Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition 38 (2005) 2256–2269
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

A statistics-based approach to control the quality of subclusters in incremental gravitational clustering

Chien-Yu Chen[a,*], Shien-Ching Hwang[a], Yen-Jen Oyang[a,b]

[a]*Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan*
[b]*Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei 106, Taiwan*

Received 30 July 2004; received in revised form 21 March 2005; accepted 21 March 2005

Abstract

As the sizes of many contemporary databases continue to grow rapidly, incremental clustering has emerged as an essential issue for conducting data analysis on contemporary databases. An incremental clustering algorithm refers to an abstraction of the distribution of the data instances generated by the previous run of the algorithm and therefore is able to cope well with the ever-growing contemporary databases. There are two main challenges in the design of incremental clustering algorithms. The first challenge is how to reduce information loss due to the data abstraction (or summarization) operations. The second challenge is that the clustering result should not be sensitive to the order of input data. This paper presents the GRIN algorithm, an incremental hierarchical clustering algorithm for numerical datasets based on the gravity theory in physics. In the design of GRIN, a statistical test aimed at reducing information loss and distortion is employed to control formation of subclusters as well as to monitor the evolution of the dataset. Due to the statistical test-based summarization approach, GRIN is able to achieve near linear scalability and is not sensitive to input ordering.
© 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Data clustering; Hierarchical clustering; Incremental learning; Gravity theory

1. Introduction

Data clustering is an important mechanism for solving various real-world problems such as segmentation, database compression, vector quantization, and pattern recognition [1–5]. Due to rapidly emerging application domains in recent years such as data mining and bioinformatics, data clustering has attracted a new round of attention [6–8]. One of the main challenges in the design of modern clustering algorithms is that, in many applications, new data instances are continuously added into an already huge database. Therefore, it is impractical to carry out data clustering from scratch whenever new data instances are added into the database. One way to tackle this challenge is to incorporate a clustering algorithm that operates incrementally.

The development of incremental clustering algorithms can be traced back to 1970s [4]. The LEADER [9] algorithm uses a threshold to determine if an instance can be placed in an existing cluster or it should form a new cluster by itself. Many incremental algorithms follow this model for clustering data instances incrementally. COBWEB [10] and CLASSIT [11] are incremental hierarchical clustering algorithms designed for categorical and numerical datasets, respectively. When processing incoming data instances, COBWEB and CLASSIT employ four operations *insert*, *create*, *split*, and *merge* to adjust the hierarchical structure locally. A clustering dendrogram is desired in many applications due to the need of taxonomies [4]. However, both COBWEB

* Corresponding author. Tel.: +886 3 4638800x2185; fax: +886 2 23688675.
E-mail addresses: cychen@mars.csie.ntu.edu.tw (C.-Y. Chen), schwang@mars.csie.ntu.edu.tw (S.-C. Hwang), yjoyang@csie.ntu.edu.tw (Y.-J. Oyang).

0031-3203/$30.00 © 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2005.03.005

and CLASSIT could result in highly unbalanced trees [6]. In recent years, several incremental clustering algorithms have been proposed for mining and monitoring evolving datasets [12–16]. Among them, Ribert's algorithm and the BIRCH algorithm keep maintaining a hierarchy as clustering outputs. Ribert's algorithm suffers a higher time complexity when compared with a linear time algorithm and therefore is not suitable for handling large datasets. On the other hand, the BIRCH algorithm [14,16] features low time and space complexity by means of grouping similar instances as a subcluster and using the derived subclusters as the primitives when generating a hierarchy.

Grouping data instances as a subcluster is considered as a process of summarization or data abstraction [4]. Data summarization keeps playing an important role in developing incremental clustering algorithms. Furthermore, as Ganti and Zhang showed in their papers [14,16], grouping data instances as a subcluster provides a good solution when maintaining hierarchies for large datasets incrementally. This idea has also been employed to scale up the hierarchical clustering algorithms successfully [17]. Since the subclusters are the primitives for generating a hierarchy as the clustering results, the quality of subclusters is crucial to the quality of the hierarchy derived.

There are three common issues associated with data summarization or abstraction. The first issue lies in how to choose a threshold while utilizing a fixed threshold to control subclusters. A new instance can be inserted into an existing subcluster as long as the dissimilarity between the new instance and the representative of the subcluster is smaller than a given threshold. Fig. 1(a) shows an example where a global threshold may fail. In Fig. 1(a), the gray balls are data instances in the database so far. As we can see, unreasonable clustering results arise due to the improper threshold. The second issue associated with data abstraction is the information loss due to data abstraction. As illustrated in Fig. 1(b), the distribution of the instances in the cluster is not consistent with the abstraction model employed to summarize a cluster. In this case, the abstraction model is the centroid and the radius of a cluster. When only the centroid and the radius of the subcluster are given, the algorithm will insert new data instances (i.e. the white balls) into wrong clusters. The third issue concerns how to properly monitor the transition of dataset. We observed that the insertions of new data points into an existing subcluster might result in the shifting of distribution within the subclusters. As exemplified in Fig. 1(c), once there are some more new instances falling in the left part of the circle, the elements inside the subcluster is not uniformly distributed any more. Splitting this subcluster would be necessary; otherwise the information loss will be amplified in the remaining clustering process.

Besides, the sensitiveness to the arriving ordering of input data is also a main problem associated with modern incremental clustering algorithms. Improper arriving order makes designing an incremental clustering a more challenging task when data abstraction is considered. Fig. 2 presents

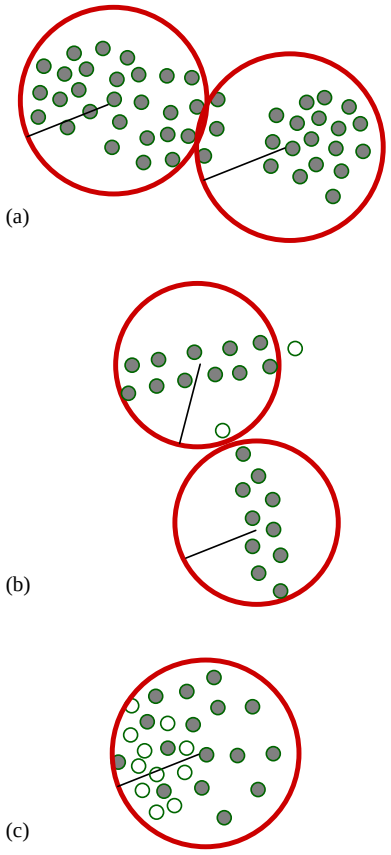

Fig. 1. Problems might happen due to data abstraction. Gray balls stand for the data instances being issued so far, and white balls stand for the new instances that will come later. (a) Flaws arise if using a fixed distance threshold to control the formation of subclusters. (b) New data instances (i.e. the white balls) will fall in wrong subclusters due to information loss after data abstraction has been executed. (c) The subcluster is not homogeneous any more after new data instances are added.

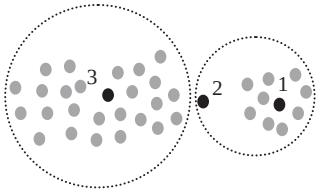

Fig. 2. A case in which the distance-based controlling approach may fail to deliver satisfactory clustering quality due to the improper arriving order of data instances.

an example where data instances arrive in unexpected order. In this example, θ denotes the threshold imposed on the diameters of leaf subclusters and it is assumed that distance (instance 1, instance 2) $< \theta$, distance (instance 2, instance 3) $< \theta$, and distance (instance 1, instance 3) $> \theta$. As the example shows, if the data enters in the following order

$1 \rightarrow 2 \rightarrow 3 \ldots$, then data instance 2 will be clustered with data instance 1 instead of with data instance 3. Such a clustering result is not in conformity with what we consider natural clusters. It is certain that this problem can be resolved by setting θ to a smaller value. However, since θ is a parameter automatically set by users heuristically or based on some system metrics, it could occur that the setting is not appropriate for some leaf subclusters.

