國立交通大學

資訊科學與工程研究所

碩 士 論 文

快速跨最佳和地理值域分群法

Efficient Joint Clustering Algorithms in Optimization and Geography Domains

研 究 生：駱嘉濠

指導教授：彭文志　教授

中 華 民 國　九 十 七　年 九 月

快 速 跨 最 佳 和 地 理 值 域 分 群 法

Efficient Joint Clustering Algorithms in Optimization and Geography Domains

研 究 生：駱嘉濠　　Student：Chia-Hao Lo

指導教授：彭文志　　Advisor：Wen-Chih Peng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年九月

摘　　要

之前的研究已點出關於最佳化和地理位置值域的跨值域分群問題。然而，之前的研究既沒有明確地定義地理位置值域連續性限制，也沒有提出足夠效率的演算法。本篇論文定義了跨值域分群問題的連續性限制。在此本篇論文著重的問題裡，使用者必須制定連續性限制的參數和分群數量。我們提出的演算法「K-means with Local Search 」（簡稱為 KLS）由三個階段組成：轉換階段、粗略分群階段以及微調分群階段。首先，將符合連續性限制的資料以 ConGraph 表示（ConGraph 意為 CONnected Graph）。接著，藉由 ConGraph 的協助，加上 K-means 和 local search 的概念，我們設計了一個有效率的粗略分群法。最後，將粗略分群的結果在最少的準確度損失下，微調成符合使用者要求的分群數。我們的實驗結果顯示出 KLS 能夠正確且有效率地完成分群。

關鍵字：地理、跨值域分群。

Abstract

Prior works have elaborated on the problem of joint clustering in the optimization and geography domains. However, prior works neither clearly specify the connected constraint in the geography domain nor propose efficient algorithms. In this paper, we formulate the joint clustering problem in which a connected constraint and the number of clusters should be specified. We propose an algorithm K-means with Local Search (abbreviated as KLS) consisting of three phases: the transformation phase, the coarse clustering phase and the fine clustering phase. First, data objects that fulfill the connected constraint is represented as the ConGraph (standing for CONnected Graph). In light of the ConGraph, by adapting the concept of K-means and local search, an algorithm is devised to coarsely cluster objects for the purpose of efficiency. Then, these coarse cluster results are fine tuned to minimize the dissimilarity of the data objects in the optimization domain. Our experimental results show that KLS can find correct clusters efficiently.

Keywords — geography, joint clustering

致　　謝

經歷了多次反覆的討論與修改，本篇論文終於得以成形定稿。首先要感謝我的指導教授彭文志教授，帶領我一步步地進行研究。讓我得以在碩士生涯裡，體會到學術文件的產生過程，還有參加學術研討會和各國的學者交流，感受到學術界的氣息。其次要感謝我的口試委員陳仲民博士與蔡明哲教授，在口試的時候，提供了許多寶貴的意見，使得本篇論文的部分缺失獲得改善的機會。

實驗室的日子，深深地影響了這兩年的研究生活。感謝博士班學長洪智傑、廖忠訓、學姊江孟芬和魏綾音的協助，充實我對學術研究的見解。感謝碩士班學長游敦皓、黃正和、李柏逸和學姊周佳欣，帶領我習慣實驗室的大小事。感謝碩士班同學郭員榕、傅道揚、蔡尚樺、蔣易杉，一起討論研究，一起玩樂，感謝你們的陪伴，讓我的研究所生活在歡樂的氣氛中渡過。感謝碩士班學妹黃琬婷、學弟王琮偉，很高興能認識你們，希望你們能一直保持積極的研究態度。在此，預祝老師順利完成手邊的研究跟計畫，祝實驗室的大家能順利畢業，保持研究的熱忱。

最後要感謝我的父母，讓我能無後顧之憂的專注於研究，真的很感謝你們為我作的一切。

Contents

List of Tables

List of Figures

Chapter 1

Introduction

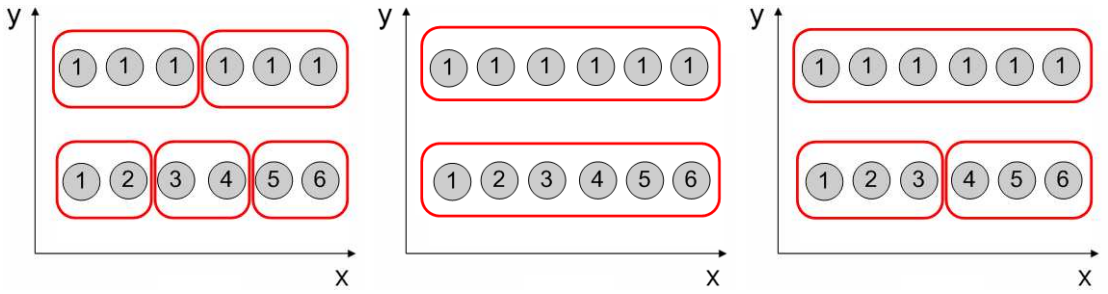
Clustering is an important technique in data mining field and has been utilized in many applications, such as biology analysis, information retrieval systems, market analysis and so forth [14, 9]. Given data objects with their attributes, clustering is to separate these data into different groups such that objects with similar attributes are in the same group while objects in different groups are dissimilar. Due to the widespread use of clustering in many applications, data clustering has been studied and a significant amount of research efforts have been elaborated on efficient clustering problems. Though data clustering has been studied for years, with a new kind of data, clustering should be developed to satisfy the requirement of applications.

Recently, with the growth of geographic information system and sensor networks, spatial data are widely collected. In general, there are two types of attributes associated with the spatial data, i.e., the geography attribute and the non-geography attribute. For example, in a traffic information system, cars equipped with GPS and wireless networks are able to upload their speeds and locations to estimate the real time traffic status or plan the fastest paths [10, 7]. Therefore, spatial data contain the geography attribute (i.e., the location of cars) and the non-geography attribute (i.e., speeds). Without loss of generality, the geography attribute is referred to the geography domain, whereas the non-geography attribute is referred to the optimization domain. To cluster spatial data for discovering interesting information, two requirements should be fulfilled in the generation of cluster results. Specifically, each data

in the same group should have similar values in the optimization domain and data objects in the same group should not be so far in the geography domain. To generalize the above scenario, a joint clustering problem over the geography domain and the optimization domain is derived. Given a set of data objects with their attributes in both the geography domain and the optimization domain, we should partition objects into several groups such that objects in the same group are connected in the geography domain while minimizing the dissimilarity of the data objects in the optimization domain.

A joint clustering problem addressed in this paper can be best understood by an illustrative example in Figure 1.1, where each data object has two attributes over the geography domain and the optimization domains. Explicitly, the attribute in the geography domain (i.e., the values of X-axle and Y-axle) of a data object indicates the location of this data object, and the other attribute in the optimization domain is the number along with each data object. Traditional clustering methods consider geography and non-geography attributes in the same domain (i.e. only one domain) and for example, in K-means [14], which is a center-based clustering method, derives cluster results as shown in Figure 1.1(a). In Figure 1.1(a), the upper six objects should be grouped together since their attribute in the optimization domain is very similar and their attributes in the geography domain are close to each other. On the other hand, Figure 1.1(b) shows the clustering results derived by contiguity-based or density-based clustering methods (e.g., DBSCAN [14]). As can be seen in Figure 1.1(b), the lower six objects are grouped into one cluster. However, their attributes in the optimization domain vary from one to six. This is due to that the contiguity-based/density-based method consider the local density only and thus cannot find center-based clusters in the optimization domain. The ideal clustering result is shown in Figure 1.1(c), where objects in each cluster have similar values in the optimization domain and their corresponding attributes in the geography domain are close to nearby objects. It is shown in [9, 13, 11] that a joint clustering problem calls for the design of new algorithms.

We mention in passing that the authors in [9] first explored the clustering problem over two domains and proposed a SVM-based approach for clustering over two domains. However, due



(a) By K-means. (b) By DBSCAN. (c) Ideal clusters.

Figure 1.1: An example to illustrate the problem.

to that the higher computation cost in complete-link, the proposed method is not feasible for a large-scale amount of data objects. The authors in [13], thus, proposed a heuristic algorithm, BINGO, to cluster objects. Note that the goal of the above two methods is to partition objects into several groups, where each group forms a compact region in the geography domain while objects in the same group have similar values in the optimization domain. Thus, the above two methods do not explicitly specify the meaning of compact regions. On the other hand, the authors in [5, 11] formulated a joint clustering problem in which data objects have two types of attributes: one is the attribute data (similar to the attribute data in the optimization domain in this paper) and the other is a relationship data. However, the relationship data is a binary relation, which cannot fully reflect the relationship in the geography domain (i.e., the various values of distances between two objects in the geography domain). Hence, the joint clustering problem addressed in this paper cannot be dealt with by any direct extension of existing joint clustering works in [5, 11].

Consequently, in this paper, we formulate a joint clustering problem in which a connected constraint is specified in the geography domain. Specifically, given data objects with attributes over the optimization domain and the geography domain, we aim to partition objects into groups such that the clustering cost is minimized and data objects in the same group satisfy a connected constraint required. Then, an algorithm KLS (standing for K-means with Local Search) is proposed. KLS consists of three phases: the transformation phase, the coarse clustering phase and the fine clustering phase. First, given the connected constraint required and the attributes of objects in the geography domain, grid-cells data structure is used to

efficiently derive the ConGraph (standing for CONnected Graph), where each vertex is an object and an edge exists between two objects if their distance in the geography domain is within a given threshold. In light of the ConGraph, by using the concept of K-means and local search, objects are coarsely clustered into several groups. The coarsely clustering approach makes a trade-off between efficiency and quality of clusters. Based on the clustering results derived, we could further fine tune clusters to minimize the dissimilarity in the optimization domain. Our experimental evaluation demonstrates that algorithm KLS is indeed able to efficiently and effectively cluster objects.

The rest of the paper is organized as follows. The related work is introduced in Chapter 2. Preliminaries are given in Chapter 3. The proposed algorithm is presented in Chapter 4. Performance evaluation is conducted in Chapter 5. This paper concludes in Chapter 6.

Chapter 2

Related Work

In this chapter, we first introduce the traditional clustering approaches, and then describe the joint clustering algorithms. Finally, a new popular area, clustering with constraints, is described.

