GAC-GEO: A Generic Agglomerative Clustering Framework for Geo-referenced Datasets

Rachsuda Jiamthapthaksin[*§] · Christoph F. Eick[§] · Seungchan Lee[§]

Received: 30 November 2009 / Revised: 29 July 2010 / Accepted: 25 September 2010

Abstract Major challenges of clustering geo-referenced data include identifying arbitrarily shaped clusters, properly utilizing spatial information, coping with diverse extrinsic characteristics of clusters and supporting *region discovery* tasks. The goal of region discovery is to identify interesting regions in geo-referenced datasets based on a domain expert's notion of interestingness. Almost all agglomerative clustering algorithms only focus on the first challenge. The goal of the proposed work is to develop agglomerative clustering frameworks that deal with all four challenges. In particular, we propose a generic agglomerative clustering framework for geo-referenced datasets (GAC-GEO) generalizing agglomerative clustering by allowing for three plug-in components. GAC-GEO agglomerates neighboring clusters maximizing a plug-in *fitness function* that capture the notion of interestingness of clusters. It enhances typical agglomerative clustering algorithms in two ways: fitness functions support task-specific clustering, whereas *generic neighboring relationships* increase the number of merging candidates. We also demonstrate that existing agglomerative clustering algorithms can be considered as specific cases of GAC-GEO. We evaluate the proposed framework on an artificial dataset and two real world applications involving region discovery. The experimental results show that GAC-GEO is capable of identifying arbitrarily shaped hotspots for different data mining tasks.

Keywords *Agglomerative clustering · Hybrid clustering · Region discovery · Spatial clustering algorithms · Hotspot detection*

[*] *Computer Science Department, Faculty of Science and Technology, Assumption University, Bangkok, THAILAND*
Phone (+66) 2719-1515 Ext. 3681
Fax (+66) 719-1639
rachsuda@scitech.au.edu

[§] *Computer Science Department, University of Houston, TX, USA*
Phone (713) 743-3350
Fax (713) 743-3335
rachsuda@cs.au.edu, ceick@cs.uh.edu, slee49@mail.uh.edu

1 Introduction

Advances in Geographic Information Systems (GIS) have brought on new capabilities in gathering, storing, editing, querying, analyzing, sharing, and displaying geographically-referenced information. In Earth Science tremendous amounts of diverse geo-referenced data have been collected at various levels of granularity. For example, the Texas Water Development Board (TWDB) collects water well data in Texas for managing the state's water resources (TWDB 2008), the U.S. Environmental Protection Agency (EPA) collects environmental and health data to execute policies in protecting human health and the environment (EPA 2008), and the National Oceanic and Atmospheric Administration's (NOAA) satellites gather ocean, coast and atmospheric data of the global ecosystem for understanding and predicting changes in the earth's environment (NOAA 2008). The rapid growth of those spatial data establishes the need for scientists to seek novel spatial data mining techniques that summarize and analyze them to find interesting but implicit patterns as well as to predict future spatial events in a highly automated fashion.

Of particular interest is task-specific spatial clustering that discovers groups of objects in spatial proximity in geo-referenced datasets based on specific characteristics; for example, earth scientists are interested in detecting geographical regions with a high co-occurrence of particular diseases. Traditional clustering algorithms are not suitable to search for task-specific clusters because they consider only domain independent characteristics to form a solution. For instance, partitioning clustering algorithms like K-Means, PAM (MacQueen 1967; Kaufman and Rousseeuw 1990) and hierarchical clustering algorithm like AGNES (Kaufman and Rousseeuw 1990) consider cluster compactness while density-based clustering algorithms, such as DBSCAN (Ester et al. 1996), construct clusters based on object density. Xiong *et al*. also addressed the similar limitation that typical clustering algorithms have no built-in knowledge of desirable patterns and may result in conflicts between implicit goals of the algorithm and expected patterns (Xiong et al. 2009).

Another related data mining methodology to support task-specific spatial clustering is *region discovery* (Eick et al. 2006; Ding et al. 2008): its goal is the discovery of interesting regions in geo-referenced datasets based on a domain expert's notion of interestingness. The problem of identifying interesting regions can be viewed as a spatial clustering problem: spatial clustering is performed to find regions that maximize an externally given *fitness function;* a function captures the notion of interestingness of clusters. Directly applying traditional clustering algorithms to cope with the first challenge is not feasible because very few of these algorithms support plug-in fitness functions: CHAMELEON (Karypis et al. 1999) and MOSAIC (Choo et al. 2007). In addition, different region discovery tasks require different fitness functions. For instance, in categorical hotspot detection, the search centers on finding regions with high class-purity while identifying discrepancy areas with respect to a single continuous attribute, the fitness function needs to reward regions whose members have high variance with respect to the continuous attribute.

In addition to task-specific spatial clustering, spatial context is one significant characteristic that guides clustering of spatial datasets. For example, instances spatially located together usually have similar patterns (Shekhar et al. 2003) and most useful patterns in spatial datasets have a geographically regional scope rather than a global scope (Ding et al. 2007). Moreover, by treating all attributes identically, traditional clustering algorithms find clusters in the complete attribute space. Spatial clustering algorithms, on

the other hand, seek for contiguous clusters in the subspace of the geo-referenced attributes; e.g. they are looking for particular places or particular periods of time containing events with some predetermined characteristics. According to our investigation, current works in spatial clustering (Ester et al. 1996; Gao, Peuquet and Gahegan 2002; Duan et al. 2007) fail to separate spatial from non-spatial attributes. Another important characteristic of spatial clustering is that clusters may have arbitrary shape, *e.g.*, in the volcano dataset (UCI repository 2008), active and inactive volcanoes locate in chain-like pattern. Clustering algorithms restricting their cluster formation to globular or convex shapes fail to identify this arbitrary pattern (Tan, Steinbach and Kumar 2005). Regarding the ability of discovering arbitrarily shaped clusters, two categories of clustering algorithms have been proposed to address this problem: agglomerative (Karypis et al. 1999; Choo et al. 2007; Chaoji et al. 2009) and density based clustering algorithms (Ester et al. 1996; Sander et al. 1998; Duan et al.2007).

This paper focuses on the development of frameworks that support task-specific clustering and which are capable of identifying arbitrarily shaped clusters. In particular, we propose a Generic Agglomerative Clustering framework for GEO-referenced datasets (GAC-GEO) that explicitly utilizes spatial information to discover arbitrarily shaped clusters, and supports plug-in fitness functions for the task-specific analysis of geo-referenced datasets. One principle of GAC-GEO is to reassemble existing clustering algorithms with minimal human effort with the goal to obtain better clustering results. GAC-GEO utilizes a two-phase clustering approach: a preprocessing phase constructs a set of initial clusters and spatial neighboring relationships among those clusters and an agglomerative phase that greedily merges neighboring clusters maximizing a given plug-in fitness function. Relying on object-oriented design principles, GAC-GEO, as depicted in Fig. 1, reassembles three generic plug-in components to generate new clustering algorithms: 1) preprocessing algorithms that generate an initial set of clusters, 2) spatial neighboring relationship among clusters and 3) fitness functions. The contributions of this paper include:

1. Provide powerful region discovery capabilities by introducing GAC-GEO, a highly generic clustering framework that enhances clustering quality of the obtained spatial clusters. GAC-GEO is capable of detecting arbitrarily shaped clusters, supports task-specific clustering and supports finding patterns in geo-referenced data at different levels of granularity.
2. Demonstrate five variations of GAC-GEO that glue one of the following five preprocessing methods with the same agglomerative algorithm in order to obtain five different clustering algorithms: a set of singleton clusters (SC), a representative based clustering algorithm (CLEVER), randomized sampling (RS), grid partitioning (GP) and divisive grid based clustering algorithm (SCMRG). The neighboring relationships for the first three variations are Gabriel graphs of cluster's representatives whereas the latter two variations use an adjacency matrix of grid cells.
3. Perform an empirical study on the five variations of GAC-GEO and evaluate their performance on an artificial dataset (DMML datasets 2008) in terms of fitness, consistency of obtained solutions and ability in detecting arbitrarily shaped clusters. Not only the benefits of the preprocessing techniques but also the benefits of the proposed agglomerative clustering framework are evaluated in the study.
4. Demonstrate the capabilities of GAC-GEO in real world case studies involving an earthquake dataset.

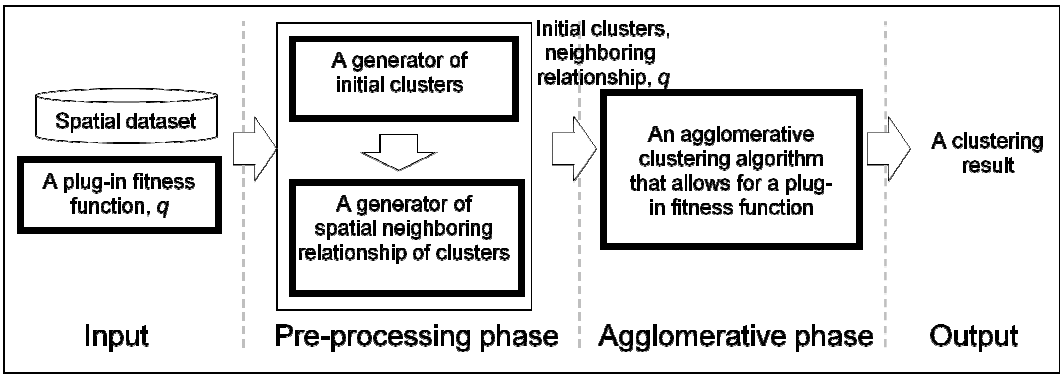


Fig. 1. Generic agglomerative clustering framework for geo-referenced datasets (GAC-GEO)

The remainder of the paper is organized as follows. In Section 2, we formally define region discovery problems, explain how clustering algorithms are used for this task and also formally define spatial agglomerative clustering algorithm. To emphasize the generality of the framework we describe five variants of the framework in Section 3. The empirical study and comparative study among different variations are performed on an artificial dataset and two real-world case studies as shown in Section 4. Sections 5 and 6 discuss related work in clustering algorithms and give conclusion, respectively.

2 Background and Problem Definition

In this section we propose a generic agglomerative clustering framework serving region discovery that supports plug-in fitness functions. We first formally define a problem of region discovery as a clustering problem to find a number of clusters that maximize a fitness function. Then, formal definitions of a generic agglomerative clustering framework and related concepts are given.

2.1 Region discovery algorithms

In region discovery, clustering algorithms allow for a plug-in fitness function to capture different notions of interestingness of clusters. A *fitness function* defines quality of a clustering as the sum of regional quality of all clusters. Given a set of k regions, $R=\{x_1,x_2,x_3,\ldots,x_k\}$ identified from a geo-referenced dataset O, we define the quality $q(R)$ of a partitioning R as a sum of the rewards obtained from each region x_j (j=1,2,3,…,k):

$$q(R) = \sum_{i=1}^{k} Reward(x_i), \quad (1)$$

where

$$Reward(x_i) = i(x_i) \times size(x_i)^{\beta}. \quad (2)$$

The *interestingness function* $i(x_i)$ computes the interestingness of the cluster x_i, $size(x_i)$ denotes the number of objects in the cluster x_i, and β is a parameter used to influence the size of the discovered regions—choosing large value for β indicates preference for finding large regions.

Let $S=\{s_1,s_2,s_3,\ldots,s_q\}$ be a set of spatial attributes. $N=\{n_1,n_2,n_3,\ldots,n_p\}$ be a set of non-spatial attributes. F=Dom(S)×Dom(N) = Dom(s_1)×…×Dom(s_q)×Dom(n_1)…×Dom(n_p) be an infinite set of objects in relational database schema. $O=\{o_1,o_2,o_3,\ldots,o_n\} \subseteq F$ be a finite set of objects in a geo-referenced dataset, and $o_i \in$ Dom(S)×Dom(N)=$\{s_1,s_2,s_3,\ldots,s_q, n_1,n_2,n_3,\ldots,n_p\}$.