In this paper we proposed the GRIN algorithm, an incremental hierarchical clustering algorithm for numerical datasets based on gravity theory in physics. Different from previous approaches, GRIN employs a statistical test to determine the subclusters. By employing the proposed statistical test, the distribution of a subcluster will be consistent with the assumption of the abstraction model. Continuing with the example shown in Fig. 2, even though instance 2 is clustered with instance 1 in the first run of the algorithm due to the unexpected sequence order, instance 2 would not be in the same subcluster with instance 1 according to the proposed statistical test. Thus, instance 2 forms a subcluster by itself. In this case, instance 2 still has the opportunity to merge with other subclusters or other instances that come later. Furthermore, the proposed statistical test can also be used to monitor the transition of dataset. This approach reduces the impact from improper input ordering. As the experiments conducted in this study reveal, the GRIN algorithm delivers favorite clustering quality in comparison with some other clustering algorithms and enjoys efficient execution time in general.

The time complexity of the GRIN algorithm is $O(nm)$, where n is the number of instances in the dataset and m is the number of leaf subclusters maintained by the algorithm. Actually, the value of m is related to the number of instances in the dataset, n. In the worst case, m equals n and each leaf subcluster contains only one instance. Our experiments show that empirically the value of m is much smaller than n in most cases. This observation holds when the dataset exhibits cluster tendency. As the space complexity is considered, the space requirement of GRIN algorithm is $O(m)$

As to the followings of this paper, Section 2 describes the agglomerative hierarchical clustering algorithm that the GRIN algorithm invokes to construct the clustering dendrogram. Section 3 presents the proposed summarization process and the statistical test. In Section 4 we discuss how the GRIN algorithm works. Section 5 reports the experiments conducted to study the characteristics of the GRIN algorithm, while the concluding remarks are given in Section 6.

2. The gravitational hierarchical clustering algorithm

This section briefly introduces the gravitational hierarchical clustering algorithm that is invoked by the GRIN algorithm for constructing the clustering dendrogram. The gravitational clustering algorithm is first proposed in Ref. [18], and has been well studied by Refs. [19–22]. In this paper, we employ the physical model defined in the paper [20]. The gravitational clustering algorithm simulates how a number of objects move and interact with each other due to the gravity force. Whenever two objects hit, which means that the distance between these two objects is less than the lumped sum of their radii, they merge to form one new and larger object. In the simulation, the merge of objects corresponds to forming a new, one-level higher cluster that contains two existing clusters. An analysis of why the gravitational clustering algorithm is guaranteed to terminate and its main characteristics can be found in [20,21]. Since the gravitational clustering algorithm generates a clustering dendrogram like other hierarchical agglomerative clustering (HAC) algorithms [2,3], we will use G-HAC to refer to the gravitational hierarchical clustering algorithm invoked by GRIN in this paper.

Fig. 3 shows the pseudo-code of the G-HAC algorithm. Basically, the G-HAC algorithm iteratively simulates the movement of each node during a time interval and check for possible merge. One key operation in the G-HAC algorithm is to compute the velocity of each disjoint node remaining in the system. In the G-HAC algorithm, Eq. (1) below is employed to compute the velocity of a node during one time interval. The derivation of Eq. (1) involves solving a differential equation under several pragmatical assumptions and is elaborated in Ref. [21].

$$v_j = \sqrt{\frac{\left\| \sum_{\text{node } n_i} \vec{F}_{g_i} \right\|}{C_r}}, \tag{1}$$

where $\sum_{\text{node } n_i} \vec{F}_{g_i}$ is the vector sum of the gravity forces that node n_j experiences from all the other disjoint nodes remaining in the physical system at the beginning of the time interval, and C_r is the coefficient of air resistance. According to gravity theory,

$$\| \vec{F}_{g_i} \| = C_g \frac{(\text{mass of } n_i)(\text{mass of } n_j)}{\text{distance}^k(n_i, n_j)}, \tag{2}$$

where C_g is a coefficient. In the physical world, $k = 2$. However, in the G-HAC algorithm, k can be any positive integer number.

3. The statistics-based summarization process

In this paper, we use centroid, radius, and the number of data instances it contains as the features of the abstraction model to represent a subcluster. The assumption of using this model for data summarization is that the data instances in each subcluster should be uniformly spread in the sphere defined by the centroid and the radius. The legitimacy of this assumption is based on the following observation: any cluster, regardless of its shape and whether it has a uniform density, can be decomposed into a number of spherical subclusters, each with virtually uniform density. Fig. 4(a) and

Procedure G-HAC
Input: W: the set of instances
{
  Repeat {
    $min\_D = MAX$; (MAX could be the maximal number of floats when implementing the algorithm.)
    $max\_V = 0$;
    For every pair of nodes $n_i, n_j \in W$ {
      calculate the distance D_{ij} between n_i and n_j;
      if ($min\_D > D_{ij}$) $min\_D = D_{ij}$;
    }
    For every $n_i \in W$ {
      calculate the new velocity V_i of n_i according to equation (2);
      if ($max\_V < V_i$) $max\_V = V_i$;
    }
    time interval $T = (min\_D / R) / max\_V$; (R is a scalar number.)
    For every $n_i \in W$ {
      calculate the new position of n_i based on V_i and T;
    }
    For every pair of nodes $n_i, n_j \in W$ {
      if (n_i and n_j hit during the time interval T){
        create a new cluster containing the clusters represented by n_i and n_j;
        merge n_i and n_j to form a new node n_h with lumped masses and merged momentum;
        delete n_i and n_j from W;
        add n_h to W;
      }
    }
  } Until (W contains only one node);
  Return the dendrogram;
}

Fig. 3. The pseudo-code of the G-HAC algorithm.

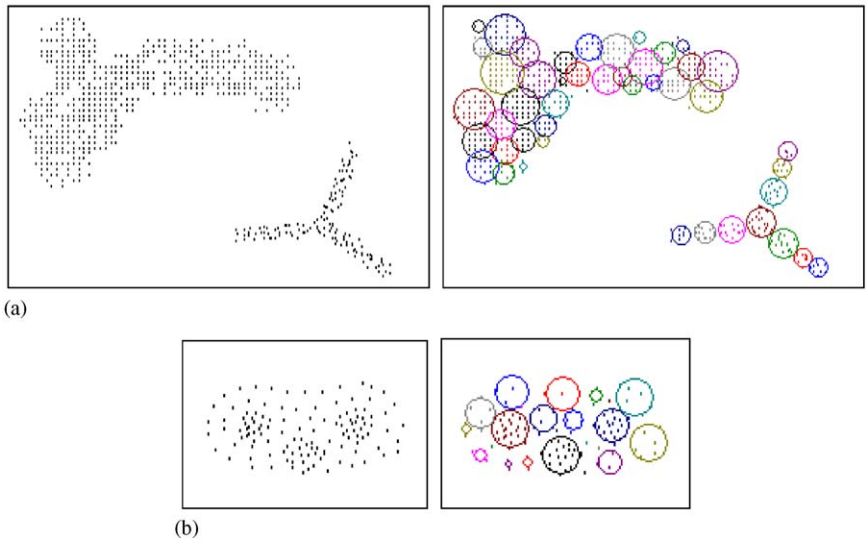

Fig. 4. An example demonstrating how an arbitrarily-shaped cluster can be represented by a set of spherical clusters. (a). Decomposition of arbitrarily-shaped clusters into a set of spherical subclusters. (b). Decomposition of a cluster with non-uniform density into a set of spherical subclusters.