2.1 Traditional Clustering

Given n objects, the goal of clustering is to cluster objects into some number of groups such that objects in the same group should be similar and objects in different groups should be dissimilar. The types of groups and definition of similarity is dependent on applications. Usually, the distance between two objects are used to define the similarity: two objects with near distance are similar.

The most famous clustering algorithm is K-means [14]. Given n objects and the number of clusters, k, K-means clusters objects into k groups. The idea of K-means is to select k different objects as seeds initially; each seed forms the initial centroid of each cluster. Each object is then assigned to the cluster of its most similar centroid. Once all objects are assigned to their corresponding clusters, compute the new centroids of clusters. A centroid is used as the mean of a cluster. Then reassign the clusters of objects until the result is not changed or after a limit number of iterations set by the user. A variation of K-means is K-medoids which use the most representative object (i.e. the object that is the most closed to the mean value)

as the centroid instead of the mean value.

In contrast to the center-based clustering, K-means, DBSCAN [14] is a density-based clustering algorithm. Instead of requiring the number of clusters, DBSCAN requires two other parameters: $MiniPts$ and Eps. In DBSCAN, an object is a core object if the number of objects within distance Eps is larger than or equal to $MiniPts$; an object is a border object if it is not a core object but is inside the neighborhood of a core object; a noise object is neither a core object nor a border object. With the definition of core, border and noise objects, DBSCAN consists of three phases: (1) label all objects, (2) filer out the noise objects, and (3) make connected core objects and their corresponding border objects into the same cluster. Two core objects are connected if their distance is within Eps. Note that although there is no need to assign the number of clusters in DBSCAN compared to K-means, DBSCAN requires appropriate $MiniPts$ and Eps to represent a suitable density. Although DBSCAN can find clusters that obey the connected constraint in our problem, DBSCAN does not consider the attributes in the optimization and thus it cannot cluster similar objects into the same cluster. Our proposed algorithm adapts the idea of DBSCAN to define the connected constraint and use the concept of K-means and local search [12] to achieve the goal.

Graph-based clustering [14] is to cluster data represented by a graph. In other words, given a graph $G = (V, E)$, a graph-based clustering is to cluster vertices into some number of groups. There are many graph clustering algorithms, and we mention a typical one here. Given a similarity threshold, the Jarvis-Patrick clustering algorithm (abbreviated as JP clustering) removed the edges with similarity that is smaller the threshold specified by users and make each connected subgraph as a cluster. The similarity value of an edge is defined as the shared nearest neighbor (SNN) similarity. For two objects, find the t nearest neighbors of each object, and then define their SNN similarity as the number of the intersection of nearest neighbors. Although the Jarvis-Patrick Clustering algorithm can cluster objects correctly even if there are different densities among objects, the correct parameters are hard to set and computing the SNN similarity graph is time-consuming when t is large. For example, when $t = n$, where n is the number of vertices, the time complexity is $O(n^3)$.

2.2 Joint Clustering

Though prior works have elaborated on proposing methods to deal with joint clustering problems, the connected constraint in the geography domain is not clearly defined. Furthermore, the proposed methods are not efficient given a huge amount of data objects.

The clustering problem with connected constraints is addressed in [9] first. Besides, an algorithm based on the hierarchical clustering method and SVM is proposed. Note that the parameters used in SVM is hard to determine, and the time complexity is high due to the usage of the complete link method. A two-phases algorithm for the clustering problem which the clusters cannot overlap to each other in the geography domain is developed in [13]. In the first phase, the original objects are transformed into non-overlapping T-regions. A T-region is a grid in the geography domain and each pair of objects in the same T-region is within distance T in the optimization domain. Each T-region can be viewed as a compact cluster with high similarity in the optimization domain and different area in the geography domain. A method to determine T automatically is also provided in [13]. In light of T-region concept, in the second phase, k regions are selected as the initial seeds of the clusters via a heuristic method. Then, two neighbor regions are merged until the number of regions (i.e. clusters) reaches k specified by users. Although authors in [13] shows that it is more efficient than [9] in their experiments, the transformation of T-regions in [13] is still not efficient since the time complexity of transformation is $O(n^3)$, where n is the number of objects.

On the other hand, the joint clustering problem of clustering objects in both the attributes and the relationships is formulated in [5, 11]; there is no attributes in the geography domain. The relationships can be represented by a graph directly, where vertices are objects and edges are relationships. Furthermore, authors in [11] focus on the joint clustering problem without the number of clusters specified. The relationships are represented by symmetric binary relations and are considered as the connected constraint similar to our problem. Although there is no need to assign the number of clusters in [11], the minimum size of each cluster is included, and this parameter affects both the efficiency and the quality of clusters a lot. The concept of their proposed algorithm, Connected X Clusters (abbreviated as CXC), is

to combine K-medoids and BFS [12]. CXC consists of two phases. In the beginning of the first phase, some number of objects are selected as medoids and form clusters. Then, for each medoid, find the nearest unclustered objects into their clusters via BFS. Therefore, the neighbors of medoids are clustered. Follow the same procedure, the unclustered objects are clustered through BFS. After all objects are clustered, clusters with sizes smaller than the minimum requirement are merged. Then new medoids are computed from new clusters and repeat the BFS-cluster procedure. The above procedure is stopped after the number of iterations specified by the user. In the second phase, the clusters are merged hierarchically until there are two clusters. Therefore, there are many sets of clusters (i.e. 2 clusters, 3 clusters, ...). The one with the best Joint Silhouette Coefficient is chosen as the final result. Compared to the above works, we define the connected constraint clearly and our target is attributes with real values in two domains instead of the binary relationships and attributes in one domain. Also, we do not require the minimum size of each cluster.

2.3 Clustering with Constraint

There are many researches about clustering with constraints introduced in [1]. The most relevant researches to our work are [15, 8, 4] which study the must-link and cannot link constraints. A must-link constraint for two object means that these two objects must be in the same cluster while the cannot-linke constraint for two objects means these two objects must not be in the same cluster. These works use the constraints to improve the qualities of clusters. Therefore, some constraints can be violated in order to acheive good quality efficiently. However, the connected constraint in this paper cannot be volated.

On the other hand, the must-link and cannot-link constraints define that two objects must be or not be in the same cluster, but our goal is to make sure the connectedness of objects in each cluster. In contrast, even if two objects are connected, they can belong to different clusters Authors in [4] propose the ϵ-constraint which is similar but not the same to our connected constraint. Although they also propose a K-means-like algorithm to ϵ-constraint,

it does not fit the constraints completely. Besides, all of these works focus on attributes in the same domain.

Chapter 3

Preliminaries

Objects considered in this paper have two domains of attributes; the two domains are the optimization domain and the geography domain. To facilitate the presentation of this paper, an object i is denoted as o_i. The corresponding set of attributes in the optimization domain (respectively, the geography domain) is expressed by S_i (respectively, L_i). Specifically, the j^{th} attribute in S_i is represented as s_i^j, whereas the j^{th} attribute in L_i denotes as l_i^j. Suppose that the dimension of the optimization domain is d_S and that of the geography domain is d_L. Based on the notations of attributes of objects, we could further formulate the similarity measurement. Same as in prior works, the distance between two objects is used as the dissimilarity measurement. Among a variety of distance functions, Euclidean distance is the most employed. Thus, we have the following two distance functions in the optimization domain and the geography domain.

Definition 1 *(Distance functions): For two objects o_i and o_j, the distance measurement in the geography domain, denoted as $dist_{geo}(o_i, o_j)$, is formulated as:*

$$dist_{geo}(o_i, o_j) = \sqrt{\sum_{k=1}^{d_L} (l_i^k - l_j^k)^2}$$

and the distance measurement in the optimization domain, denoted as $dist_{opt}(o_i, o_j)$, is for-

mulated as:

$$dist_{opt}(o_i, o_j) = \sqrt{\sum_{k=1}^{d_S}(s_i^k - s_j^k)^2}$$

.

Based on the definition of distance functions above, the cost of a cluster in the optimization domain is further formulated. Assume that a cluster C_j has a set of objects (e.g., $(o_1, o_2, \ldots, o_{|C_j|})$), where $|C_j|$ is the number of objects in C_j. The cost of cluster C_j in the optimization domain is thus formulated as:

$$g(C_j) = \frac{1}{|C_j|}\sum_{i=1}^{|C_j|} dist_{opt}(o_i, cen_j)$$

where cen_j is the centroid of C_j and is derived as

$$(\frac{1}{|C_j|}\sum_{o_i \in C_j} s_i^1, \frac{1}{|C_j|}\sum_{o_i \in C_j} s_i^2, \ldots, \frac{1}{|C_j|}\sum_{o_i \in C_j} s_i^{d_S})$$

Consequently, the average cost of a set of clusters is defined as:

Definition 2 *(Average cost of clusters) Given a set of clusters $SC = (C_1, C_2, ..., C_k)$, the cost of SC is defined as:*

$$f(SC) = \sum_{i=1}^{k} \frac{|C_i|}{n} g(C_i)$$

where $n = \sum_{i=1}^{k} |C_i|$.

The constraint in the geography domain is used to cluster objects such that their distance in the geography domain is within a threshold required such that objects in the same cluster are *connected*. The definition of the connected constraint is defined as:

Definition 3 *(Connected constraint on cluster) Given a clusters C_t, where $|C_t| > 1$, and a threshold r, $\forall o_i, o_j \in C_t \wedge o_i \neq o_j$, $dist_{geo}(o_i, o_j) \leq r$ or there is a sequence of objects $o_{u1}, o_{u2}, \ldots, o_{un} \in C_t$ such that $dist_{geo}(o_i, o_{u1}) \leq r$, $dist_{geo}(o_{u1}, o_{u2}) \leq r$, $\cdots$, and $dist_{geo}(o_{un}, o_j) \leq r$. C_t is fit the constraint inherently when $|C_t| = 1$.*

Problem formulation: From the definitions above, the problem addressed in this paper is that given the number of clusters k, a distance threshold r, n objects $o_1, o_2, ..., o_n$ with their attributes in the optimization domain and the geography domain, the goal is to derive a set of clusters $SC = (C_1, C_2, ..., C_k)$ such that (1) each object o_i belongs to only one cluster C_j, (2) objects in the same cluster are connected, and (3) the average cost (i.e., $f(SC)$) is minimized.

