The VLDB Journal (2007) 16(2): 201–217
DOI 10.1007/s00778-005-0164-6

REGULAR PAPER

Bi-Ru Dai · Cheng-Ru Lin · Ming-Syan Chen

Constrained data clustering by depth control and progressive constraint relaxation

Received: 31 July 2004 / Accepted: 6 May 2005 / Published online: 11 January 2007

Abstract In order to import the domain knowledge or application-dependent parameters into the data mining systems, constraint-based mining has attracted a lot of research attention recently. In this paper, the attributes employed to model the constraints are called constraint attributes and those attributes involved in the objective function to be optimized are called optimization attributes. The constrained clustering considered in this paper is conducted in such a way that the objective function of optimization attributes is optimized subject to the condition that the imposed constraint is satisfied. Explicitly, we address the problem of constrained clustering with numerical constraints, in which the constraint attribute values of any two data items in the same cluster are required to be within the corresponding constraint range. This numerical constrained clustering problem, however, cannot be dealt with by any conventional clustering algorithms. Consequently, we devise several effective and efficient algorithms to solve such a clustering problem. It is noted that due to the intrinsic nature of the numerical constrained clustering, there is an order dependency on the process of attaining the clustering, which in many cases degrades the clustering results. In view of this, we devise a *progressive constraint relaxation* technique to remedy this drawback and improve the overall performance of clustering results. Explicitly, by using a smaller (tighter) constraint range in earlier iterations of merge, we will have more room to relax the constraint and seek for better solutions in subsequent iterations. It is empirically shown that the progressive constraint relaxation technique is able to improve not only the execution efficiency but also the clustering quality.

Keywords Data mining · Data clustering · Constrained clustering

B.-R. Dai (✉) · C.-R. Lin · M.-S. Chen
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC
E-mail: {brdai, owenlin}@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw

1 Introduction

Data clustering is a useful technique for many applications, including similarity search, pattern recognition, trend analysis, marketing analysis, grouping, classification of documents, and so forth [1–4]. In data clustering, similar data points are grouped together in a cluster. In general, there are two types of attributes associated with the data points, i.e., numerical attributes and categorical attributes. Numerical attributes are those with ordered values, such as the height of a person and the speed of a moving vehicle. Categorical attributes are those with unordered values, such as the blood type of a person and the brand of a car.

Since data mining is an application-dependent technology, the information involving domain knowledge is usually imposed on the mining systems as various constraints. For example, the temperature or time attribute may possess extremely important meaning in some experiments and should be regarded as a constraint while analyzing the data. In the conventional clustering problems, all the attributes of the data set are regarded as having the same role. In contrast, a specific constrained clustering model is introduced in this paper to cope with these user-specified constraints. Similar to most prior works [5–10], only the numerical attributes are considered in this paper. Specifically, those attributes employed to model the constraints are called constraint attributes whereas those attributes involved in the objective function to be optimized, similar to those in most prior works, are called optimization attributes. Note that an attribute could be a constraint attribute and an optimization attribute at the same time. The constrained clustering considered in this paper is conducted in such a way that the objective function of optimization attributes is optimized subject to the condition that the imposed constraint is satisfied. Explicitly, we address in this paper the problem of constrained clustering with numerical constraints, in which the constraint attribute values of any two data items in the same cluster are required to be within the corresponding constraint range. Note that the results in [7] are focused on solving conventional clustering problems

X	Y	Age
6	20	25
10	13	23
14	8	26
14	24	27
17	12	36
19	18	35
20	25	22
23	8	33
29	18	24
32	23	37
35	11	22
37	18	25
41	10	36
42	26	34
43	15	35
47	7	33
49	17	38

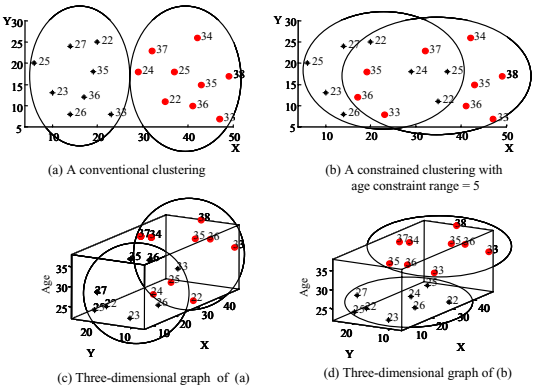


Fig. 1 The difference between a conventional clustering and a numerical-constrained clustering

without any constraint, which is different from the goal of our work. This numerical-constrained clustering problem can be best understood by the following example. Consider the data pointsin Fig. 1 where X and Y are the conventional optimization attributes which form the coordinates of the residential location and age is the constraint attribute with the constraint range being 5 years. The nodes in Fig. 1a are identical to those in Fig. 1b and the number next to each node represents the age of that person. By conventional clustering methods which are designed to cluster nearby nodes together, these nodes may be partitioned into the two clusters as shown in Fig. 1a, in which, however, the age constraint is not obeyed. Instead, one possible solution to this constrained clustering problem is shown in Fig. 1b, where members within 22–27 years old are in one cluster, and people within 33–38 years old are in the other cluster. The main objective of this study is not to improve the clustering result in general, but aims to developing techniques to handle constraints. Note that such a clustering with numerical constraints is called for in many real applications. For example, we may apply this numerical constrained clustering on the basic data of people in a club to group people provided that we require the range of ages in each group to be no more than 5 years. Also, in order to observe the trends of any data sets with a timestamp, such as Web logs, video images, or CDR (i.e., Call Detail Record) of a moving person, a time constraint range is expected to be specified to monitor the behavior of each data group, e.g., 5 min for video images or 24 h for a Web logs or a CDRs. Such problems do require the clustering with numerical constraints. In addition, in a chemical experiment, some reactions are only informative or useful within a specific range of time or temperature. Therefore, we can set the time and/or temperature as constraint attributes. Note that, if the differences in time and/or temperature attributes affect the clustering result (for example, smaller differences imply better results), these attributes can also be included as optimization attributes. On the other hand, if the differences in time and/or temperature attributes are not irrelevant to the clustering result, these attributes are part of optimization attributes. For example, suppose that we only need the reaction to occur in 10 min, but do not make a distinction whether it actually occurs in 3 min or 5 min. In this occasion, the time attribute is a constraint attribute instead of an optimization attribute. The constraint attributes and optimization attributes should be selected according to the domain knowledge of the application considered. Another example application of this constrained clustering model is to identify objects from video images. For example, if there is a person in a video image at time t, it is likely that an object with similar shape and similar location may be viewed as the same person in a sequence of video images within 10 s, but as a different person after a longer time interval. Therefore, we can cluster video images with time constraint range being 10 s to identify clusters from the video. After clusters are identified, other algorithms can be applied on the clustering results to obtain more information implied in the video.

This numerical-constrained clustering problem, however, cannot be dealt with by any conventional clustering algorithms. With respect to the numerical constrained problem, the constraint ranges put bounds to the possible destination clusters of each point. Notice that we do not indicate a fixed boundary of a cluster explicitly. Instead, the boundaries of constraint attributes of each cluster are dynamically determined by the data points existing in the cluster. In the example in Fig. 1, once a person joins a cluster, the age range of this cluster is adjusted immediately. For example, assume that the first person joining a group is 23 years old, then people between 18 and 28 years old are allowed to join this group afterward. However, if the second person joining this group is 27 years old, the age range of this group is narrowed down to 22–28 years old, meaning that the age boundary of this cluster is revised. Finally, groups with similar people are generated based on the 5-year range. Fig. 1c and 1d show the age in the third dimension, and it can be seen that a conventional clustering method cannot resolve the constraint directly. Consequently, new methods are called for to handle this constrained clustering problem.

In view of this, we devise several effective and efficient algorithms to solve the problem in this paper. First, we design a partition-based algorithm called *clustering with depth control* (abbreviated as CDC), which is able to obtain the local optimal solution of the constrained clustering problem. However, it incurs relative long time in the searching procedure. Therefore, the hierarchical clustering algorithms, which merge most similar clusters one by one, are taken into consideration to solve this problem without searching the optimal solution iteratively. The algorithm of *constrained clustering with complete-link* (abbreviated as CCL), which is revised from a well-known complete-link hierarchical clustering algorithm, is performed to show the efficiency resulting from the merging procedure. However, complete-link algorithms tend to generate spherical clusters, and such a problem also occurs on the algorithm CCL. Therefore, another hierarchical clustering algorithm, namely the *constrained clustering with spring-model* (abbreviated as CSP), is introduced. Algorithm CSP is based on the gravity theory, which can generate clusters in various shapes. In addition,

while algorithm CCL only merges two clusters in each iteration, algorithm CSP is more efficient since it can merge several clusters in one iteration. It is noted that due to the intrinsic nature of the numerical-constrained clustering, there is an order dependency on the process of attaining the clustering, which in many cases degrades the clustering results. In view of this, we devise a *progressive constraint relaxation* technique to remedy this drawback and improve the overall performance of clustering results. The basic idea of the progressive constraint relaxation is that by using a smaller (tighter) constraint range in earlier iterations of merge, we will have more room to relax the constraint and seek for better solutions in subsequent iterations. It is empirically shown that the progressive constraint relaxation technique is able to improve not only the execution efficiency but also the clustering quality. As shown in the complexity analyses and also validated by our empirical studies, the partition-based algorithm CDC is able to achieve a low clustering cost, while incurring more time for data points to traverse between clusters so as to achieve a stable result with the minimum cost. In contrast, the hierarchical algorithms which are further enhanced by progressive constraint relaxation technique, i.e., both progressive CCL and progressive CSP, are executed much more efficiently than algorithm CDC in the moderate size of data sets with a slight increase in clustering costs. However, the partition-based constrained clustering algorithm CDC outperforms the hierarchical ones when dealing with a very large data set. We also conduct a series of experiments on these algorithms to exhibit their properties.

The rest of this paper is organized as follows. The problem description and related works are given in Sect. 2. Sect. 3 presents the proposed algorithms to deal with this constrained clustering problem. The issue of order dependency is identified and solved by the progressive relaxation technique in Sect. 4. The extensions to multiple constraint attributes are discussed in Sect. 5. We empirically evaluate the performance of several algorithms in Sect. 6. This paper concludes with Sect. 7.

2 Problem description

The problem of pair-wise constrained clustering studied in this paper is defined in Sect. 2.1. Then, related works of clustering with constraints are summarized in Sect. 2.2.