Region Discovery Algorithms. Given O and q, a region discovery algorithm searches for a set of regions (clusters) $\{x_1,x_2,x_3,\ldots,x_k\} \in R$ that maximizes $q(R)$. Regions discovered must satisfy the following criteria:

1. $x_i \subseteq O$, $i=1,2,3,\ldots,k$.
2. $x_i \cap x_j = \emptyset$, $i \neq j$.
3. $contiguous(x_i)$, $i=1,2,3,..,k$.
4. $x_1 \cup x_2 \cup x_3 \cup \ldots \cup x_k \subseteq O$.
5. $x_1,x_2,x_3,\ldots,x_k$ are ranked based on their reward values.

In summary, the discovered regions must be a disjoint set of contiguous regions ($contiguous(x_i)$) which obtained from O. The generated regions are not required to be exhaustive with respect to the geo-referenced dataset O. The objects belonging to regions that receive no reward are treated as outliers in our approach. In many applications, we are interested in the top k regions with respect to q, similar to a search-engine. Consequently, we rank the regions based on the rewards.

Region contiguity is one of the constraints required by region discovery algorithms. We derive a concept of the contiguity of a region from its spatial neighboring relationship, *no*. A region x is *contiguous* if for each pair of objects u and v in x there is a path between u and v that solely traverses x and no other regions. The contiguity constraint guarantees that region discovery algorithms generate contiguous clusters. More formally, contiguity of region is defined as a predicate over subsets x of O:

$$contiguous(x) \Leftrightarrow \forall u \in x\ \forall v \in x\ \exists m \geq 2\ \exists w_1,w_2,w_3,\ldots,w_m \in x: u=w_i \wedge v=w_{i+1} \wedge no(w_i, w_{i+1}) \quad (i=1,2,3,\ldots,m).$$

As a result, we also obtain contiguity of regions: $contiguous(R) \Leftrightarrow \forall x \in R: contiguous(x)$.

Region discovery algorithms are restricted to be applied on geo-referenced datasets, but in this paper we formally define a slightly simpler version of the framework that just operates in spatial datasets. We define a geo-referenced dataset O as a set of objects in an attribute space F. Objects belonging to O are tuples characterized by two kinds of attributes: spatial attributes S and non-spatial attributes N. In another word, each object in the geo-referenced dataset is a tuple that takes values in both spatial attributes' space (Dom(S)) and non-spatial attributes' space (Dom(N)). Datasets discussed in this paper conform to the notation of the geo-referenced dataset; as roles specified spatial attributes are used for calculating spatial neighboring distance whereas non-spatial attributes are used in fitness function computation.

2.2 An agglomerative clustering framework in spatial domain

This section provides formal definitions necessary for solving a problem of the spatial agglomerative clustering. First, we propose a generic agglomerative clustering framework in spatial domain, and give details of its preprocessing and agglomerative phases. Then, we provide discussion of incremental updating in the agglomerative phase and time complexity of the propose framework.

Given $O=\{o_1,o_2,o_3,\ldots,o_n\}$, $O \subseteq F$. $R=\{x_1,x_2,x_3,\ldots,x_k\}$, $x_i \cap x_j = \emptyset$, $\bigcup_{i=1}^{k} x_i \subseteq O$, where O is a geo-referenced dataset, F is relational database schema, $S=\{s_1,s_2,s_3,\ldots,s_p\}$, $s_i \in \Re^p$ is a set of spatial attributes, $N=\{n_1,n_2,n_3,..,n_q\}$, $n_i \in \Re^q$ is a set of other non-spatial attributes.

DEFINITION 1 (Initial Clusters). Initial clusters are defined as clusters of objects that are non-overlapping and contiguous (traversable objects in cluster x_i'):

$R_{initial} = \{x_1', x_2', x_3', \ldots, x_m'\}, x_i' \subseteq O$, $x_i' \cap x_j' = \emptyset$, and $\bigcup_{i=1}^{m} x_i' \subseteq O$, $m>k$ and *contiguous*($R_{initial}$).

DEFINITION 2 (Agglomerative Clusters). Agglomerative clusters are defined as clusters that are unions of initial clusters$R_{initial}$ that have to be non-overlapping and contiguous:

$R_{agglo} = \{x_1, x_2, x_3, \ldots, x_k\}$, $x_i \cap x_j = \emptyset$ and $\forall x_l' \in R_{initial}\ x_l' \subseteq x_i$, $\bigcup_{i=1}^{k} x_i = \bigcup_{l=1}^{m} x_l'$, and *contiguous*($R_{agglo}$).

DEFINITION 3 (The Spatial Agglomerative Clustering Problem). Given a spatial dataset O in an attribute space F, a set of initial clusters $R_{initial}$, a matrix of pairwise spatial neighboring relationship among clusters NC[1] and a fitness function q, we define a problem of spatial agglomerative clustering as finding a set of clusters R_{agglo} that maximizes the fitness function $q(R_{agglo})$ subject to the following constraints:

1. $x_1 \cup x_2 \cup x_3 \cup \ldots \cup x_k = O$.
2. $x_i \cap x_j = \emptyset$, $i \neq j$.
3. *contiguous*(x_i), i=1,2,3,...,k.
4. $\forall x \in R_{agglo}, \exists m \geq 1, \exists x_1' \exists x_2' \exists x_3' \ldots \exists x_m' \in R_{initial}: x = x_1' \cup x_2' \cup x_3' \cup \ldots \cup x_m'$.

In other words, the solution obtained by the clustering algorithm must be exhaustive (all objects in O included), exclusive (non-overlapping clusters), *contiguous* (traversable objects in region x_i) and agglomerative (new clusters are obtained by merging initial clusters).

A pseudo code of the GAC-GEO algorithm is depicted in Fig. 2. The algorithm composes of two phases: The preprocessing phase forms a set of initial clusters $R_{initial}$; Section 3 will discuss several strategies for form initial clusters. The agglomerative phase iteratively merges neighboring clusters in $R_{initial}$ obtaining a final clustering R_{agglo} whose clusters are unions of the clusters in $R_{initial}$. We provide detailed description of the two phases of GAC-GEO algorithm in the following sub-sections. It should be mentioned that a matrix of pairwise spatial neighboring relations a clusters $NC(R_{initial})$ has to be constructed beforehand to perform the agglomerative phase.

1. Preprocessing Phase
   $R_{initial}$ = *createInitialClusters*(O, F, NO, q)
   $NC(R_{initial})$ = *createNeighboringRelationship*($R_{initial}$)
2. Agglomerative Phase
   R_{agglo} = *agglomerateClusters*($R_{initial}$, NC, q)

Fig. 2. An Overview of GAC-GEO Algorithm

2.2.1 GAC-GEO's Agglomerative Clustering Phase

The major role of the agglomerative phase is to merge some clusters in $R_{initial}$ obtained from the preprocessing phase. Selecting merging candidates (clusters) is a challenging task. Traditional agglomerative clustering algorithms like AGNES merge two closest clusters to form large clusters. This merging strategy may result in generating bad clusterings, which is illustrated in Fig. 3. A hierarchical clustering algorithm that uses

[1] Two clusters, x_i and x_j are said to be neighboring clusters if there exists a spatial distance between the clusters that satisfies the user-defined neighboring constraint, *e.g.*, nearest neighboring.

average linkage[2] would merge clusters $C3$ and $C4$, although the two clusters are not contiguous and hence, are not considered neighboring. This example emphasizes the need to disallow merging of non-neighboring clusters in agglomerative clustering.

GAC-GEO, on the other hand, copes with this challenge of determining merging candidates which are clusters that are neighboring, captured by a matrix *NC*. An advantage of considering multiple merging candidates is that the clustering algorithm conducts a wider search in the agglomerative phase. For example, the clusters $C1$ and $C2$ in Fig. 3 are contiguous and considered neighboring. These two clusters would be merged, if it results in enhancing the clustering fitness. $C3$ and $C4$, on the other hand, would never be merged, because they are not neighboring—guaranteeing contiguity of clustering results.

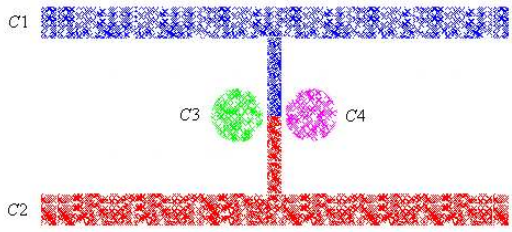


Fig. 3. Merging Elongated Clusters

We provide of the two approaches for agglomerative clustering whose pseudo code is given in Fig. 4 and 5. In general, the algorithm obtains initial clusters with the corresponding spatial neighboring relations and a fitness function q as inputs. Then, it greedily merges a pair of merging candidates that enhance q the most.

```
1. R_agglo AgglomerateClusters (R_initial, NC, and q)
2.       R = R_initial
3.       NC = NC(R_initial)
4.       R_best=R_initial "initial clustering"
5.       WHILE there are merge-candidates (x_i ,x_j) left in NC
6.       BEGIN
7.          Merge the pair of merge-candidates (x_i,x_j), that enhances fitness function q
8.          the most, into a new cluster x': x' = x_i∪x_j
9.          Update R_best by merging x_i and x_j;
10.         Update merge-candidates in NC by removing merging candidates
11.            x_i, and x_j and by adding candidates for x':[3]
12.            Merge-Candidate(x',x)⇔(Merge-Candidate(x_i,x)∨Merge-Candidate(x_j,x))
13.                                   ∧ (Reward(x∪x')≥Reward(x)+Reward(x'))
14.      END
15. RETURN R_best;
```

Fig. 4. A Pseudocode of the First Approach of the Agglomerative Process

In the step of updating merge-candidates in the spatial neighboring relations of clusters *NC* (at line 10 of Fig. 4 and 5):

$$Merge\text{-}Candidate(x',x) \Leftrightarrow Merge\text{-}Candidate(x_i,x) \vee Merge\text{-}Candidate(x_j,x),$$

the neighbors of a newly merged cluster is the union of the neighboring clusters of the clusters that were merged. Consequently, *NC* can be updated incrementally.

---

[2] *Average linkage* uses the average distance between the members of two clusters as its distance function.

[3] This step is used for all clusters in R.

1. R_{agglo} *AgglomerateClusters* ($R_{initial}$, *NC*, and q)
2. $R = R_{initial}$
3. $NC = NC(R_{initial})$
4. $R_{best}=R_{initial}$ "initial clustering"
5. $R_{current} =R_{initial}$
6. WHILE there are merge-candidates (x_i ,x_j) left in *NC*
7. BEGIN
8. Merge the pair of merge-candidates (x_i,x_j), that enhances fitness function q
9. the most, into a new cluster x': $x' = x_i \cup x_j$
10. Update merge-candidates in *NC* by removing merging candidates for
11. x_i, x_j and adding candidates for x':[4]
12. *Merge-Candidate*(x',x) $\Leftrightarrow$ *Merge-Candidate*(x_i,x)$\vee$*Merge-Candidate*(x_j,x)
13. Update $R_{current}$;
14. Update R_{best} if a better clustering was found;
15. END
16. RETURN the best clustering R_{best} found.

Fig. 5. A Pseudocode of the Second Approach of the Agglomerative Process

2.2.2 Complexity of GAC-GEO

The computational time of GAC-GEO varies and depends on the number of objects in the datasets, and on the number of cluster k that were obtained in the preprocessing. However, in general, it requires $O(O(f(n)+O(g(k))+O(k^2*O(q(X)))$ where $O(f(n))$, $O(g(k))$ and $O(k^2+O(q(X))$ are the time complexity of the preprocessing algorithm, the neighboring relationship construction and the agglomeration of k clusters based on a fitness function $q(X)$, respectively. As we will see later different preprocessing techniques have different complexities $f(n)$. Different neighborhood graphs could be used whose construction $g(k)$ cost vary, *e.g.*, one of our current implementations uses Gabriel graphs to identify neighboring clusters whose construction cost are $O(k^3)$.

