

# On the Optimal Clustering of Sequential Data

*Cheng-Ru Lin and Ming-Syan Chen<sup>\*†</sup>*

## Abstract

Data clustering has attracted a lot of attention in the field of computational statistics and data mining. Notice, however, that in many applications not only the attributes but also the sequence of the objects has to be considered for clustering. Specifically, in some applications, a set of sequential data has to be partitioned into clusters in such a way that all the data points in each cluster form a continuous region. This clustering capability, termed sequential clustering in this paper, can be applied to analyze the moving pattern of an object or the status log of a running machine. Due to its continuous constraint, prior results on data clustering cannot be applied to solve this problem, thus calling for the design of new algorithms. To remedy this, we shall explore the problem of optimal sequential clustering in this paper. Specifically, we first prove that this optimal sequential clustering problem addressed in this paper possesses the *optimal substructure property* which means that an optimal solution to this problem is composed of the optimal solutions to its subproblems. In light of this property, we devise algorithm **SC<sub>OPT</sub>** to obtain the optimal solution of the problem. The time complexity of algorithm **SC<sub>OPT</sub>** is  $O(kn^2)$ , where  $n$  is the size of the dataset and  $k$  is the number of clusters. To further reduce its complexity, we devise a greedy algorithm, **SC<sub>GD</sub>**, which solves the problem in linear time. Extensive experimental studies are conducted to evaluate the performance and the effectiveness of these two algorithms. It is shown that algorithm **SC<sub>GD</sub>** is able to obtain the solution clustering of very high quality which is in fact very close to that obtained by algorithm **SC<sub>OPT</sub>**.

---

<sup>\*</sup>Electrical Engineering Department of National Taiwan University, Taipei, Taiwan

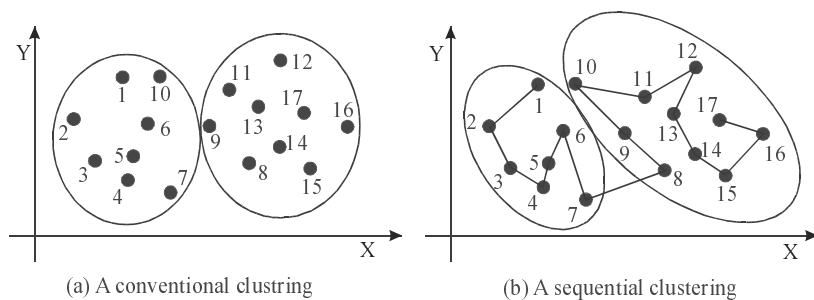
<sup>†</sup>Email: {owenlin, mschen@arbor.ee.ntu.edu.tw}

# 1 Introduction

Recently, data mining has attracted a significant amount of research attention due to its usefulness in many applications, including selective marketing, decision support, business management, and user profile analysis, to name a few [4][5][9][12]. Among others, data clustering is one of the most active research areas [14][16]. The data clustering techniques can be used to do similarity search, pattern recognition, trend analysis, grouping, classification, and so forth [5][12][20]. Traditionally, objects are clustered mainly according to their similarity. Given the definition of distance (i.e., measure of dissimilarity) between any two objects, clustering is the process that aims to minimize the cost (i.e., dissimilarity) of each cluster, so that similar objects are grouped into the same cluster. Several techniques have been devised for data clustering, including hierarchical clustering algorithms [22], partitional algorithms [8], nearest neighbor clustering algorithms [17], fuzzy clustering algorithms [2], and neural networks algorithms [13]. Generally, there are two types of attributes associated with the objects discussed: numerical ones [1][3][19][23][25] and categorical ones [10][24]. Since clustering is a very service dependent issue, its potential applications usually impose certain constraints. In [3], the authors proposed a modified k-means clustering algorithm which ensures each cluster to contain at least a predetermined number of objects. The work in [23] proposed a more general method for constrained clustering which is based on the *existential constraints*, meaning that each cluster contains at least a certain number of objects from a specific subset of objects.

Notice, however, that in many applications not only the attributes but also the sequence of the objects has to be considered for clustering. Specifically, in some applications, a set of sequential data has to be partitioned into clusters in such a way that all the data points in each cluster form a continuous region. This clustering capability is termed *sequential clustering* in this paper. Sequential clustering can be applied to analyze the moving pattern of an object or the status log of a running machine.

The sequential clustering problem can be best understood by the following illustrative example. As shown in Figure 1, the black nodes in Figure 1(a) are identical to those in Figure 1(b) and the number next to each node represents the order of that node in the sequence. For example, in a cellular phone service provider, it is desirable to partition and analyze the user moving patterns in different areas across the time sequence [21]. In such an application, the number next to each node can be viewed as a time stamp of a traced point of a moving user according to the CDR (i.e., Call Detail Record) database. In Figure 1(b), nodes are connected with a line according to their sequence numbers. Suppose one would like to partition those nodes into two clusters. By conventional clustering methods, such as k-means algorithm [18] or CLARANS [19], these nodes may be partitioned into the two clusters as shown in Figure 1(a). However, with the constraint of sequential clustering, all the nodes in a cluster have to form a continuous region, and the clustering in Figure 1(a) is thus not a legitimate solution since the sequence numbers  $\{1, 2, 3, 4, 5, 6, 7, 8, 10\}$  do not form a continuous region. Instead, one possible solution to this sequential clustering problem is shown in Figure 1(b),



**Figure 1.** *The difference between a conventional clustering and a sequential clustering.*

where the sequence numbers  $\{1, 2, 3, 4, 5, 6, 7\}$  are continuous, and so are those in  $\{8, 9, 10, \dots, 16\}$ , showing an instance where objects are clustered not only by their attributes but also the time sequence numbers. Note that with the sequential clustering constraint, a node in a cluster is not always closer to the centroid of its cluster than to the centroid of another cluster.

A similar problem to the sequential clustering is the signal segmentation problem, which is mostly studied in the field of signal processing [6][7][11], where one would like to partition the signals into several segments and each segment has its own distribution model. However, the objective of forming a segment in that signal processing application is different from the one we consider in the sequential clustering problem, thus calling for the design of different solution procedures.