2.1 Pair-wise constrained clustering

As mentioned earlier, a specific constrained clustering model is introduced in this paper to cope with the user-specified constraints. Among all attributes of the data set, some are specified as *constraint attributes* and some are *optimization attributes*. Constraint attributes and optimization attributes may overlap because some attributes are possibly important in both considerations. Similar to the conventional clustering problems, there is an objective function operating on the optimization attributes to measure the cost of clustering. In addition, an additional *constraint range* R_{a_c} is set for each constraint attribute a_c. For any pair of objects o_i, o_j in a cluster, the constraint distance $d_{a_c}(o_i, o_j)$ of constraint attribute a_c between any two objects o_i and o_j is required to be less than or equal to R_{a_c}. For example, in Fig. 1, age is a constraint attribute, denoted by a_{age}, and its constraint range $R_{a_{\text{age}}}$ is 5. The constraint distance $d_{a_{\text{age}}}(.,.)$, which is the difference of ages, should be no more than 5 years. Formally, we have the following problem definition.

Definition of clustering with numerical constraints:

Given a data set D of n objects $\{o_1, o_2, \ldots, o_n\}$, and a predetermined number of clusters k, the numerical constrained clustering problem is defined as the problem of determining the k-clustering $Cl = \{C_1, C_2, \ldots, C_k\}$ in such a way that the total cost $\text{Cost}(Cl)$ is minimized subject to the condition that for any pair of objects (o_i, o_j) in a cluster, we have $d_{a_c}(o_i, o_j) \leq R_{a_c}$, where a_c is any constraint attribute, R_{a_c} is the constraint range of a_c, and the value of $R_{a_c} \times k$ is larger than or equal to the whole range of the constraint attribute a_c. The cost function $\text{Cost}(Cl)$ is calculated based on the optimization attributes.

A typical cost function is

$$\text{Cost}(Cl) = sq(Cl) = \frac{\sum_{C_i} \sum_{p \in C_i} (p - C_i.\text{center})^2}{\sum_{C_i} \|C_i\|},$$

where C_i is a cluster of the clustering result Cl and C_i.center is the center of cluster C_i. Without loss of generality, this cost function is used to evaluate the clustering results in our performance studies later, and the number of constraint attributes is assumed to be one for ease of exposition. However, the algorithms can be easily extended to multiple constraints. We have the following theorem for the complexity of the numerical-constrained clustering problem.

Theorem 1 *The problem of finding the optimal numerical constrained clustering is NP-hard.*

Proof We can transform this problem into a graph partition problem. Each data point can be regarded as a vertex of the graph. For each pair of vertexes u and v, there is an edge connecting them if no attribute of them exceeds the range constraints. In other words, there is an edge between vertexes u and v if these two data points can be allocated into the same cluster without violating any constraint. The weights of edges can be calculated by some cost functions of the clustering problem. In this manner, this problem is equivalent to an NP-hard problem named minimum edge deletion k-partition problem [11] below, and this theorem follows.
Minimum edge deletion k-partition
INSTANCE: Graph $G = (V, E)$ and a weight function w on the edges.
SOLUTION: A k-partition, i.e., a color assignment c: $V \rightarrow [1 \ldots k]$. In other words, each vertex v_i is assigned a color $c(v_i)$, where $c(v_i)$ is an integer between 1 and k.

MEASURE: The weight of the monochromatic edges, i.e.,

$$\sum_{(v_1,v_2)\in E:c(v_1)=c(v_2)} w(v_1, v_2),$$

where a monochromatic edge is an edge with both vertices assigned the same color. □

Theorem 1 justifies our efforts in the following sections to explore efficient heuristics to solve this numerical-constrained clustering problem.

2.2 Related works

Since the early work in *k*-means algorithm, the data clustering has been studied for many years and several technologies on data clustering have been developed, including the nearest neighbor clustering [12], fuzzy clustering [13], partitional clustering [14], hierarchical clustering [15], artificial neural networks for clustering [16], and so on. In addition, several works have been conducted for studying the constrained data clustering problems. The work in [9] defines a taxonomy of constraints for clustering with the focus on exploring the constraints which can be formulated with SQL aggregates and imposed on individual clusters. The work [6] considers the constraint of each cluster having at least a minimum number of points in it. In the work [17], a set of representatives is given to represent data points, and groups of representatives will converge to the same points, which are defined as clusters. The total mass constraint specifies the constraint of a fixed total number of representatives. The clustering techniques for spatial data in the presence of physical constraints are discussed in [18–20]. The constraints between instances in the data set are discussed in [21], where the predefined "must-link" and "cannot-link" constraints established from background knowledge are considered. These prior works do not set constraints on the ranges of attributes.

Some existing works also have range constraints on attributes, but the objectives are different from our work. The work in [22] focuses on the continuous constraint: all the data points in each cluster form a continuous region. The work in [23] on fascicles also considers clustering with range constraints. Some attributes are given the range constraints for numeric attributes (or the number of distinct values for categorical attributes), where the widths of the ranges of all records in a cluster do not exceed these given constraint ranges. The main difference between our work and the one in [23] is that the distances of those attributes without constraints are considered in the cost function and are minimized in our study, whereas the latter does not take those attributes into consideration and thus records in the same cluster could have totally dissimilar values of those attributes.

The works for segmenting a video into story units try to collapse visually similar and temporally local shots into a compact structure by introducing the time-constraint into clustering problems [24]. This one can be regarded as an application of our range constraint on the video segmentation problem that utilized a simple complete-link algorithm.

3 Algorithms for pair-wise constrained clustering

Several algorithms are devised in this section to solve the constrained clustering problem. To cope with these constraints, a partitional clustering algorithm is developed in Sect. 3.1. In Sect. 3.2, two different hierarchical clustering methods are proposed to explore the relationship between each pair of data points.

3.1 Partition-based clustering

The *k*-means algorithm is a well-known partitional algorithm for data clustering. Several algorithms, such as CLARA and CLARANS [8], are variations of the *k*-means algorithm. However, these clustering algorithms assign a point into a cluster only according to the distance between the point and the cluster center, thus not considering the constraints among points in the same cluster. Consequently, to deal with the numerical constraints, we devise a partition-based clustering algorithm with depth control (abbreviated as CDC), which in essence tries to minimize the cost of clustering step by step. For simplicity, we first outline algorithm CDC in Fig. 2. Note that except the centers of clusters, all of the points discussed in this paper are from the input data set. As shown in Fig. 2, algorithm CDC repetitively rearranges the clusters and tries to allocate as many data points into clusters as possible by Steps 3 and 4. Therefore, these two steps will iteratively improve the clusters until no more data points can be moved to reduce the total clustering cost subject to the imposed constraint.

However, in addition to minimizing the cost of partitioning as in the original clustering problems, the constraints have to be observed in our study. Note that if we select cluster seeds according to the constraints or the distribution of the optimization attributes of the data points, the whole data set should be parsed once to collect the information of the constraints before the clustering procedure. Moreover, some preprocessing techniques, which may be more complicated than sorting the data set, could be applied to analyze the information collected. However, the constraint attributes and

Algorithm CDC: Partition-Based Clustering with Depth Control

//Input: an input data set, the number of clusters, k, and the constraint range R_{ac}

1. Assign all input data points into the pool.
2. Randomly select *k* points from the pool and insert them into the *k* clusters respectively.
3. For each point in the pool, try to insert it into a cluster that minimizes the cost. If it cannot be inserted into any cluster, retain it in the pool.
4. For each point not in the pool, first remove it from the original cluster and then try to insert it into a cluster that minimizes the clustering cost. (This cluster must exist because at least it can be reinserted into the original cluster.)
5. Repeat Steps 3 and 4 until no point moves between clusters in the two steps.

Fig. 2 The outlines of algorithm CDC

Point Insertion Procedure of Algorithm CDC
// Input: the inserting point, p, the inserted cluster C[i], the maximum traverse depth, d, and the limit of the size of conflict set, c.
//Output: a boolean value indicating the success of this insertion
1. Insert p into $C[i]$.
2. Find the conflict set according to the new inserted point p.
3. If the size of the conflict set exceeds the predefined value c then undo all the actions (of Steps 1 and 2) and return false.
4. For each point of the conflict set:
   4.1 Remove it from $C[i]$.
   4.2 If the traverse depth, d, is larger than 0 then try to insert it into another cluster that minimize the cost with the traverse depth as d-1. If no such cluster exists, undo all the actions (of Steps 1 to 4) and return false.
5. Return true to indicate the success of this insertion.

Fig. 3 The outlines of point insertion procedure of algorithm CDC

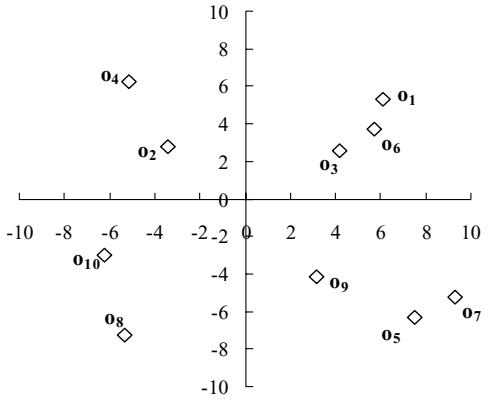


Fig. 4 An example set of data points for constrained clustering

the optimization attributes may possess any kind of distribution and they do not always have similar locality. Therefore, it is not always true that selecting cluster seeds according to preprocessing will improve the subsequent clustering. Consequently, the cluster seeds are generated randomly to avoid extra time complexity. Note that the distribution of all attributes can be analyzed by techniques like sampling. Relevant work on sampling can be found in [25]. In addition, we may observe the phenomenon that an incoming point may violate the constraints with some points in the cluster. In this situation, the set of those points is named as the *conflict set*. Algorithm CDC will try to solve these violations by moving those points in the *conflict set* to other clusters. Note that the moving process may trigger a sequence of movements. To avoid an endless chained reaction, a maximum traverse depth is specified to restrict the level of movement. As the algorithm terminates, the points which still remain in the pool and cannot be inserted into any cluster will be regarded as outliers. The detail of the insertion procedure is listed in Fig. 3.