Chapter 4

Algorithm KLS: K-means with Local Search

We propose algorithm KLS consisting of three phases: the transformation phase, the coarse clustering phase, and the fine clustering phase. In the transformation phase, the Congraph (standing for CONnected Graph) is derived for efficiently verifying the connected constraint. Then, in the coarse clustering phase, rough clusters are efficiently derived via the ConGraph. The number of clusters is greater than or equal to k and the clusters may lose some qualities due to the efficient consideration. Finally, the clusters found in the second phase are merged according to $f(SC)$ iteratively until the number of clusters is k. In the following sections, each phase will be described in details.

4.1 Transformation Phase

In this phase, given a set of objects, the goal is to derive the ConGraph that captures the connected features among objects in the geography domain. In this section, the distance between objects in the geography domain is named as distance in short. The definition of the ConGraph is as follows:

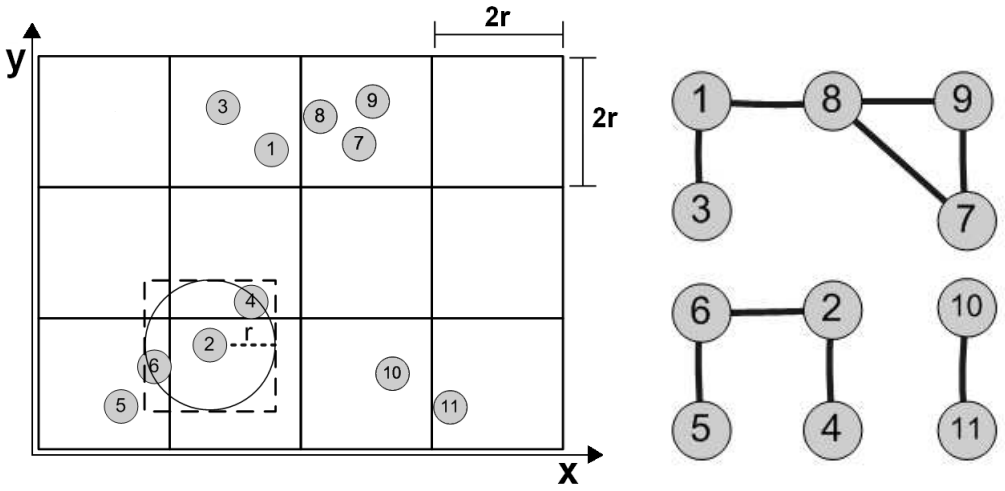
Definition 4 *(ConGraph) Given a set of n data objects $O = \{o_1, o_2, \ldots, o_n\}$ and a threshold r, a ConGraph is a graph $G = (V, E)$, where each vertex v_i is referred to each object o_i and an*

edge $e(v_i, v_j)$ *between* v_i *and* v_j *exists if and only if* $dist_{geo}(o_i, o_j) \leq r$. *In addition,* $E(v_i) = \{v_j \mid e(v_i, v_j) \in E\}$ *is the neighbors of vertex* v_i.

One naive method of generating the ConGraph is that distances of all object pairs are computed and then each object pair is verified whether its corresponding distance is within r or not. Clearly, this is very time consuming.

Consequently, we divide the geography domain into cells [2] with equal size. Objects are hashed into cells according to their attributes in the geography domain. Through cells, given an object o_i, we are able to quickly find out possible objects nearby o_i. Since the threshold of the connected constraint is set to r, the width and the height of a cell can be set to $2r$ such that only 2^{d_L} neighbor cells are required to explore when finding the neighbors of a vertex. Figure 4.1(a) is an example to illustrate the transformation phase, where there are eleven objects, the numbers along objects are object identifications, and the coordinate values represent their attributes in the geography domain. The geography domain is partitioned into $2r$-by-$2r$ cells. Note the $2r$-by-$2r$ rectangle with dashed line and has o_2 as its center. Only the four cells overlapped by this rectangle are necessary to further verify for $E(v_2)$. Then, the distances between those objects in these four cells are computed. Finally, edge $e(v_2, v_4)$ and $e(v_2, v_6)$ are added into E since their distance is within r. By utilizing cells, only a limited amount of objects are searched. Given the objects in Figure 4.1(a), the ConGraph is generated in Figure 4.1(b).

Time and Space Analysis. Assume that there are n objects. Let m be the maximum size of cells. The time complexity of constructing the hash table is $O(n)$, and the space complexity is also $O(n)$. Finding all edges for a vertex requires $O(2^{d_L} \cdot m)$ which is smaller or equal to $O(n)$, so the total time complexity is $O(2^{d_L} \cdot m \cdot n)$ or $O(n^2)$. When the density of each cell and the dimension of the geography domain are low, d_L and m can be viewed as constants. In such situation, finding all edges for a vertex requires only $O(1)$ and thus the total time complexity of finding E is $O(n)$. Since the graph is probably sparse, we use adjacency list to represent the graph. Therefore, the space overhead of the graph is $O(E)$.



(a) Cells. (b) Congraph.

Figure 4.1: An example of transformation.

4.2 Coarse Clustering Phase

Same as in K-means, we first select k vertices as initial centroids. Adapting the concept of local search [12], these vertices are used as the representative objects for their clusters. Then, those neighbors of these representative objects are extracted. The distances between each centroid and the corresponding neighbors are calculated. Then, only the neighbor with the smallest distance will be selected into the nearest cluster and the centroid of the corresponding cluster will be updated. Moreover, the representative object for the corresponding cluster is replaced by the new neighbor. The above procedure will be repeated until there is no unclustered neighbor. After this procedure, the unclustered objects are assigned to the nearest cluster if the connected constraint is not violated. The pseudo code is written in Algorithm 1.

Figure 4.2(a) shows an example of seven objects, where the ConGraph is provided and the coordinate values of objects are the attributes in the optimization domain. Assume that we would like to cluster the seven objects into three groups. The execution of KLS is shown in Figure 4.2(b). As shown in Figure 4.2(a), the initial seeds are v_1, v_2 and v_3 and each seed forms a new cluster (i.e., C_a, C_b and C_c) in Step 0; Step 0 is described from line 3 to 7. Then, through line 9 to 15, the centroid of each cluster is calculated in the optimization domain. PQ is the priority queue to store data pairs (d^2, v_i). We use d^2 here to calculate more easily.

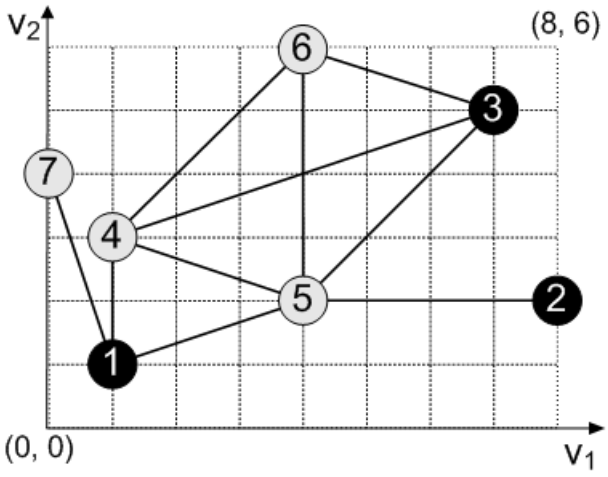
Algorithm 1: Coarse Clustering Algorithm

Input: An integer k, a graph $G = (V, E)$ and attributes of V
Output: a set of clusters SC, where $|SC| \geq k$

```
1  let E_u(v_i) (respectively, E_c(v_i)) be the unclustered (respectively, clustered) neighbors of
   v_i
2  /* initialization */
3  select k vertices as the initial seeds and each seed forms a cluster C_t
4  foreach seed v_i do
5      let d be the smallest distance dist_opt(v_i, v_j), v_j ∈ E_u(v_i)
6      add pair (d, v_i) into a priority queue PQ
7  end
8  /* local search */
9  while PQ.not_empty do
10     remove (d, v_i) from PQ with the smallest d
11     let C_t be the cluster of v_i, and cen_t be the centroid of C_t
12     let d, u be the smallest distance dist_opt(cen_t, v_j) and the corresponding vertex,
       where v_j ∈ E_u(v_i)
13     add u into cluster C_t and update cen_t
14     add pair (d, u) into PQ
15 end
16 /* fix unclustered objects */
17 foreach unclustered v_i ∈ V do
18     if |E_c(v_i)| > 0 then add v_i to the nearest cluster of v_j ∈ E_c(v_i) and update the
       corresponding centroid
19     else v_i forms a new cluster
20 end
21 return the union of all clusters SC
```

As can be seen in Step 1, v_4 is selected and included into cluster C_a. Then, the unclustered neighbors of v_4 (i.e., v_5, v_6) are searched in the ConGraph. Since the distance between v_5 and the centroid of C_a is the smallest, the data pair $(9, 5)$ will be inserted into PQ. Following the same operation, we could have the cluster results shown in Step 6. Note that in the end of Step 5, there is no representative object in PQ and v_7 has not been clustered. This special situation is handled by line 17 to 20 in Algorithm 1 and shown in Step 6.

As point out in prior works, the selection of initial seeds in K-means has great impact on the cluster results [3, 14, 11]. In our works, good initial seeds requires that distances of two seeds should be as far as possible in both the optimization and the geography domains. It is apparent that seeds should be much different from each other in the optimization domain



Step 0
C_a = (1), C_b = (2), C_c = (3)
cen_a = (1,1), cen_b = (8,2), cen_c = (7,5)
PQ = {(4,1), (10,3), (16,2)}

(a) Setting.

Step 1
C_a = (1,**4**), C_b = (2), C_c = (3),
cen_a = **(1,2)**, cen_b = (8,2), cen_c = (7, 5),
PQ = {**(9,4)**, (10,3), (16,2)}

Step 4
C_a = (1,4,5), C_b = (2), C_c = (3,6),
cen_a = (2,2), cen_b = (8,2),
cen_c = (5.5,5.5), **PQ = {(20,5)}**

Step 2
C_a = (1,4,**5**), C_b = (2), C_c = (3),
cen_a = **(2,2)**, cen_b = (8,2), cen_c = (7, 5),
PQ = {(10,3), (16,2), (20,5)}

Step 5
C_a = (1,4,5), C_b = (2), C_c = (3,6),
cen_a = (2,2), cen_b = (8,2),
cen_c = (5.5,5.5), **PQ = {}**

Step 3
C_a = (1,4,5), C_b = (2), C_c = (3,**6**),
cen_a = (2,2), cen_b = (8,2),
cen_c = **(5.5,5.5), PQ = {(16,2), (20,5)}**

Step 6
C_a = (1,4,5,**7**), C_b = (2), C_c = (3,6)
cen_a = **(1.5,2.5)**, cen_b = (8,2),
cen_c = (5.5,5.5), PQ = {}

(b) Execution process.