Because of this continuous constraint, prior results on data clustering cannot be applied to solve this problem. Consequently, we shall explore the problem of optimal sequential clustering in this paper. Specifically, we first prove that this optimal sequential clustering problem addressed in this paper possesses the *optimal substructure property* which means that an optimal solution to this problem is composed of the optimal solutions to its subproblems. In light of this property, we devise an algorithm, denoted by  $\mathbf{SC}_{\text{OPT}}$ , to obtain the optimal solution of this sequential clustering problem. The time complexity of algorithm  $\mathbf{SC}_{\text{OPT}}$  is proved to be  $O(kn^2)$ , where  $n$  is the size of the dataset and  $k$  is the number of clusters. Though algorithm  $\mathbf{SC}_{\text{OPT}}$  is able to handle up to thousands of data points efficiently, it is desirable to further reduce the complexity, since the order of many transaction databases, such as the CDR one, is in general very high and is in fact fast growing in recent applications. To further reduce the complexity, we design an algorithm, denoted by algorithm  $\mathbf{SC}_{\text{GD}}$  to obtain greedy solutions. It will be shown that by exploiting the hill climbing technique, algorithm  $\mathbf{SC}_{\text{GD}}$  leads to solution clustering with the quality very close to that by algorithm  $\mathbf{SC}_{\text{OPT}}$  while only incurring linear time complexity.

We conduct several experiments on the synthetic workloads for performance studies. It is shown by our experimental results that compared to algorithm  $\mathbf{SC}_{\text{OPT}}$ , algorithm  $\mathbf{SC}_{\text{GD}}$  not only incurs a very scalable execution time but also leads to

the clustering results of very good quality. From the experiments, it is shown that algorithm  $\mathbf{SC}_{\mathbf{GD}}$  is able to deal with a huge amount of data very efficiently, showing the very advantage of its linear time complexity.

The rest of the paper is structured as follows. Section 2 presents the problem formulation of sequential clustering. Algorithm  $\mathbf{SC}_{\mathbf{OPT}}$  and algorithm  $\mathbf{SC}_{\mathbf{GD}}$  are presented in Section 3. Performance studies are conducted in Section 4. We conclude this paper in Section 5.

## 2 Problem Definition

In this paper, we develop our algorithms mainly based on the partitional algorithms on numerical attributes. Several well-known algorithms, including k-means clustering algorithm, PAM [15], CLARA, and CLARANS [19], belong to this category. The similarity measurement is the foundation of forming clusters. Same as in prior works, we use the distance between two objects as the measurement of their dissimilarity. Among others, Euclidean distance and squared Euclidean distance are two most popular measurements whose definitions are given below.

**Definition 1** (Euclidean and squared Euclidean distance):

Let  $o_i$  and  $o_j$  be two objects of the dataset with  $(x_{i,1}, x_{i,2}, \dots, x_{i,d})$  and  $(x_{j,1}, x_{j,2}, \dots, x_{j,d})$  as their coordinates, where  $d$  is the degree of dimension. The Euclidean and squared Euclidean distance are defined as:

$$D_E(o_i, o_j) = \left( \sum_{k=1}^d (x_{i,k} - x_{j,k})^2 \right)^{\frac{1}{2}} = \|o_i - o_j\|,$$

$$D_{E^2}(o_i, o_j) = \sum_{k=1}^d (x_{i,k} - x_{j,k})^2 = \|o_i - o_j\|^2.$$

Note that  $(x_{i,1}, x_{i,2}, \dots, x_{i,d})$  represents the features or attributes of the object  $o_i$ .

In this paper, without loss of generality, we adopt squared Euclidean distance function as the dissimilarity measurement. The quality of a cluster can then be determined to reflect the goodness of clustering and a cost function is used as a quality measurement. Generally, the cost function is so designed to represent the penalty of dissimilarity of the objects within a cluster. The clustering problem is defined as the one to determine a partition of the dataset that minimizes the sum of the costs of all clusters. The cost function used in this paper is given below, which is in essence the same as those in related works [18].

**Definition 2** (Cost of a cluster):

Given a cluster  $Cl$  with  $k$  data points  $\{o_1, o_2, \dots, o_k\}$ , where  $o_i$  can be represented by  $(x_{i,1}, x_{i,2}, \dots, x_{i,d})$  and  $d$  is the degree of dimension, the cost of  $Cl$

is defined as

$$Cost(Cl) = \sum_{i=1}^k D_{E^2}(o_i, c),$$

where  $c$  is the centroid of these points, i.e.,  $c = \frac{\sum_{i=1}^k o_i}{k}$ .

Note that the centroid is the point that minimizes the cost when it is measured in squared Euclidean distance. This explains the choice of  $c$ .

As mentioned earlier, since clustering is a very service dependent issue, its potential applications usually impose certain constraints. The sequential cluster problem is similar to the traditional clustering problem, except that all the objects considered contain not only their attributes but also individual sequence numbers, and all objects belonging to the same cluster have to be continuous with respect to that sequence. In this paper, a *partition* refers a division of an integer interval into several continuous regions, each of which is called a *segment*. An example partition of  $\{1, 2, 3, 4, 5\}$  could be  $\{\{1, 2, 3\}, \{4, 5\}\}$ , where  $\{1, 2, 3\}$  and  $\{4, 5\}$  are also called segments. A partition composed of  $k$  segments is called a  $k$ -partition. The terms *data point* and *object* are used interchangeably in this paper. The definition of sequential clustering problem is stated as follows.

**Definition 3** (Sequential clustering problem)

Given a dataset  $D$  of  $n$  objects  $\{o_1, o_2, \dots, o_n\}$  and a desired number of clusters  $k$ , each object  $o_i$  is given a unique sequence number  $s_i \in [1, n]$ .  $\{Cl_1, Cl_2, \dots, Cl_k\}$  is said to be a  $k$ -partition of the dataset, if it is a disjoint partition of the dataset, i.e.,  $Cl_i \cap Cl_j = \phi, \forall i \neq j$ , and the sequence numbers of all objects within each  $Cl_i$  are continuous. The sequential clustering problem is defined as the problem of determining the  $k$ -partition in such a way that the total cost, i.e.,  $\sum_{i=1}^k Cost(Cl_k)$ , is minimized.

