; Zhang, A. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers Inc.: 1998; pp 428-439.

19. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* **1977**, *39*, 1-38.

20. Gennari, J.H.; Langley, P.; Fisher, D. Models of incremental concept formation. *Artificial Intelligence* **1989**, *40*, 11-61.

21. Kohonen, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics* **1982**, *43*, 59-69.

22. Schikuta, E. In *Grid-clustering: An efficient hierarchical clustering method for very large data sets*, Pattern Recognition, 1996., Proceedings of the 13th International Conference on, 25-29 Aug 1996, 1996; pp 101-105 vol.102.

23. Pei, T.; Zhu, A.X.; Zhou, C.; Li, B.; Qin, C. A new approach to the nearest-neighbour method to discover cluster features in overlaid spatial point processes. *International Journal of Geographical Information Science* **2006**, *20*, 153-168.

24. Tsai, C.-F.; Tsai, C.-W.; Wu, H.-C.; Yang, T. Acodf: A novel data clustering approach for data mining in large databases. *Journal of Systems and Software* **2004**, *73*, 133-145.

25. Estivill-Castro, V.; Lee, I. *Amoeba: Hierarchical clustering based on spatial proximity using delaunaty diagram*. 2000.

26. Estivill-Castro, V.; Lee, I. Argument free clustering for large spatial point-data sets via boundary extraction from delaunay diagram. *Computers Environment & Urban Systems* **2002**, *26*, 315–334.

27. Wei, C.P.; Lee, Y.H.; Hsu, C.M. Empirical comparison of fast partitioning-based clustering algorithms for large data sets. *Expert Systems with Applications* **2003**, *24*, 351-363.

28. Li, D.; Yang, X.; Cui, W.; Gong, J.; Wu, H. A novel spatial clustering algorithm based on delaunay triangulation</title>. **2008**, *7285*, 728530-728530-728539.

29. Liu, D.; Nosovskiy, G.V.; Sourina, O. Effective clustering and boundary detection algorithm based on delaunay triangulation. *Pattern Recognition Letters* **2008**, *29*, 1261-1273.

30. Nosovskiy, G.V.; Liu, D.; Sourina, O. Automatic clustering and boundary detection algorithm based on adaptive influence function. *Pattern Recognition* **2008**, *41*, 2757-2776.

31. Xu, D.; Tian, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science* **2015**, *2*, 165-193.

32. Zhao, Q.; Shi, Y.; Liu, Q.; Fränti, P. A grid-growing clustering algorithm for geo-spatial data. *Pattern Recognition Letters* **2015**, *53*, 77-84.

33. Bolaños, M.; Forrest, J.; Hahsler, M. Clustering large datasets using data stream clustering techniques. In *Data analysis, machine learning and knowledge discovery*, Spiliopoulou, M.; Schmidt-Thieme, L.; Janning, R., Eds. Springer International Publishing: Cham, 2014; pp 135-143.

34. Preparata, F.P.; Ian, S.M. *Computational geometry:An introduction*. 1985.

35. Žalik, B. An efficient sweep-line delaunay triangulation algorithm. *Computer-Aided Design* **2005**, *37*, 1027-1038.

36. Alfred, U. A mathematician's progress. *The Mathematics Teacher* **1966**, *59*, 722-727.

37. Shamos, M.I.; Hoey, D. Geometric intersection problems. *Foundations of Computer Science Annual Symposium on* **1976**, 208-215.

38. Bentley, J.L.; Ottmann, T.A. Algorithms for reporting and counting geometric intersections. *Computers IEEE Transactions on* **1979**, *C-28*, 643-647.

39. Fortune, S. A sweepline algorithm for voronoi diagrams. *Algorithmica* **1987**, *2*, 153-174.

40. Zhou, P. Computational geometry algorithm design and analysis(fourth edition). In *Computational geometry algorithm design and analysis(fourth edition)*, Beijing:Tsinghua University Press: 2011.

41. Žalik, K.R.; Žalik, B. A sweep-line algorithm for spatial clustering. *Advances in Engineering Software* **2009**, *40*, 445-451.

42. Biniaz, A.; Dastghaibyfard, G. A faster circle-sweep delaunay triangulation algorithm. *Advances in Engineering Software* **2012**, *43*, 1-13.

43. Adam, B.; Kauffmann, P.; Schmitt, D.; Spehner, J.-C. In *An increasing-circle sweep-algorithm to construct the delaunay diagram in the plane*, CCCG, 1997.

44. Guha, S.; Mishra, N.; Motwani, R.; O'Callaghan, L. In *Clustering data streams*, Symposium on Foundations of Computer Science, 2000; pp 359-366.

45. O'Callaghan, L.; Mishra, N.; Meyerson, A.; Guha, S.; Motwani, R. In *Streaming-data algorithms for high-quality clustering*, Data Engineering, 2002. Proceedings. 18th International Conference on, 2002; pp 685-694.

46. Zengyou, H.E.; Xiaofei, X.U.; Deng, S. Squeezer: An efficient algorithm for clustering categorical data. *Journal of Computer Science & Technology* **2002**, *17*, 611-624.

47. Guha, S.; Meyerson, A.; Mishra, N.; Motwani, R.; O'Callaghan, L. Clustering data streams: Theory and practice. *Knowledge & Data Engineering IEEE Transactions on* **2003**, *15*, 515-528.

48. Ding, S.; Zhang, J.; Jia, H.; Qian, J. An adaptive density data stream clustering algorithm. *Cognitive Computation* **2016**, *8*, 30-38.

49. Zheng, L.; Huo, H.; Guo, Y.; Fang, T. Supervised adaptive incremental clustering for data stream of chunks. *Neurocomputing* **2016**.

50. Tobler, W.R. A computer movie simulating urban growth in the detroit region. *Economic Geography* **1970**, *46*, 234-240.

51. Ellis, W.D. *A source book of gestalt psychology*. Kegan Paul, Trench, Trubner & Company: London, England, 1938; p xiv, 403.
52. Hathaway, R.J.; Bezdek, J.C. Extending fuzzy and probabilistic clustering to very large data sets. *Computational Statistics & Data Analysis* **2006**, *51*, 215-234.
53. Cho, K.; Jo, S.; Jang, H.; Kim, S.M.; Song, J. Dcf: An efficient data stream clustering framework for streaming applications. In *Database and expert systems applications: 17th international conference, dexa 2006, kraków, poland, september 4-8, 2006. Proceedings*, Bressan, S.; Küng, J.; Wagner, R., Eds. Springer Berlin Heidelberg: Berlin, Heidelberg, 2006; pp 114-122.