As far as the agglomerative clustering phase is concerned, because neighborhood graphs are planar graphs we have $O(k)$ initial merge-candidates. In each iteration, i is decreased by one. Therefore, $O(k–i)$ merge-candidates have to be evaluated at the i^{th} iteration and the complexity of total fitness function evaluations can be represented as: $O(k–1)+O(k–2)+O(k–3)+…+1$ which adds up to $O(k^2)$. Considering all these, the time complexity for GAC-GEO's agglomerative phase becomes $O(k^2*O(q(X)))$ where $O(q(X))$ denotes the time complexity of the fitness function. However, this complexity can be significantly reduced by computing the fitness value of a clustering incrementally, where previous fitness function values of unchanged clusters are reused instead of recalculating their fitness values.

3 Discussion of Five Preprocessing Generic Agglomerative Clustering Framework for Geo-Referenced Datasets (GAC-GEO)

There are two motivations of supporting plug-and-play of initial clusters and neighboring relations in GAC-GEO. The first one is to reassemble existing clustering algorithms with minimal human effort with the goal to obtain better clustering results. There exist plenty of clustering algorithms that can generate good clusters, and a use of agglomeration on those clusters can further enhance clustering quality. The second one is to find arbitrarily shaped clusters, which is one of characteristics found in spatial domain. Many clustering

[4] This step is used for all clusters in R.

algorithms, particularly prototype-based, limit themselves to identify non-convex shaped clusters. The agglomerative process can enable identifying arbitrarily shaped clusters.

GAC-GEO provides an architecture of plug-and-play components to users; the three plug-in components of GAC-GEO are: 1) different types of initial clusters, 2) spatial neighboring relationships among clusters, and 3) a plug-in fitness function. Regarding the first component, we define three types of clusters used by GAC-GEO as following:

Prototype-based Clusters. Prototype-based clusters are formed by using a set of representatives. The clustering is created by assigning individual objects to clusters having its nearest representative. The shape of this clusters' type is convex-polygon, which is formed by generating Voronoi cells of clusters.

Grid-based Clusters. Rectangle plays the same role as Voronoi cell but rectangle is much more restricted in shape. Given a set of grid cells in spatial space as a clustering, individual objects are assigned to grid cells covering them. The region contiguity of this type of initial clusters is determined from the boundary of the grid cells to which the cluster belongs, *e.g.*, two grid-based clusters sharing k–1 edges of grid cells in k spatial dimension.

Singleton Clusters. Singleton clusters is also defined as a restricted form of prototype-based clusters in which a single object is assigned to a cluster.

We also illustrate five variations of GAC-GEO to produce different types of initial clusters and spatial neighboring relationships in Table 1. The first variation (No. 1 (a) in Table 1) employs single-object clusters, and the spatial neighboring cluster relations are constructed by computing a proximity graph for the objects in the dataset, *i.e.*, a Gabriel graph of spatial attributes of n objects. The AGglomerative NESted clustering algorithm (AGNES) [10] is similar to this approach except that 1) AGNES does not differentiate between spatial and non-spatial attributes and 2) AGNES uses single linkage and merges the closest pair of clusters, instead of using a plug-in fitness function and neighboring relations to agglomerate clusters.

The second variation (No. 1 (b) in Table 1) uses a representative-based clustering algorithm that generates a set of initial clusters with corresponding representatives. Representative-based clustering algorithm that allows for a plug-in fitness function is used here to guarantee that initial clusters have good quality. Examples of eligible representative-based clustering algorithms are SPAM and CLEVER [20]. The set of representatives obtained is used to construct a proximity graph to identify a neighboring relation among clusters; our current implementation uses Gabriel graphs [26] to construct neighboring relations among clusters. The third variation (No. 1 (c) in Table 1) is similar to the second one except that it randomly selects representatives from a dataset and uses them to create initial clusters; non-representative objects are assigned to the cluster with closest representative to form initial clusters.

The fourth and fifth variation (No. 2 (a) and (b) in Table 1) use grid cells as initial clusters; the fourth variation directly overlays a grid on a dataset, whereas the fifth variation runs a divisive grid-based clustering algorithm to create grid-based initial clusters. It is noted that for these alternatives the initial clusters are simply rectangular. A spatial neighboring relation is constructed by considering the contiguity of grid cells; two cells are neighboring if they share one edge.

Table 1. The Alternative Plug-in Components in GAC-GEO

No.	Preprocessing Methods	Cluster Representatives	Types of Initial Cluster	Spatial Neighboring Clusters Relationships
1	(a) Individual objects	Individual objects defined as initial clusters	Objects	The proximity graph of representatives, *e.g.*, Voronoi cells, Gabriel graphs, the closest clusters, …
	(b) Representative-based clustering algorithms	Representatives of clusters		
	(c) Randomize representatives			
2	(a) Individual grid cells	Grid cell ids	Grid cells	Directly neighboring relationship of grid cells
	(b) Grid-based clustering algorithms			

We also explain in detail the five preprocessing methods to generate initial clusters (as shown in Table 2), which will be used in the experimental section as follows.

Table 2. Five Variations of GAC-GEO

Item	Name	Preprocessing Methods	Spatial Neighboring Clusters Relationships
1	Singleton Clusters (SC)	No. 1 (a) of Table 1	Gabriel graphs
2	CLEVER	No. 1 (b) of Table 1	
3	Randomized Sampling (RS)	No. 1 (c) of Table 1	
4	Grid Partitioning (GP)	No. 2 (a) of Table 1	Directly neighboring relationship of grid cells
5	SCMRG	No. 2 (b) of Table 1	

Preprocessing Method 1: Singleton Clusters (SC) (Item 1 in Table 2)**.** We construct singleton clusters from the original dataset; each cluster contains one object.

Preprocessing Method 2: CLEVER (Item 2 in Table 2). We run CLEVER (Eick et al. 2008), the representative-based clustering that allows for plug-in fitness functions with different parameters: *k'*, neighborhood size, distance measure, *P* and *P'*. Then the clustering with the best fitness value is selected.

Preprocessing Method 3: Randomized Sampling (RS) (Item 3 in Table 2). Given a number of initial clusters *k,* this preprocessing method randomly selects *k* representatives from objects in a dataset, which individually represents a cluster. Then, it assigns non-representative objects to the cluster with its nearest representatives to form initial clusters and calculates a fitness value of the clustering. This process is repeated 1,000 times, and the clustering having maximum fitness value is selected as the set of initial clusters in the agglomerative phase. The numbers of initial clusters *k* are given by users. We can consider SC as applying RS where *k* is equal to the number of objects in the dataset.

Preprocessing Method 4: Grid Partitioning (GP) (Item 4 in Table 2). We generate initial clusters using user-defined grid resolutions; different resolutions produce different

numbers of initial clusters, *e.g.*, a grid resolution of 12x12 generates 144 initial clusters including non-empty clusters. Moreover, empty grid cells are eliminated.

Preprocessing Method 5: SCMRG (Item 5 in Table 2). We run SCMRG (Eick et al. 2006)—a divisive grid-based clustering algorithm that allows for plug-in fitness functions—to generate initial clusters. SCMRG input parameters include: minimum cell size, dividing resolution, initial resolution and looking ahead level.

4 Experimental Results

We conducted three experiments to demonstrate the capabilities of the proposed agglomerative clustering framework GAC-GEO. The first experiment evaluated the performance of GAC-GEO among its variations using an artificial dataset. The second and third experiment evaluated the performance of GAC-GEO using a real-world problem: identifying high variance hotspots and high correlation hotspots of earthquakes. All of the experiments were conducted on a desktop with Intel(R) Pentium(R) M 1.6 GHz processor with 2.0 GB of RAM.

Experiment 1. Supervised Clustering using GAC-GEO with Purity Fitness Function on a Complex9 Dataset.

Dataset. The Complex9 dataset (DMML datasets 2008) depicted in Fig. 6 had several unique characteristics: the shape of some clusters was non-convex, some clusters were nested, and clusters were very pure and highly dense. The dataset contained 3,032 objects with 3 attributes: *x*, *y* and *class_id*, where *x* and *y* were coordinates and *class_id* had 9 classes.

Experimental Objectives. There were two main objectives of this experiment: to demonstrate the flexibility of GAC-GEO with respect to different sources of input provided by different preprocessing methods and to perform a comparative study on the variations of GAC-GEO on an artificial dataset. In particular, for the latter objective, this experiment demonstrated the ability of GAC-GEO to form arbitrarily shaped clusters, and investigated the capabilities and limitations of GAC-GEO based on the quality of initial input clusters and final output clusters.

We applied five preprocessing methods to form initial input clusters, as discussed in Section 3, including randomized sampling (RS), singleton clusters (SC), grid partitioning (GP), the SCMRG clustering algorithm (Eick et al. 2006) and the CLEVER clustering algorithm (Eick et al. 2008). In order to create a comparable set of initial clusters, we anticipated an approximate number of initial clusters k using four different granularity levels (resolutions) using the scaling factor $f = \{1, 3, 9, 27\}$ in Eq. (3):

$$k = \frac{n}{f} \tag{3}$$

where n is the number of objects in the dataset. Therefore, k={3032, 1010, 336, 112}. For two-dimensional datasets, we computed a number of grid cells g proportional to $\sqrt{k}$ obtaining g={55, 31, 18, 10}. It should also be noted that SCMRG did not require a number of initial input clusters, whereas CLEVER likely generated a number of initial clusters different from the number of clusters generated by RS, SC, and GP. We set SCMRG parameters as follows: minimum cell size=0.1, dividing resolution={2, 2},

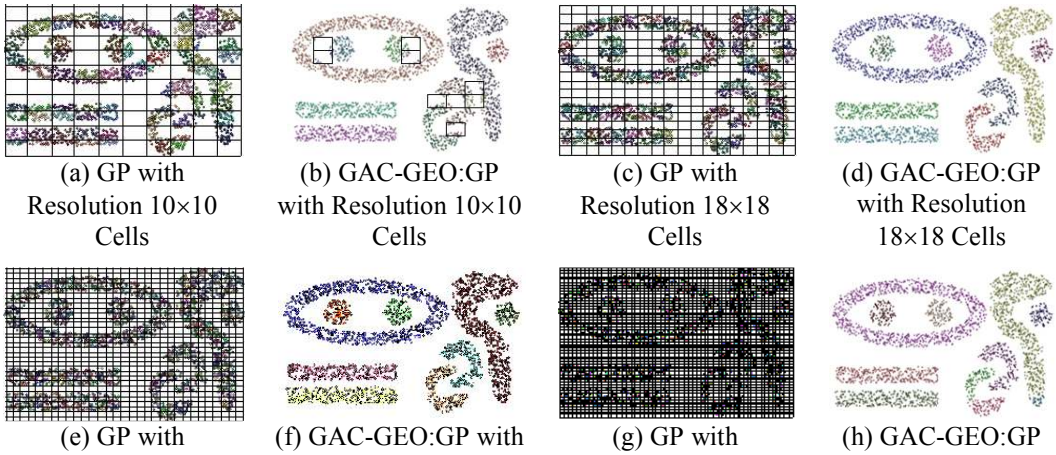
looking ahead level=2; we set CLEVER parameters as follows: β=1.1, η=1.0, pTh=0.7, neighbor size=3, distance measure=2, p=10, p'=10, k={112, 336, 1010}. After obtaining initial input clusters from the 5 preprocessing methods, we applied GAC-GEO to those initial clusters with the following parameters: pTh={0.7, 0.8, 0.9}, η=1.0, β={1.0000000001, 1.01, 1.1, 1.2} obtaining 12 different experimental results.

Employed Interestingness Function. This interestingness function rewards class purity in individual clusters:

$$i(x) = \begin{cases} (MAX_{j \in J}(purity_j(x)) - th)^{\eta} & MAX_j(purity_j(x)) > th \\ 0 & otherwise. \end{cases} \quad (4)$$

$purity_j(x)$ denoted the normalized ratio of the number of objects belonging to class $j \in J$ to the number of objects in the cluster x. Then, the class purity of the cluster x was the maximum purity calculated from all classes in J. η controlled the level of importance of purity; the greater value η was, the more importance of purity we emphasized on the clusters obtained.