This rearrangement procedure picks up each object in the conflict set and tries to insert it into the cluster that minimizes the cost of clustering. The insertion aborts if it is impossible to insert this point into any cluster within $d - 1$ levels. More specifically, when the insertion of a new point to a cluster causes the expulsion of existing points, the traverse depth decreases by one. As long as the traverse depth is larger than zero, the point is allowed to be inserted into a cluster with possible expulsion of other points. However, in the situation of traverse depth being zero, this insertion is permitted only if no conflict occurs. Note that, in Step 5 of the insertion procedure, the points in the conflict set cannot be reinserted into the original cluster $C[i]$, because the point, say q, will cause the original point p to form a new conflict set, which may in turn cause the point p to be reinserted into this cluster and q to form a new conflict set, leading to an endless loop. In this algorithm, points are moved between clusters to minimize the total clustering cost. A single movement only takes place when this movement can reduce clustering cost. Therefore, if no further single movement can reduce clustering cost, algorithm CDC completes. Since there are only finite combinations of clustering results for a given data set for CDC to find a clustering result with fewer cost, algorithm CDC is guaranteed to terminate. We show an example to illustrate the detail of algorithm CDC below.

Example 1 Consider the example shown in Fig. 4. There are 10 data points in the data set and the timestamp is labeled next to each node. Table 1 exhibits the timestamp and coordinates of each point. In this example, we shall partition these points into 3 clusters with the time constraint range being 4. For example, points $\{o_1, o_3, o_4, o_6\}$ do not form a valid cluster since $6 - 1 > 4$. The limitation of the traverse depth is set as two and the size of conflict set is set as one. At the beginning of algorithm CDC, points o_1, o_3, and o_6 are randomly chosen as cluster centers and all the other points are put in a pool. The partition is refined by moving points to clusters with the lowest cost.

Moving o_1: o_1 remains in the first cluster since one cannot reduce the cost by moving o_1 to another cluster.

Moving o_2: The clustering cost will become 48.59, 28.98, and 42.26 if o_2 is inserted into the first (with o_1), second (with o_3), and third (with o_6) cluster, respectively. Therefore, o_2 is inserted into the second cluster since it results in the minimal clustering cost.

Moving o_3: Since o_3 is already in the second cluster, it will be removed from the original cluster and reinserted into a cluster with the minimum cost. The clustering cost will become 5.52, 28.98, and 1.83 if o_3 is inserted into the first (with o_1), second (with o_2), and third (with o_6) cluster, respectively. Therefore, o_3 is inserted into the third cluster.

Table 1 The coordinates of the points shown in Fig. 4

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
Timestamp	1	2	3	4	5	6	7	8	9	10
X-coordinate	6.09	−3.44	4.16	−5.15	7.50	5.70	9.27	−5.33	3.17	−6.25
Y-coordinate	5.28	2.76	2.57	6.21	−6.34	3.72	−5.21	−7.26	−4.17	−3.03

Similarly, o_4–o_7 are inserted into clusters with the corresponding minimum costs. We use the insertion of o_8 to illustrate the effects of traverse depth and conflict set in algorithm CDC. As shown in Fig. 5a, before o_8 is inserted, the clusters are $\{o_1\}$, $\{o_2, o_4\}$, and $\{o_3, o_5, o_6, o_7\}$, and the traverse depth $d = 2$ initially. If o_8 is inserted into the first cluster, it will conflict with o_1 (as shown in b) and decrease d from 2 to 1. Then we have to move o_1 to the second cluster (as shown in e) or the third cluster (as shown in f) by decreasing d to 0. Note that the conflict set of f contains two points (o_6 and o_7) and exceeds the size limitation of one, and this movement is hence aborted (undo the insertion of o_1 to the second cluster). If o_8 is inserted into the second cluster, it will conflict with o_2 (as shown in c) and decrease d from 2 to 1. Then we have to move o_2 to the first cluster (as shown in g) or the third cluster (as shown in h) by decreasing d to 0. Although the conflict set of h does not exceed the size limitation of one, it is not successful because we cannot move o_7 to other clusters when d is 0. Therefore, this movement is also aborted (undo the insertion of o_2 to the third cluster). Similarly, o_8 can be inserted into the third cluster (as shown in d) by moving o_3 to the first (as shown in i) or the second (as shown in j) cluster. We can find that e, g, i, and j are possible solutions of inserting o_8 and e has the minimum cost. Finally, o_8 is inserted into the first cluster (as shown in b) and o_1 is moved to the second cluster (as shown in e).

In the subsequent steps of the first iteration, all the points in the pool are inserted in the same manner. The clustering result is $\{o_1, o_2, o_3, o_4\}$, $\{o_5, o_6, o_7, o_9\}$, and $\{o_8, o_{10}\}$ after the first iteration finished. The second iteration attempts to change the position of each point to achieve lower clustering cost. At the end of the second iteration, the clustering cost is still 194.31 with the clustering result of $\{o_1, o_2, o_3, o_4\}$, $\{o_5, o_6, o_7, o_9\}$, and $\{o_8, o_{10}\}$. Consequently, algorithm CDC ends up with this partition.

The following two theorems state the average time and space complexities of algorithm CDC.

Theorem 2 *The time complexity of a single iteration of algorithm CDC is $O(n(kc)^d \log(n))$, where d and c are the predefined maximum limits of the traverse depth and the size of conflict set.*

Proof In each iteration of algorithm CDC, each point determines its destination cluster by evaluating the clustering costs of all possible movements, including the removal from the original cluster and the insertion into the new cluster. However, at each insertion, the algorithm may push out at most c conflicting points at each depth, and each conflicting point has to try k possible destinations. Since we use a tree structure to store the points of a cluster for better execution efficiency, each insertion takes time complexity of $O(\log \|C\|)$, where $\|C\|$ is the size of the cluster. The size of a cluster can be approximated as $O(n)$. Thus, the time complexity is $O(n(kc)^d \log(n))$. □

Theorem 3 *The space complexity of algorithm CDC is $O(n \log n + c^d)$, where d and c are the predefined maximum limits of the traverse depth and the size of conflict set.*

Proof In this algorithm, we use a tree structure to store the points of a cluster so that we can easily remove or insert a point into a cluster while keeping the information of constraint attribute, such as the minimum value and maximum value of the constraint attribute of the cluster, at the same time. Thus, the space complexity required is $O(\sum \|C\| \log \|C\|) \cong O(n \log n)$. At Step 4, if the cost of the partition cannot be improved, we need to restore all the actions involved in inserting this node. Therefore, we need at most $O(c^d)$ space to keep the undo information. As a result, the space complexity of the algorithm is $O(n \log n + c^d)$. □

3.2 Hierarchical clustering algorithms

We have introduced algorithm CDC to search for a local optimal solution in the previous section. However, it incurs relatively long time in the searching procedure. Therefore, the hierarchical clustering algorithms, which merge most similar clusters one by one, are taken into consideration to solve this problem without searching the optimal solution iteratively. In Sect. 3.2.1, we present algorithm CCL which is devised by utilizing a well-known hierarchical clustering algorithm, complete-link [26], to show the efficiency resulting from the merging procedure. As mentioned before, complete-link algorithms tend to generate spherical clusters, and such a problem also occurs on the algorithm CCL. Therefore, another hierarchical clustering algorithm CSP is introduced In Sect. 3.2.2. Algorithm CSP is based on the spring model, which can generate clusters in other shapes. In addition, while algorithm CCL only merges two clusters in each iteration, algorithm CSP is more efficient since it can merge several clusters in one iteration.

3.2.1 CCL: clustering with complete link

The complete-link algorithm uses the distance of two farthest points as the intercluster distance, i.e.,

$$\text{dist}(C_i, C_j) = \max\{d(o_i, o_j) | o_i \in C_i, o_j \in C_j\}.$$

Here we take the time constraint attribute a_{time} as example, i.e., the constraint attribute a_c represents a_{time}. For a cluster C, we define the start constraint value, denoted by $C.ts$, and end constraint value, denoted by $C.te$, as the smallest and largest constraint attribute values of data points in the cluster, respectively, i.e., $C.ts = \min\{o.\text{time} | o \in C\}$ and $C.te = \max\{o.\text{time} | o \in C\}$. Then, the constraint distance between two clusters C_i and C_j is determined as

$$d_{a_c}(C_i, C_j) = \max\{C_i.te - C_j.ts, C_j.te - C_i.ts\}$$

The distance measurement of two clusters is modified for this constrained clustering problem. Instead of the original

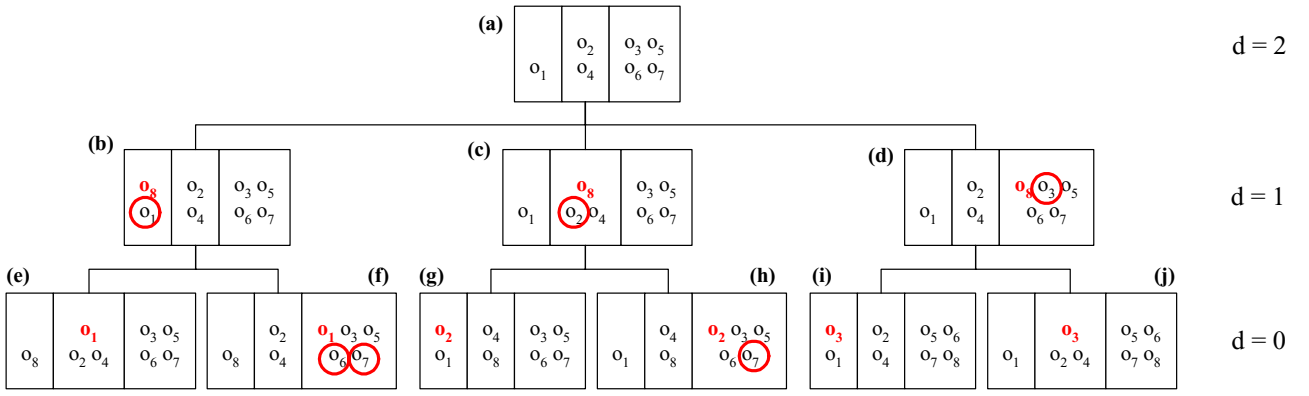


Fig. 5 The insertion of o_8 in Example 1. The *red object* is the inserted one and the *circles* represent the conflict set caused by the inserted object

Algorithm CCL: Constrained Clustering with Complete Link
//Input: an input data set, the number of clusters, k, and the constraint range R_{ac}
1. Initially, each data point forms a cluster by itself.
2. The algorithm repetitively merges the two closest clusters.
3. Repeat Step 2 until exactly k clusters left or all pairs of points exceed the constraint range.