Figure 4.2: An example for the coarse clustering phase.

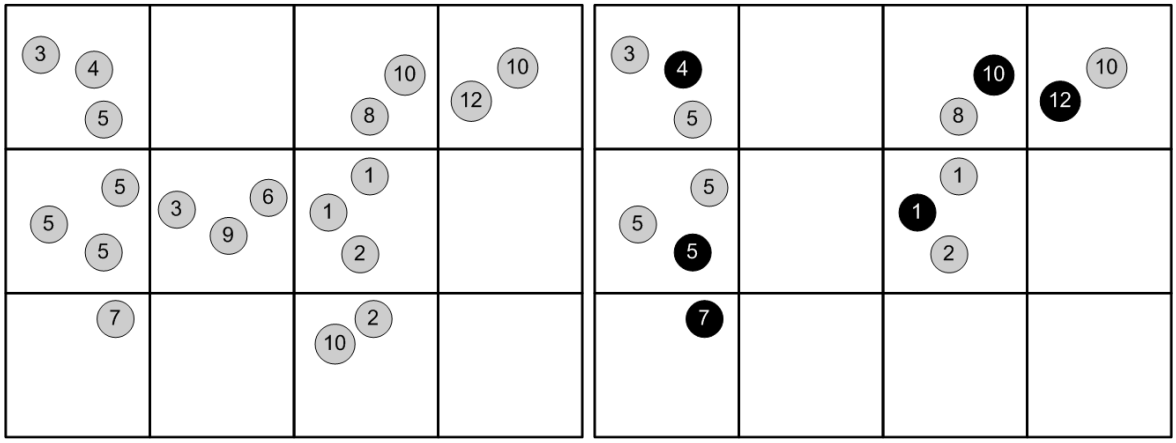
since this fits the objective function $f(SC)$. However, seeds may not need to be different from each other in the geography domain intuitively. We argue that seeds that are far to each other in the geography domain may be better because clusters must obey the connected constraint. Consider Figure 4.1(a) again, assume that we want to choose four medoids, and the most four different objects in the optimization domain are o_1, o_3, o_8, o_7. If we use these objects as the medoids, there are two problems. Let the cell in the down-left corner be cell $(0, 0)$. First, it cannot cover the other disconnected objects, and therefore there will be four bad clusters among cell $(1, 2)$ and $(2, 2)$, and the rest objects form other clusters. Apparently the number of clusters and the quality of each cluster are not good. Moreover even if there are some

objects in the cell $(1, 1)$ and $(2, 1)$ such that there are only one connected graph instead of three connected subgraphs, the representative objects of clusters that contain o_3 and o_8 may not "move" out in the local search of KLS because the other seeds, o_1, o_7 are in their way.

To remedy this, we can take the advantage of cells which have been computed before. Since the objects have been placed into cells which are small region in the geography domain, we can view these cells as temporary clusters and therefore they are inherently far in the geography domain. For each cell, the variance of objects in the optimization domain is derived first. Here, the variance of a cell is defined as $\sum_{o \in cell} dist_{opt}(o, cen)^2$, where cen is the centroid of the cell. Then the median of the variance is chosen as the pivot such that cells with higher variance than the pivot are filtered. Note that to make sure that there are at least k seeds, the filtering operation is stopped when the number of cells is smaller than k. After the filtering, the medoid, the nearest object to the centroid, of each cell is chosen according to the attributes in the optimization domain. Then, the most different k initial seeds (i.e. medoids) are selected. We randomly select one medoid as the first seed. Then for each medoid, the minimum distance among all selected seeds is calculated, and the one with the maximum distance is selected as the new seed. The selection operation is stopped until the number of seeds is k.

Note that the definition of difference among seeds affects the quality of seeds selection. We choose the minimum distance since this operation can reveal the most difference among seeds. For example, after selecting t seeds, the new selected $(t+1)_{th}$ seed is at least the most different from all selected seeds compared to the other unselected seeds. If we use a sum of distances instead of the minimum, a new seed that is much different from a selected seed but similar to some selected seeds may be selected, and thus it is a redundant seed.

Figure 4.3 shows an example of the seeds selection. The number in each circle indicates its attribute in the optimization domain, and the location indicates its attributes in the geography domain. Figure 4.3(a) is the objects before filtering. The median of variances of cells is 2, and the results after filtering is shown in Figure 4.3(b), where black nodes are the medoids of cells. In the following content, we use their attribute in the optimization domain as their identification numbers. Assume that o_{10} is selected as the first seed. Then, o_1 would be



(a) Before filtering. (b) After filtering.

Figure 4.3: An example of seeds selection.

selected since its the farthest object in the optimization domain compared to o_{10}. If we use a sum of distances to select seeds, o_{12} would be selected. However, o_{12} has a similar attribute in the optimization domain as o_{10}. On the other hand, with the minimum distance, o_5 would be selected as the third seed.

Time Analysis of seeds selection. For n objects, b cells and k cluster, computing the variance of all cells requires $O(n)$, filtering the cells requires $O(b)$, and finding k most different medoids from $max(k, b/2)$ cells requires $O(b \cdot k^2)$. Therefore, the total time complexity is $O(n + b + b \cdot k^2)$. In the worst case, each object forms an unique cell (i.e., $b = n$), and thus the time complexity is $O(n \cdot k^2)$.

Time Analysis of the Coarse Clustering. For n objects, k clusters, the initialization costs $O(n \cdot k^2)$. A priority queue can be implemented by a heap which requires $O(k)$ to insert a new object or remove the top object. Therefore, the total time complexity about operating the priority queue is $O(n \log k)$. We do not find the best representative object in each iteration. Instead, the new appended object is chosen as the representative object. This makes a trade-off between the quality of clusters and running time, and thus the time complexity of the local search is $O(|E| + n \log k)$. Finding unclustered objects requires $O(|E|)$, and generate SC requires only $O(n)$. The overall time complexity of this phase is $O(n \cdot k^2 + |E|)$.

4.3 Fine Clustering Phase

Before explaining the fine clustering method, the connectedness of two clusters is defined as follows:

Definition 5 *(Connected constraint among clusters) Two clusters C_i and C_j are connected if and only if $\exists o_t \in C_i$ and $\exists o_u \in C_j$ such that $dist_{geo}(o_t, o_u) \leq r$.*

The goal of this phase is to merge clusters until the number of clusters is k. To minimize the average cost, we adapt the agglomerative hierarchical clustering with the mean distance [6]. The fine clustering is to recursively merge the two connected clusters with the smallest distance between their centroids until the number of clusters is k.

Specifically, if there are more than k disconnected subgraphs in the ConGraph, the number of clusters is also more than k due to the connected constraint. Therefore, there is no suitable solution in this situation.

Time Analysis. The agglomerative hierarchical clustering with average link requires $O(k'^3)$, where k' is the number of clusters found by the coarse clustering. If sorted lists are used to maintain the the distances from each cluster to the other clusters, the time complexity could be reduced to $O(k'^2 \log k')$.

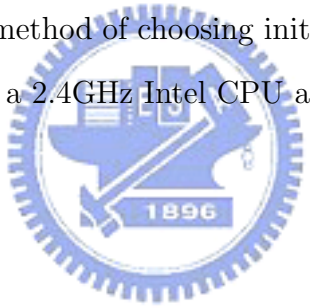
4.4 Overall Time and Space Complexity

Given n objects, the number of clusters k, and the threshold of the connected constraint r, the transformation phase requires $O(n^2)$ while the coarse clustering phase requires $O(n \cdot k^2 + |E|)$ The agglomerative hierarchical clustering with the mean distance requires $O(k'^2 \log k')$, where k' is the number of rough cluster results and $k < k' << n$ generally. Thus, the overall time is bounded by the transformation time, $O(n^2)$. The space overheads are cells and graphs which requires $O(E)$ space.

Chapter 5

Performance Study

In this chapter, we first describe the environment of the experiments including the data generator, the suits of test cases and the methods we use to evaluate in Section 5.1. Then the cost evaluation is conducted in Section 5.2 and the running time comparison is conducted in Section 5.3. Finally, the heuristic method of choosing initial seeds is evaluated in Section 5.4. All experiments are executed with a 2.4GHz Intel CPU and 8GB of memory.