Experimental Results. The experimental results showed that given a sufficient number of initial clusters, GAC-GEO was capable of identifying arbitrarily shaped clusters irrespective of the initial input clusters generated by the five preprocessing methods. Fig. 11 depicted four different initial clusters generated by GP and the corresponding final clusters generated by GAC-GEO. Due to the agglomerative nature of GAC-GEO global purity never decreased; *in general,* the number of minority class examples remains the same or increases as clusters were merged—the objects in original clusters stay together. Since the quality of final clusters is directly influenced by the quality of initial clusters, obtaining pure initial clusters is important. According to our experimental results depicted in Fig. 7, using a number of initial clusters greater than 335 in RS and using a number of initial clusters larger than 240 in GP could create initial clusters with 100% purity.



(a) GP with Resolution 10×10 Cells | (b) GAC-GEO:GP with Resolution 10×10 Cells | (c) GP with Resolution 18×18 Cells | (d) GAC-GEO:GP with Resolution 18×18 Cells

(e) GP with Resolution 31×31 Cells | (f) GAC-GEO:GP with Resolution 31×31 Cells | (g) GP with Resolution 55×55 Cells | (h) GAC-GEO:GP with Resolution 55×55 Cells

Fig. 6. Initial Clusters on Complex9 Dataset Preprocessed by GP, and the Corresponding Final Clusters Post-processed by GAC-GEO

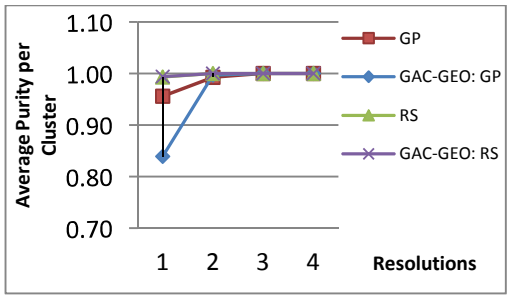


Fig. 7. A Comparison of Average Purity Per Cluster Before and After Applying GAC-GEO to Two Preprocessing Methods: GP and RS

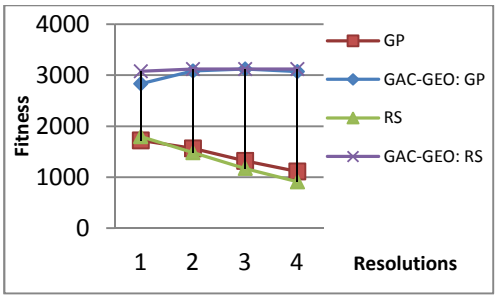


Fig. 8. A Comparison of Fitness Before and After Applying GAC-GEO to Two Preprocessing Methods: GP and RS

Additional evidence depicted in Fig. 8 illustrated the enhancement of fitness values before and after applying GAC-GEO to RS and GP. We found that, if used with proper parameter setting, SC, SCMRG and CLEVER could generate pure initial clusters. In summary, GAC-GEO could easily detect natural clusters from these high quality inputs.

Experiment 2. Using GAC-GEO for Variance Analysis and Co-location Mining[5].

Dataset. We used an earthquake dataset available on the website of the U.S. Geological Survey Earthquake Hazards Program, http://earthquake.usgs.gov/. We sampled 4,132 examples of earthquakes dated from January 1986 to November 1991. Information extracted included the location (longitude, latitude) and the depth (kilometers) of the earthquake. The statistics of the dataset were calculated as follows: $Mean_{depth}$=48.518, $StdDev_{depth}$=87.589 and $Variance_{depth}$=7,671.809. Fig. 9 depicts the dataset, whose depth was divided equally into 20 ranges. Looking at the dataset we could visually identify three areas with high variance in earthquake depth in the south Pacific Ocean, and in the eastern and southeastern Asia.

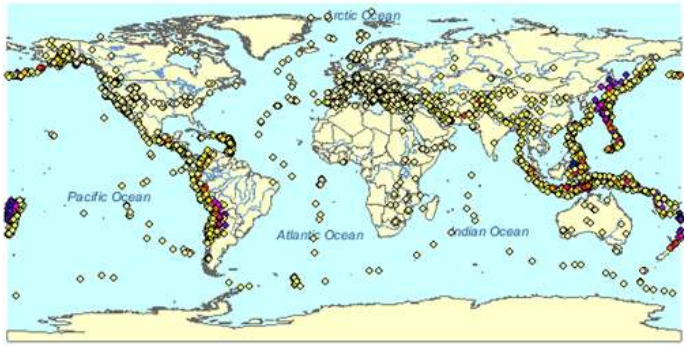


Fig. 9. A Visualization of the Earthquake Dataset; Depth of Earthquakes Is Discretized into 20 Ranges from Shallow (Light Yellow) to Deep (Dark Purple).

Experimental Objectives. In this case study, we were interested in discovering hotspots in geographical space where deep earthquakes were in close proximity to shallow earthquakes. The High Variance function given in Eq. (4) was employed to find such regions. The objectives of this experiment were similar to the previous ones except that we evaluated GAC-GEO on a real dataset. In particular, we depicted how GAC-GEO enhances the quality of the initial clusters, and analyzed how different preprocessing methods affected the quality of the final clustering. We compared the quality of initial

[5] In this experiment, GAC-GEO was used to identify high variance clusters with respect to earthquake depth for an earthquake dataset. Due to the fact that AGNES clusters datasets solely on distance information it is impossible to use AGNES for such a problem.

and final clusters using the following measures: fitness, reward per object (*RO*), average variance per cluster (*AVC*), number of clusters and the visualization of top five interesting regions discovered by the different variations. Fitness q defined in Eq. (1) measured quality of a clustering. *RO* measured the average reward per object of positive reward clusters:

$$RO(X) = \frac{q(X)}{k\_po}, \tag{5}$$

where *k_po* was the total number of objects in all positive reward clusters. The *AVC* measures raw interestingness, which was the average variance of earthquake depth for all clusters obtained:

$$AVC(X) = \frac{\sum_{x=1}^{k} Var(x)}{k\_pc}, \tag{6}$$

where *Var*(*x*) was the variance of earthquake depth in a cluster *x,* and *k_pc* was the number of positive reward clusters.

We applied the five preprocessing methods to form initial clusters as performed in the first set of experiments. Details of the preprocessing methods are as follows. For GP, we used four grid resolutions: 12×12, 21×21, 37×37, and 64×64 to produce a different number of initial clusters: 104, 192, 344, and 572, respectively. The rationale for choosing the specified grid resolutions was discussed in the first set of experiments. Then, all empty grid cells were eliminated. For RS, the numbers of initial clusters *k* were as follows: 1377, 459, and 153. For SCMRG, we ran SCMRG using various parameter settings and chose one that gave the best fitness value as initial clusters. The best parameter setting of the SCRMG clustering algorithm was minimum cell size=0.0001, dividing resolution=2, initial resolution={2, 2}, looking ahead level=6, β=1.01, η=1.0, and variance threshold (*th*)=1.01. Finally, we ran CLEVER with different parameter settings and selected the one having the best fitness value, which had neighbor size=3, distance measure=2, *P*=10, *P'*=10, β=1.01, η=1.0, *th*=1.1. After that, we ran GAC-GEO to generate final clusters by plugging in a High Variance function defined in Eq. 7 with different parameter settings: β={1.01, 1.2}, η=1.0 and *th*={1.1, 1.2, 2.0}; the higher beta value, the larger regions were produced by GAC-GEO. In this paper, we discussed one of the good results using β=1.2 and *th*=2.0.

Employed Interestingness Function. High variance function maximizes variance of a numeric attribute (Rinsurongkawong and Eick 2008). In other words, we were interested in areas where extremely high and low values of the attribute were in close proximity. The interestingness function was defined as follows:

$$i(x) = \begin{cases} \left(\frac{Var(x,z)}{Var(O,z)} - th\right)^{\eta} & \frac{Var(x,z)}{Var(O,z)} > th \\ 0 & otherwise, \end{cases} \tag{7}$$

where

$$Var(x,z) = \frac{1}{|x|-1}\sum_{o \epsilon x}(z(o) - \mu_z)^2. \tag{8}$$

z denotes earthquake's depth in the dataset O; th>1 was a threshold; 0<η<∞ is a form parameter; μ_z was an average value of z for all objects in the cluster x; *Var*(x,z) denoted a regional variance of z in a cluster x whereas *Var*(O,z) denoted a global variance of z in the dataset.

Experimental Results. First, we analyzed how different resolutions of initial clusters influence final clusters. For RS and GP, we defined 4 resolutions (1 to 4) from rough (k=153 and grid resolution=12×12) to fine granularity (k=4,132 and grid resolution=64×64). As shown in Fig. 10, the large initial clusters had higher fitness q than the small ones (as depicted by the blue and green bars). As shown in Fig. 11, the results measured by using RO[6] were also comparative except the result generated by GP using the third resolution. However, as far as final clusters were concerned, small initial clusters led to higher q and RO of the final clustering (as depicted by the red and purple bars in Fig. 10 and 11, respectively). Fig. 10 and 11 also illustrate the improvement of clustering quality in terms of fitness q and RO, when GAC-GEO was applied to the initial clusters generated by RS and GP. By employing the two simple preprocessing methods, GAC-GEO obtained better final clusters when giving small initial clusters.

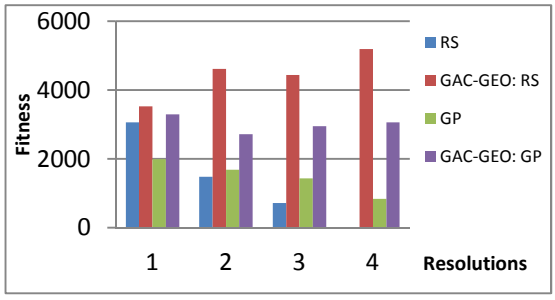


Fig. 10. Comparison of Fitness Values of Clustering Before and After Applying GAC-GEO to Four Resolutions of RS and GP

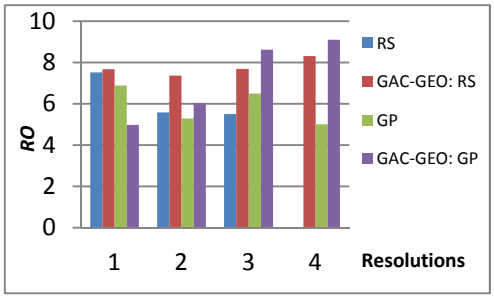


Fig. 11. Comparison of RO Before and After Applying GAC-GEO to Four Resolutions of RS and GP

Next, we compared the results of clusters before and after applying GAC-GEO to initial clusters generated by the five preprocessing methods. GAC-GEO and the preprocessing methods with different parameter settings produced a large number of results, and therefore, we chose the best results with the highest fitness q from each preprocessing method. The objective was to see which preprocessing approach performed the best in choosing initial clusters for GAC-GEO. As seen in Fig. 12, CLEVER, SCMRG, and GP were the top three preprocessing methods that gave the highest fitness q of initial clusters. After the agglomerative step, GAC-GEO using the initial clusters of SC led to the best result. Taking a broader view, using representative-based approaches (SC, CLEVER and RS) to create initial clusters, GAC-GEO obtained the better results than using grid-based approaches (GP and SCMRG). Regardless of the size of clusters, GP, RS, and CLEVER were the top three preprocessing methods that gave the highest reward per object of initial clusters, as depicted in Fig. 13. (This measure considers positive reward clusters only.)

However, in the agglomerative step, the result of this measure was similar to the one measured by using fitness. Considering AVC of clustering, CLEVER and RS are comparative methods that returned superior initial clusters, as depicted in Fig. 14. However, in the agglomerative step, GAC-GEO with CLEVER and SCMRG became comparative methods that returned superior final clusters compared to SC, RS and GP. Fig. 15 showed the number of initial and final clusters before and after employing GAC-GEO to the five preprocessing methods.

---

[6] RO is calculated using the total number of objects in positive reward clusters only.