From the definition of the sequential clustering problem, it follows that each segment corresponds to one cluster of data points. In addition, the sum of the objects' features of a cluster  $\{o_1, o_2, \dots, o_n\}$  is defined as  $\overrightarrow{\text{sum}} = (\sum_{o \in Cl} o.x_1, \sum_{o \in Cl} o.x_2, \dots, \sum_{o \in Cl} o.x_d)$ , and the *square sum* of the cluster is defined as  $\overrightarrow{\text{sqsum}} = (\sum_{o \in Cl} o.x_1^2, \sum_{o \in Cl} o.x_2^2, \dots, \sum_{o \in Cl} o.x_d^2)$ . According to these definitions, we can derive the following theorem which is important for determining the cost of a cluster.

**Theorem 1:** Given a cluster  $Cl = \{o_1, o_2, \dots, o_n\}$ , the cost of the cluster can be determined by the following formula.

$$Cost(Cl) = \left( \overrightarrow{\text{sqsum}} - \frac{\overrightarrow{\text{sum}}^2}{n} \right) \cdot (1, 1, 1, \dots, 1),$$

where  $\overrightarrow{\text{sum}}^2$  is defined to be  $\overrightarrow{\text{sum}} \cdot \overrightarrow{\text{sum}}$ .

*Proof:* Without loss of generality, the degree of dimension is assumed to be two, i.e., there are only two features,  $o.x$  and  $o.y$ , of an object  $o$ . Then center  $c$  of a cluster  $Cl$  can be expressed as  $c = (\bar{x}, \bar{y}) = \frac{1}{n} \sum_{o \in Cl} (o.x, o.y)$ . Thus, the cost of the cluster can be expressed as

$$\begin{aligned} Cost(Cl) &= \sum_{o \in Cl} \left( (o.x - \bar{x})^2 + (o.y - \bar{y})^2 \right) = \sum_{o \in Cl} (o.x - \bar{x})^2 + \sum_{o \in Cl} (o.y - \bar{y})^2 \\ &= \left( \sum_{o \in Cl} o.x^2 \right) + \left( \sum_{o \in Cl} o.y^2 \right) - n \times \bar{x}^2 - n \times \bar{y}^2 \\ &= \left( \overrightarrow{\text{sqsum}} - \frac{\overrightarrow{\text{sum}}^2}{n} \right) \cdot (1, 1, 1, \dots, 1). \end{aligned}$$

**Q.E.D.**

For example, given a cluster  $\{(1, 1), (2, 3), (4, 2)\}$ , we have  $\overrightarrow{\text{sum}} = (7, 6)$  and  $\overrightarrow{\text{sqsum}} = (21, 14)$ . Then the cost of the cluster is  $\left( (21, 14) - \frac{(49, 36)}{3} \right) \cdot (1, 1) = 6.667$ . From Theorem 1, it can be seen that to determine the cost of a cluster, one would only need the values of three parameters, i.e.,  $n$ ,  $\overrightarrow{\text{sum}}$ , and  $\overrightarrow{\text{sqsum}}$ , for this cluster. To facilitate our description, we define a new terminology, *information triplet* of a cluster, as follows.

**Definition 4** (Information Triplet)

Given a cluster of  $n$  objects and their features, the 3-attribute tuple,  $(n, \overrightarrow{\text{sum}}, \overrightarrow{\text{sqsum}})$ , is called the *information triplet* of the cluster.

Theorem 1 leads to the following corollary.

**Corollary 1.1:** Given the information triplets of two clusters,  $(\overrightarrow{\text{sum}}_i, \overrightarrow{\text{sqsum}}_i, n_i)$  of  $Cl_i$  and  $(\overrightarrow{\text{sum}}_j, \overrightarrow{\text{sqsum}}_j, n_j)$  of  $Cl_j$ , the information triplet of the new cluster formed by merging the two disjoint clusters can be expressed as

$$\left( \overrightarrow{\text{sum}}_i + \overrightarrow{\text{sum}}_j, \overrightarrow{\text{sqsum}}_i + \overrightarrow{\text{sqsum}}_j, n_i + n_j \right).$$

According to Corollary 1.1, we can obtain the information triplet of a new cluster formed by the merge of two clusters in only constant time, which in turns, means that the cost of this newly formed cluster can be obtained in constant time. This property is important and will be fully utilized in  $\text{SC}_{\text{OPT}}$  and  $\text{SC}_{\text{GD}}$  to reduce their execution complexity.

### 3 Algorithms for Sequential Clustering

In this section, two algorithms are devised to solve the sequential clustering problem. Specifically, we first prove that this optimal sequential clustering problem possesses the *optimal substructure property* which means that an optimal solution to the problem contains the optimal solutions to its subproblems. In light of this property,

we devise algorithm  $\mathbf{SC}_{\text{OPT}}$  to obtain the optimal solution of the problem. The time complexity of algorithm  $\mathbf{SC}_{\text{OPT}}$  is  $O(kn^2)$ , where  $n$  is the size of the dataset and  $k$  is the number of clusters. To further reduce its complexity, we devise a greedy algorithm, algorithm  $\mathbf{SC}_{\text{GD}}$ , based on the hill climbing technique to solve the problem in the linear time complexity.

### 3.1 Development of Optimal Algorithm $\mathbf{SC}_{\text{OPT}}$

We first derive Lemma 1 which states that this sequential clustering problem possesses the property of optimal substructure.

**Lemma 1** (Optimal substructure property): Given a dataset  $D$  of  $n$  sequential objects  $o_1, o_2, \dots, o_n$ , suppose  $\{Cl_1, Cl_2, \dots, Cl_k\}$  is an optimal  $k$ -partition. Then, for any  $i, j$ , such that  $1 < i < j < k$ ,  $\{Cl_i, Cl_{i+1}, \dots, Cl_j\}$  is an optimal  $(j - i + 1)$ -partition for the subdataset  $D'$ , where  $D' = \cup_{t=i}^j Cl_t$ .