5.1 Experiment

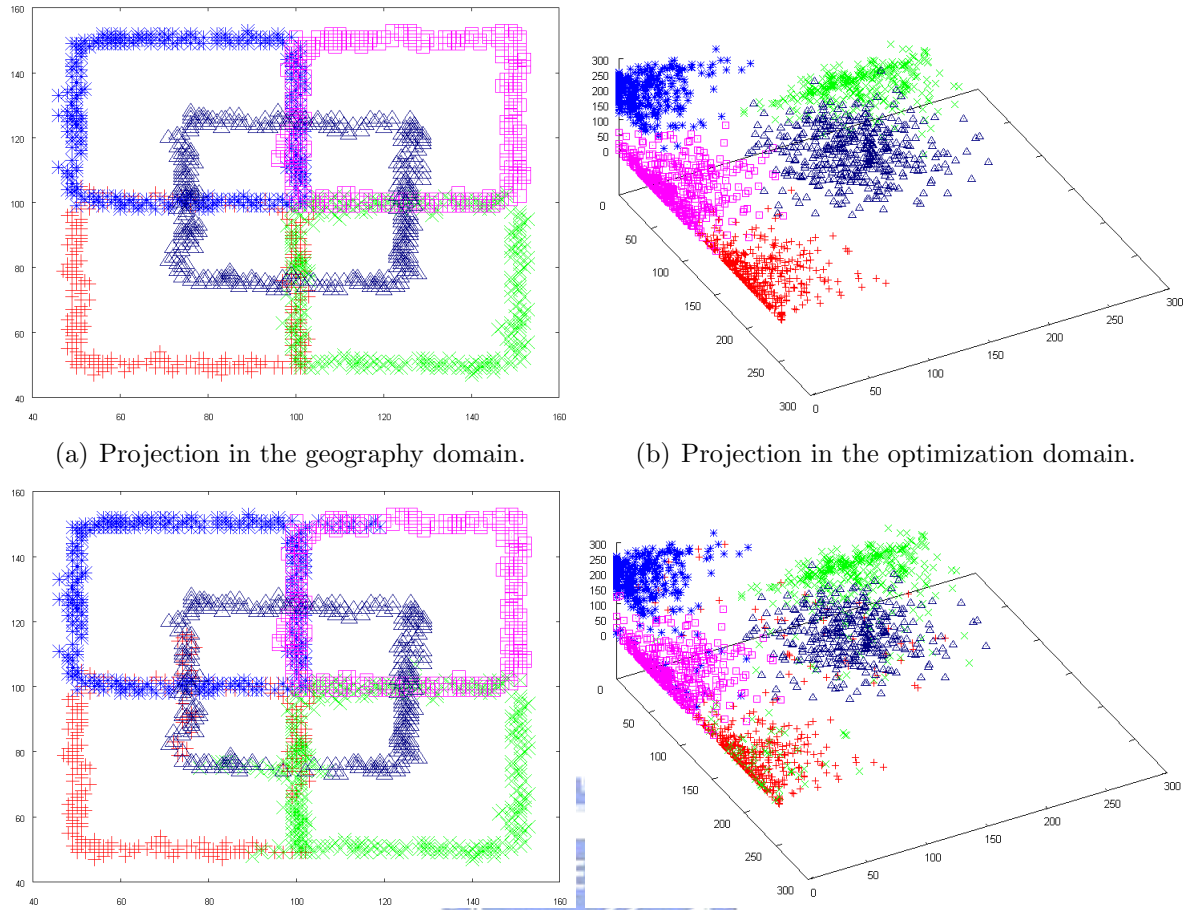
5.1.1 Simulation Model

The testing data have two domains of attributes. In order to view these data easier, we utilize the RGB color model to represent the attributes in the optimization domain, that is, three types of attributes represent red, blue and green, and their range are between 0 and 255. For example, value (255, 255, 0) in the optimization domain represents the color yellow. In the geography domain, the range of each attribute is between 0 and 799. Therefore, information of both domains of data could be shown in a 800x800 figure with different colors in the optimization domain while regarding the attributes in the geography domain as their coordinate. In the following content, we use (x, y) and (cr, cg, cb) to represent the attributes in the geography domain and the ones in the optimization domain, respectively.

A synthetic data generator is made to prepare test data with the ground truth. Our generator requires parameters k and r, where k indicates how many clusters should be generated, and objects with the same clusters are within a distance r in the geography domain. For each cluster, we generate the pivot points uniformly from 0 to 799 first, and then attributes in the geography domain is generated to pass through these pivot points iteratively. The number of pivot points are generated randomly, and each pivot point has different x or y from the previous one. After these pivot points are generated, the first pivot point is the attribute of the first object, and the next value of x is $x + dir(x, x')w(t)$, where x' is the value of the next pivot point, $dir(x, x') = \frac{x'-x}{|x-x'|}$ and $w(t) = t$ with the probability $\frac{(t+r)}{2r^2}$, $t \in [-r, r]$ and t is an integer. $w(t)$ generates larger integers with higher probability, and $dir(x, x')$ is 1 or -1 according to the relative position between x and the destination x'. Therefore, the generated points are toward to x' with higher probability, but they may be backward locations, too. y is also generated by the same function. Each (x, y) is the attribute in the geography domain of a new object. After reaching the pivot point (x', y'), the next pivot point is used until there is no pivot point remained. This approach can generate random points in the form of many intersected "horizontal and vertical lines". In order to get a denser points such that objects in each clusters have more neighbors, we use the same pivot points again after all pivot points are passed, and this may cause a "diagonal line" between the last pivot and the first pivot.

After objects in each cluster have their attributes in the geography domain, the attributes in the optimization domain are generated according to the normal distribution $N(\mu, \sigma)$. Our data generation in the optimization domain is similar to the method in [9]. cr, cg, cb are generated according to the same process, so we explain the process by one attribute here. First, μ and σ are generated uniformly from 0 to 255 and from 1 to 32, respectively. Then, each object has the value from $N(\mu, \sigma)$. The value is replaced by 0 (or 255) if it belows 0 (or above 255). Therefore, objects in each cluster have similar attributes in the optimization domain, and all clusters obey the connected constraint.

Figure 5.1 is an example of synthisized data with $r = 5$ and $k = 5$. There are five clusters: each cluster forms a "closed line" and clusters are overlapped to each other. The data over



(a) Projection in the geography domain.

(b) Projection in the optimization domain.

(c) Projection in the geography domain.

(d) Projection in the optimization domain.

Figure 5.1: An example of generated data with $r = k = 5$ and the result of KLS: (a) and (b) show the data over two domains; (c) and (d) show the result of KLS over two domains.

two domains are shown in Figure 5.1(a) and Figure 5.1(b), where objects in the same cluster are represented by the same symbol and color. Figure 5.1(c) and Figure 5.1(d) show the result of KLS over two domains, and the clustering result of KLS is just a little different from the truth. We will conduct a detailed experiments in the following content to evaluate KLS deeply.

5.1.2 Test Cases

We setup eight suits of test cases, and each suit has five test cases, so there are forty test cases in total. For example, Figure 6 shows forty clusters and each color curve represent a cluster. As mentioned in Section 5.1.1, the ranges of all attributes in the geography domain and in

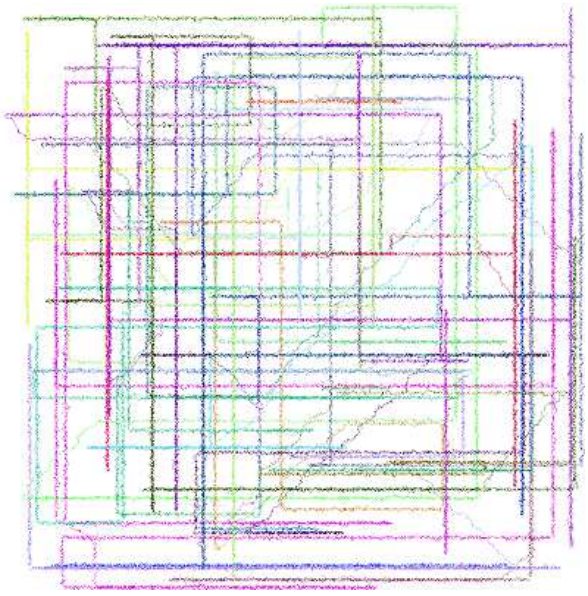


Figure 5.2: An example of forty clusters.

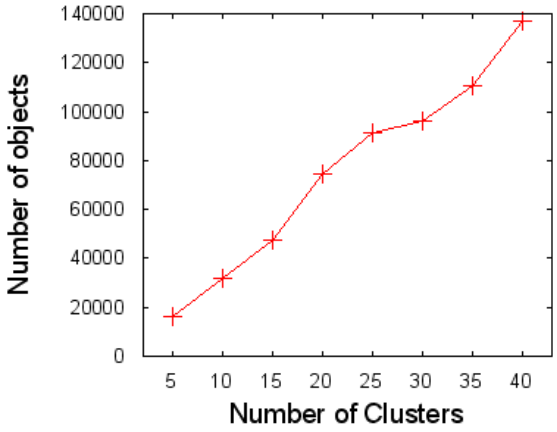


Figure 5.3: Setting of test suits.

the optimization domain are $[0, 799]$ and $[0, 255]$, respectively. We let $r = 5$ for all test cases. If each object has more neighbors, the test cases would be more complicated and the running time would increase. Therefore, we use $r = 1$ as the parameter of the data generator in order to make the objects closer. The other parameter k, the number of clusters, is different among test suits. Figure 5.3 shows the average number of objects of each test suit.

5.1.3 Implementation of Existing Works

Currently, BINGO [13] is the best algorithm for joint clustering in the optimization domain and geography domain. We implement BINGO in order to compare it with KLS. The main idea of BINGO is to consider the cost in both optimization domain and geography domain simultaneously by using T-regions. A T-region is a square region in the geography domain and each pair of data in this geography domain have distance in the optimization domain within T. Through analysis, BINGO provides a way to calculate the desired T. With compact T-regions, BINGO forms a graph named NeiGraph $G = (V, E)$ where each vertex is a T-region and there is an edge between two vertices if the two T-regions are near by. Then, BINGO selects k vertices as seeds and then merges remaining vertices to form the final k clusters. However, BINGO cannot cluster successfully when some data objects are far in the optimization domain

and are near in the geography domain. In such cases, some T-regions are not near by other T-regions and thus they are not connected. The problem is that BINGO assumes that all T-regions should be connected in order to finish the whole algorithm. In order to fix the unconnected problem, we use minimum spanning tree (MST) to make sure T-regions form a connected graph. We compute the distances between all pairs of T-regions to form a temporal graph $G'(V, E')$, and then use MST on G' to get $|V| - 1$ edges. By adding these $|V| - 1$ edges to NeiGraph, NeiGraph is connected and thus BINGO can finish clustering successfully. Although using MST solves the unconnected problem, it increases the time and space cost to $\Theta(n^2)$, where n is the number of data objects. Therefore, the revised BINGO cannot finish the cases with larger size of objects in our environment.

For the purpose of comparison, we implement one naive algorithm Connected K-means (abbreviated as CK-means) which has three phases as KLS does. In CK-means, the operations in the transformation phase and the fine cluster phase are the same as KLS. Recall that the ConGraph $G = (V, E)$ was derived in the transformation. In the coarse clustering phase of CK-means, objects are first partitioned by K-means with the attributes in the optimized domain only. After clusters OC are derived by K-means, we generate a new graph $G' = (V, E')$, where $E' = \{e(v_i, v_j) \mid e \in E \wedge v_i, v_j \in C_t, C_t \in OC\}$. We exploit BFS [12] to traverse G' and get connected subgraphs G_1, G_2, ..., $G_{k'}$. Then each connected subgraph is an equivalent cluster which fits the connected constraint. Moreover, objects in the same clusters have similar attributes in the optimization domain since they are clustered by K-means previously. Therefore, $k' \geq k$ clusters are found.