Fig. 6 The outlines of algorithm CCL

distance dist(., .), we define a new distance measurement of two clusters as below:

$$\widehat{\text{dist}}(C_i, C_j) = \begin{cases} \text{dist}(C_i, C_j), & \text{if } \ d_{a_c}(C_i, C_j) \le R_{a_c}, \\ \infty, & \text{otherwise.} \end{cases}$$

Hence, with these provisions, the complete-link clustering algorithm is revised to deal with the numerical constraints, as shown in Fig. 6.

As the algorithm terminates, if the resulting cluster number $k\prime$ is larger than k, those points in the smallest $(k\prime - k)$ clusters that contain the minimal number of points will be regarded as outliers so that exactly k large clusters are obtained. Note that algorithm CCL can also be modified to a constrained clustering method based on single-link [15] by changing the intercluster distance measurement. However, this method based on single-link is found to incur a much larger clustering cost than other algorithms we devised and will thus not be explored in the following discussion.

Example 2 We also apply algorithm CCL to cluster the points in Fig. 4. Recall that the cluster number is 3 and the time constraint range is 4. At the beginning of the algorithm, a distance matrix of the 10 points is built and some distances are set to be infinity because the time intervals between these pairs of points exceed the time constraint range 4. Points o_3 and o_6 are merged first since they are two closest points. Then the distances between these two points and other points are updated. Similarly, CCL will merge o_5 and o_7, o_2 and o_4, o_8 and o_{10}, o_7 and o_9 in turn, and finally o_6 and o_4. (Note that we use a point to represent the cluster to which the point belongs.) Algorithm CCL merges these 10 points into four clusters, which are $\{o_1\}$, $\{o_2, o_3, o_4, o_6\}$, $\{o_5, o_7, o_9\}$, and $\{o_8, o_{10}\}$. It can be seen that any pair of these clusters exceeds the time constraint range. As a result, no further merging is possible and algorithm CCL completes.

Note that for a data point p, not all other data points can be put into the same cluster with it because some of them may violate the constraint range. The fraction of data points that can be put into the same cluster with p is represented by the ratio r_c. For example, assume that there are totally 100 data points in a data set, and 60 of them exceed the constraint distances to p. In this situation, only 40 data points are possible to be in the same cluster with p. Therefore, we only have to calculate these 40 distances to p rather than 99 distances to p. In other words, only 0.4 of the whole data points should be considered when merging with p. In this example, the constraint ratio r_c is 0.4. This ratio is between 0 and 1, and it is usually smaller when the constraint range is tighter. In our complexity analyses, we use this ratio to approximate the number of data points that can be merged with a data point. That is, about $O(nr_c)$ can be merged with a data point. The following two theorems state the average time and the space complexities of algorithm CCL.

Theorem 4 *The time complexity of algorithm CCL is* $O(n^2 r_c \log(n^2 r_c))$.

Proof The time complexity of original complete-link algorithm is $O(n^2 \log n)$, which is bounded by sorting the distances of all pairs of points. In this constrained clustering problem, we can use $O(nr_c)$ to represent possible neighbors of each point, where r_c is between 0 and 1. Therefore, only $O(n^2 r_c)$ distances are required to be taken into consideration. Similar to the original complete-link algorithm, it requires $O(n^2 r_c \log(n^2 r_c))$ for sorting those $O(n^2 r_c)$ distances. Consequently, the time complexity of algorithm CCL is $O(n^2 r_c \log(n^2 r_c))$. □

Theorem 5 *The space complexity of algorithm CCL is* $O(n^2 r_c)$.

Proof In algorithm CCL, only $O(n^2 r_c)$ distances should be stored. Therefore, the space complexity is $O(n^2 r_c)$.

3.2.2 CSP: clustering with spring model

We next devise a spring model for the clustering with numerical constraints, where attractive forces of objects are employed to merge similar clusters. Assume the data items are distributed in the space. There are springs connecting each pair of objects if and only if they are in the range of the specified constraint. Note that when the spring is stretched, it pulls the two objects on the separate ends to approach each other and the force is large if the distance between them is long. In our problem, to model the case where the nearest ones should merge together first and the attractive force should increase as the distance between them gets shorter, we adapt the shrinking strength of a spring by the gravity force according to the gravity model presented in [27].

Specifically, the magnitude of gravity force $\vec{F}_{\mathrm{g}}$ applied to two nodes apart by a distance r is modelled as:

$$\|\vec{F}_{\mathrm{g}}\| = \frac{c_{\mathrm{g}} \times m_1 \times m_2}{r^p},$$

where c_{g} is a constant coefficient, m_1 and m_2 are the masses of these two objects, and p is a positive integer. By theorems in physics, the air resistance force that a moving node experiences is equal to $\vec{F}_{\mathrm{r}} = c_{\mathrm{r}} \times \| \rightarrow v\| \rightarrow v$, where c_{r} is the coefficient of air resistance and $\rightarrow v$ is the velocity of the objects. By these two kinds of forces, the terminal velocity v_j of an object o_j can be derived as:

$$v_j = \sqrt{\frac{\left\| \sum_{\text{object } o_i} \vec{F}_{\mathrm{g}i} \right\|}{c_r}},$$

where $\sum_{\text{object } o_i} \vec{F}_{\mathrm{g}i}$ is the sum of the gravity forces that object o_j experiences.

When objects o_i and o_j collide with each other, i.e., their distance is shorter than a predetermined *collision distance*, they are merged into a cluster, and the spring S_{ij} connecting o_i and o_j is then removed. The position of this new cluster is the average of objects o_i and o_j, and the mass is the summation of the merged clusters. For another object o_{m}, if the springs $S_{i\mathrm{m}}$ and $S_{j\mathrm{m}}$ between the object o_{m} and the merged cluster containing o_i and o_j both exist, only one of them is retained. Otherwise, the link is removed. Given the desired cluster number k, this algorithm will terminate *either* when the remaining object number is equal to k, *or* when all springs are removed. In the latter situation, the disconnected objects resulted are considered as the natural clusters. For example, with the constraint range $R = 3$, Fig. 7 shows the merging procedure of this spring model. Note that the collision distance can be regarded as the distance within which two objects are regarded so similar that they should be merged into the same cluster directly. Note that the collision distance is application-dependent and is considered as a user-defined parameter decided according to the domain knowledge and the distribution of data sets.

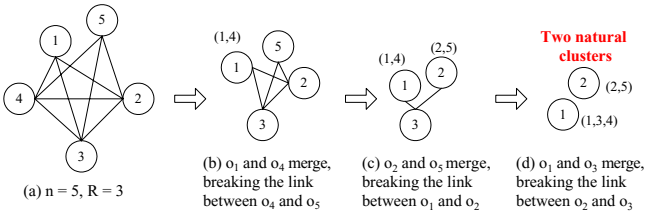


Fig. 7 An illustrative clustering procedure of attractive spring model

Algorithm CSP: Clustering with a Spring Model

//Input: an input data set, the number of clusters, k, and the constraint range R_{ac}

1. Establish links between each pair of points within the constraint range.
2. For each pair of points within the constraint range, calculate their distance and merge them if they are within a collision distance.
3. For each point remained, calculate the attractive forces and the new velocity.
4. Calculate the new position of each point.
5. Repeat Step 2 to Step 4, until exactly k points remained or all pairs of points exceed the constraint range.

Fig. 8 The outlines of algorithm CSP

The outlines of algorithm CSP are listed in Fig. 8. Same as in CCL, some points may be regarded as outliers to achieve exactly k valid clusters.

Example 3 Consider Fig. 4 again. In this example, we adopt the parameter p as 7, c_{g} as 9.8, c_{r} as 10, 000, the collision distance as 0.01, and the initial mass of each point as $\frac{1}{c_{\mathrm{g}}}$. At the beginning of the algorithm, we can obtain the force between o_1 and o_2 as $(-1.05 \times 10^{-6}, -2.77 \times 10^{-7})$ in the (X, Y) directions, respectively by the formula $\|\vec{F}_{\mathrm{g}}\| = \frac{c_{\mathrm{g}} \times m_1 \times m_2}{r^p}$. The total force imposed on o_1 is $(-1.27 \times 10^{-3}, -1.79 \times 10^{-3})$, which is the sum of the subforces from those points within the time constraint range of o_1, i.e., o_2, o_3, o_4, and o_5. Then the resulting velocity of o_1 is $(-2.72 \times 10^{-4}, -3.82 \times 10^{-4})$, which is derived by the formula, $v_j = \sqrt{\frac{\| \sum_{\text{object } o_i} \vec{F}_{\mathrm{g}i} \|}{c_r}}$. After obtaining the velocities of all the data points, the maximal velocity generated is 3.25×10^{-3} and the two closest points are o_3 and o_6, whose distance is 1.92. Thus, the simulation granularity, $\triangle t$, is set as $\frac{\min\_D}{2 \max\_V} = \frac{1.92}{2 \cdot 3.25 \times 10^{-3}} = 295$. After this iteration, the position of o_1 becomes (6.00758, 5.16508), which is obtained by the formula $\triangle x = \triangle t \times v$. The new positions of all the other points are calculated similarly. According to their new positions, those points within the collision distance are merged together. In this example, o_3 and o_6, whose distance is $0.0057 < \text{collision\_}d = 0.01$, are merged, as shown in Fig. 9a. Then o_5 and o_7 are merged in the subsequent iteration. Same as algorithm CCL, algorithm CSP can only merge the 10 points into four clusters, which are $\{o_1\}$, $\{o_2, o_3, o_4, o_6\}$, $\{o_5, o_7, o_9\}$, and $\{o_8, o_{10}\}$, as shown in Fig. 9b. This is the same result as the one by algorithm CCL in this case.

The following two theorems state the average time and space complexities of algorithm CSP.

Theorem 6 *The time complexity of algorithm CSP is* $O(n^2 r_{\mathrm{c}})$.