(b) show two examples. In Fig. 4(a), two arbitrarily-shaped clusters with virtually uniform density are decomposed into a number of spherical subclusters. Fig. 4(b) shows an example of decomposing a cluster with non-uniform density into a set of spherical subclusters, each with virtually uniform density. Therefore, spherical subclusters will be the primitive building blocks of the clustering dendrogram derived. We call it a leaf cluster or a leaf subcluster. In particular, the term *leaf spherical cluster* is alternatively used in this paper, because the leaf clusters identified by our algorithm is in spherical shape. Note that, though any two clusters contain disjoint sets of data instances, the spheres defined by their respective centroids and radii may overlap. The centroid of a cluster is defined to be the geometric center of the data instances in the cluster and the radius is defined to be the maximum distance between the centroid and the data instances in the cluster (Figs. 3). and 4

This summarization process is supposed to be applied to a dendrogram generated by a HAC algorithm [2,3]. Each node in the clustering dendrogram corresponds to a cluster of data instances. A cluster is said to be in the spherical shape, if the cluster satisfies either one of the following two conditions:

(1) The cluster contains less than *Min* data instances, where *Min* is a parameter to be set based on the statistical sense discussed in the following.
(2) The cluster contains *Min* or more data instances and passes the statistical test.

A cluster containing less than *Min* data instances is considered as a spherical cluster by default, because such a cluster does not contain sufficient number samples for any meaningful statistical test to be conducted. Therefore, we just trust the employed HAC algorithm for its capability of identifying spherical clusters of small size. Concerning a cluster containing *Min* or more data instances, a χ^2 goodness of fit test [23] is conducted to check whether the data instances in a cluster are uniformly distributed or not. The hypothesis of the statistical test is that the data instances in the cluster are uniformly distributed in the sphere defined by the centroid and the radius of the cluster. The test is applied to each node in the dendrogram in order of their formation. In other words, a node is tested only after all of its descendents have been tested. A node that has passed the test will be marked as a spherical cluster. Fig. 5 presents an example that illustrates the statistical test. In both Fig. 5(a) and (b), there is a cluster containing three subclusters, each of them is already identified as a spherical cluster and is not covered by another parent spherical cluster. Conceivably the new cluster in Fig. 5(a) should have a higher chance to pass the statistical test when compared with that of Fig. 5(b). In the statistical test, each subcluster in the new cluster is considered as containing a set of random samples from it. Take Fig. 5(a) as example, the χ^2 test of goodness of fit is applied to determine whether the distributions of the data instances in the following 4 subspaces conform with the hypotheses or not:

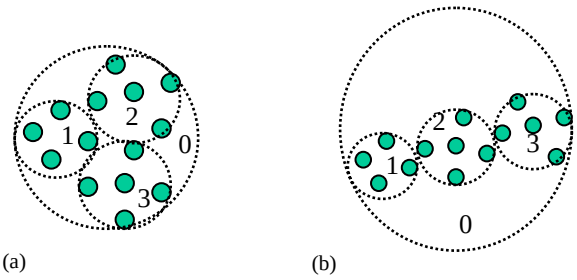

Fig. 5. An example illustrating the statistical test of spherical clusters. (a) The node will pass the spherical test. (b) The node will not pass the spherical test.

(1) The subspace enclosed by the sphere of subcluster 1.
(2) The subspace enclosed by the sphere of subcluster 2.
(3) The subspace enclosed by the sphere of subcluster 3.
(4) The subspace enclosed by the sphere of the parent cluster but outside subclusters 1, 2, and 3.

The parent cluster is said to be in the spherical shape, if

$$\frac{(k_1 - V_1\mu_0)^2}{V_1\mu_0} + \frac{(k_2 - V_2\mu_0)^2}{V_2\mu_0} + \frac{(k_3 - V_3\mu_0)^2}{V_3\mu_0} + \frac{[0 - (V_0 - V_1 - V_2 - V_3)\mu_0]^2}{(V_0 - V_1 - V_2 - V_3)\mu_0} \leqslant \chi^2_\alpha, \quad (3)$$

where k_i is the number of points in subcluster i, V_i is the subspace enclosed by cluster i, $\mu_0 = (k_1 + k_2 + k_3)/V_0$ and χ^2_α is a threshold for the χ^2 distribution of 3 degrees of freedom. In general, if the parent cluster of concern contains m spherical subclusters, then the χ^2 test with m degrees of freedom is conducted. In the statistical test, if one of the child subclusters contains only one single data instance, then the radius of the child cluster is defined to be one half of the distance between the instance and its nearest neighbor among the instances in the parent cluster. Note that applying the χ^2 goodness of fit in identifying spherical clusters is an empirical mechanism in some sense, as inequality (3) above assumes that the spheres of subclusters 1, 2, and 3 do not overlap and each of them is completely enclosed by the sphere of the parent cluster.

After all the nodes in the dendrogram are examined by the statistical test, the next operation performed in the summarization process is to flatten the bottom levels of the clustering dendrogram in order to derive the so-called *abstract dendrogram*. In the flattening process, a spherical cluster in the original dendrogram will become a *leaf cluster* in the dendrogram, if it satisfies both of the following conditions:

(1) The cluster is of spherical shape according to the statistical test.

(2) All of its ancestors do not satisfy condition (1). The ancestors of a node are defined as the nodes on the path from the root of the dendrogram to the current node.

The structure under a leaf cluster in the original dendrogram is then flattened so that all the data instances under the leaf cluster become its children. An example of the summarization process will be given in the next section.

4. The GRIN algorithm

This section describes the GRIN algorithm. The GRIN algorithm operates in two phases, initial phase and incremental phase. In both phases, it invokes the gravitational agglomerative hierarchical clustering algorithm presented in Section 2 to construct clustering dendrograms. Fig. 6 shows the flowchart and Fig. 7 provides the pseudo-code of GRIN algorithm.

4.1. Initial phase

In the GRIN algorithm, it is assumed that all the incoming data instances are first buffered in an *incoming data pool*. In the initial phase of the algorithm, the first *s* data instances in the incoming data pool are collected and the G-HAC algorithm, the gravity-based agglomerative hierarchical clustering algorithm described in Section 2, is invoked to build a clustering dendrogram for these instances. After the G-HAC algorithm terminates, the summarization process presented in Section 3 is invoked to identify the spherical subclusters and flatten the dendrogram. The parameter *s* can be set by the users under the consideration that *s* is the input size of a HAC algorithm. In fact, the initial phase could be omitted entirely if an abstract dendrogram is available from the previous run.