In addition to use CK-means, we also use the JP clustering algorithm in order to demonstrate that traditional clustering algorithms cannot work. After the transformation, we get the ConGraph $G = (V, E)$ and set the weight of edges between v_i and v_j as $dist_{opt}(v_i, v_j)$. Then, after applying JP clustering to the graph, the coarse clusters are derived. Finally, as KLS and CK-means, the fine clustering phase is applied to the coarse clusters in order to get the final k clusters.

5.1.4 Performance Metrics

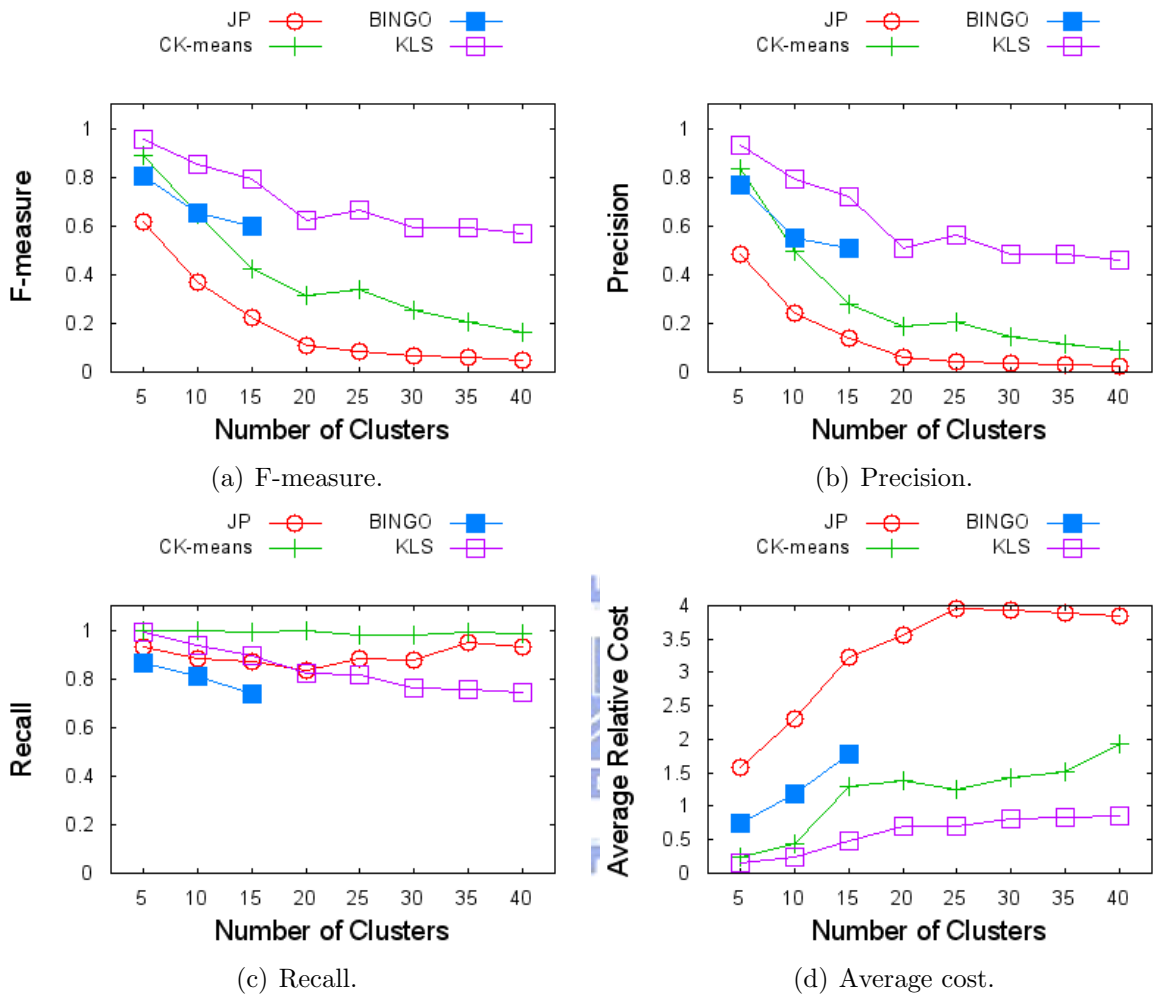
Since the objects in each test case are generated under control, the true clusters SC_{true} can be known. According to $f(SC_{true})$, the average cost of all test cases in each test suit is computed first. Instead of using $f(SC)$ as the measurement of the quality in the optimization domain, the relative cost $\frac{f(SC)-f(SC_{true})}{f(SC_{true})}$ is used.

In addition to the traditional cost measurement, we can use more precise measurement to understand the degree of correctness. Let $C_1, C_2, ..., C_{k'}$ be the clusters which are found by our algorithm, $CT_1, CT_2, ..., CT_k$ be the true clusters according to the setting of the experiments, and $m_{i,j}$ be the size of the intersection of C_i and CT_j. Then, the precision of the pair of cluster C_i and CT_j is $P(i,j) = \frac{m_{i,j}}{|C_i|}$. Similarly, the recall $R(i,j) = \frac{m_{i,j}}{|CT_j|}$, and the F-measure [14] is $F(i,j) = \frac{2P(i,j)R(i,j)}{P(i,j)+R(i,j)}$. Note that the range of F-measure is still $[0,1]$ as the precision and recall, and 1.0 is the best value which indicates that the precision and recall are also 1.0. Considering the two extreme cases: each object forms an unique cluster and there is only one cluster. In the first case, because the size of each cluster is small, it reaches the best precision, 1.0, for its corresponding true cluster. However, the recall is the worst. On the other hand, the second case reaches the best recall, 1.0, but the worst precision. Normally, a cluster with larger size may have high recall but lower precision. Because F-measure can reflect the overall quality between the precision and recall, it is chosen as our main measurement.

By using the F-measure, for each true cluster CT_j, the cluster C_i with the largest $F(i,j)$ is chosen to represent it. Then, the overall precision P is computed by $\frac{\sum_{i=1}^{k'} m_{i,j}}{\sum_{i=1}^{k'} |C_i|}$, where C_i is the chosen cluster for each true cluster CT_j. Similarly, the overall recall $R = \frac{\sum_{j=1}^{k} m_{i,j}}{\sum_{j=1}^{k} |CT_j|}$, and the overall F-measure $F = \frac{2PR}{P+R}$. Consider the two extreme cases again, their F-measures are near zero because one of the precision or the recall is near zero. For example, assume that there are $100,000$ objects, the correct result is 100 clusters and the largest cluster have $2,000$ objects. If each object forms an unique cluster, $P = 1.0$ and $R = \frac{100}{100000} = 0.001$. Therefore, $F = 0.002$. On the other hand, if there is only one cluster which contains all objects, $P = \frac{2000}{100000} = 0.02$, and $R = 1.0$. Therefore, $F = 0.04$.

5.2 Correctness Verification

5.2.1 Performance Comparisons



(a) F-measure.

(b) Precision.

(c) Recall.

(d) Average cost.

Figure 5.4: Overall results between CK-means and KLS.

Figure 5.4 shows the results of the experiments with KLS and its competitors. Note that we use $2k$ as the number of seeds in KLS instead of k. This small modification increases the precision of clusters found by KLS since it is hard to choose the correct seeds even a useful heuristic method is used. Then, the fine clustering merge more than $2k$ clusters to k clusters. For CK-means, we use k as the number of seeds and set the iteration of K-means as one. Since the check of connected constraint would divide clusters into many smaller pieces of clusters, set the number of clusters as $2k$ would not help. Also, in our preliminary experiments, we

found that the number of iterations does not help the performance after checking connected constraint, so we use one iteration in order to save computation time. For JP clustering, the similarity threshold is set as the median of weights of edges, and the number of nearest neighbors is set as $2 \cdot \log(|V|)$ in order to save the computation time.

According to Figure 5.4(a), clusters found by KLS are the best. Specifically, when the number of clusters increases, KLS performs much better than JP clustering and CK-means. On the other hand, even there are forty clusters and more than 100,000 objects, KLS still achieves above 50% correctness. Note that BINGO cannot finish the cases with $k \geq 20$ due to memory limit.

Figure 5.4(b) and 5.4(c) shows the corresponding value of the overall precision and recall. The performance of the precision and the F-measure are very similar while the variation of recall is more stable than the other measurements. With the concept of T-regions, BINGO has better precision than CK-means and JP clustering. However, it still has worse precision than KLS due to the selection of T is hard: a general T is not reliable when the distributions of data objects are mess. On the other hand, since CK-means perform good recall but very bad precision, we know that CK-means resulted some large clusters. The behind reason will be explained in 5.2.2.

In Figure 5.4(d), the common measurement, the average relative cost compared to the true average cost, is shown. KLS has at most double cost than the true cost; CK-means has double cost after $k = 15$ and has at most triple cost; and the worst result of JP clustering supports the necessity of reserving attributes into two domains. Note that from Figure 5.4(d), BINGO performs worse than CK-means, and this result is opposite to Figure 5.4(a). Figure 5.4(d) indicates that although CK-means finds clusters that are much different from the true clusters compared to BINGO, CK-means still achieve better clusters (note that true clusters may not be the best clusters.).

Algorithm Name	Avg. F	Avg. P	Avg. R	Avg. cost
CK-means w/o fine	0.756	0.806	0.720	22.042
CK-means	0.404	0.294	0.990	53.174
KLS w/o fine	0.673	0.778	0.598	36.864
KLS	0.705	0.618	0.841	38.967

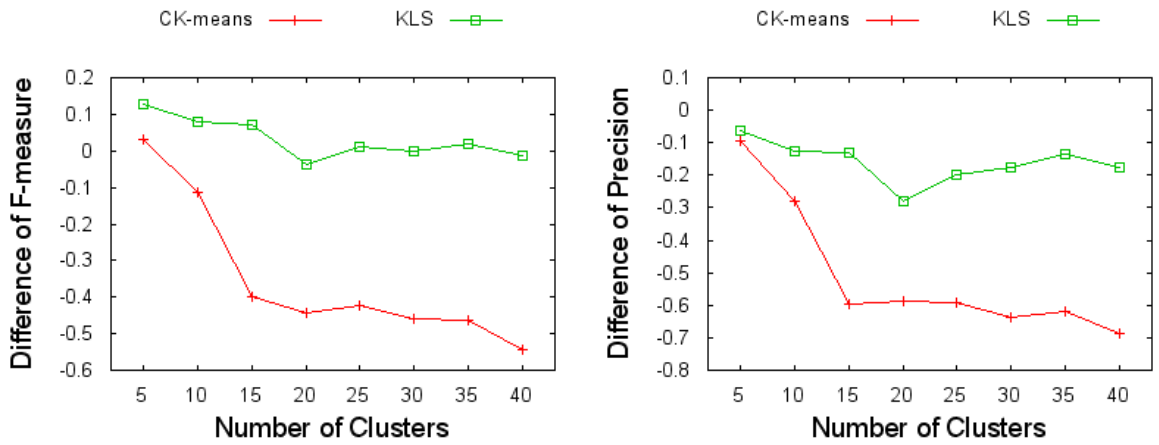
Table 5.1: Summary of the quality of the coarse and fine clustering.