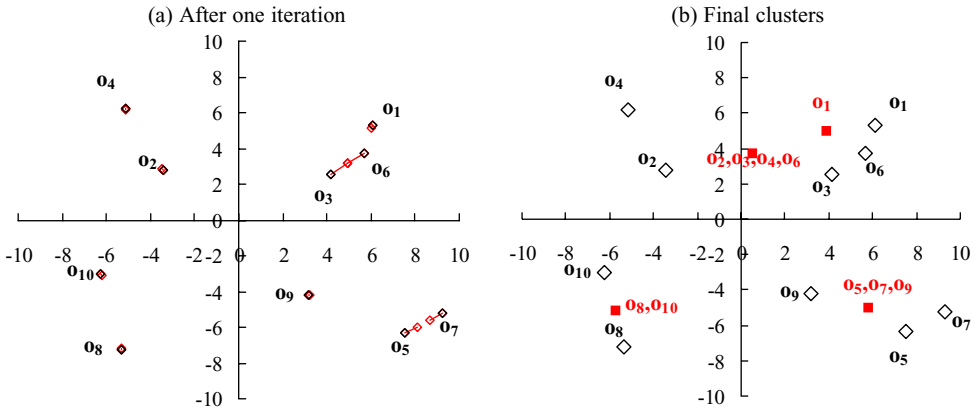


Fig. 9 The illustration of algorithm CSP. **a** The black objects are the original data points of Example 3, and the red ones are the new positions of these objects after one iteration of algorithm CSP. **b** The red objects are the final positions of the resulting clusters after algorithm CSP stops

Proof Similar to that of algorithm CCL, the time complexity of algorithm CSP is reduced from $O(n^2)$ to $O(n^2 r_c)$ by calculating merely $O(n^2 r_c)$ distances rather than $O(n^2)$ distances of the original gravity-based algorithm [27]. This theorem follows. □

Theorem 7 *The space complexity of algorithm CSP is* $O(n)$.

Proof The space required in this algorithm is smaller than algorithm CDC and CCL because the distances are calculated in each iteration and the information stored is merely the positions of remaining points. The space complexity of algorithm CSP is therefore $O(n)$. □

4 Progressive constraint relaxation

The problem of order dependency when dealing with numerical-constrained clustering is addressed in Sect. 4.1. In Sect. 4.2, a progressive constraint relaxation technique is proposed to address this issue and improve the performances of all the hierarchical clustering algorithms.

4.1 Order dependency

It is noted that the hierarchical clustering algorithms always search for the nearest pair of clusters and merge them into a new single cluster. However, in the numerical-constrained clustering, the objects in the same cluster must obey the constraint imposed. That is, If object o_i and o_j are merged in to a new object, then this new object can be only merged with those objects within both the constraint ranges of o_i and o_j. In other words, the merging process in an early iteration will reduce the number of possible merges in the subsequent iterations. This problem is referred to as order dependency in this paper. Note that the order dependency is different from the well-known chaining-effect in the single-link clustering algorithm, which means that two different

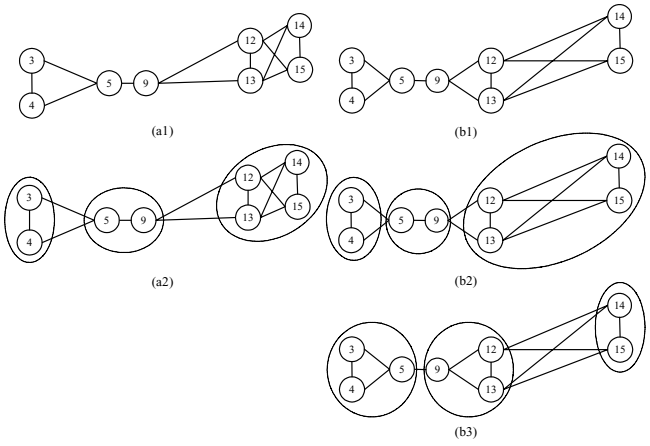


Fig. 10 Exhibition of order dependency. **a2** is the clustering result of **a1**. **b2** and **b3** are two clustering results of **b1**. Although **b3** is of better clustering quality, most clustering algorithms tend to generate such a result as in **b2**

clusters may be merged by a very slim noise link. An illustrative example of order dependency is shown in Fig. 10. As shown in Fig. 10a1, assume the constraint range, R, is four, the cluster number, k, is three, and the nearest pair of objects is o_5 and o_9. Thus, in the first iteration, o_5 and o_9 are merged into a cluster. This merging will break the connections between o_5 and o_3 and that between o_5 and o_4. Similarly, the connections between o_9 and o_{12} and between o_9 and o_{13} are disconnected. Finally, three natural clusters are obtained as shown in Fig. 10a2.

In the example shown in Fig. 10b1, the distances to objects o_5 and o_9 from objects o_3, o_4, o_{12}, and o_{13} are shorter than those in Fig. 10a1. The nearest pair of objects is still o_5 and o_9. By using the same merging scheme, the final cluster distribution is shown in Fig. 10b2. However, the clustering cost is obviously larger than that shown in Fig. 10a2. This is because the two small clusters containing $\{o_5, o_9\}$ and $\{o_3, o_4\}$ are generated in earlier iterations, causing $\{o_{12}, o_{13}, o_{14}, o_{15}\}$ to form a sparse cluster. It can be noted that although the structure and links of this data

set are similar to those in the Fig. 10a1, the characteristic of the point distribution is changed. Clearly, it is more beneficial to group the data points into the clusters as shown in Fig. 10b3, where objects with large distance are not put in the same cluster. The reason of the misclustering in Fig. 10b2 is the order of point merging, which is what we refer to as order dependency. Note that if the objects o_5 and o_3 are merged first, the only link disconnected by this merge is the connection between objects o_5 and o_9. Most of the future merges are not interfered. Thus, after the clustering algorithm stops, the clustering result is like the one shown in Fig. 10b3.

Order dependency problem is a common problem that most of the hierarchical clustering methods will encounter. However, without considering possible future merges, one may make incorrect merging decisions. To remedy this, we devise a progressive constraint relaxation technique which is able to take the future merges into consideration by progressively relaxing the constraints in early iterations.

4.2 Progressive constraint relaxation

As pointed out in the previous section, the order of merging will affect the quality of clustering results. If the difference of the constraint attribute a_c between two objects o_i and o_j is close to the constraint range R_{a_c}, merging them in the early iterations will put rigid bounds to the constraint attribute of this cluster and prevent further merging with other objects. As a result, we devise a technique, called *Progressive Constraint Relaxation*, to achieve the goal of diminishing the effect of order dependency.

The basic idea of the progressive constraint relaxation is that by using a tighter (smaller) constraint range in early iterations of merging, we will have more room to seek for better solutions in subsequent iterations. In addition to the relaxation of the constraint range, the desired cluster number should also be temporarily modified accordingly. The whole relaxation process starts with a small local constraint range and a large local desired cluster number. The procedure continuously relaxes the constraint range and reduces the number of desired clusters, and eventually results in the actual constraint range and the cluster number specified by the user. Explicitly, a new parameter $level$ is imported to control the number of relaxation steps. A constrained clustering algorithm is executed $level$ times to achieve the final clustering result. If the value of $level$ is equal to one, the algorithm is executed only once with the real constraint range and the cluster number. Otherwise, the temporary constraint range and cluster number, named $\text{local\_}R_{a_c}$ and $\text{local\_}k$, are assigned by the following expression,

$$\text{local\_}k = \frac{n \times (\text{level} - i) + k \times i}{\text{level}},$$

$$\text{local\_}R_{a_c} = R_{a_c} \times \frac{k}{\text{local\_}k},$$

where n is the size of the data set, i means the current iteration number (from 1 to $level$), and k is the desired cluster number. Note that by this definition, the product of $\text{local\_}R_{a_c}$ and $\text{local\_}k$ is a constant, i.e., $\text{local\_}R_{a_c} \times \text{local\_}k = R_{a_c} \times k$.

Progressive Constraint Relaxation Technique
//Input: an input data set, the number of clusters, k, the relaxation level, and the constraint range R_{ac}
1. For $i = 1$ to *level*, do Step 2 and Step 3.
2. Calculate the *local_k* and the *local_R_{ac}*.
3. Run the constrained clustering algorithm based on *local_k* and *local_R_{ac}*.

Fig. 11 The outlines of Progressive Constraint Relaxation Technique

As will be shown in our experiments, this technique improves not only the clustering quality but also the execution efficiency of all the hierarchical algorithms. The Progressive Constraint Relaxation Technique is outlined in Fig. 11.

Algorithm CCL (respectively, CSP) enhanced by the progressive constraint relaxation technique is referred to as progressive CCL (respectively, progressive CSP). We then have the following theorem.

Theorem 8 *With the progressive constraint relaxation technique, the average time complexities of algorithm CCL and algorithm CSP become*

(i) $O\left(nr_c k \log\left(\frac{nr_c k}{\text{level}}\right)\right) + O\left(\frac{n^2 r_c}{\text{level}^2} \log\left(\frac{n^2 r_c}{\text{level}^2}\right)\right) + O(f\ level)$, *for progressive CCL,*

(ii) $O(nr_c k) + O\left(\frac{n^2 r_c}{\text{level}^2}\right) + O(f \times \text{level})$, *for progressive CSP.*

Proof Since algorithm progressive CCL (progressive CSP) is intrinsically the one to apply algorithm CCL (algorithm CSP, respectively) $level$ times on different constraint ranges and different number of remaining data points, the time complexity will be the summation of the $level$ times of algorithm CCL. The number of data points left at the beginning of each iteration (n_i) is actually the number of clusters obtained in the previous iteration ($\text{local\_}k_{i-1}$). According to the time complexity of algorithm CCL, which is $O(n^2 r_c \log(n^2 r_c))$, the complexity of algorithm progressive CCL will be the summation of $\sum_{i=1}^{\text{level}} [O(n_i^2 r_{ci} \log(n_i^2 r_{ci})) + O(f)]$, where n_i is the number of remaining points at the beginning of iteration i, r_{ci} is the constraint ratio of this iteration, and $O(f)$ represents the lower order terms which can be ignored in the analysis of CCL. Note that n_i is equal to the $\text{local\_}k$ of its previous iteration, which can be obtained by the equation of $\text{local\_}k_{i-1} = \frac{n \times (\text{level} - (i-1)) + k \times (i-1)}{\text{level}}$, and r_{ci} is approximated by $r_c \times \frac{k}{\text{local\_}k_i}$ of this iteration according to $\text{local\_}R_{a_c} = R_{a_c} \times \frac{k}{\text{local\_}k}$. Also, we have

$$\sum_{i=1}^{\text{level}} \left[O\left(n_i^2 r_{ci} \log\left(n_i^2 r_{ci}\right)\right) + O(f) \right]$$
$$= \sum_{i=1}^{\text{level}} \left[O\left(\left(\text{local\_}k_{i-1}^2 \times r_c \frac{k}{\text{local\_}k_i} \right) \times \log\left(\text{local\_}k_{i-1}^2 \times r_c \frac{k}{\text{local\_}k_i} \right) \right) + O(f) \right],$$

where

$$O\left(\text{local\_}k_{i-1}^2 \times r_c \frac{k}{\text{local\_}k_i}\right)$$
$$= O\left(\left(\frac{n(\text{level}-(i-1))+k(i-1)}{\text{level}}\right)^2 r_c \frac{k}{\frac{n(\text{level}-i)+ki}{\text{level}}}\right)$$
$$= O\left(\frac{r_c k}{\text{level}} \times \frac{(n(\text{level}-i)+ki+(n-k))^2}{n(\text{level}-i)+ki}\right).$$