*Proof:* We prove this lemma by contradiction. Suppose that  $\{Cl_i, Cl_{i+1}, \dots, Cl_j\}$  is not an optimal  $(j - i + 1)$ -partition of the dataset  $D'$ , i.e., there exists another  $(j - i + 1)$ -partition of  $D'$ ,  $\{Cl'_i, Cl'_{i+1}, \dots, Cl'_j\}$ , that is different from  $\{Cl_i, Cl_{i+1}, \dots, Cl_j\}$  and  $\sum_{t=i}^j \text{Cost}(Cl'_t) < \sum_{t=i}^j \text{Cost}(Cl_t)$ . Consider this new  $k$ -partition of  $D$ ,  $\{Cl_1, Cl_2, \dots, Cl_{i-1}, Cl'_i, Cl'_{i+1}, \dots, Cl'_j, Cl_{j+1}, \dots, Cl_k\}$ . The total cost of the new  $k$ -partition is  $\sum_{t=1}^k \text{Cost}(Cl_k) + (\sum_{t=i}^j \text{Cost}(Cl'_t) - \sum_{t=i}^j \text{Cost}(Cl_t))$ . Since  $\sum_{t=i}^j \text{Cost}(Cl'_t) < \sum_{t=i}^j \text{Cost}(Cl_t)$ , the cost the new partition is less than that of the original one, i.e.,  $\sum_{t=1}^k \text{Cost}(Cl_k)$ . This is a contradiction to the fact that  $\{Cl_1, Cl_2, \dots, Cl_k\}$  is an optimal  $k$ -partition. This lemma follows. **Q.E.D.**

From Lemma 1, it follows that the dynamic programming technique can be employed to obtain the optimal solution of this sequential clustering problem.

#### Algorithm $\mathbf{SC}_{\text{OPT}}$ : $\mathbf{SC}_{\text{OPT}}(D, k)$

//Find an optimal  $k$ -partition of the dataset  $D = \{o_1, o_2, \dots, o_n\}$

1. **set**  $n = |D|$ ;
2. **for**  $j = 1$  **to**  $n$
3.   **set**  $\text{left}[1][j] = \text{Cost}(\{o_1, o_2, \dots, o_j\})$ ;
4.   **set**  $\text{size}[1][j] = (j)$ ;
5. **for**  $i = 2$  **to**  $k$
6.   **for**  $j = n$  **to**  $i + 1$  **desc**
7.     **set**  $\text{left}[i][j] = \infty$ ;
8.     **for**  $l = 1$  **to**  $j - k$
9.       **if**  $(\text{Cost}(\{o_{j-l+1}, o_{j-l+2}, \dots, o_j\}) + \text{left}[i-1][j-l] < \text{left}[i][j])$
10.        **set**  $\text{left}[i][j] = \text{Cost}(\{o_{j-l+1}, o_{j-l+2}, \dots, o_j\}) + \text{left}[i-1][j-l]$ ;
11.        **set**  $\text{size}[i][j] = (\text{size}[i-1][j-l], l)$ ;
12. **return**  $\text{size}[k][n]$ ;

An important technique of algorithm  $\mathbf{SC}_{\text{OPT}}$  is the reuse of stored information in previous iterations, i.e., arrays  $\text{left}$  and  $\text{size}$ . To exploit the property of optimal substructure, two arrays,  $\text{left}$  and  $\text{size}$ , are maintained during the execution of the

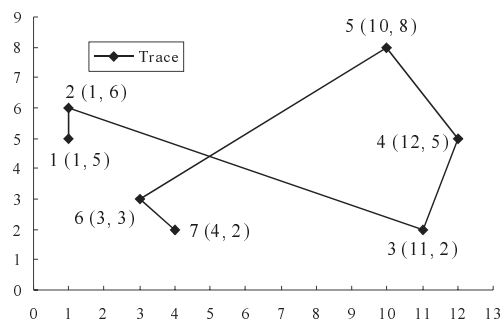


Figure 2. An example trace log of a moving object.

algorithm. The minimal cost of all possible  $i$ -partitions of the subdataset formed by the first  $j$  objects, i.e.,  $\{o_1, o_2, \dots, o_j\}$ , is stored in  $left[i][j]$ , and the size of each cluster of the optimal solution is stored in  $size[i][j]$  of which each element is a list. For example, if  $\{Cl_1, Cl_2, \dots, Cl_h\}$  is the optimal  $h$ -partition of  $D'$ , where  $D'$  is formed by the first  $m$  objects in the dataset, i.e.,  $D' = \{o_1, o_2, \dots, o_m\}$ , then  $left[h][m] = \sum_{i=1}^h Cost(Cl_i)$  and  $size[h][m] = (|Cl_1|, |Cl_2|, \dots, |Cl_h|)$ . After the execution of the algorithm, the minimal cost of all the  $k$ -partitions of the whole dataset  $D$  will be stored in  $left[k][n]$ , and the corresponding clusters partitioned can be directly derived from  $size[k][n]$ .

Explicitly, from Step 1 to Step 4, algorithm **SC<sub>OPT</sub>** sets the initial value of  $left$  and  $size$ . Each  $left[1][j]$ , for  $j$  from 1 to  $n$ , is initially set as the cost of the cluster  $\{o_1, o_2, \dots, o_j\}$ , and each  $size[1][j]$ , for  $1 \leq j \leq n$ , is set as a list of only one element with the value being the size of that cluster, i.e.,  $size[1][j] = (j)$ . Then, the algorithm enters the main iteration. At the  $t$ -th iteration (i.e.,  $i = t + 1$ ), by considering all the possible sizes of the last cluster together with the information stored in the previous iterations, the algorithm will determine the optimal  $i$ -partition for the objects in the front. An illustrative example for the operations of algorithm **SC<sub>OPT</sub>** is given below.

$left[i][j]$	7	6	5	4	3	2	1
1	169.7	158.2	140.8	119.8	75.3	0.5	0
2	96.5	71.5	20.5	5.5	0.5	0	–
3	21.5	20.5	5.5	0.5	0	–	–

Table 1. Values in the array  $left$  in Example 3.1

$size[i][j]$	7	6	5	4	3	2	1
1	(7)	(6)	(5)	(4)	(3)	(2)	(1)
2	(2, 5)	(2, 4)	(2, 3)	(2, 2)	(2, 1)	(1, 1)	–
3	(2, 3, 2)	(2, 3, 1)	(2, 2, 1)	(2, 1, 1)	(1, 1, 1)	–	–