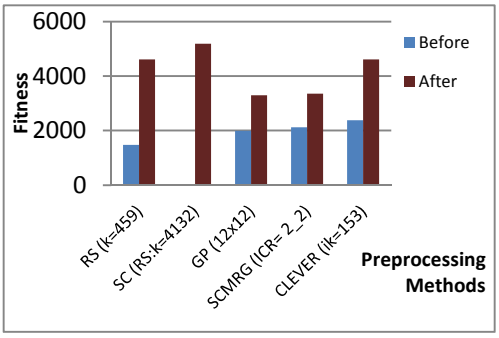


Fig. 12. Comparison of Fitness Values of Clustering Before and After Applying GAC-GEO to the Five Preprocessing Methods

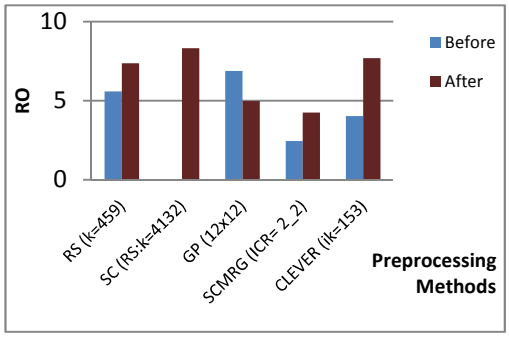


Fig. 13. Comparison of *RO* Before and After Applying GAC-GEO to the Five Preprocessing Methods

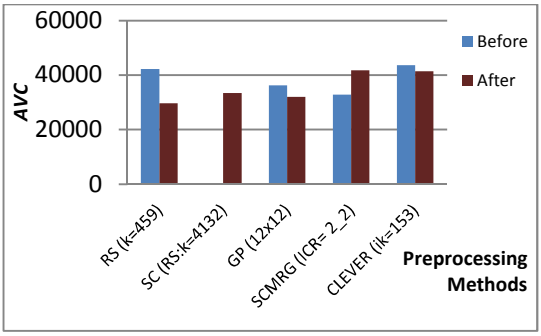


Fig. 14. Comparison of *AVC* Before and After Applying GAC-GEO to the 5 Preprocessing Methods

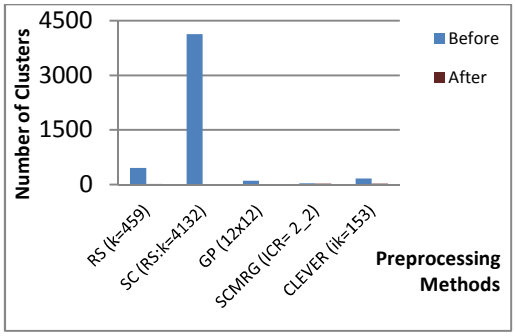


Fig. 15. Number of Clusters Before and After Applying GAC-GEO to the 5 Preprocessing Methods

Next, we performed a comparison using the same resolution. Since SCMRG always returned a number of initial clusters less than 100, we only provide the comparison with respect to the roughest resolution. As depicted in Fig. 16 and 17, the results were identical to the previous comparison; GAC-GEO gave the better result, when representative-based approaches (RS and CLEVER) were used in creating initial clusters, as compared to grid-based approaches (GP and SCMRG).

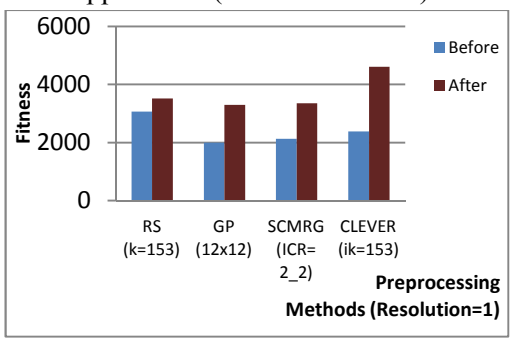


Fig. 16. Comparison of Fitness Values of Clustering Before and After Applying GAC-GEO to the Four Preprocessing Methods (Resolution=1)

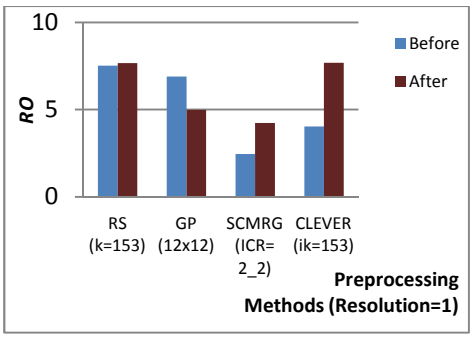


Fig. 17. Comparison of *RO* Before and After Applying GAC-GEO to the Four Preprocessing Methods (Resolution=1)

In summary, for all the variations, GAC-GEO was capable of forming larger clusters while enhancing the clustering quality. In particular, GAC-GEO could improve the quality of the final result (in terms of fitness) even with initial clusters generated by simple preprocessing methods. For example, as shown in Fig. 18, for RS, GAC-GEO on average increased the quality of the final result 3.49 times that of the initial clusters, and for GP, it on average increased the quality of the final result 3 times that of the initial

clusters. Even for the two clustering algorithms SCMRG and CLEVER that produced high quality initial clusters, GAC-GEO could increase quality of the final result 1.7 times and 2.1 times that of the initial clusters. It should be noted that SCMRG weeded out low quality clusters as outliers, and hence, the fitness value of the final results was small compared to the other results. However, for SC, GAC-GEO merged very small clusters, and thus it obtained the highest fitness value. Fig. 19 showed the enhancements of GAC-GEO using *RO*. Fig. 20 showed that the *AVC* of both initial and final clusters generated by the five preprocessing methods. It was observed that *AVC*s obtained from all of the five preprocessing methods were greater than the global variance of earthquake depth (7,672) in the dataset.

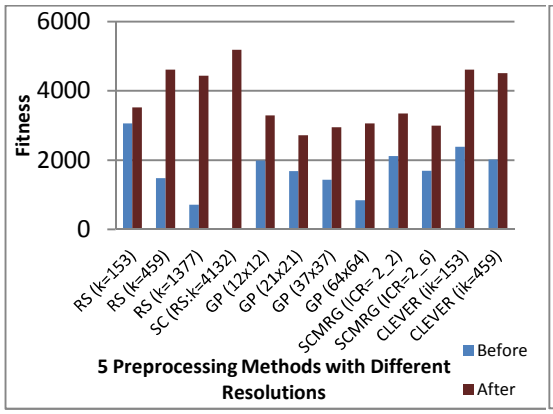


Fig. 18. Comparison of Fitness Before and After Applying GAC-GEO to the Five Preprocessing Methods with Different Resolutions

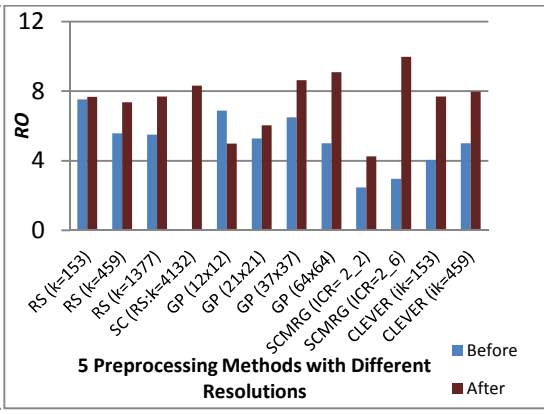


Fig. 19. Comparison of *RO* Before and After Applying GAC-GEO to the Five Preprocessing Methods with Different Resolutions

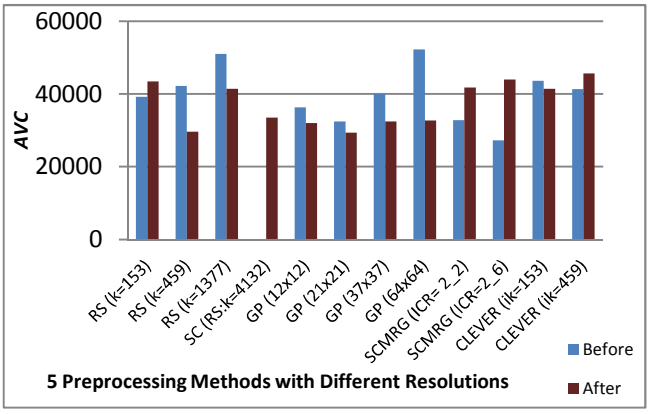
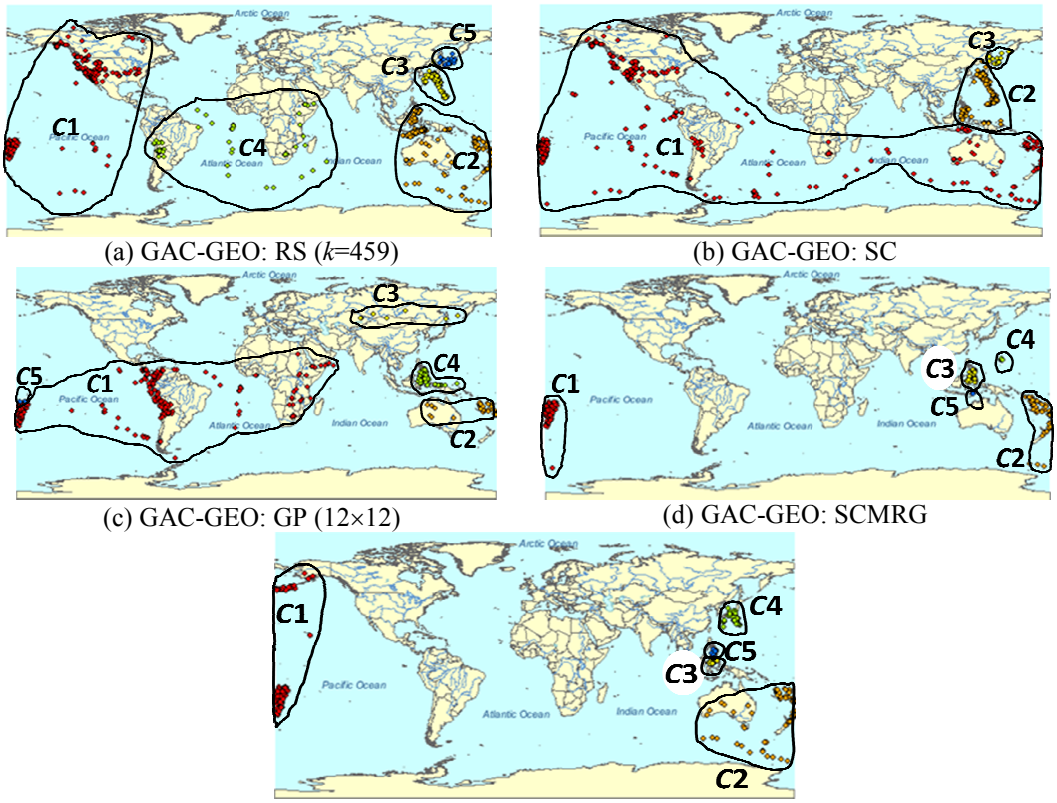


Fig. 20. Comparison of *AVC* Before and After Applying GAC-GEO to the 5 Preprocessing Methods with Different Resolutions

Fig. 21 illustrates hotspots obtained from the five variations of GAC-GEO by querying the top five regions as per the reward. Overall, after applying GAC-GEO, we were able to discover four regions of the dataset where deep and shallow earthquakes are located in proximity (see also Fig. 9). GAC-GEO: CLEVER returned regions with the scope most similar to the ground truth (Fig. 21 (e)). GAC-GEO: RS and GAC-GEO: GP returned similar results in the larger region (Fig. 21 (a) and (c), respectively). GAC-GEO: SCMRG returned a similar result in the smaller regions (Fig. 21 (d)). Finally, GAC-GEO: SC (Fig. 21 (b)) agglomerated the interesting regions into 3 big regions. Considering variance and size of the top five hotspots (shown in Fig. 22), almost all of them were comparably high except GAC-GEO: SCMRG, which identified two small, very high variance regions.