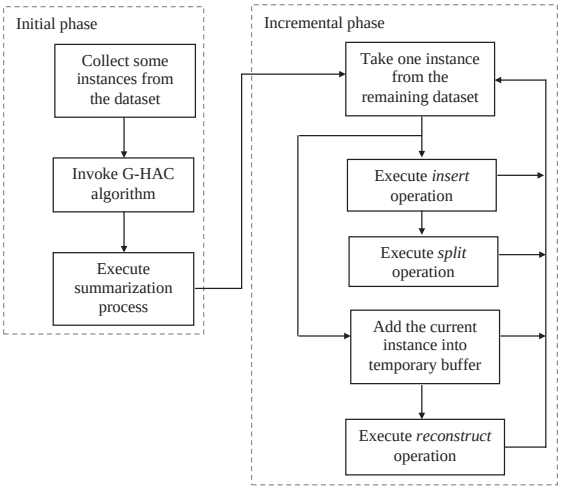

Fig. 6. The flowchart of the GRIN algorithm.

Fig. 8 demonstrates an example that illustrates the operations carried out by the initial phase. Fig. 8(a) shows a dataset and Fig. 8(b) shows the first 100 data instances in the dataset. Fig. 8(c) depicts the dendrogram built based on these first 100 data instances by G-HAC. In Fig. 8(c), the clusters that pass the criterion of spherical cluster are marked by "*". Fig. 8(d) shows the dendrogram derived from flattening the dendrogram depicted in Fig. 8(c).

Each leaf spherical cluster in the abstract dendrogram is represented by three features: the centroid, the radius, and the mass of the cluster. The radius of a cluster is defined as the maximum distance between the centroid and the data instances in this cluster. The mass of a cluster is defined to be the number of data instances that it contains. In other words, it is assumed that the mass of each single data instance is equal to unity. In addition, to facilitate the procedure of conducting the proposed statistical test on a leaf spherical subcluster in the incremental phase of GRIN algorithm, we accumulate the number of instances in each subspace defined by the child spherical clusters.

4.2. Incremental phase

In the second phase of the GRIN algorithm, the not-yet clustered instances in the incoming data pool are examined one by one. Let the new coming data instance move in the space based on the gravity theory, provided that all the leaf spherical clusters are fixed. Like the G-HAC algorithm, GRIN iteratively simulates the movement of the new instance during a time interval and stops when the new data instance falls in the sphere enclosed by its nearest leaf spherical cluster in the current iteration.

4.2.1. Insert operation

If the distance between the new data instance and its nearest leaf spherical cluster in the final iteration is smaller than the radius of the leaf spherical cluster, the new data instance is inserted into that leaf spherical cluster. On the other hand, if the distance between the new data instance and the leaf spherical cluster that absorbs it is larger than the radius of the leaf spherical cluster, the statistical test is conducted first to determine whether this leaf cluster is still qualified to be a spherical cluster given the data instance were added into the cluster. If yes, then the data instance can be added into that leaf cluster. If no, the data instance is treated as an outlier to the abstract dendrogram and will be put into the *temporary buffer*. The data instance, however, may form a cluster with other data instances that are already in the temporary buffer or that come in later.

4.2.2. Split operation

The GRIN algorithm checks whether a leaf spherical cluster should be split whenever a new instance has been added into it. For large datasets, this procedure could be hold until the number of new data instances added into the leaf spherical cluster exceeds one half of the number of original data

```
Procedure GRIN
Input: I : an input buffer
       T : an abstract dendrogram
{
    /* Phase I: Initial phase */
    If (T is null) {
        Let S = {p | some data instances in I};
        I = I – S;
        T = G-HAC(S);
        Apply the summarization process to T;
    }
    /* Phase II: Incremental phase */
    While (I is not empty) {
        Take a new instance p from I;
        Repeat {
            Let p move in the space based on the gravity theory, provided that all the spherical clusters
            are fixed.
        } until (a leaf spherical cluster c absorbs p)
        Let Dis be the distance between p and the centroid of c.
        If (Dis is smaller than the radius of c) {
            /* insert operation */
            Add instance p as a member of the leaf spherical cluster c.
            If ( c is not satisfied as a spherical cluster any more) {
                /* split operation */
                S = {n | all the instances in leaf spherical cluster c}
                T' = G-HAC(S);
                Apply the summarization process to T' ;
                Replace the leaf cluster c in T with the sub abstract dendrogram T' ;
            }
        }
        Else if (c is still satisfied as a spherical cluster after absorbing p) {
            Add instance p as a member of the leaf spherical cluster c;
        }
        Else {
            Add p into the temporary buffer TempBuffer;
        }
        Remove p from I;
        If (TempBuffer is full ) {
            /* reconstruct operation */
            S = {n | all the leaf spherical clusters in the dendrogram T}; Each node n is summarized by
            its centroid, radius, and the number of data instances it contains.
            T = G-HAC(S ∪ TempBuffer).
            Apply the summarization process to T;
        }
    }
    Return T;
}
```

Fig. 7. The pseudo-code of the GRIN algorithm.

instances. To do that, the statistical test will be conducted to verify whether the cluster is still satisfied as a spherical cluster. If not, a *split* operation will be executed on this cluster. The split operation is performed by executing the G-HAC algorithm to cluster all the data instances inside the cluster, executing the summarization process to construct a sub abstract dendrogram, and finally replacing the current cluster in the abstract dendrogram with the new sub abstract dendrogram.

4.2.3. Reconstruct operation

Once the number of data instances in the temporary buffer exceeds a threshold, the G-HAC algorithm described in Section 2 is invoked to construct a new abstract dendrogram. In this reconstruction process, the primitive objects are the leaf spherical clusters in the current abstract dendrogram and the data instances in the temporary buffer, where a leaf spherical cluster is treated as an object located at the centroid of it and with the mass equal to the number of data instances it contains. After new dendrogram is generated, the same summarization process invoked in the initial phase is conducted to generate a new abstract dendrogram.

4.2.4. Time and space complexities

The time complexity of the first-phase is a constant, as long as the number of data instances taken in the first phase

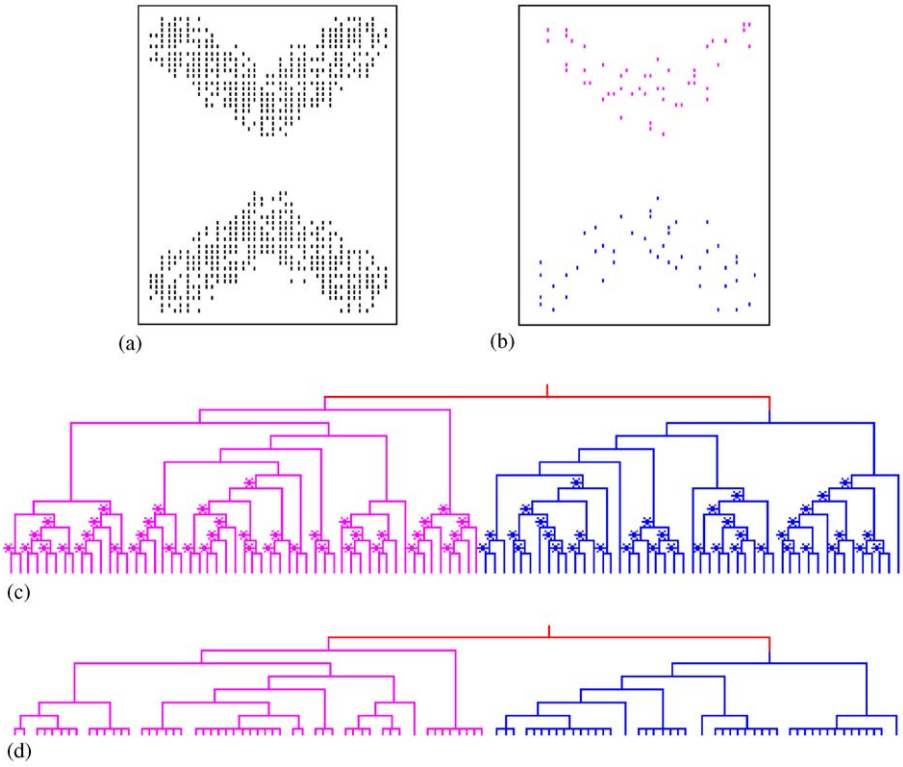

Fig. 8. An example employed to demonstrate the operations of the GRIN algorithm. (a) The dataset containing 1500 data instances. (b) The first 100 instances of the dataset in (a). (c) The dendrogram built based on the 100 instances in (b). (d) The flattened dendrogram derived from (c).