Table 2. Values in the array  $size$  in Example 3.1

**Example 3.1:** Consider the trace of a moving object shown in Figure 2. The coordinates of these points are  $(1, 5)$ ,  $(1, 6)$ ,  $(11, 2)$ ,  $(12, 5)$ ,  $(10, 8)$ ,  $(3, 3)$ , and  $(4, 2)$ . Values in the array  $left$  are shown in Table 1 and those in the array  $size$  are shown in Table 2. The procedure to determine the values of array  $left$  and  $size$  is described below. At the beginning of last iteration, i.e.,  $i = 3$  and  $j = 7$ , the values of  $left[3][7]$  and  $size[3][7]$  are determined at the algorithm from Step 7 to Step 11. Note that, the values of  $size[2][j]$  and  $left[2][j]$ , for  $j$  from 2 to 7, are obtained in the previous iteration. The answer of  $left[3][7]$  and  $size[3][7]$  could be obtained directly by evaluating all of the possible sizes of the last cluster. When the size of last cluster is one, we have  $Cl_3 = \{o_7\}$ . Thus the information triplet of  $Cl_3$  is  $((4, 2), (16, 4), 1)$ , and  $Cost(Cl_3) = 0$  since  $Cl_3$  has only one object. Then, from Lemma 1, it follows that the first  $(7 - 1) = 6$  objects should be optimally partitioned into two clusters. From  $size[2][6] = (2, 4)$ , we have its optimal 2-partition of as  $\{\{o_1, o_2\}, \{o_3, o_4, o_5, o_6\}\}$ . From  $left[2][6] = 71.5$ , it follows that the total cost of these two cluster is 71.5. Hence, as the size of the last cluster is 1, the optimal cost is  $(0 + 71.5) = 71.5$ . When the size of last cluster becomes 2, the information triplet of the last cluster is updated to  $((7, 5), (25, 13), 2)$  whose cost is 1. Then the first  $(7 - 2) = 5$  objects are partitioned into two clusters. Since  $left[2][5] = 20.5$  and  $size[2][5] = (2, 3)$ , one of the candidate value of  $left[3][7]$  is  $(1 + 20.5) = 21.5$ . After evaluating all of the possible sizes of the last cluster, it can be shown that 21.5 is the minimal value. Details are shown in Table 3. From  $size[3][7] = (2, 3, 2)$ , we obtain the optimal 3-partition of the 7 objects as  $\{\{o_1, o_2\}, \{o_3, o_4, o_5\}, \{o_6, o_7\}\}$ .

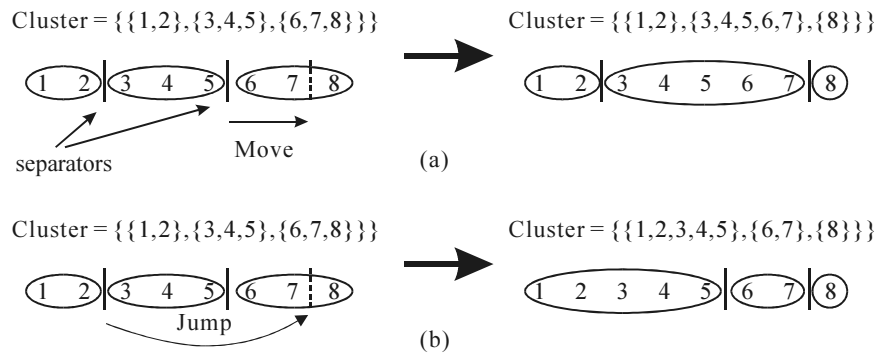
Size of last cluster	1	2	3	4	5
Matched $left$	71.5	<b>20.5</b>	5.5	0.5	0
Matched $size$	(2, 4)	(2, 3)	(2, 2)	(2, 1)	(1, 1)
Cost of last cluster	0	<b>1</b>	49.3	79.8	96
Total cost	71.5	<b>21.5</b>	54.8	80.3	96

Table 3. The profile when the last cluster is examined

Note that in our design, the  $Cost$  function used in Step 9 and Step 10 can be determined in only constant time, as proved by Theorem 1. This means that we can obtain the cost of a new cluster only by the information triplet instead of recalculating it.

**Theorem 2:** The time complexity of algorithm  $\mathbf{SC}_{OPT}$  is  $O(kn^2)$ .

*Proof:* In the algorithm, the statements from Step 9 to Step 11 will be executed for  $\sum_{i=2}^k \sum_{j=i+1}^n (j - k) = \frac{1}{6} (k - 1) (2k^2 - 6kn + 3n^2 + 2k + 3n - 6)$  times. As shown



**Figure 3.** Examples of move and jump of a separator.

in Theorem 1, each iteration of these statements is executed in constant time. Since  $k$  is in general smaller than  $n$ , the time complexity of this algorithm is  $O(kn^2)$ . **Q.E.D.**

According to the time complexity, the algorithm can be applied to a dataset of up to thousands of objects.

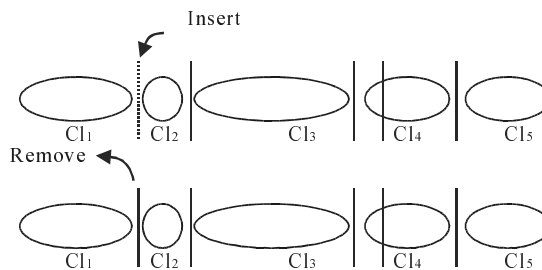
**Theorem 3:** The space complexity of algorithm  $\mathbf{SC}_{\text{OPT}}$  is  $O(kn)$ .

*Proof:* Note that the only information we need to calculate  $size[i][j]$  and  $left[i][j]$  is stored in  $size[i-1][t]$  and  $left[i-1][t]$ , for  $t \in \{k, k+1, \dots, j-1\}$ . Moreover, the value of  $j$  is decreased from  $n$  to  $i+1$  in the algorithm. Hence, we can reduce array  $size$  and  $left$  to only 1 dimension by removing the first dimension  $[i]$ , leading to the fact that the space complexity of algorithm  $\mathbf{SC}_{\text{OPT}}$  is only  $O(kn)$ . **Q.E.D.**

### 3.2 Development of Greedy Algorithm $\mathbf{SC}_{\text{GD}}$

Although algorithm  $\mathbf{SC}_{\text{OPT}}$  is able to obtain optimal solutions of the sequential clustering problems, as conformed by our experimental results, the time complexity of  $\mathbf{SC}_{\text{OPT}}$  allows it of handling only up to thousands of data points in reasonable time. However, as pointed out earlier, this might not be adequate for some applications. Thus, we propose a greedy hill climbing algorithm, algorithm  $\mathbf{SC}_{\text{GD}}$ , for better execution efficiency.