(a) GAC-GEO: RS (k=459)

(b) GAC-GEO: SC

(c) GAC-GEO: GP (12×12)

(d) GAC-GEO: SCMRG

(e) GAC-GEO: CLEVER

Fig. 21. Visualization of Top Five Regions Produced by GAC-GEO Using the Five Preprocessing Methods (Ordered by Reward)

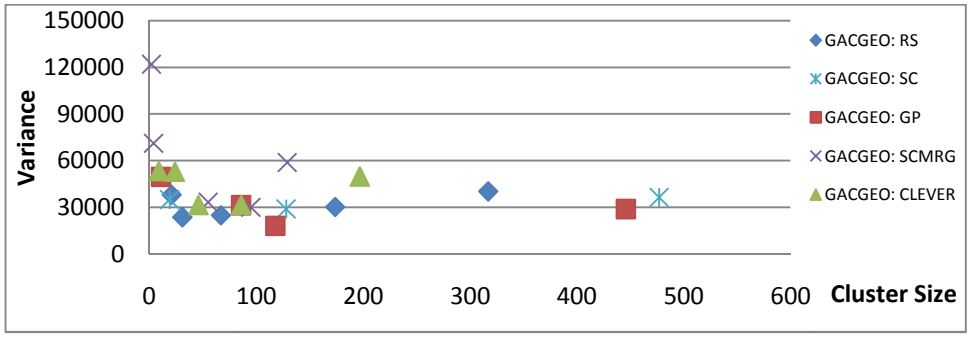


Fig. 22. Variance and Size of Top Five Positive Reward Regions Generated by GAC-GEO with the Five Preprocessing Methods

Experiment 3. Apply GAC-GEO on an Earthquake Dataset in Identifying Regions with High Correlation[7].

Dataset. According to the earthquake dataset in the experiment 2, we queried two attributes: A_0=depth and A_1=severity and then pre-computed z-score: $z_{A_0}(o)$ and $z_{A_1}(o)$. Fig. 23 shows the dataset used. The statistics of the dataset are calculated as follows: $\text{Mean}_{\text{ProductOfZ-score}}$=0.194, $\text{Stdev}_{\text{ProductOfZ-score}}$=1.014, $\text{Variance}_{\text{ProductOfZ-score}}$=1.029 and Correlation ($\text{z-score}_{\text{depth}}$,$\text{z-score}_{\text{severity}}$)= −0.113.

[7] In this experiment, GAC-GEO was used to identify regions with high correlation for an earthquake dataset. Due to the fact that AGNES clusters datasets solely on distance information it is impossible to use AGNES for such a problem.

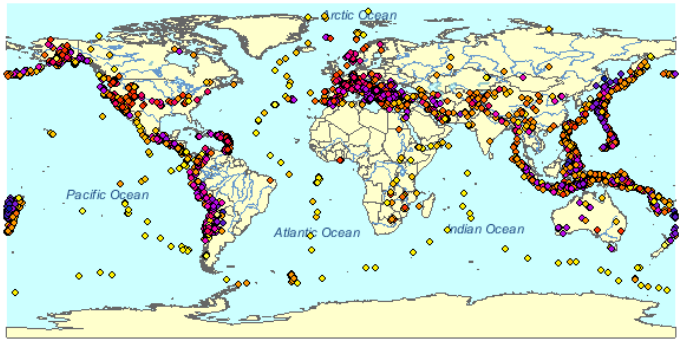


Fig. 23. Earthquake Dataset; the Product of z-score of Depth and Severity of Earthquakes is Discretized into 20 Ranges from Low (Light Yellow) to High (Dark Purple)

Experimental Description. Objective of this study was to see which preprocessing methods suitably work with GAC-GEO in finding regions with high correlation between two continuous attributes: depth and severity of earthquakes. In general, we compared the results of clusters before and after applying GAC-GEO to the initial clusters generated by the five preprocessing methods. We chose the best results from each variation that gave the highest fitness value. We compared the quality of initial and final clusters using the following measures: the fitness, the reward per object (*RO*), the average raw interestingness of a clustering (*AP*1), the average raw interestingness of a cluster (*AP*2), number of clusters and the visualization of top five interesting regions discovered by the different variations. The formula of *AP*1 and *AP*2 were given in Eq. (9) and (10), respectively,

$$AP1(X) = \frac{\sum_{i=1}^{k\_pc} \left(\frac{\left| \sum_{o=1}^{m} z_a z_b \right|}{size(x_i)} \right)}{k\_pc} \tag{9}$$

$$AP2(x) = \frac{\sum z_a z_b}{size(x)} \tag{10}$$

where z_a and z_b were the z-score of depth of earthquake and severity, respectively, and k_pc was the number of positive reward clusters.

We applied the five preprocessing methods to form initial clusters as performed in the other experiments. Details of the preprocessing methods were the same as in the experiment 2. For SCMRG, the parameter settings that gave the best fitness value were that minimum cell size=0.0001, dividing resolution=2, initial resolution={2, 6} looking ahead level=2, β=1.01, η=1.0 and z threshold (z_{th})=0.3, and for CLEVER, the parameter settings were that neighbor size=3, distance measure=2, P=20, P'=20, β=1.01, η=1.0, z_{th} =0.3. Then, we ran GAC-GEO by plugging in the interestingness function in Eq. (12) with the parameter setting: β=1.2, η=1.0 and z_{th} =0.3.

Employed Interestingness Function. This intererestingness function (Ding et al. 2008) sought for localized regions with high positive/negative correlation; the regions where continuous non-spatial attributes of objects attained together the values from the wings of their respective distributions.

Given a set of continuous attributes $A=\{A_1,A_2,A_3,\ldots,A_q\}$ the interestingness of an object $o \in O$ was measured as follows:

$$i(A,o) = \prod_{j=1}^{q} z_{A_j}(o). \quad (11)$$

$z_{A_j}(o)$ was the z-score of the continuous attribute A_j. Objects with $|i(A,o)|>>0$ were clustered as hotspots, where the attributes in A happened to attain sum of products of z-scores. The products of z-scores were considered an estimate of a correlation of A. Then, the definition of interestingness to regions was extended as the absolute value of the sum of the interestingness of the objects belonging to it:

$$i(A,x) = \begin{cases} \left(\frac{|\sum_{o \in r} i(A,o)|}{size(x)} - z_{th}\right) & if \frac{|\sum_{o \in r} i(A,o)|}{size(x)} > z_{th} \\ 0 & otherwise. \end{cases} \quad (12)$$

Eq. (12) estimates the absolute value of a regions' correlation. In this formula, the interestingness threshold z_{th} was introduced to weed out regions with $i(x)$ close to 0, which prevented clustering solutions from containing only large clusters of low interestingness.

Experimental Results. As seen in Fig. 24, CLEVER, RS, and GP are the top three preprocessing methods that gave the highest fitness value of the initial clusters. After the agglomerative step, GAC-GEO: SC returned the highest fitness values. In a broader view, using representative-based approaches (SC, CLEVER and RS) in creating initial clusters, GAC-GEO gave better results than those obtained using grid-based approaches (GP and SCMRG). Regardless of the size of clusters, CLEVER, GP, and SCMRG were the top three preprocessing methods that gave the highest reward per object of initial clusters, as depicted in Fig. 25. (This measure considered positive reward clusters only.)

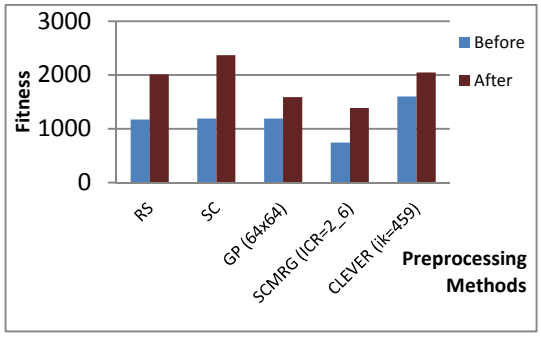


Fig. 24. Comparison of Fitness Values of Clustering Before and After Applying GAC-GEO to Five Preprocessing Methods

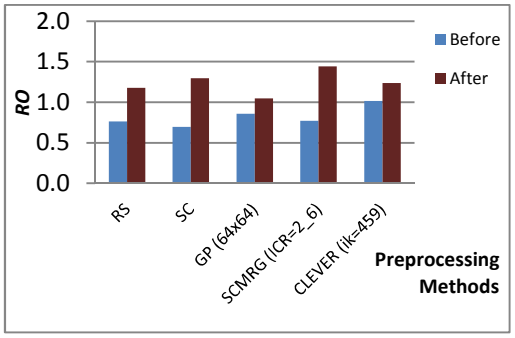


Fig. 25. Comparison of *RO* Before and After Applying GAC-GEO to Five Preprocessing Methods

However, in the agglomerative step, GAC-GEO: SCMRG, GAC-GEO: SC and GAC-GEO: CLEVER returned the top three best results. Consider *AP*1, the average difference of correlation of clusters before and after applying the four variations of GAC-GEO slightly decreased (Fig. 26); this result showed that the agglomerative process sometimes created a tradeoff between the quality and size of clusters. However, the tradeoff was not significant (the average difference of *AP*1 was less than 0.2). On the other hand, GAC-GEO could effectively increase the size of clusters (Fig. 27); it was able to reduce more than 83% of the number of initial clusters when applying with representative-based approaches, and was able to reduce 69% and 29% for GP and SCMRG, respectively. Considering all of the measures, CLEVER is the best preprocessing method that returned superior initial clusters and results in superior final clusters when applying to GAC-GEO.

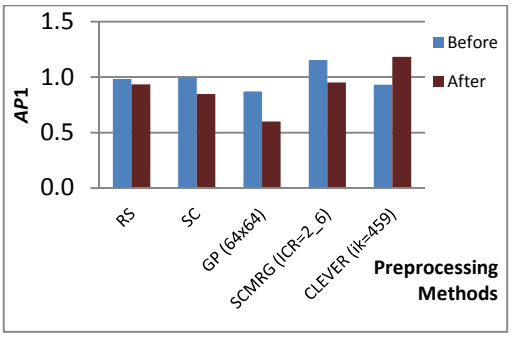


Fig. 26. Comparison of *AP*1 Before and After Applying GAC-GEO to Five Preprocessing Methods