For $i = 1$ to $level - 1$,

$$\left(\frac{r_c k}{\text{level}} \times \frac{(n(\text{level}-i)+ki+(n-k))^2}{n(\text{level}-i)+ki}\right)$$
$$= O\left(\frac{n r_c k}{\text{level}}\right).$$

For $i = level$,

$$O\left(\frac{r_c k}{\text{level}} \times \frac{(n(\text{level}-i)+ki+(n-k))^2}{n(\text{level}-i)+ki}\right)$$
$$= O\left(\frac{n^2 r_c}{\text{level}^2}\right).$$

Therefore, the time complexity is

$$\sum_{i=1}^{\text{level}} \left[O\left(n_i^2 r_{ci} \log\left(n_i^2 r_{ci}\right)\right) + O(f)\right]$$
$$= \left(\sum_{i=1}^{\text{level}-1} O\left(\frac{n r_c k}{\text{level}} \log\left(\frac{n r_c k}{\text{level}}\right)\right)\right)$$
$$+O\left(\frac{n^2 r_c}{\text{level}^2} \log\left(\frac{n^2 r_c}{\text{level}^2}\right)\right)$$
$$+O(f \times \text{level})$$
$$= O\left(n r_c k \log\left(\frac{n r_c k}{\text{level}}\right)\right) + O\left(\frac{n^2 r_c}{\text{level}^2} \log\left(\frac{n^2 r_c}{\text{level}^2}\right)\right)$$
$$+O(f \times \text{level}).$$

Similarly, the time complexity of algorithm progressive CSP is

$$\sum_{i=1}^{\text{level}} \left[O\left(n_i^2 r_{ci}\right) + O(f)\right]$$
$$= O(n r_c k) + O\left(\frac{n^2 r_c}{\text{level}^2}\right) + O(f \times \text{level}).$$

It is important to see that according to Theorem 8, except the last iteration, the time required is reduced by about the order of $O(n)$. Although the complexity of the last iteration is almost equal to the original one without progressive constraint relaxation enhancement, the number of points actually involved is merely about $\frac{n}{\text{level}}$. Consequently, the progressive constraint relaxation technique significantly reduces the time complexity of the constrained clustering algorithms. Note that utilizing the progressive constraint relaxation technique will not increase the space complexity. Table 2 shows the symbols used in modeling the constrained clustering problem.

5 Extension to multiple constraint attributes

For the situation of using multiple constraint attributes, we have to check the constraints whenever a distance is calculated. According to the definition of numerical-constrained clustering problem, all the clusters generated should conform to the condition that the constraint distances of all the constraint attributes between any two objects in that cluster are required to be less than or equal to their corresponding constraint ranges. In other words, a cluster is generated only if all the constraints are satisfied. Let A_c be the set of constraint attributes, and $|A_c|$ be the number of constraint attributes. When being calculated, a distance is set to be infinity once a constraint distance exceeds its constraint range. Clearly, at most $|A_c|$ constraint attributes will be checked while the distance between two objects is computed. Accordingly, the time complexity of calculating pairwise distances will be increased from $O(n^2 r_c)$ to $O(|A_c| n^2 r_c)$ when the $|A_c|$ constraint attributes are used instead of only one constraint. With the progressive constraint relaxation technique on multiple constraint attributes, all constraints are relaxed simultaneously. In each level, new local constraint ranges are calculated for generating local clusters by a hierarchical-constrained clustering algorithm (algorithm CCL or CSP). Note that additional time is required for the calculation of new constraint ranges and the corresponding pairwise distances. Such additional time in fact approximates to the time increase from $O(n^2 r_c)$ to $O(|A_c| n^2 r_c)$ in the pairwise distance calculation of algorithm CCL and CSP. Similarly, when a point is inserted into a cluster with $O(n)$ points by the time complexity of $O(\log(n))$ in algorithm CDC, as analyzed in Theorem 2, the time complexity will become $O(|A_c| \log(n))$ when $|A_c|$ constraint attributes are taken into consideration. The time complexities of the proposed algorithms for multiple constraint attributes are shown in Table 3.

Note that for the complexity of algorithm CCL, the former part is the complexity of calculating $O(n^2 r_c)$ distances, and the latter part is that of sorting $O(n^2 r_c)$ distances. Since the former is much smaller than the latter, the time complexity is dominated by the latter and can be approximated by $O(n^2 r_c \log(n^2 r_c))$. Similarly, the time complexity of algorithm progressive CCL can be approximated by the one with single constraint attribute.

Consider an extreme case that several contradictory constraints have to be satisfied according to the domain knowledge, which causes the desired number of clusters not to be generated. That is, we cannot force any two clusters to merge

Table 2 Meanings of symbols used

Symbol	Description
a_c	Constraint attribute
A_c	The set of constraint attributes, where $a_c \in A_c$.
$\|A_c\|$	The number of constraint attributes in A_c
R_{a_c}	Constraint range of constraint attribute a_c
$d_{a_c}(.,.)$	Distance of the constraint attribute between objects o_i, o_j or clusters C_i, C_j
r_c	Constraint ratio
CDC	Partition-based clustering algorithm with depth control
CCL	Constrained clustering algorithm with complete link
CSP	Constrained clustering algorithm with spring model

Table 3 The time complexities for multiple constraint attributes

Algorithm	Time complexity
CDC (one iteration)	$O(n(kc)^d \|A_c\| \log(n))$
CCL	$O(\|A_c\| n^2 r_c + n^2 r_c \log(n^2 r_c))$
CSP	$O(\|A_c\| n^2 rc)$
Progressive CCL	$O\left(nr_c k \log\left(\frac{nr_c k}{\text{level}}\right)\right) + O\left(\frac{n^2 r_c}{\text{level}^2} \log\left(\frac{n^2 r_c}{\text{level}^2}\right)\right) + O(f \times \text{level})$
Progressive CSP	$O(\|A_c\| nr_c k) + O\left(\frac{\|A_c\| n^2 r_c}{\text{level}^2}\right) + O(f \times \text{level})$

further without violating any constraint. In this situation, users can observe the clustering hierarchy to obtain some satisfactory clustering results (including clustering cost and cluster number), which satisfy all the specified constraints. Afterward, the users will decide either to accept the generated clustering results or to modify some constraints for reconstructing clusters. Users can interact with the constrained clustering model to refine constraint ranges or just build a clustering hierarchy according to the specified constraints and select appropriate clustering results on the clustering hierarchy.

6 Performance studies

To assess the performance of these algorithms, we have conducted a series of experiments. These experiments are performed on a computer with an 800 Mhz Intel CPU and 1 GB of memory. In order to generate the synthetic data used in our experiments, we adopt a method similar to the one in [22] and [10], by which n data points for k clusters are generated. Without loss of generality, every generated data point contains two optimization attributes and one constraint attribute, which is assigned by a timestamp according to the exponential distribution. Several such synthetic data sets are combined together to simulate the phenomenon of clusters with time constraint. Both of the optimization attributes are in the range of {−100,100}, and the range of the constraint attribute is {0, 10000}. Four such data sets, each of which contains 2500 data points for five clusters, are combined into 10, 000 data points with 20 clusters which may overlap with one another in optimization and constraint attributes. All the data sets in our experiments are random samples of this data set. Therefore, these data sets have the same distribution as described above. We also adjust the distribution of the synthetic data sets in our experiments, and observed that the performance results will be primarily dependent upon the characteristics of these algorithms, which is mainly consistent with the conclusion we obtained from the data set presented in this paper.

In Sect. 6.1, we will show the effect of progressive constraint relaxation technique on improving both the clustering quality and the execution efficiency of algorithms CCL and CSP. In Sect. 6.2, we conduct a series of experiments to exhibit the properties of algorithm CDC. The parameters of algorithm CSP are investigated in Sect. 6.3. Next, an overall performance comparison among all algorithms is conducted in Sect. 6.4. Finally, we will perform a scale up experiment on the size of data set.

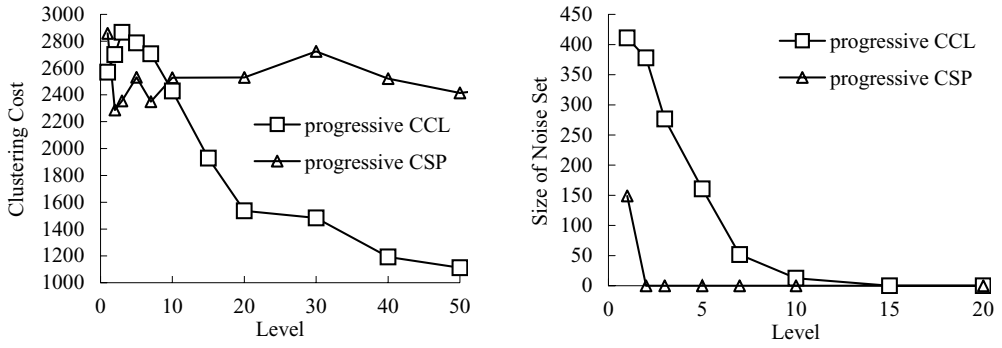
In the following experiments, we use the average squared error of all points as the evaluation function for clustering results, i.e.,

$$\text{Cost}(Cl) = sq(Cl) = \frac{\sum_{C_i} \sum_{p \in C_i} (p - C_i.\text{center})^2}{\sum_{C_i} \|C_i\|},$$

where C_i denotes a cluster of the clustering result Cl, and C_i.center is the center of cluster C_i.