To facilitate our description of algorithm  $\mathbf{SC}_{\text{GD}}$ , we use the term *separator* to refer to the division of two neighboring clusters. For example, in Figure 3(a), there are two separators (shown in solid lines) that divide the numbers into 3 segments, i.e.,  $\{\{1,2\}, \{3,4,5\}, \{6,7,8\}\}$ . We next define two actions of a separator, namely *move* and *jump*, to perform re-clustering. In Figure 3(a), a move action moves the second separator (between 5 and 6) to a new position (between 7 and 8), resulting in a new clustering of  $\{\{1,2\}, \{3,4,5,6,7\}, \{8\}\}$ . In Figure 3(b), a jump action relocates the first separator (between 2 and 3) to a new position (between 7 and 8), leading to another clustering, i.e.,  $\{\{1,2,3,4,5\}, \{6,7\}, \{8\}\}$ . Note that a move



**Figure 4.** *The definition of insertion and removal.*

action can only move a separator and insert it into neighboring clusters, whereas a jump action is allowed to relocate a separator to nonadjacent clusters.

In fact, either of the two actions, i.e., move or jump, can be decomposed into two basic operations, i.e., insertion and removal. The two operations are shown in Figure 4. However, for the move action, the removal has to be immediately before the insertion, and there is thus unnecessary to perform such a decomposition for the move action since no extra flexibility will be allowed. On the other hand, there is no removal/insertion order imposed for a jump action and we hence decompose a jump action into a removal and an insertion. As such, we can estimate the cost reduction resulted by a jump action in advance by evaluating all of the possible insertion and removal operations. With the decomposition of the jump action, we are to use the move actions, insertion operations and removal operations to perform the clustering. In algorithm  $\mathbf{SC}_{\text{GD}}$ , the move actions and insertion operations are designed to attain the local optimal, i.e., a separator will be moved to and/or inserted to a position that maximizes the cost reduction.

**Algorithm  $\mathbf{SC}_{\text{GD}}$ :**  $\mathbf{SC}_{\text{GD}}(D, k)$

//Find a sub-optimal  $k$ -partition of the dataset  $D = \{o_1, o_2, \dots, o_n\}$

1. Set the  $k - 1$  separators to arbitrary locations. That is, the  $n$  objects are arbitrarily divided into  $k$  clusters.
2. Calculate the cost reduction of all possible jump (decomposed into insertion and removal) and move actions, and store the actions with their cost reductions into **Action-Q** in order of their cost reduction.
3. Identify the maximum cost reduction from **Action-Q**, if the cost reduction is negative then goto Step 5.
4. Perform the action, and then update **Action-Q** by removing invalid actions as well as inserting new actions. Then goto Step 3.
5. Output the total cost of all clusters, and then stop.

Since the total cost of all clusters is monotonic decreasing, algorithm  $\mathbf{SC}_{\text{GD}}$  will finally complete and reach a local minimum. Experimental results show that the algorithm converges very fast. In the section of performance studies, we will show that algorithm  $\mathbf{SC}_{\text{GD}}$  leads to solution clustering of very good quality, which is in fact very close to that by  $\mathbf{SC}_{\text{OPT}}$ . The following two theorems show the time and space complexity of algorithm  $\mathbf{SC}_{\text{GD}}$ .

**Theorem 4:** The time complexity of algorithm  $\mathbf{SC}_{\text{GD}}$  is  $O\left(\frac{nl}{k} + n\right)$ , where  $l$  is the times of execution of Step 3 and Step 4.

*Proof:* With the decomposition of the jump action, Step 2 only needs to consider  $k$  insertion operations,  $k - 1$  removal operations, and  $k - 1$  move actions instead of considering  $(k - 1)(k - 2)$  jump and  $k - 1$  move actions. The time needed to construct each insertion operation and move action is proportional to the sizes of related clusters, and is constant for each removal operation. By utilizing the heap data structure, an insertion into **Action-Q** only consumes the time in the order of  $\ln(|\text{Action} - Q|)$ , which is smaller than the order of average cluster size, meaning that the time complexity of Step 2 is  $O(n)$ . Also, an extraction from **Action-Q** only takes a constant time and the time complexity of Step 3 is thus  $O(1)$ . After the execution of the extracted action, the partition of clusters is changed. Some actions in **Action-Q** become invalid, and some new actions need to be added. The time complexity of Step 4 is also proportional to the sizes of related clusters since some move actions and insertion operations have to be reconstructed. Thus the time complexity of Step 4 is  $O\left(\frac{n}{k}\right)$ . It follows that the time complexity of algorithm  $\mathbf{SC}_{\text{GD}}$  is  $O\left(\frac{nl}{k} + n\right)$ . **Q.E.D.**

**Theorem 5:** The space complexity of algorithm  $\mathbf{SC}_{\text{GD}}$  is  $O(k)$ .

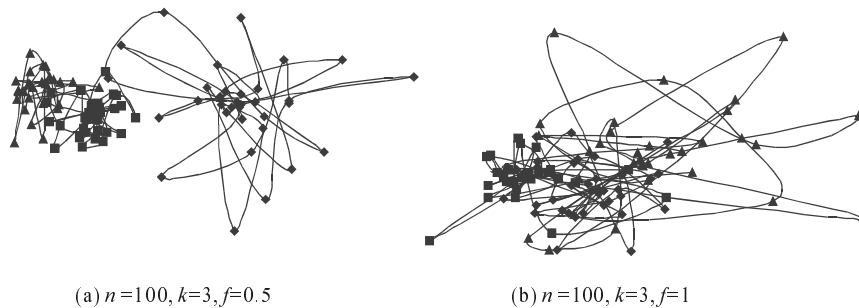
*Proof:* This theorem follows from the fact that only  $k$  insertion operations,  $k - 1$  removal operations, and  $k - 1$  move actions need to be stored in **Action-Q**, and by utilizing the information triplet structure, each action can be stored in constant space. **Q.E.D.**

An illustrative example for the operations of algorithm  $\mathbf{SC}_{\text{GD}}$  is given below.