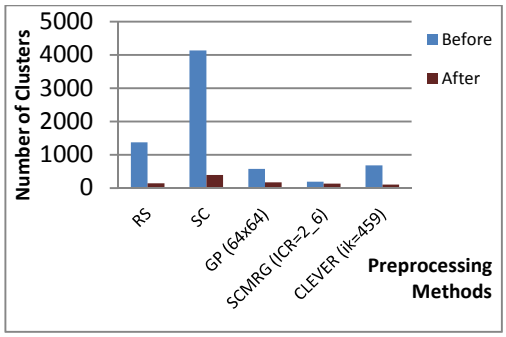


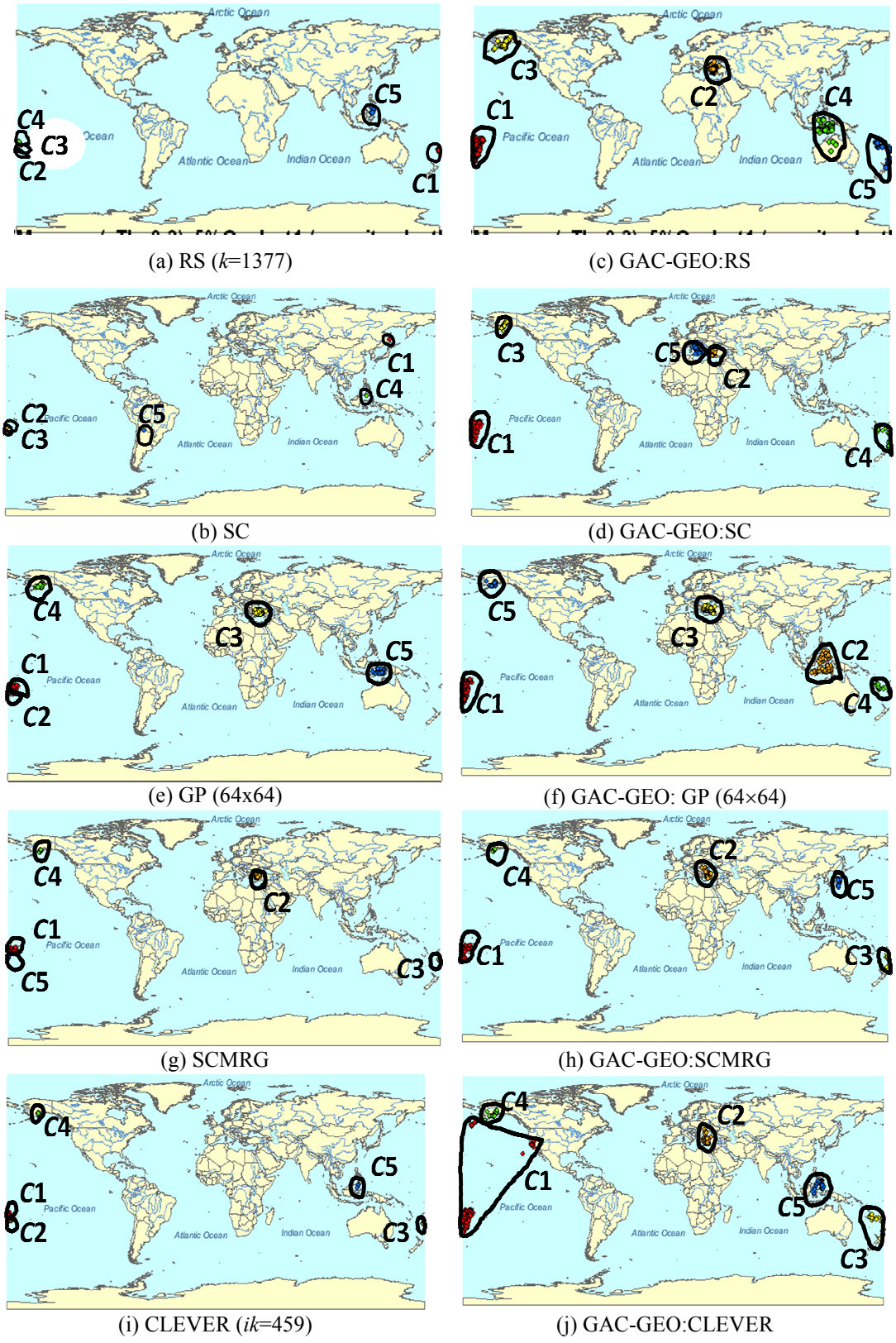
Fig. 27. Comparison of Number of Clusters Before and After Applying GAC-GEO to Five Preprocessing Methods

Fig. 28 illustrated the hotspots obtained from the five variations of GAC-GEO by querying the top five regions as per reward. Results obtained before and after applying GAC-GEO were arranged on the left and right panel, respectively. In summary, the five variations of GAC-GEO found similar regions which had the extreme density of extremely high (low) severity and depth of earthquakes. SC and SCMRG identified fewer regions; especially for SCMRG, the size of the hotspots was smaller than those found by other approaches. These figures also convinced us of the capability of GAC-GEO to form larger clusters. The interesting point was that after the agglomerative process, GAC-GEO further identified new hotspots in the top ranks. This was because some small hotspots in close proximity were merged to form the larger ones.

The comparison of the top five regions in Table 3 showed in particular that GAC-GEO was able to identify high positive correlation of earthquakes' severity and depth in the southern part of Alaska (except for GAC-GEO: SCMRG, which due to the small size of the region, identified this hotspot as a negative correlation). The other four hotspots were considered positive correlation.

Table 3. *AP*2 and Cluster Size of the Top Five Regions Produced by GAC-GEO Using the Five Preprocessing Methods: Ordered by Reward

Algorithms\ (*AP*2, size)	**$Rank_1$= $region_{red}$**	**$Rank_2$= $region_{orange}$**	**$Rank_3$= $region_{yellow}$**	**$Rank_4$= $region_{green}$**	**$Rank_5$= $region_{blue}$**
RS(*k*=1377)	(5.64, 7)	(5.80, 5)	(7.49, 4)	(5.26, 5)	(4.22, 6)
GAC-GEO: RS	(2.63, 118)	(0.52, 284)	(-0.64, 155)	(0.82, 92)	(1.81, 38)
SC	(13.99, 1)	(12.41, 1)	(-10.86, 1)	(9.38, 1)	(-8.26, 1)
GAC-GEO: SC	(2.80, 111)	(0.62, 222)	(-0.77, 135)	(2.47, 27)	(0.63, 125)
GP (64×64)	(4.66, 30)	(4.50, 20)	(-0.67, 80)	(1.13, 39)	(5.68, 7)
GAC-GEO: GP	(2.42, 123)	(0.81, 159)	(0.41, 391)	(1.78, 32)	(-0.66, 100)
SCMRG	(4.44, 64)	(0.49, 262)	(5.43, 11)	(-0.78, 62)	(1.03, 31)
GAC-GEO: SCMRG	(4.77, 37)	(5.25, 20)	(4.98, 12)	(1.42, 23)	(6.92, 4)
CLEVER (*ik*=459)	(5.14, 29)	(5.40, 19)	(5.86, 9)	(-0.83, 56)	(6.46, 5)
GAC-GEO: CLEVER	(1.67, 175)	(0.52, 281)	(1.92, 36)	(-0.71, 106)	(1.39, 45)



(a) RS (k=1377)

(c) GAC-GEO:RS

(b) SC

(d) GAC-GEO:SC

(e) GP (64x64)

(f) GAC-GEO: GP (64×64)

(g) SCMRG

(h) GAC-GEO:SCMRG

(i) CLEVER (ik=459)

(j) GAC-GEO:CLEVER

Fig. 28. Top Five Regions Produced by GAC-GEO Using the Five Preprocessing Methods (Ordered by Reward)

In general, the GAC-GEO framework integrates fitness functions, preprocessing methods and clustering algorithms each of which needs parameters. This raises the question how those parameters are determined when using GAC-GEO? As far as fitness function parameters are concerned, their values are chosen in close collaboration with a

domain expert—which parameters are selected is determined based on what kind of clusters the domain expert is interested in. For example, the parameter β, which is used in all experiments discussed earlier, determines how much reward is put on cluster size; consequently, if the domain expert is interested in obtaining a lot small clusters containing a few objects, a small value for β would be selected.

As far as parameters of preprocessing methods and clustering algorithms are concerned, their values are chosen maximizing the plug-in fitness function q—the fitness function serves as with the "ground truth". Basically, we select the parameter setting that leads to a clustering X for which $q(X)$ is higher than $q(X')$ for other clusterings X' that were obtained using other parameter setting. This leads to question how these parameter settings are actually determined. In the presented work, we employed the following three strategies to select preprocessing and clustering algorithm parameters:

1. Exhaustive approaches that a fixed number of values for each parameter are preselected manually, and then all combinations of parameter values are evaluated by running the clustering algorithm, and selecting the parameter combination that leads to the best clustering.
2. Round robin approaches that change one parameter while keeping all the other parameters fixed; this process continues until there is no more improvement in $q(X)$.
3. "One after the other" approaches, which select values of a subset of the parameters first, and then the values of the remaining parameters are chosen without backtracking.[8]

In the presented experimental results we mostly used methods 1 and 3 for selecting parameters. However, it is worth mentioning that in the recent Netflix contest more sophisticated techniques (Pilotte and Chabbert 2009), which can be downloaded at Netflix Prize official website (Netflix 2009), such as the Nelder-Mead Simplex Method (Nelder and Mead 1965), have been successfully used to select prediction algorithm parameters, and it might be interesting to evaluate their usefulness in conjunction with GAC-GEO.

5 Related Work

In general, the idea of two-phase clustering is to create a set of small clusters using some preprocessing methods and then, merge them to form larger clusters. Most research emphasizes the preprocessing phase while keeping the agglomerative phase simple, *i.e.*, merging the closest pair of clusters. As far as the preprocessing phase is concerned, the Hybrid Cell Density Clustering method (HyCeltyc) (Otoo et al. 2001) proposed by Otoo *et al.* obtains initial clusters by sampling and performing dimensional reduction to identify cell-density clustering of samples in lower dimensional space. Lin and Chen (Lin and Chen 2002) obtain initial clusters by running representative-based clustering, and Zhong and Ghosh (Zhong and Ghosh 2003) partition data using a model-based partitioning clustering algorithm. Xiong *et al*. use hyper-clique patterns to define the initial clusters to preserve interesting patterns used in hierarchical clustering with pattern preservation algorithm (Xiong et al. 2009). As far as the agglomeration phase is concerned, the algorithm proposed by Lin and Chen (Lin and Chen 2002) employs different merging criterion, *cohesion* rather than *single linkage* whereas Zhong whereas

[8] For example, in most experiments, we selected the preprocessing methods parameters prior to selecting the agglomerative clustering algorithm parameters.

Ghosh (Zhong and Ghosh 2003) propose modified *Kullback-Leibler* (KL) distances to be employed with the *single linkage*.

In comparison, GAC-GEO provides a very generic preprocessing phase that it accepts different forms of initial clusters, such as grid cells, clusters of objects and polygons. Therefore, it is capable to interface with any algorithms to generate initial clusters. Many clustering algorithms can be considered variations of GAC-GEO, such as AGNES, MOSAIC and some divisive-and-agglomerative grid-based algorithms. Agglomerative Nesting algorithm (AGNES) (Kaufman and Rousseeuw 1990) uses *single linkage* and merges the closest pair of clusters. MOSAIC (Choo et al. 2007) applies a representative-based clustering algorithm to form initial clusters. It forms a neighboring relationship using Gabriel graphs and employs plug-in fitness functions. Regarding the agglomerative phase, GAC-GEO considers all spatial neighboring clusters, and selects to merge the pair that returns the maximum fitness value. This merging criterion differs from the others because GAC-GEO aims to compromise the optimization of both spatial and non-spatial attributes space, in form of spatial distance and fitness, separately and simultaneously.

One challenge that we focus in this paper is capability to identify arbitrarily shaped clusters. In particular to our case study, we aim to identify interesting non-convex shaped clusters having similar characteristics, such as high-variance and high-correlation with respect to non-spatial attribute(s). 2 categories of clustering algorithms providing this capability are hierarchical and density-based. Generally hierarchical clustering algorithms applying *single linkage* claim to provide the same capability[9] (Kaufman and Rousseeuw 1990). Chaoji *et al*. proposed SPARCL that runs a modified version of K-Means considering local outlier factor to generate sufficient number of seed clusters, and then iteratively merges neighboring clusters considering three cluster similarity criteria: closeness in Euclidean space, comparable densities and wide hyperplane of two clusters. (Chaoji et al. 2009). Jiang (Jiang 2004) proposes spatial clustering techniques that employ hierarchical clustering accompanied by tree-like diagrams beneficial for visualizing cluster hierarchies at different levels of detail. Various neighborhood graphs are employed to define coarse-to-fine hierarchical segmentation of data. For instance, Anders (Anders 2003) proposed Hierarchical Parameter-free Graph Clustering (HPGCL), an unsupervised graph-based clustering algorithm that utilizes the hierarchy of proximity graphs for cartographic generalization. CHAMELEON, a graph-partitioning-based clustering method, integrates the agglomerative phase to form arbitrarily shaped clusters; its merging criteria determine the homogeneous density and closeness of merging candidates (Karypis et al. 1999).

A major drawback of agglomerative clustering is high computational time, especially, if agglomerative clustering algorithms uses single objects as initial clusters; traditional hierarchical agglomerative algorithms—using *single linkage* and *complete linkage* to merge clusters—require the time complexity of $O(n^2)$. Many agglomerative clustering algorithms aim to reduce their computational times using different techniques. Cohesion-based Self-Merging algorithm runs a faster algorithm, *e.g.*, K-Means to produce a number of initial clusters in linear time (Lin and Chen 2002). Another popular technique employed in CURE is to cluster randomly-sampled data instead of the original dataset and use these clusters to label the remaining objects (Guha, Rastogi and Shim 1998).

---

[9] In practice there is a lack of evidence that supports this claim. There has been no report of experimental results where traditional agglomerative clustering algorithms have been successfully used to identify arbitrarily shaped clusters for some popular benchmarks.

Another agglomerative clustering algorithm proposed by Hisashi *et al*. uses locality-sensitive hashing to find the nearest clusters (Hisashi et al. 2006).