is not a function of the size of the dataset. As to the second phase, the time taken to process each incoming data instance is $O(m)$, where m is the number of leaf clusters kept by the algorithm. The time taken to issue an *insertion* operation is constant. Given that the maximum data instances in a leaf cluster is q, the time taken to issue a *split* operation is $O(q^2)$ because the time complexity of G-HAC is $O(n^2)$ [20]. Similarly the time taken to reconstruct the dendrogram in the incremental phase is $O(m^2)$, provided that the size of the temporary buffer is fixed. Thus, the time complexity of the GRIN algorithm is $O(nm + a + bq^2 + cm^2)$, where n is the number of instances in the dataset, and a, b, and c are the number of times that *insert*, *split*, and *reconstruct* operations occur, respectively. One important observation in the experiments regarding the operations performed the incremental phase is that, $b \ll n, c \ll n$, and a is bounded by n. Given this observation, the time complexity of GRIN algorithm is $O(nm)$. Even though the value of m is related to the number of data instances n, our experimental results show that the ratio of m to n is getting smaller when n increases. On the other hand, the space complexity of the GRIN algorithm is $O(m)$ because the size of the abstract dendrogram is bound by the number of leaf subclusters.

5. Experiments

This section reports the experiments conducted to study the following three issues concerning the GRIN algorithm:

(1) How the GRIN algorithm compares with the BIRCH algorithm in terms of clustering quality.
(2) Whether the clustering quality of GRIN algorithm is influenced by the order of input data?
(3) How the GRIN algorithm performs in terms of execution time in real applications?

Table 1 shows how the parameters in the GRIN and G-HAC algorithms are set in the experiments. The values of parameters in G-HAC algorithm are selected according to the suggestions in [20,21]. The parameter *Min* is set to 3 because we observed that the proposed statistical test performs well on clusters containing more than 3 data instances. Similarly the significance of the χ^2 test, α, is set to 0.01 after a large number of experiments have been performed on several skewed datasets for different values of α from 0.1 to 0.001. The parameter s as well as the size of temporary buffer could be set by the users. As mentioned before, these parameters should

Table 1
Parameter settings of the GRIN and G-HAC algorithms in the experiments

G-HAC	
k: Order of the distance term in gravity force formula	10
C_r: Coefficient of air resistance	10000
GRIN	
s: number of instances used in the first phase	500
size of temporary buffer	500
Min	3
α: significance of the χ^2 test	0.01

be set under the consideration that they directly or indirectly determine the input size of the G-HAC algorithm.

Fig. 9(a) shows a dataset used in the experiments. In Fig. 9(a), natural clusters are annotated according to human's intuition. Fig. 9(b) depicts the clusters identified by the GRIN algorithm and the dendrogram constructed. In this experiment, data is fed to the GRIN algorithm one natural cluster by another natural cluster in the following order $1 \rightarrow 2 \rightarrow 3 \rightarrow \cdots$. For providing better visualization quality, only the clusters at the top few levels of the dendrogram are plotted. The result in Fig. 9(b) shows that the GRIN algorithm is able to identify natural clusters flawlessly. Several runs of the GRIN algorithm were executed with the data fed to the algorithm in different orders. In one particular run, the data was fed to the algorithm also one natural cluster by another natural cluster but in the reverse order. In the remaining runs, data was fed to the algorithm randomly. The outputs of all these separate runs of the GRIN algorithm are basically identical to what is depicted in Fig. 9(b). The experiment results show that the clustering quality of the GRIN algorithm is irrelevant to the order of input data.

Fig. 9(c) and (d) depict the data clusters identified by the BIRCH algorithm with different hierarchical clustering algorithms incorporated. The BIRCH algorithm is employed for comparison, because it is a well-known incremental hierarchical clustering algorithm that features $O(n)$ time complexity. In Fig. 9(c), we enlarge the portion of the dataset in which the BIRCH algorithm with the complete-link algorithm incorporated fails to deliver reasonable clustering quality, and data instances belonging to different clusters are marked by different symbols. In this case, the BIRCH algorithm mixes the data instances from natural clusters 2, 3, and 4. BIRCH's imperfection is due to two factors. First, BIRCH uses a distance threshold to determine the leaf subclusters. Therefore, as exemplified in Fig. 2, it could occur that no optimal value for this parameter can be found when the local distributions of the dataset are skewed. Second, the complete-link algorithm itself suffers bias towards spherical clusters [20]. Fig. 9(d) reveals that the clustering quality of BIRCH is improved when the G-HAC algorithm is incorporated instead of the complete-link algorithm. The G-HAC algorithm contributes to the improvement of clustering quality, because the complete-link algorithm suffers bias towards spherical clusters in a much higher degree than the G-HAC algorithm [20]. Nevertheless, there are still a few flaws in Fig. 9(d) due to the poor parameter chosen within the BIRCH algorithm.

Fig. 10(a) depicts the clusters identified by the GRIN algorithm and the dendrogram constructed for another dataset. Fig. 10(b) shows the clusters outputted by the BIRCH algorithm with the G-HAC algorithm incorporated. Several flaws are observed as marked by the squares. Fig. 11(a) and (b) show the leaf subclusters identified by the GRIN algorithm and the BIRCH algorithm, respectively. In Fig. 11(b), more leaf subclusters are used to summarize the same dataset when compared with Fig. 11(a), and some flaws happened in the boundary of natural clusters. Again, in order to test the sensitivity of the GRIN algorithm to the input ordering, several runs of the GRIN algorithm were executed with various arriving orders.

In the third experiment, we first use a subset of the Sequoia 2000 benchmark [24] to test how the GRIN algorithm performs while dealing with a real application. The subset contains the locations of all the high schools in California. Fig. 12(a) plots the 989 location instances in the subset. Fig. 12(b) depicts the outlook of the remaining 946 schools after outliers are removed manually. Fig. 12(c) shows the clusters outputted by the GRIN algorithm, where different clusters are plotted using different symbols. We also use the dataset shown in Fig. 12(b) to test the how the BIRCH algorithm performs when operating with the complete-link algorithm and the G-HAC algorithm, respectively. Here, we would like to emphasize the right bottom section of the results. As shown in both Fig. 12(d) and (e), the largest natural cluster (marked by X) in Fig. 12(c) is divided into two parts and the left part is clustered with the location instances further to the left.