**Example 3.2:** Consider the trace log shown in Figure 2 again. Assume the initial partition, which is arbitrarily chosen at the beginning of algorithm  $\mathbf{SC}_{\text{GD}}$ , is  $\{\{o_1, o_2, o_3, o_4\}, \{o_5, o_6\}, \{o_7\}\}$ . Recall that all the actions are designed to attain the local optimal. The legitimate actions and operations in **Action-Q** with their cost reductions are listed below.

Action Name	<i>mv_1</i>	<i>mv_2</i>	<i>ins_1</i>	<i>ins_2</i>	<i>rm_1</i>	<i>rm_2</i>
Cost Reduction	85.25	36	114.25	37	-1.417	-12.333

Note that we have only two insertion operations since a separator cannot be inserted into a cluster with only one object. The symbol *mv\_1* means the move action of the first separator, and its cost reduction is 85.25 since the difference between the original cost, i.e.,  $Cost(\{o_1, o_2, o_3, o_4\}) + Cost(\{o_5, o_6\})$ , and the local optimal cost, i.e.,  $Cost(o_1, o_2) + Cost(o_3, o_4, o_5, o_6)$ , is  $156.75 - 71.5 = 85.25$ . Similarly, *ins\_1* means the insertion of a separator into the first cluster, *rm\_1* means the removal of



**Figure 5.** The generated synthetic data of different divergence factors.

the first separator, and so on. From these actions and operations, it can be verified that the jump of the second separator into the first cluster (combination of  $ins\_1$  and  $rm\_2$ ) is the action that maximizes the cost reduction. After the execution of this action, the partition will become  $\{\{o_1, o_2\}, \{o_3, o_4\}, \{o_5, o_6, o_7\}\}$ . Since the clustering is changed after the jump, we need to reconstruct **Action-Q**. The new elements stored in **Action-Q** are listed below.

Action Name	$mv\_1$	$mv\_2$	$ins\_1$	$ins\_2$	$ins\_3$	$rm\_1$	$rm\_2$
Cost Reduction	0	33	0.5	5	48.333	-114.25	-41.667

Again, we take the action that maximizes the cost reduction, i.e.,  $mv\_2$ , and then the partition will become  $\{\{o_1, o_2\}, \{o_3, o_4, o_5\}, \{o_6, o_7\}\}$ . Note that this action does not change the first cluster, so that we need not recalculate the  $insert\_1$  operation. After this action, the elements in **Action-Q** are shown below.

Action Name	$mv\_1$	$mv\_2$	$ins\_1$	$ins\_2$	$ins\_3$	$rm\_1$	$rm\_2$
Cost Reduction	0	0	0.5	15	1	-120.3	-75

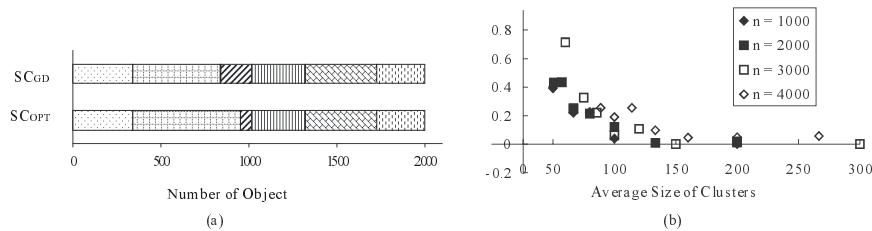
It can be verified that no further move or jump actions in **Action-Q** are available to reduce the cost. Algorithm  $\mathbf{SC}_{GD}$  thus completes. In this case, the final partition of the dataset by algorithm  $\mathbf{SC}_{GD}$  is  $\{\{o_1, o_2\}, \{o_3, o_4, o_5\}, \{o_6, o_7\}\}$  which is identical to the one obtained by algorithm  $\mathbf{SC}_{OPT}$  in Example 3.1.

## 4 Performance Studies

To assess the performance of algorithm  $\mathbf{SC}_{OPT}$  and  $\mathbf{SC}_{GD}$ , we have conducted a series of experiments on these two algorithms. These experiments are performed on a computer with 700Mhz Intel CPU and 256M of memory, running Microsoft Windows 2000.

### 4.1 Synthetic Data Generation

In order to generate the synthetic data used in our experiments. We adopt a method similar to the one in [25], by which  $n$  data points for  $k$  clusters are generated.



**Figure 6.** (a) The sizes of clusters by  $SC_{OPT}$  and  $SC_{GD}$ . (b) Cost difference between  $SC_{OPT}$  and  $SC_{GD}$  (in percentage).

Without loss of generality, every generated data point is 2 dimensional and assigned a sequence number from 1 to  $n$ . We also use a parameter  $f$ , called divergence factor, to model the divergence of data points in each cluster. That is, the larger the value of  $f$ , the further away points in each cluster (after clustering) are from their centroid. Two generated patterns with different random factors are shown in Figure 5, where objects of different clusters are represented by different symbols, i.e., the first cluster is represented by diamonds, and the second is by squares, and the third is by triangles.

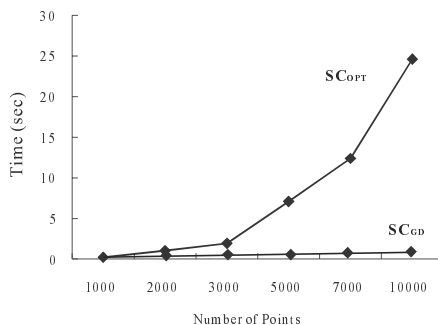
## 4.2 Experiment I: On the clustering quality of $SC_{GD}$

A series of synthetic data are generated to assess the clustering quality of  $SC_{GD}$ . The results are shown in Figure 6(a).