In addition to the hierarchical clustering techniques, density-based clustering methods (Ester et al. 1996; Ankerst et al. 1999; Kriegel and Pfeifle 2005; Hinneburg and Keim 1998) are also capable of discovering dense arbitrarily shaped clusters. However, the main drawbacks of the density-based clustering algorithms are difficulties of parameters tuning, and non-robustness with respect to datasets with varying density.

Another challenge in clustering addressed in this paper is the use of plug-in fitness function. Typically, clustering algorithms do not allow for plug-in fitness functions except work by (Karypis et al. 1999) and (Choo et al. 2007). In addition, some work on semi-supervised clustering takes prior knowledge into account enabling them to obtain clusters that satisfy a given set constraints; Davidson and Ravi propose an agglomerative hierarchical clustering that incorporates *must-link* and *cannot-link* constraints in clustering. (Davidson and Ravi 2005).

6 Conclusion

We proposed an agglomerative clustering geo-referenced framework (GAC-GEO) for geo-referenced datasets that cleverly utilizes spatial information to discover arbitrarily shaped clusters in geo-referenced datasets. The framework relies on a two-phase clustering methodology consisting of preprocessing and agglomerative clustering phases. For the preprocessing phase, we introduced efficient ways to generate initial clusters: by running representative-based clustering or grid-based clustering algorithms. The former option is preferable when dealing with large datasets due to its fast computational time. On the other hand, the latter option trades off speed for clustering quality.

For the agglomerative phase, we introduced concepts of neighboring relationships compatible with different forms of initial clusters: proximity graphs like Gabriel graphs for the clusters with representatives and the contiguity of grid cells for grid-based clusters. Neighboring relationships are the powerful component of GAC-GEO that seamlessly bridge clustering algorithms in the two phases and guarantee that contiguous clusters have to be neighboring. Relying on the neighboring relationships, the GAC-GEO conducts a much wider search which, we claim, results in clusters of higher quality. Moreover, the expensive, agglomerative clustering algorithm is only run for usually less than 1,000 iterations; therefore, the impact of its high complexity on the overall run time is alleviated, particularly for very large datasets.

In addition, the use of plug-in fitness functions enables GAC-GEO to successfully serve different region discovery tasks. Our broader perspective on the generalization of the proposed framework is that with minimal effort potential users will employ it as a base framework for other combinations of clustering algorithms in future investigations. Many hybrid clustering algorithms can be considered variations of GAC-GEO, *e.g.*, MOSAIC and CHAMELEON.

Experimental results on artificial datasets illustrated the ability of five variations of GAC-GEO in detecting arbitrarily shaped clusters. We found that the quality of initial clusters has a direct affect on the quality of final clusters generated by the post-processing phase. In other words, any flaw occurring in the initial clusters degrades quality of the final clusters. Therefore, a choice of good clustering algorithms in the preprocessing phase is important. Nevertheless, even for low quality clusters, the post-processing phase enhances clustering quality.

We also conducted experiments on the five variations of GAC-GEO for two region discovery applications on an earthquake dataset: to identify hotspots of high variance of depth in earthquake occurrences, and to identify hotspots with high positive/negative correlation of earthquakes' depth and severity. Our method was able to identify larger hotspots of higher clustering quality. Comparing results among the variations for preprocessing methods, representative-based clustering algorithms are superior to the grid-based clustering algorithms in terms of clustering quality. On the other hand, the grid-based algorithms are superior in terms of computational time. According to our investigation, the CLEVER algorithm is the best choice for region discovery tasks in that it returns both superior initial clusters and final clusters. Surprisingly, using singleton initial clusters lead to superior final clusters for some datasets.

Our current implementation uses Gabriel graphs to identify neighboring clusters which is quite expensive. As far as future work is concerned, we plan to explore using other neighborhood graphs, such as spanning trees. We also plan to explore more flexible neighboring relationships for grid-based preprocessing methods; for example which that allow small gap between cells. In addition, the current version of GAC-GEO limits merging only to a single pair of clusters that leads to suboptimal solutions for some datasets. Allowing merging of multiple neighboring in parallel may lead to better cluster quality and might also speed up the agglomerative phase.

Acknowledgements This research was supported in part by a grant from the Environmental Institute of Houston (EIH).

References

Anders KH (2003) A hierarchical graph-clustering approach to find groups of objects. Technical Paper. In: ICA commission on map generalization, the 5th workshop on progress in automated map generalization

Ankerst M, Breunig MM, Kriegel HP, Sander J (1999) OPTICS: Ordering points to identify the clustering structure. In: Proceedings of the 1999 ACM-SIGMOD international conference on management of data. ACM Press, Philadelphia, Pennsylvania, pp 49–60

Chaoji V, Hasan MA, Salem S, Zaki MJ (2009) SPARCL: An Effective and Efficient Algorithm for Mining Arbitrary Shape-based Clusters. Knowledge and Information Systems, 21(2):201–229

Choo J, Jiamthapthaksin R, Chen C, Celepcikay O, Giusti C, Eick CF (2007) MOSAIC: a proximity graph approach to agglomerative clustering. In: Proceedings of the 9th international conference on data warehousing and knowledge discovery (DaWaK), pp 231–240

Davidson I, Ravi SS (2005) Hierarchical clustering with constraints: Theory and practice. In: Proceedings of the 9th European conference on machine learning and principles and practice of knowledge discovery in databases, pp 59–70

Ding W, Eick CF, Yuan X, Wang J, Nicot J-P (2007) On regional association rule scoping. In: Proceedings of international workshop on spatial and spatio-temporal data mining (SSTDM), pp 595–600:30

Ding W, Jiamthapthaksin R, Parmar R, Jiang D, Stepinski T, Eick CF (2008) Towards region discovery in spatial datasets. In: Proceedings of pacific-asia conference on knowledge discovery and data mining (PAKDD), pp 88–99

DMML datasets (2008) Datasets. In: Data mining and machine learning group website, University of Houston, Texas. http://www.tlc2.uh.edu/dmmlg/Datasets. Accessed 1 Jul 2008

Duan L, Xu L, Guo F, Lee J, Yan B (2007) A local-density based spatial clustering algorithm with noise, Information Systems 32(7):978–986

Eick CF, Parmar R, Ding W, Stepinki T, Nicot J-P (2008) Finding regional co-location patterns for sets of continuous variables in spatial datasets. In: Proceedings of the 16th ACM SIGSPATIAL international conference on advances in GIS (ACM-GIS)

Eick CF, Vaezian B, Jiang D, Wang J (2006) Discovery of interesting regions in spatial datasets using supervised clustering. In: Proceedings of the 10th European conference on principles and practice of knowledge discovery in databases (PKDD), pp 127–138

EPA (2008) Databases and software. In: U.S. Environmental Protection Agency (EPA) official website. http://www.epa.gov/epahome/data.html. Accessed 1 Aug 2008

Ester M, Kriegel HP, Sander J, Xu X (1996) Density-based spatial clustering of applications with noise. In: Proceedings of the international conference on knowledge discovery and data mining, pp 2976–2981

Gao D, Peuquet D, Gahegan M (2002) Opening the black box: interactive hierarchical clustering for multivariate spatial patterns. In: Proceedings of the 10th ACM international symposium on advances in geographic information systems, pp 131–136

Guha S, Rastogi R, Shim K (1998) CURE: an efficient clustering algorithm for large databases. In: Proceedings of ACM SIGMOD international conference on management of data, pp 73–84

Hinneburg A, Keim D (1998) An efficient approach to clustering large multimedia databases with noise. In: Proceedings of the 4th ACM SIGKDD, pp 58–65

Hisashi Koga, Tetsuo Ishibashi, Toshinori Watanabe (2006) Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing. Knowledge and Information Systems, 12(1):25–53

Jiang B (2004) Spatial clustering for mining knowledge in support of generalization processes in GIS. In: ICA workshop on generalization and multiple representation

Karypis G, Han EH, Kumar V (1999) CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, In: IEEE Computer 32(8):68–75

Kaufman L, Rousseeuw PJ (1990) Finding groups in data: an introduction to cluster analysis. Wiley, New York

Kriegel HP, Pfeifle M (2005) Density-based clustering of uncertain data. In: Proceedings of international conference on knowledge discovery in data mining, pp 672–677

Lin C, Chen M (2002) A robust and efficient clustering algorithm based on cohesion self-merging. In: Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining, pp 582–587

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, University of California Press, pp 281–297

Nelder JA, Mead R (1965) A simplex method for function minimization, The Computer Journal, 7(4):308–313

Netflix (2009) Netflix prize official website. http://www.netflixprize.com/. Accessed 1 October 2009

NOAA (2008) Explore NOAA. In: National Oceanic and Atmospheric Administration (NOAA) official website. http://www.noaa.gov/. Accessed 1 Sep 2008

Otoo EJ, Shoshani A, Hwang S-W (2001) Clustering High Dimensional Massive Scientific Datasets. Intelligent Information Systems 17(2–3):147–168

Piotte M, Chabbert M (2009) The Pragmatic Theory solution to the Netflix Grand Prize (Report from the Netflix Prize Winners)

Rinsurongkawong V, Eick CF (2008) Change analysis in spatial datasets by interestingness comparison. ACM-SIGSPATIAL Newsletter, pp 33–38

Sander J, Ester M, Kriegel HP, Xu X (1998) Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. Data mining and knowledge discovery 2(2):169–194

Tan PN, Steinbach M, Kumar V (2005) Introduction to Data Mining. Addison Wesley

TWDB (2008) TWDB data. In: Texas Water Development Board (TWDB) official website, http://www.twdb.state.tx.us/data/data.asp. Accessed 1 Sep 2008

UCI repository (2008) UCI Machine Learning Repository. http://www.ics.uci.edu/~mlearn/MLRepository.html. Accessed 1 Aug 2008

Vargas-Irwin C, Donoghue JP (2007) Automated spike sorting using density grid contour clustering and subtractive waveform decomposition. Neuroscience methods, 164(1):1–18

Xiong H, Steinbach M, Ruslim A, Kumar V (2009) Characterizing pattern preserving Clustering. Knowledge and Information Systems, 19(3):311–336

Zhong S, Ghosh J (2003) A unified framework for model-based clustering. Machine Learning Research, 4:1001–1037

Rachsuda Jiamthapthaksin received her Ph.D. in 2009 from the University of Houston, Texas, United States. She obtained a bachelor degree in Computer Science in 1997 and graduated with Honors in master degree in Computer Science, Dean's Prize for Outstanding Performance, in 1999 from Assumption University, Bangkok, Thailand. She is currently a lecturer in the Department of Computer Science and also a researcher of the Intelligent Systems Laboratory at the Assumption University. Her research interests include intelligent agents, fuzzy systems, cluster analysis, data mining and knowledge discovery. She has served as a reviewer for many conferences and served as a volunteer staff in an organization of the 2005 IEEE ICDM Conference, November 2005, Houston, Texas. She also served as a session chair for the 2009 Symposium on Computational Intelligence and Data Mining.

Christoph F. Eick received a Ph.D. in Computer Science from the University of Karlsruhe in Germany. He is an Associate Professor in the Department of Computer Science at the University of Houston and the Director of the UH Data Mining and Machine Learning Group. His areas of expertise include spatial data mining, clustering, machine learning, evolutionary computing, geographical information systems, databases and artificial intelligence. He published more than 100 papers in these areas. His current research centers on region discovery in spatial datasets, repository and correspondence clustering, density-based clustering and clustering with plug-in fitness functions, association analysis, geo-regression techniques, trajectory mining, distance function learning, and on the application of data mining to challenging problems in medicine, astronomy, geology, and environmental sciences. He also serves in program committees of major data mining conferences.

Seungchan Lee received his master degree in Computer Science in 2008 from the University of Houston, Texas, United States. He is a programmer at the University of Kansas. He had worked as programmer analyst at the University of Missouri from 2009 to 2010, software developer at University of Houston from 2007 to 2009, research assistant at Mississippi State University in 2006 and assistant engineer at Samsung Electronics in 2005. His special interests include software engineering in C++ and Java, and web development using ASP.NET C#, MS-SQL Server, and Visual Studio.