Two other real datasets from UCI Machine Learning Repository [25] are further used to test the capability of GRIN in summarizing data incrementally. The first one is the famous Iris Plant data, a small dataset that contains 150 instances with four attributes. And the second dataset is a larger one, the Image Segmentation data from the Statlog Project. The Image Segmentation data contains 2310 instances and originally has 19 attributes. Here we organized it into a reduced dataset with only four attributes because the proposed statistical test is not able to handle that high dimensionality of the feature set. The selected features show their strong correlation with the categorization of data instances and are supposed to be non-redundant to each other. The UCI datasets are considered for comparison because they provide categorization for each data instance that can be used to judge the clustering results objectively. Several clustering algorithms are employed in the experiment to

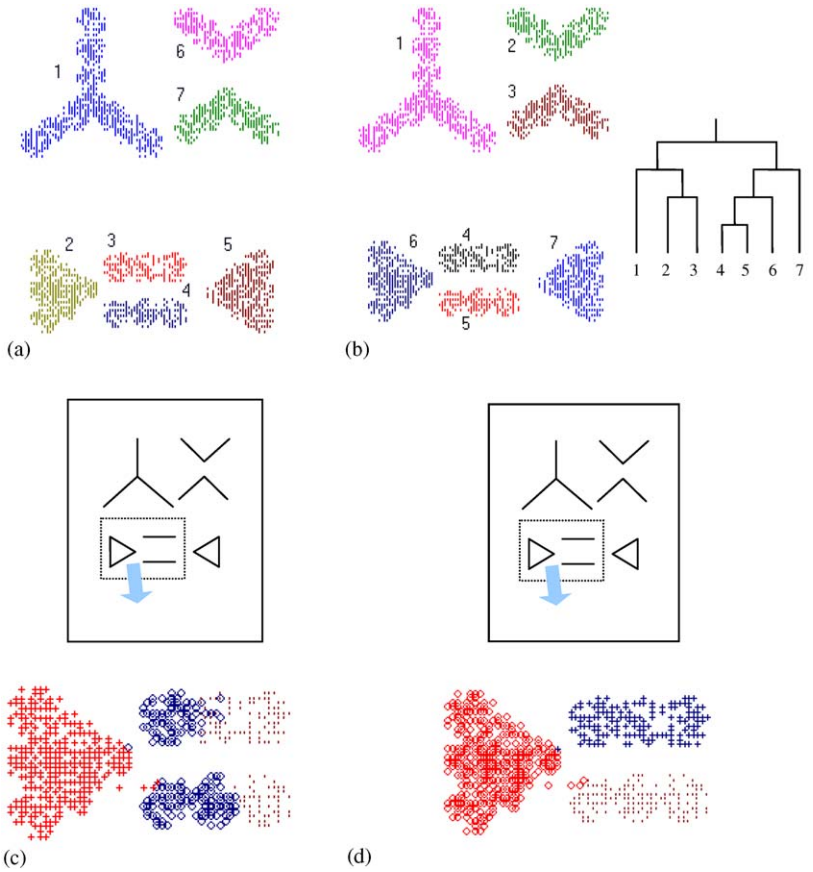

Fig. 9. The first experiment conducted to evaluate clustering quality. (a) A dataset containing 3000 data instances. (b) The clusters identified by the GRIN algorithm and the dendrogram constructed. (c) Flaws in the output of the BIRCH algorithm with the complete-link algorithm incorporated. (d) Flaws in the output of the BIRCH algorithm with the G-HAC algorithm incorporated.

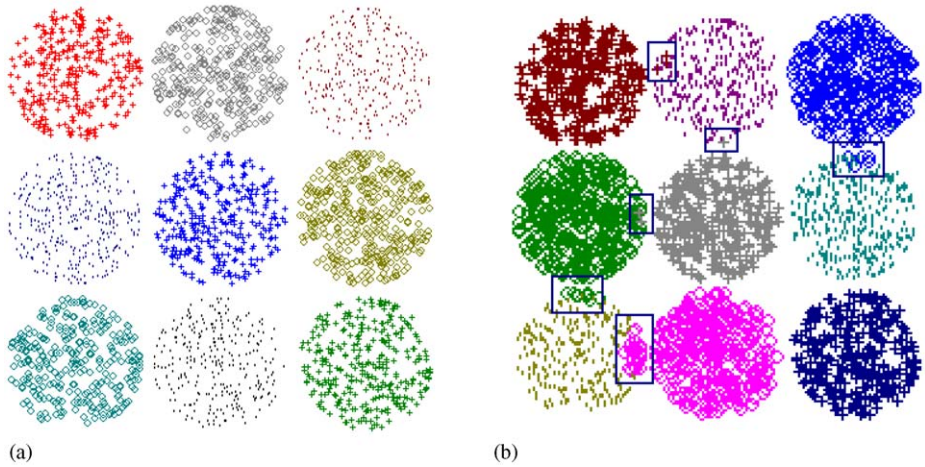

Fig. 10. The second experiment conducted to evaluate clustering quality. (a) The clusters identified by the GRIN algorithm and the dendrogram constructed. (b) Clusters identified by the BIRCH algorithm with the G-HAC algorithm incorporated.

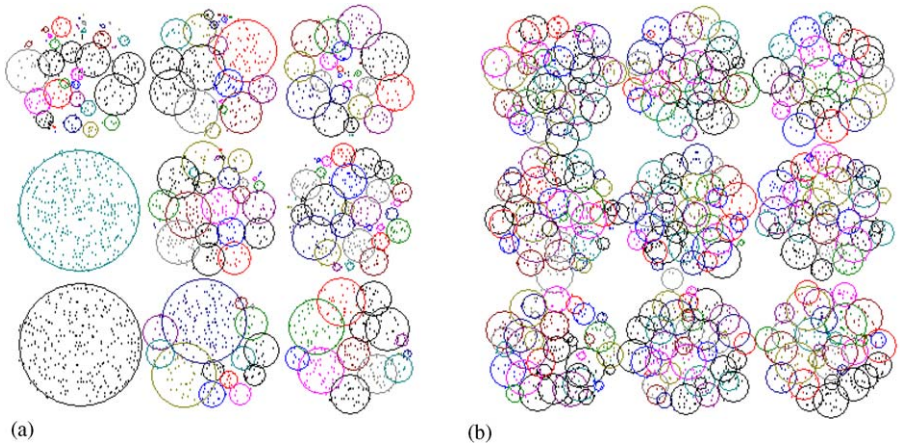

Fig. 11. The spheres identified by BIRCH overlap with each other more seriously than the spheres identified by the GRIN algorithm. Some spheres identified by BIRCH cross through the regions of low densities. (a) 180 leaf spherical subclusters identified by the GRIN algorithm. (b) 367 leaf subclusters identified by the BIRCH algorithm.