6.1 On progressive constraint relaxation

In this experiment, we apply the hierarchical algorithms to a data set of 5000 points with their timestamp in the range of (0, 10000). Here, we set the time constraint as 4000 and try to partition the data set into 20 clusters. Note that if the value of *level* is equal to one, the original algorithm without relaxation is performed. As shown in Fig. 12, by solving the order dependency, the progressive constraint relaxation technique not only improves the clustering results of algorithm



(a) clustering cost (b) size of noise set

Fig. 12 The clustering costs and sizes of noise sets of progressive CCL and progressive CSP at different levels

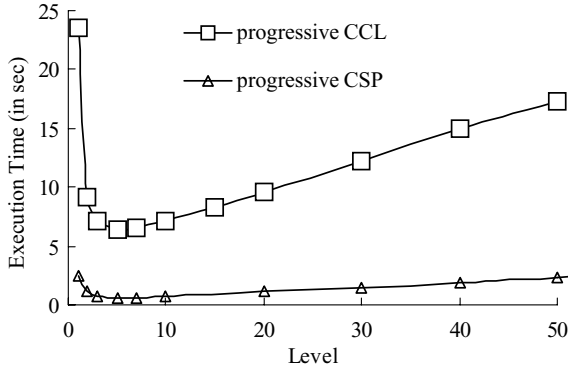


Fig. 13 The execution times of the progressive CCL and progressive CSP at different levels

CCL but also reduces the numbers of outliers of algorithm CCL. It is also interesting to note that the progressive constraint relaxation technique enhancement does not affect the algorithm of spring model very much. This can be explained by the reason that at the beginning of algorithm progressive CSP, we only consider partial of the real links. Thus, one point may be moved far away from its original position and forbidden from merging with its similar object in later iterations of the progressive relaxation algorithm.

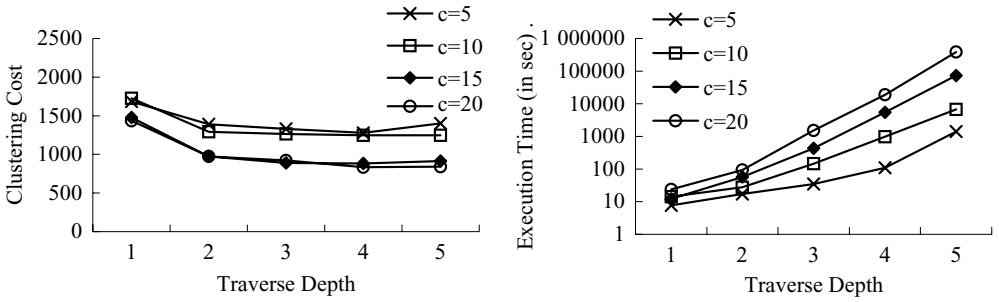
As shown in Fig. 13, the progressive constraint relaxation technique enhancement can also be used to reduce the execution time with smaller levels. Because in the early iterations, with the smaller time constraint values, only a few points are considered to be grouped. In the later iterations, the number of clusters has become much smaller than the original so that it can reduce the execution time. The improvement could be as much as 75%, i.e., takes only $\frac{1}{4}$ of the original execution time. However, with larger values, it takes too much time to rebuild the distance matrix at the beginning of each iteration, thus lengthening the execution time of the whole algorithm.

6.2 On partition-based clustering

In this experiment, we use the same data set and same time constraint as in the previous experiment. Since the time complexity of algorithm CDC is exponential to the maximum traverse depth, a large value will not be applicable in practice. As shown in Fig. 14b, the execution time increases sharply with the maximum traverse depth. We can also observe that when the maximum traverse depth increases to two, the clustering cost decreases significantly. However, the clustering cost becomes stable when the maximum traverse depth is larger than two, as shown in Fig. 14a. Therefore, from the complexity analyses and empirical experiments, setting the maximum traverse depth to be two suffices in practical usage. In our experiment, we observe that the execution time of algorithm CDC is less when the time constraint is either smaller or larger. This is because that when the time constraint is strict, the size of conflict set exceeds the limits easily and the algorithm will not go into higher depth. On the other hand, when the time constraint is loose, there will be no points violating the constraint and thus no need for a larger depth.

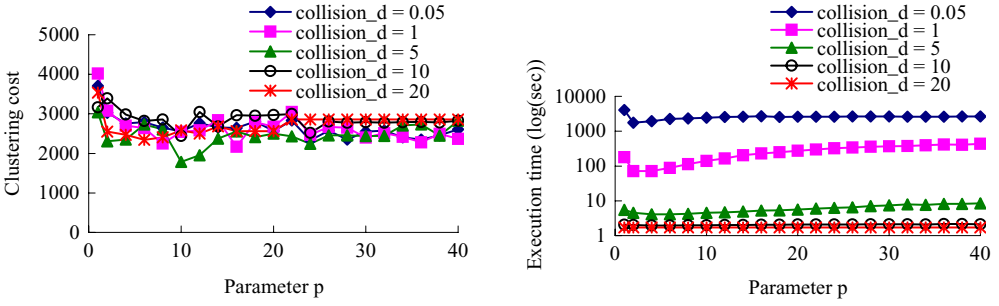
6.3 On hierarchical clustering

The sensitivity analyses on parameters of hierarchical-constrained clustering algorithms are investigated in this experiment. There is no parameter for algorithm CCL, therefore, we observe parameter p and the collision distance of algorithm CSP on the same data set and same time constraint as in previous experiments. Parameter p is a positive integer used in the calculation of the gravity force $\|\vec{F}_g\| = \frac{c_g \times m_1 \times m_2}{r^p}$, and the collision distance is the distance between two points which can be merged. As shown in Fig. 15, there are some differences in the clustering costs due to the distribution of data points when using different values of collision distance. However, the number of points merged in each iteration is fewer with smaller collision distance. Consequently, more iterations are needed to merge all the data points into the desired number of clusters. On the other hand, with smaller parameter p, the gravity forces are less sensitive to the distances between objects, thereby resulting in larger clustering costs but shorter execution time. However, clustering costs and execution time tend toward stable as parameter p increases. Therefore, this parameter



(a) clustering cost (b) execution time

Fig. 14 The clustering costs and execution times of algorithm CDC when varying the traverse depth and sizes of the maximum conflict set (**c**). Note that the execution times shown in **b** are in log scale for ease of illustration



(a) clustering cost (b) execution time

Fig. 15 The clustering costs and execution times of algorithm progressive CSP with different values of parameter p and collision distances

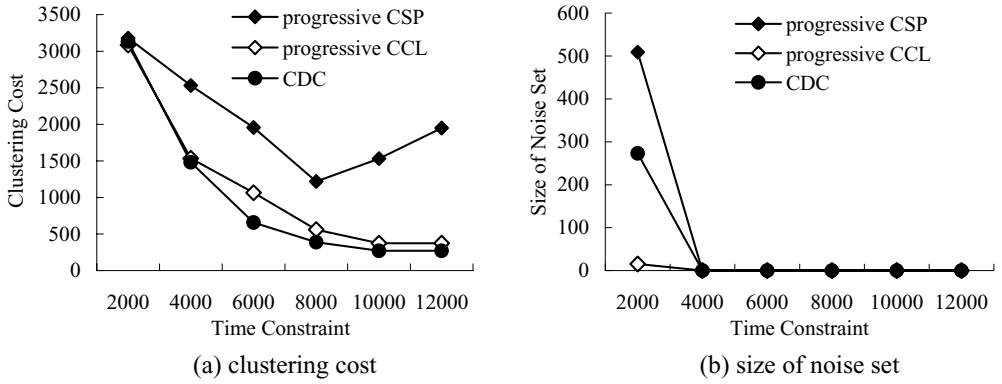
can be regarded as a control value in most applications. If distances are more important in the process of merging clusters, a larger value of p should be used. On the other hand, if the number of objects in a cluster is more important in the process of merging clusters, a smaller value of p is preferred. Note that the execution times of algorithm CSP shown in Fig. 15b are in log scale for ease of illustration.

6.4 Overall comparison between these algorithms

In this experiment, we use the same data set as in the previous experiments. For comparison reason, we choose suitable parameters for each algorithm. The traverse depth of algorithm CDC is set as 2; the level of the progressive constraint relaxation is set as 20 for both algorithm progressive CSP and CCL. The qualities of clustering results and sizes of noise sets are shown in Fig. 16. As shown, the clustering quality of progressive CCL is very similar to that of algorithm CDC. However, it produces a much smaller noise set. It is noted that algorithm CDC is especially suitable for loose time constraints.

The execution times of these algorithms are shown in Fig. 17, where the execution time of algorithm progressive CSP is shown to be much less than the others. We will show the superiority of algorithm CSP more clearly in the next experiment. It is interesting to notice that the execution time of algorithm CDC is less when the time constraint is either smaller or larger. This is because that when the time constraint is strict, the size of conflict set exceeds the limits easily and the algorithm will not go into higher depth. On the other hand, when the time constraint is loose, there will be no points violating the constraint and thus no need for higher depth. In both cases, the algorithm will run much faster.

In addition, we can also observe the sensitivity of the clustering result against varying the value of the constraint attribute in this experiment. When using a strict constraint or very scattered distribution of the timestamp values, we have to sacrifice the clustering cost in order not to violate the imposed constraint. In the extreme case, we may encounter the situation that the clustering cost is too large, or that the desired cluster number cannot be generated successfully. Hence, when some constraints have to be strictly followed, the users will have to accept the corresponding high clustering cost. On the other hand, if we do not have precise values of the constraints but are allowed to adjust them slightly, better clustering quality may be obtained by refining the constraints or cluster number according to the clustering results. For example, according to the constraints and the cluster number k specified by users, the hierarchical constrained clustering algorithm CCL and CSP will build a clustering hierarchy in the clustering process until there are k clusters left on the top of the tree. However, if the constraints are too tight or k is too small, the cluster number on the top of clustering hierarchy might be larger than k when the algorithm terminates because no clusters can be merged further without violating any constraint. In this situation, users can decide either to refine constraint ranges or to just build a clustering hierarchy according to the specified constraints



(a) clustering cost (b) size of noise set

Fig. 16 The clustering costs and sizes of noise sets of different algorithms when time constraint varies

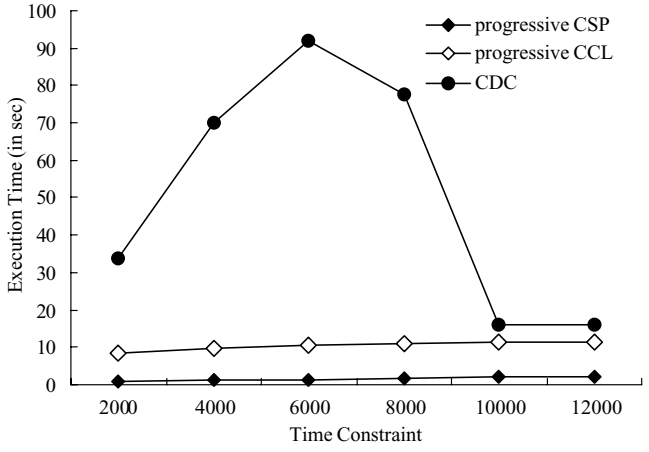


Fig. 17 The execution times of different algorithms when time constraint varies

and select appropriate clustering results, including the cluster number and the clustering cost, on the clustering hierarchy.