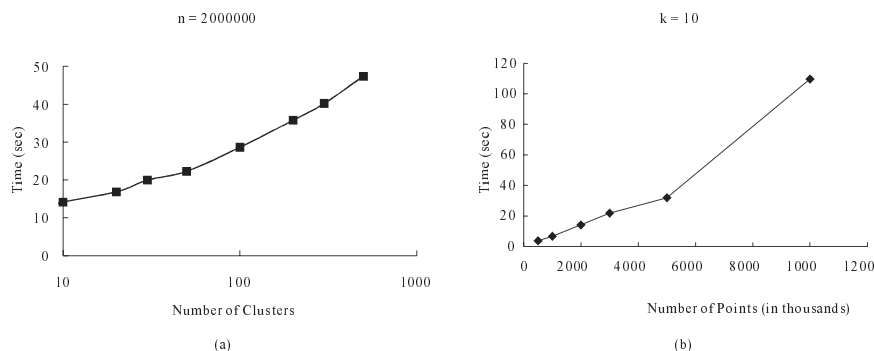
It can be seen from the figure that the quality of the clustering by algorithm  $SC_{GD}$  is very high and in fact very close to that achieved by algorithm  $SC_{OPT}$ . Note that the effectiveness of algorithm  $SC_{GD}$  is also affected by the total number of objects to be clustered and also the average size of clusters. More experiments are conducted to investigate the cost difference between  $SC_{OPT}$  and  $SC_{GD}$  and the results are shown in Figure 6(b), where the x-axis represents the average size of clusters, the y-axis is the cost difference between  $SC_{OPT}$  and  $SC_{GD}$  in percentage (i.e.,  $\frac{\text{total\_cost\_by\_}SC_{GD} - \text{total\_cost\_by\_}SC_{OPT}}{\text{total\_cost\_by\_}SC_{OPT}} \times 100\%$ ), and four curves for different values of  $n$  are examined. As shown in Figure 6(b), it is seen that as average size of clusters increase, the solution quality of algorithm  $SC_{GD}$  approaches to that of  $SC_{OPT}$ .

## 4.3 Experiment II: On the scalability of clustering algorithms

As mentioned before, due to its time complexity, algorithm  $SC_{OPT}$  can only be applied to a small dataset (up to thousands of objects). Execution efficiency of  $SC_{OPT}$  and  $SC_{GD}$  is shown in Figure 7 which in fact conforms with the complexity derived by Theorem 2, i.e.,  $O(kn^2)$  for algorithm  $SC_{OPT}$  and Theorem 4, i.e.,  $O(\frac{nl}{k} + n)$  for algorithm  $SC_{GD}$ . The linear time of algorithm  $SC_{GD}$  is again validated by the experiments conducted in Figure 8 fro a huge number of data



**Figure 7.** Execution time of algorithms  $SC_{OPT}$  and  $SC_{GD}$ .



**Figure 8.** Scalability of algorithm  $SC_{GD}$ .

points. From Figure 8, it is noted that algorithm  $SC_{GD}$  is able to deal with a large amount of data very efficiently, showing the very advantage of its linear time complexity.

## 5 Conclusion

We studied the problem of optimal sequential clustering in this paper. Specifically, we first proved that this optimal sequential clustering problem possesses the *optimal substructure property*. In light of this property, we devised algorithm  $SC_{OPT}$  to obtain the optimal solution of the problem. The time complexity of algorithm  $SC_{OPT}$  was proved to be  $O(kn^2)$ , where  $n$  is the size of the dataset and  $k$  is the number of clusters. To further reduce its complexity, we devise a greedy algorithm,  $SC_{GD}$ , which solves the problem in linear time. Extensive experimental studies were conducted to evaluate the performance and the effectiveness of these two algorithms. It has been shown that algorithm  $SC_{GD}$  is able to obtain the solution clustering of very high quality which is in fact very close to that obtained by algorithm  $SC_{OPT}$ .

# Bibliography

- [1] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, Philips S. Yu, and J.-S. Park. Fast Algorithms for Projected Clustering. *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1999.
- [2] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, NY, 1981.
- [3] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. MSR-TR-2000-65, Microsoft Research, May 2000.
- [4] A. G. Buchner and M. Mulvenna. Discovery internet marketing intelligence through online analytical web usage mining. *ACM SIGMOD Record*, 27(4):54–61, Dec. 1998.
- [5] M.-S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from Database Perspective. *IEEE Trans. on Knowledge and Data Engineering*, 5(1):866–883, December 1996.
- [6] P. Djuric. Segmentation of nonstationary signals. *In Proceedings of ICASSP 92*, V:161–164, 1992.
- [7] P. Djuric. A MAP solution to off-line segmentation of signals. *In Proceedings of ICASSP 94*, IV:505–508, 1994.
- [8] R. C. Dubes. How many clusters are best? - an experiment. *Pattern Recognition* 20, 6:645–663, Nov. 1987.
- [9] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurasamy. *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, MA, 1996.
- [10] S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. *Proceedings of the 15th International Conference on Data Engineering*, 1999.
- [11] F. Gustafsson. Segmentation of signals using piecewise constant linear regression models. *IEEE Transactions on Signal Processing*, 1996.
- [12] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

- [13] J. Hertz, A. Krogh, and R. G. Palmer. Introduction to the theory of neural computation. *Santa Fe Institute Studies in the Sciences of Complexity lecture notes*. Addison-Wesley Longman Publ. Co., Inc., Reading, MA., 1991.
- [14] A. K Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computer Surveys*, 31(3), Sept. 1999.
- [15] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, 1990.
- [16] D. A. Keim and A. Hinneburg. Clustering techniques for the large data sets - from the past to the future. *Tutorial Notes for ACM SIGKDD*, pages 141–181, Aug. 1999.
- [17] S. Y. Lu and K. S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Trans. Syst. Man Cybern.* 8, pages 381–389, 1978.
- [18] J. McQueen. Some methods for classification and analysis of multivariate observations. *Proc. Of the Fifth Berkeley Symposium on Mathematical Statics and Probability*, 1967.
- [19] R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. *Proceedings of the 20th VLDB Conference*, 1994.
- [20] J. O. Pedersen, D. R. Cutting, D. R. Karger, and J. W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. *Proceedings of the 15th Annual International ACM Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.
- [21] W.-C. Peng and M.-S. Chen. Mining user moving patterns for personal data allocation in a mobile computing system. *Proc. of the 29th International Conference on Parallel Processing (ICPP-2000)*, pages 573–580, Aug 2000.
- [22] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy*. Freeman, London, UK, 1973.
- [23] A. K. H. Tung, L. V. S. Lakshmanan, J. Han, and R. T. Ng. Constraint-based clustering in large databases. *Proc. 2001 Int. Conf. On Database Theory*, Jan 2001.
- [24] C.-H. Yun, K.-T. Chuang, and M.-S. Chen. An efficient clustering algorithm for market basket data based on small-large ratios. *Proceedings of the 25th International Computer Software and Applications Conference (COMPSAC 2001)*, Oct. 2001.
- [25] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Database. *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 103–114, 1996.