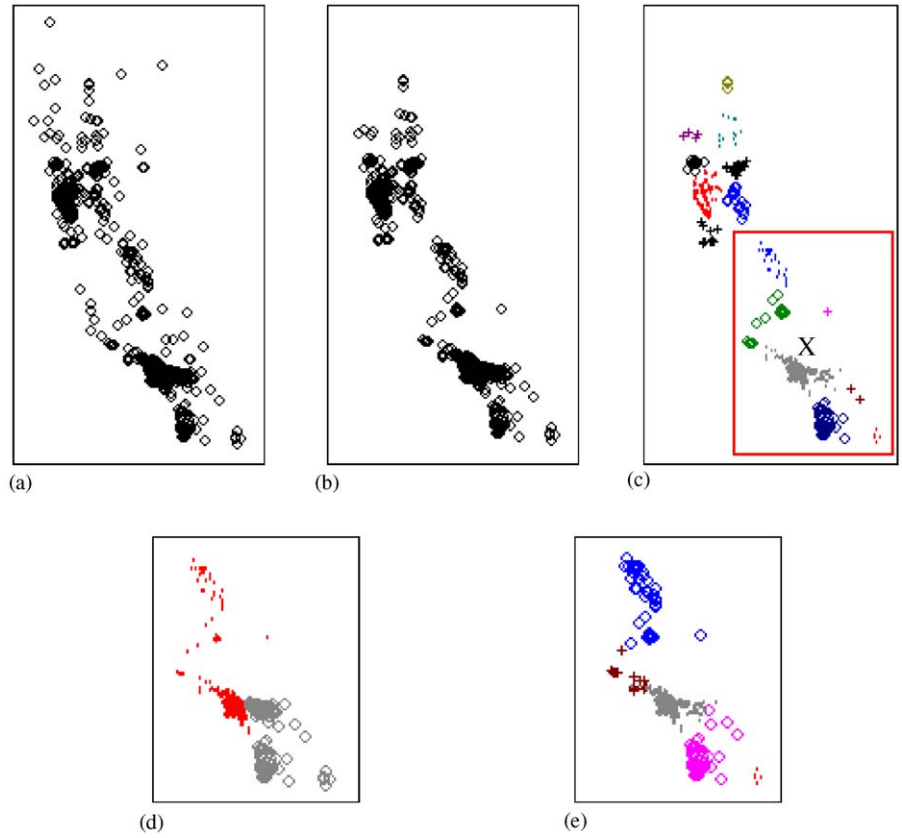

Fig. 12. Experiment conducted to show how the GRIN algorithm performs with real datasets. (a) A subset of the Sequoia 2000 benchmark. (b) The dataset after outliers are removed by GRIN. (c) Clusters outputted by GRIN. (d) The clustering result outputted by BIRCH when the complete-link algorithm is incorporated. (e) The clustering result outputted by BIRCH when the G-HAC algorithm is incorporated.

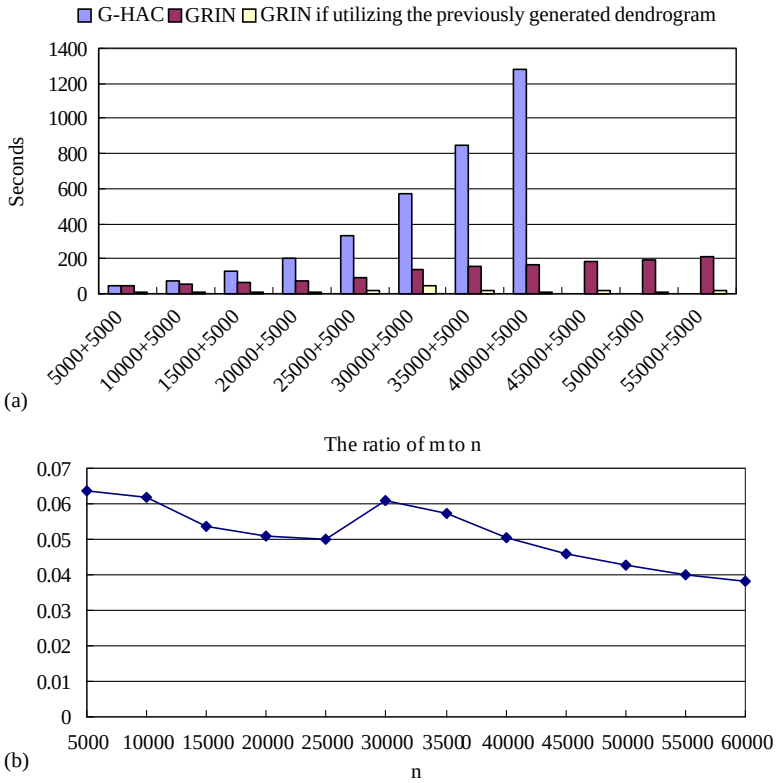

Fig. 13. Experiment conducted to test the performance of the GRIN algorithm. (a) The performance of the GRIN algorithm as the number of instances in the dataset increases. (b) The behavior of the GRIN algorithm as the number of instances in the dataset increases.

Table 2
Comparison of the capability in identifying natural subclusters on two real datasets from UCI Machine Learning Repository [25]

Clustering algorithm	SLINK	CLINK	ALINK	K-means[a]	LEADER[b]	BIRCH[b]	GRIN[b]
Number of flaws for the Iris dataset (150 *instances*, 4 *attributes*, 3 *classes*)							
Number of cluster = 25	32	7	9	7.6 ± 1.51	11.2 ± 1.21	7.2 ± 1.46	4.6 ± 1.48
Number of cluster = 30	27	5	6	7.9 ± 1.52	9.0 ± 1.73	6.8 ± 1.34	4.6 ± 1.48
Number of flaws for the Image Segmentation dataset (2310 *instances*, 4 *attributes*, 7 *classes*)							
Number of cluster = 200	479	177	187	204.4 ± 16.5	243.2 ± 10.92	202.2 ± 11.6	197.6 ± 9.86
Number of cluster = 250	448	164	156	181.6 ± 8.15	201.8 ± 10.71	192.6 ± 9.53	173.6 ± 8.96

[a]Each record of K-means algorithm is the average (followed by the standard deviation) of ten runs with different initial centers.
[b]In the experiment of incremental clustering algorithms, the original dataset was shuffled randomly to generate five datasets with different orders. Each record in the table is the average (followed by the standard deviation) of the five runs.

generate a set of clusters that summarize the original dataset. An instance is marked as a flaw if its class attribute is different to the majority of the cluster it belongs to. Table 2 shows that GRIN algorithm delivers slightly better clustering results than the BIRCH algorithm on these datasets, where BIRCH is incorporated with the complete-link algorithm. GRIN also outperforms the K-means algorithm [3] and the LEADER algorithm [4], a distance-based incremental clustering algorithm. The parameter of LEADER is set to produce about the same number of clusters with the other clustering algorithms. We also provide in Table 2 the clustering results of three traditional hierarchical clustering algorithms, single-link algorithm (abbreviated as SLINK in Table 2), complete-link algorithm (abbreviated as CLINK), and average-link algorithm (abbreviated as ALINK). The results in Table 2 reveal that GRIN algorithm even performs better than the non-incremental hierarchical clustering algorithms on the Iris dataset due to the summarization process

Table 3
The accumulated number of times that *insert*, *split*, and *reconstruct* operation occurred versus the number of data instances in the dataset

n	# of *insert* operation	# of *split* operation	# of reconstruct operation
5000	3312	64	1
10000	7529	119	1
15000	11824	158	1
20000	16246	207	1
25000	20659	246	1
30000	25085	286	1
35000	29534	323	2
40000	33941	324	2
45000	38451	336	2
50000	42959	351	2
55000	47574	379	2
60000	52234	414	2

it employs. However, as shown in Table 2, the incremental approaches do degrade the performance a little bit on the Segmentation dataset.

Fig. 13(a) shows how the execution time of the GRIN algorithm increases with the number of data instances in the dataset. The experiment was conducted on a machine equipped with a 1 GHz Intel Pentium-III CPU and running Microsoft Window 2000 operating system. The dataset used is the point data in Sequoia 2000 Earth benchmark [24], which contains 62556 data instances in total. The experiment is conducted by executing the clustering process whenever there have been 5000 instances added into the dataset. The results in Fig. 13(a) reveal that the GRIN algorithm generally features linear scalability. Fig. 13(b) provides the ratio of the number of leaf subclusters to the number of data instances in the dataset. Table 3 shows that the *split* and *reconstruct* operation happens much less frequently when compared with the *insert* operation (Tables 2 and 3).