The experiments of varying cluster number k are also conducted in this section, as shown in Fig. 18. As the number of clusters increases, the clustering cost decreases. However, the execution time of CDC increases significantly since the time complexity of CDC is proportional to k^d, as shown in Theorem 2. On the other hand, the execution time of progressive CCL and progressive CSP increases almost linearly, which conforms to the complexity analyses shown in Theorem 8.

From all the experiments conducted, we can also observe that there are some differences between the clusters generated by these methods. As illustrated in Sect. 3, algorithm CDC tries to minimize the distances of data points to their corresponding centroid, and algorithm CCL is based on the Complete-link algorithm, which merges two clusters to generate a new cluster with the minimum diameter in each iteration. Therefore, these two algorithms are both very suitable in determining spherical clusters, but algorithm CDC takes more time to adjust the clusters iteratively and results in better clustering results. On the other hand, algorithm CSP can generate clusters with arbitrary shapes. Further, as discussed in Sect. 4, the progressive relaxation technique allows algorithms CCL and CSP to relax the constraint ranges of clusters iteratively. Therefore, the clusters generated will be more compact since more data points are allowed to join a cluster before the ranges of the constraint attributes in a cluster reach their maximum value.

6.5 On the scalability

We next apply our algorithm on a data set consisting of 10, 000 points with timestamp ranging from 0 to 10, 000. We also make several smaller data sets consisting of {1000, 2000, 5000, 7000} points by sampling the original data set. So that these data sets have the same distribution with each other.

The execution time of these algorithms is shown in Fig. 19a. As shown, the algorithm progressive CSP is the fastest one, and algorithm CDC runs slowest. However, if we show the growth of trends of execution time of these algorithms, i.e., execution time is normalized to the one of the same algorithm for the 1000-point data set. The result is shown in Fig. 19b. As shown, algorithm CDC grows slower than any other algorithm. However, after the analysis of the growth trend, we envision that algorithm CDC will outperform algorithm progressive CSP when the size of data set grows to 10^8 points. Thus, progressive CSP is the most efficient algorithm in most cases.

7 Conclusions

In this paper, we proposed a new constrained clustering problem, named numerical-constrained clustering. Explicitly, we addressed in this paper the constrained clustering with numerical constraints. We devised several effective and efficient algorithms to solve the problem in this paper. In addition, we devised a *progressive constraint relaxation* technique to handle the order dependency and improve the overall performance of clustering results. As shown in the complexity analyses and also validated by our empirical studies, the partition-based algorithm CDC performed

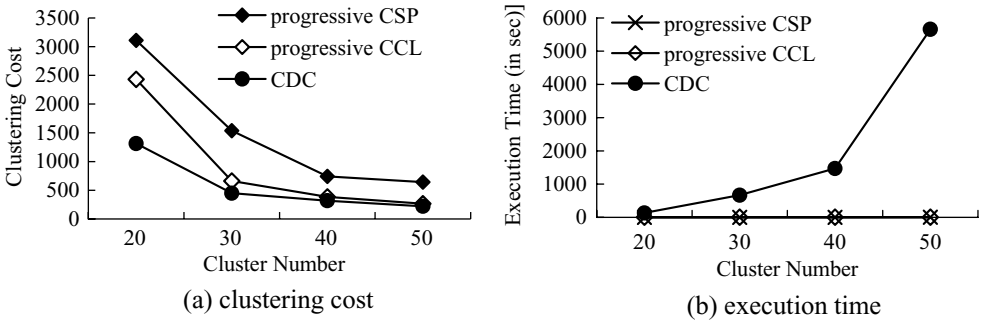


Fig. 18 The clustering costs and execution time of different algorithms when the number of clusters varies

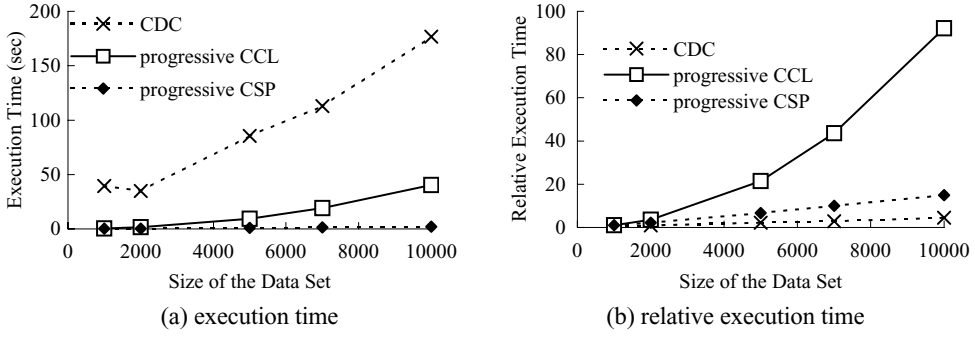


Fig. 19 The execution times and relative execution times of different algorithms with various sizes of data set

well in the cost of clustering results, but spent more time for data points to traverse between clusters to achieve a stable result with the minimum cost. In contrast, the hierarchical algorithms which were further enhanced by progressive constraint relaxation technique, i.e., both progressive CCL and progressive CSP, were executed much more efficiently than algorithm CDC in the moderate size of data sets with a slight increase in clustering costs.

References

1. Büchner, A.G., Mulvenna, M.D.: Discovery internet marketing intelligence through online analytical web usage mining. ACM SIGMOD Rec. **27**(4), 54–61 (1998)
2. Chen, M.-S., Han, J., Yu, P.S.: Data mining: An overview from database perspective. IEEE Trans. Knowl. Data Eng. **8**(6), 866–883 (1996)
3. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R.: Advances in Knowledge Discovery and Data Mining. MIT Press, Cambridge, MA (1996)
4. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2000)
5. Aggarwal, C.C., Procopiuc, C., Wolf, J.L., Yu, P.S., Park, J.-S.: Fast algorithms for projected clustering. In: Proceedings of ACM SIGMOD (1999)
6. Bradley, P.S., Bennett, K.P., Demiriz, A.: Constrained *K*-Means Clustering. MSR-TR-2000-65, Microsoft Research, May (2000)
7. Lin, C.-R., Chen, M.-S.: A robust and efficient clustering algorithm based on cohesion self-merging. In: Proceedings of the ACM SIGKDD (2002)
8. Ng, R.T., Han, J.: Efficient and effective clustering methods for spatial data mining. In: Proceedings of the VLDB (1994)
9. Tung, A.K.H., Han, J., Lakshmanan, L.V.S., Ng, R.T.: Constraint-based clustering in large databases. In: Proceedings of the 2001 International Conference on Database Theory (2001)
10. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An efficient data clustering method for very large database. In: Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 103–114 (1996)
11. Crescenzi, P., Kann, V.: A compendium of NP optimization problems [http://www.nada.kth.se/∼viggo/problemlist/compendium.html]
12. Lu, S.Y., Fu, K.S.: A sentence-to-sentence clustering procedure for pattern analysis. IEEE Trans. Syst. Man Cybern. **8**, 381–389 (1978)
13. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum, New York (1981)
14. Dubes, R.C.: How many clusters are best?—an experiment. Pattern Recognit. **20**(6), 645–663 (1987)
15. Sneath, P.H.A., Sokal, R.R.: Numerical Taxonomy. Freeman, London (1973)
16. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the Theory of Neural Computation. Addison-Wesley Longman, Reading, MA (1991)
17. Rose, K., Gurewitz, E., Fox, G.: Constrained clustering as an optimization method. IEEE Trans. Pattern Anal. Mach. Intell. **15**(8), 785–794 (1993)
18. Estivill-Castro, V., Lee, I.: Autoclust+:automatic clustering of point-data sets in the presence of obstacles. In: Proceedings of the TSDM, pp. 133–146 (2000)
19. Tung, A.K.H., Hou, J., Han, J.: Spatial clustering in the presence of obstacles. In: Proceedings of the ICDE, pp. 359–367 (2001)
20. Zaïane, O.R., Foss, A., Lee, C.-H., Wang, W.: On data clustering analysis: Scalability, constraints, and validation. In: PAKDD, pp. 28–39 (2002)

21. Klein, D., Kamvar, S.D., Manning, C.: From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In: Proceedings of the the Nineteenth International Conference on Machine Learning (ICML-2002), Sydney, Australia (2002)
22. Lin, C.-R., Chen, M.-S.: On the optimal clustering of sequential data. In: Proceedings of the 2nd SIAM International Conference on Data Mining (2002)
23. Jagadish, H.V., Madar, J., Ng, R.T.: Semantic compression and pattern extraction with fascicles. In: Proceedings of the VLDB, pp. 186–198 (1999)
24. Yeung, M., Yeo, B.L.: Time-constrained clustering for segmentation of video into story units. In: International Conference on Pattern Recognition, pp. 375–380, May (1996)
25. Palmer, C.R., Faloutsos, C.: Density-biased sampling: An improved method for data mining and clustering. In: ACM SIGMOD Int. Conf. Manage. Data (2000)
26. King, B.: Step-wise clustering procedures. J. Am. Stat. Assoc. **69**, 86–101 (1967)
27. Oyang, Y.-J., Chen, C.-Y., Yang, T.-W.: A study on the hierachical data clustering algorithm based on gravity theory. In: Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 350–361 (2001)