5.2.2 Comparison between Coarse and Fine Clustering

From the above experimental results, we found that CK-means is the best competitor compared to KLS. We futher analyze the detailed result in each phase in this section. The average F, P and R of eight test suits are listed in Table 5.1. "CK-means w/o fine" means the clusters found by CK-means before the fine clustering phase, and "CK-means" means the clusters found by CK-means with all three phases. "KLS w/o fine", and "KLS" have the same meaning. Although "CK-means w/o fine" has the best F-measure, it gets much worse result after the fine clustering due to the low precision.

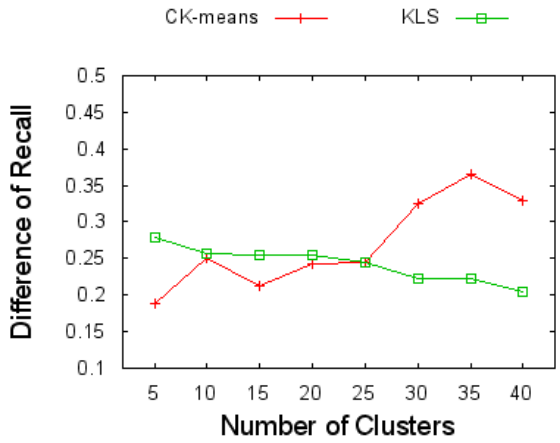
Figure 5.5 shows the detailed results between the coarse and fine clustering. Figure 5.5(a) is the difference between the F-measure after the fine clustering and the one before the fine clustering. Apparently, CK-means has worse F-measure after the fine clustering in all test suits except $k = 5$. On the other hand, KLS has similar F-measure after the fine clustering; when the number of clusters are small (i.e. $k \leq 15$), the quality is improved by the fine clustering for KLS. The difference between clustering with and without fine clustering phase shows the usefulness of our proposed coarse clustering in KLS. In Figure 5.5(d), the difference of average cost also indicates the same result.

The reason behind the variation of the F-measure can be understood by Figure 5.5(b) and 5.5(c). The goal of the fine clustering is to reduce the number of clusters to k, to maintain the precision and to increase the recall simultaneously. Since fine clustering decreases the number of clusters, the number of objects in each cluster increases. Therefore, the recall would be increased basically. However, the increases of the recall are small compared to the decreases of the precision for CK-means according to Figure 5.5(b) and 5.5(c). The result of decreasing precision explains why CK-means have good recalls but bad overall results.

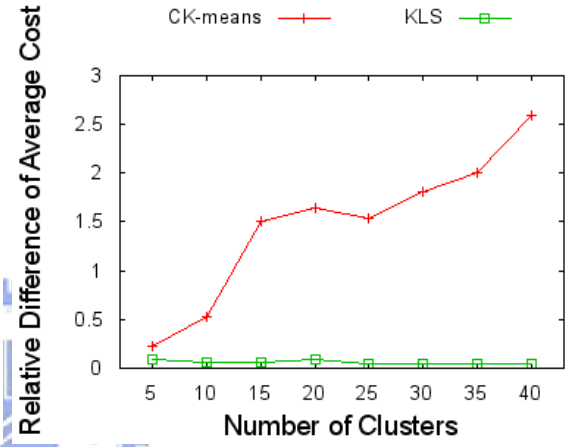


(a) Difference between F-measure.

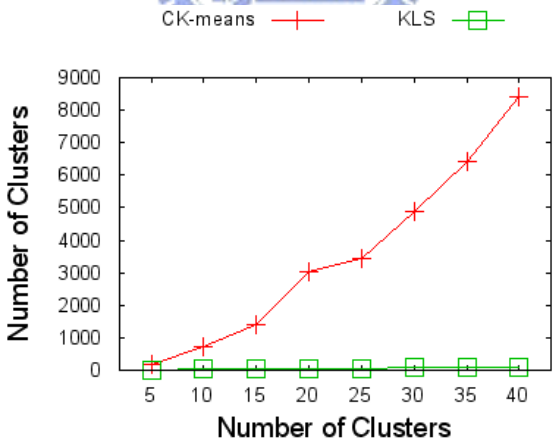
(b) Difference between precision.



(c) Difference between recall.



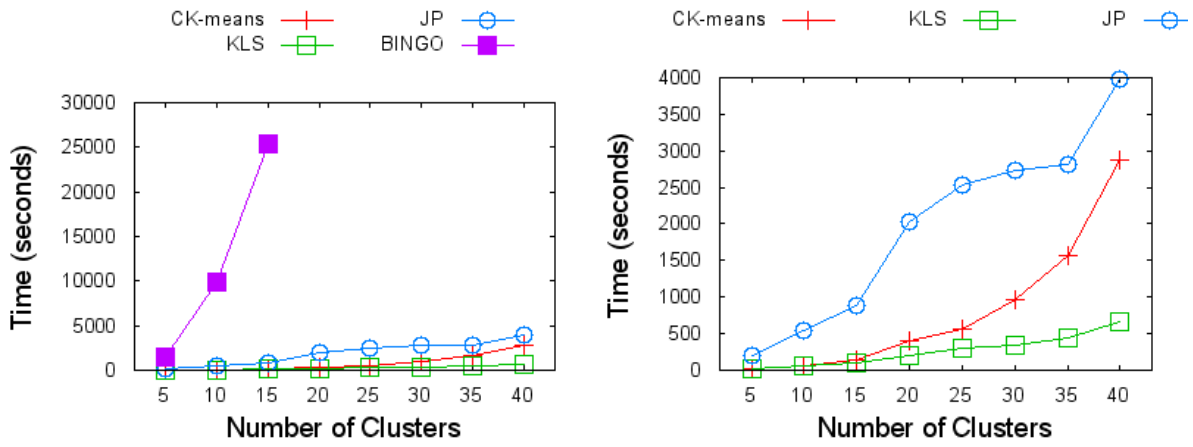
(d) Difference between average cost.



(e) Number of clusters found.

Figure 5.5: Comparison between the coarse and fine clustering for CK-means and KLS.

From the above discussion, we find that CK-means performed bad because of its decreasing precision, so what is the behind reason? Note that the clusters found by the coarse clustering



(a) Execution time of all algorithms

(b) Execution time of all except BINGO

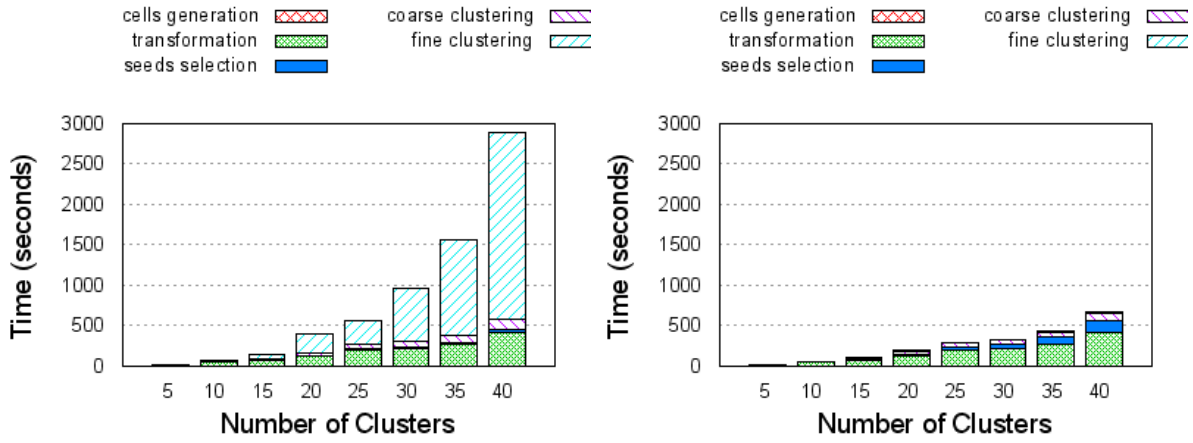
Figure 5.6: Comparison among the running time of all algorithms.

cannot be divided in the fine clustering. The idea of fine clustering is to combine two similar clusters which have good precision to a new larger cluster such that it can maintain a high precision and result a higher recall. However, when the number of clusters increases, clusters with similar attributes in the optimization domain increase, too. This results that many objects which have similar attributes in the optimization domain but dissimilar attributes in the geography domain are clustered by K-means in CK-means, and generates large number of clusters (as shown in Figure 5.5(e)) after dividing the clusters which violate the connected constraint. The large number of clusters increase the difficulty of merging clusters. Therefore, the quality loses. Moreover, the large number of clusters also hurts the computation time as we will see in the next section.

5.3 Time Comparison

As shown in Figure 5.6(a), BINGO is much slower than the other algorithms. Except T-regions generation is not efficient, our revision for connected problem prevent its efficiency. Figure 5.6(b) shows the average running time of algorithms except BINGO. Apparently, KLS is more efficient than the others. JP is not efficient due to the generation of SNN.