6. Conclusion

This paper presents the GRIN algorithm, an incremental hierarchical clustering algorithm based on gravity theory in physics. The incremental nature of the GRIN algorithm implies that it is particularly suitable for handling the already huge and still growing databases in modern environments. In addition, its hierarchical nature is a highly desirable estate for many applications in biological, social, and behavior studies due to the need to construct taxonomies. In GRIN algorithm, the leaf subclusters are determined according to a statistical test. The derived spherical subclusters are the basis in further clustering process and will be updated dynamically. Employing the proposed statistical test as well as invoking the *split* and *reconstruct* operations based on the testing results can alleviate the damage caused from ill input ordering. Furthermore, the proposed summarization process decomposes the original dataset into several spherical clusters, which can provide a good representation for clusters in arbitrary shape.

In this paper, we demonstrate that the idea of decomposing a distribution into several spherical clusters with different densities works well on low-dimensional datasets. However, further studies are required to determine whether the same assumption and approach work for high-dimensional datasets. Though currently, the G-HAC algorithm works well on the datasets studied in this paper, we would like to see whether the proposed summarization process could be incorporated with other HAC algorithms for different applications.

7. Summary

One of the main challenges in the design of modern clustering algorithms is how to cope with new data instances that are continuously added into an already huge database. The importance of incremental clustering algorithms arises as a result, since it is impractical to carry out data clustering from scratch whenever there are new data instances added into the database. An incremental clustering algorithm refers to an abstraction of the distribution of the data instances generated by the previous run of the algorithm and therefore is able to deliver reasonable efficiency. This paper presents the GRIN algorithm, an incremental hierarchical clustering algorithm for numerical datasets based on the gravity theory in physics. Due to its hierarchical nature, GRIN outputs a clustering dendrogram, which is a highly desirable feature for some applications due to the need to create taxonomies. The main distinction in the design of the GRIN algorithm is how abstraction (or summarization) of the distribution of the dataset is carried out. Abstraction (or summarization) is an essential operation for incremental clustering algorithms. The main challenge in this regard is that information loss and distortion may result. In the design of GRIN, a statistical test aimed at reducing information loss and distortion is employed to control formation of subclusters as well as to monitor the evolution of the dataset. Due to the statistical test-based summarization approach, GRIN is able to achieve near linear scalability and is not sensitive to input ordering.

References

[1] R.J. Brachman, T. Anand, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), The Process of Knowledge Discovery in Databases, Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, MA, 1996, pp. 37–57.
[2] B. Everitt, Cluster Analysis, Halsted Press, New York, 1980.

[3] A.K. Jain, R.C. Dubes, Algorithms for clustering data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[4] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323.
[5] M.N. Murty, G. Krishina, A computationally efficient technique for data clustering, Pattern Recognition 12 (1980) 153–158.
[6] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco, 2000.
[7] J. Han, M. Kamber, A.K.H. Tung, Spatial clustering methods in data mining: a survey, in: H. Miller, J. Han (Eds.), Geographic Data Mining and Knowledge Discovery, Taylor and Francis, London, 2001.
[8] I.H. Witten, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, San Francisco, CA, 2000.
[9] J.A. Hartigan, Clustering Algorithms, Wiley, New York, 1975.
[10] D. Fisher, Improving inference through conceptual clustering, Proceedings of the sixth National Conference on Artificial Intelligence (AAAI-87), 1987, pp. 461–465.
[11] J. Gennari, P. Langley, D. Fisher, Models of incremental concept formation, Artif. Intell. 40 (1989) 11–61.
[12] M. Charikar, C. Chekuri, T. Feder, R. Motwani, Incremental clustering and dynamic information retrieval, Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC-97), 1997, pp. 626–634.
[13] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu, Incremental clustering for mining in a data warehousing environment, Proceedings of the 24th International Conference on Very Large Data Bases (VLDB-98), 1998, pp. 323–333.
[14] V. Ganti, J. Gehrke, R. Ramakrishnan, DEMON: mining and monitoring evolving data, IEEE Trans. Knowledge Data Eng. 13 (1) (2001) 50–63.
[15] A. Ribert, A. Ennaji, Y. Lecourtier, An incremental hierarchical clustering, Proceedings of the 1999 Vision Interface Conference, 1999, pp. 586–591.
[16] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, Proceedings of 1996 ACM-SIGMOD International Conference on Management of Data (SOGMOD-96), 1996, pp. 103–114.
[17] L. Talavera, J. Bejar, Using multistrategy learning to scale up hierarchical clustering algorithms, Proceedings of the Second International Workshop on Extraction of Knowledge from Data Bases (EKDB-99), 1999.
[18] W.E. Wright, Gravitational clustering, Pattern Recognition 9 (1977) 151–166.
[19] S. Kundu, Gravitational clustering—a new approach based on the spatial distribution of the points, Pattern Recognition 32 (1999) 1149–1160.
[20] Y.-J. Oyang, C.-Y. Chen, T.-W. Yang, A study on the hierarchical data clustering algorithm based on gravity theory, Proceedings of the Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-01), 2001, pp. 350–361.
[21] Y.-J. Oyang, C.-Y. Chen, S.-C. Hwang, C.-F. Lin, Characteristics of a Hierarchical Data Clustering Algorithm Based on Gravity Theory, Technical Report of NTUCSIE 02-01, 2001 (available at http://mars.csie.ntu.edu.tw/∼cychen/publications_on_dm.htm).
[22] T.V. Ravi, C.K. Gowda, Clustering of symbolic objects using gravitational approach, IEEE Trans. Syst. Man Cybern.—Part B: Cybernetics 29 (6) (1999) 888–894.
[23] R.V. Hogg, E.A. Tanis, Probability and Statistical Inference, Prentice-Hall, Englewood Cliffs, NJ, 2001.
[24] M. Stonebraker, J. Frew, K. Gardels, J. Meredith, The Sequoia 2000 storage benchmark, Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data (SIGMOD-93), 1993, pp. 2–11.
[25] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, Technical report, University of California, Department of Information and Computer Science, Irvine, CA, 1998.

About the Author—CHIEN-YU CHEN received the B.S. degree in Electrical Engineering from National Taiwan University in 1996, the M.S. degree in Electrical Engineering from the Stanford University in 1998, and the Ph.D. degree in Computer Science and Information Engineering from National Taiwan University in 2003. She is currently an Assistant Professor in the Graduate School of Biotechnology and Bioinformatics, Yuan Ze University. Her research interests include bioinformatics, data mining and machine learning.

About the Author—SHIEN-CHING HWANG received the B.S. degree in Mathematics from Fu Jen Catholic University, Taiwan, in 1993, and the M.S. and Ph.D. degrees in computer science and information engineering from National Taiwan University, Taiwan, in 1995 and 2000, respectively. He is currently an Assistant Professor in the Department of Information Science and Applications, Taichung Healthcare and Management University, Taiwan. His research interests include machine learning, data mining and bioinformatics.

About the Author—YEN-JEN OYANG received the B.S. degree in Information Engineering from National Taiwan University in 1982, the M.S. degree in Computer Science from the California Institute of Technology in 1984, and the Ph.D. degree in Electrical Engineering from Stanford University in 1988. He is currently a Professor in the Department of Computer Science and Information Engineering, National Taiwan University. His research interests include data mining/machine learning and bioinformatics.