Figure 5.7 shows the detailed running time for CK-means and KLS. In Figure 5.7(a), the



(a) Execution time of using CK-means

(b) Execution time of using KLS

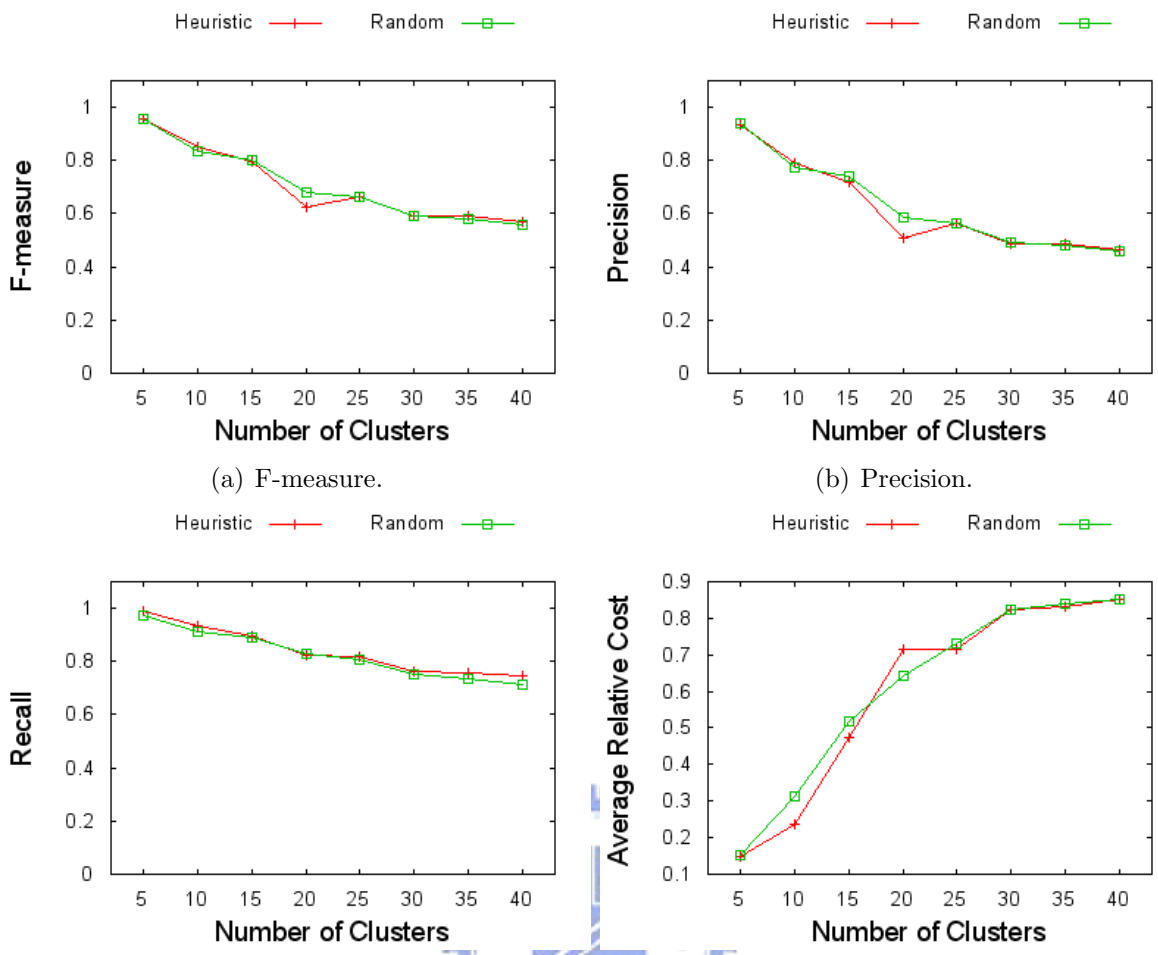
Figure 5.7: Comparison between the running time of CK-means and KLS.

execution time of the fine clustering dominates the overall execution time of CK-means. This can be understood by Figure 5.5(e) again since the time complexity of the fine clustering is determined by the number of clusters. On the other hand, in Figure 5.7(b), the transformation dominates the execution time which is the necessary computation dominates the execution time. Note that even there are more than 100,000 objects, KLS can finish in about ten minutes.

5.4 Comparison of Initial Seeds

In order to judge the quality of the heuristic method of choosing the initial seeds by cells. For each test case, KLS with random seeds is executed 10 times. The average values of F, P, R and costs are compared to KLS with the heuristic seeds selection. The results of all values are highly similar, and the difference between the average F of eight test suits is only 0.003. This implies that the heuristic method can find the seeds with the average quality. Although it cannot increase the quality much, it still saves the execution time of repeating the experiments with different random seeds.

Figure 5.9 shows the detailed comparison in test case 38 with $k = 40$. We choose test case 38 as the example of the detailed comparison because it has large number of objects

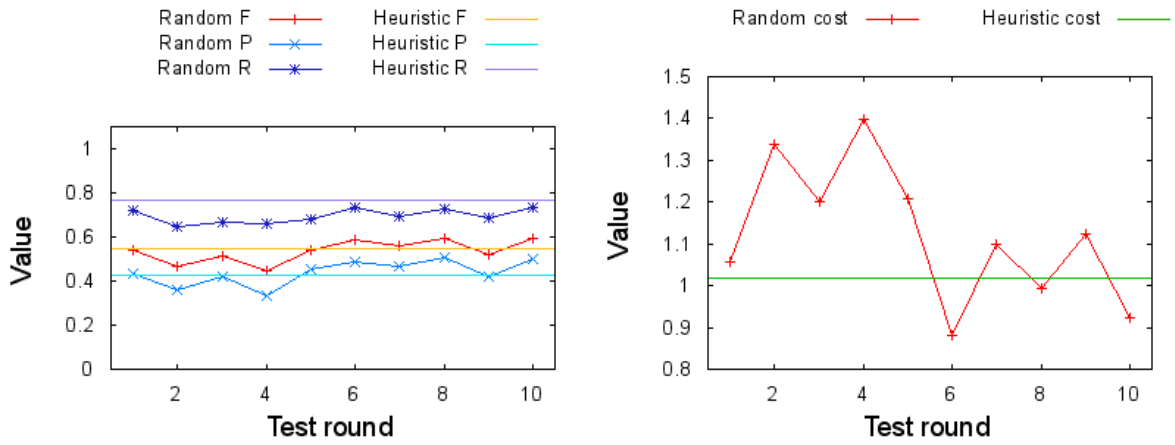


(a) F-measure. (b) Precision.

(c) Recall. (d) Average cost.

Figure 5.8: Overall results between our heuristic seeds selection and random selection.

and clusters, and the average F-measures of random and heuristic seeds selections are similar. The average values of random seeds selection and the corresponding values of heuristic seeds selection are shown in Table 5.2. From Figure 5.9(a), heuristic selection has stable result and higher precision, but the result of higher precision is not significant in the other test cases. Figure 5.9(b) shows the same result. In summary, random seeds selection can achieve higher quality sometimes, but it costs more time to compute. On the other hand, heuristic seeds selection can achieve stable and good result generally. In large scale data, the usefulness of heuristic seeds selection would be more significant since the time cost of one execution is very expensive.



(a) F-measure, precision and recall.

(b) Average cost.

Figure 5.9: Detailed results between our heuristic seeds selection and random selection in test case 38.

Seeds selection methods	Avg. F	Avg. P	Avg. R	Avg. cost
Random	0.536	0.438	0.694	1.123
Heuristic	0.548	0.426	0.770	1.02

Table 5.2: Result of Test case 38: comparison between random and heuristic seeds selection.

Chapter 6

Conclusion

In this paper, we formulated the joint clustering problem in which a connected constraint and the number of clusters should be specified. We proposed the algorithm K-means with Local Search (abbreviated as KLS). Specifically, algorithm KLS consists of three phases: the transformation phase, the coarse clustering phase, and the fine clustering phase. In the transformation phase, we only consider the connected constraint and then derive the ConGraph. Thus, in the coarse clustering phase, by exploring local search in the ConGraph with a global priority queue, rough cluster results are derived. In the fine clustering phase, these cluster results are able to further adjust so as to fit the goal. Experiments were conducted and the results show the effectiveness and efficiency of our proposed algorithm.

Bibliography

[1] S. Basu and I. Davidson. Clustering with Constraints: Theory and Practice. In *ACM SIGKDD tutorial*, 2006.

[2] J. L. Bentley and J. H. Friedman. Data Structures for Range Searching. *ACM Comput. Surv.*, 11(4):397–409, December 1979.

[3] P. S. Bradley and U. M. Fayyad. Refining Initial Points for K-Means Clustering. In *Proceedings of the 15th International Conference on Machine Learning (ICML 1998)*, Madison, Wisconson, USA, July 24-27, 1998.

[4] I. Davidson and S. S. Ravi. Clustering with Constraints: Feasibility Issues and the K-Means Algorithm. In *Proceedings of the 5th SIAM International Conference on Data Mining (SDM 2005)*, Newport Beach, CA, USA, April 21-23, 2005.

[5] M. Ester, R. Ge, B. J. Gao, Z. Hu, and B. Ben-Moshe. Joint Cluster Analysis of Attribute Data and Relationship Data: the Connected k-Center Problem. In *Proceedings of the 6th SIAM International Conference on Data Mining (SDM 2006)*, Bethesda, MD, USA, April 20-22, 2006.

[6] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, September 2000.

[7] E. Kanoulas, Y. Du, T. Xia, and D. Zhang. Finding Fastest Paths on A Road Network with Speed Patterns. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006)*, Atlanta, GA, USA, April 3-8, 2006.

[8] D. Klein, S. Kamvar, and C. Manning. From Instance-Level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering. In *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, University of New South Wales, Sydney, Australia, July 8-12, 2002.

[9] C. R. Lin, K. H. Liu, and M. S. Chen. Dual Clustering: Integrating Data Clustering over Optimization and Constraint Domains. *IEEE Trans. on Knowledge and Data Engineering*, 17:628–637, May 2005.

[10] C.-H. Lo, W.-C. Peng, C.-W. Chen, T.-Y. Lin, and C.-S. Lin. CarWeb: A Traffic Data Collection Platform. In *Proceedings of the 9th International Conference on Mobile Data Management (MDM 2008)*, Beijing, China, April 27-30, 2008.

[11] F. Moser, R. Ge, and M. Ester. Joint Cluster Analysis of Attribute and Relationship Data Withouta-Priori Specification of the Number of Clusters. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2007)*, San Jose, CA, USA, August 12-15, 2007.

[12] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, December 2002.

[13] C. H. Tai, B. R. Dai, and M. S. Chen. Incremental Clustering in Geography and Optimization Spaces. In *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007)*, Nanjing, China, May 22-25, 2007.

[14] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. ADDISON WESLEY, June 2006.

[15] K. Wagstaff and C. Cardie. Clustering with Instance-Level Constraints. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, Stanford University, Standord, CA, USA, June 29 - July 2, 2000.