Knowledge Media Institute

Hierarchical clustering speed up using position lists and data position hierarchy

Jiri Komzak

KMI-TR-115

June, 2001

www.kmi.open.ac.uk/papers/kmi-tr-115.pdf

The Open University

Hierarchical clustering speed up using position lists and data position hierarchy

Jiri Komzak

Abstract

The aim of this paper is to address the nature of hierarchical clustering problems in systems with very large numbers of entities, and to propose specific speed improvements in the clustering algorithm. The motivation for this theme arises from the challenge of visualising the geographic and logical distribution of many tens of thousands of distance-learning students at the UK's Open University. A general algorithm for solving hierarchical clustering is mentioned at the beginning. Then the paper describes (i) a speed-up technique based on lists sorted according to particular dimensions or attributes of the entities to be visualised and (ii) a speed-up technique based upon hierarchical partitioning into regions. At the end, the paper discusses the algorithm's complexity and presents experimental results.

Keywords

hierarchical clustering, position hierarchy, position list, geographical information system

Introduction

The UK's Open University is one of the world's largest Universities, with over 160,000 currently-enrolled distance-learning students distributed throughout the world. It has been described a 'Mega-University' [1] and as such takes particular care with regard to the extensive support mechanisms required for dealing with students at a distance. This support includes a robust enrolment and record-tracking system, as well as numerous custom online support tools for managing over 16,000 separate student discussion forums (see, [2, 3] for descriptions of the Open University's online mentoring support, tools, and related experiments).

In this context, we have begun to develop prototype support tools for visualising the geographical location of students for both offline demographic analysis and online 'presence' and instant messaging support (see [4] for up-to-date presence/messaging discussions). With over 160,000 students, and our interest in real-time updating for presence/messaging support, any such system faces significant scalability issues: many thousands of entities have to be displayed, and symbols representing entities will overlap (especially in long-distance views) and cause loss of information. The standard approach in cartography and Geographical Information Systems (GIS) such as SAS [5], MapInfo [6] and MapPoint [7] is to cluster the symbols according to some accepted convention (e.g. large dot represents 1,000 students, or color-shaded regions represent >N students, etc.). The challenge facing us is two-fold: (i) to perform such clustering rapidly and in nearly-real-time, and (ii) to allow this clustering to take place dynamically depending upon the ‘zoom level’ of the map display. Hierarchical clustering of entities addresses these challenges, reducing the number of symbols by grouping entities and representing them in clusters. Then, each cluster is displayed as a symbol with a parameter (for example symbol size) determining the number of entities included in the cluster.

At the beginning of this paper, the basic principle of hierarchical clustering is described. The next two sections present techniques for speeding up the clustering algorithm. This can be achieved by reduction of number of the potential closest neighbours. The first method is based on lists sorted following particular dimensions and the second one uses hierarchical partitioning into regions. At the end of paper, the complexity is discussed and experimental results are presented.

Hierarchical clustering principle

A hierarchical structure completely describing relationships between entities can be constructed by a technique called *hierarchical clustering*. Then any suitable distance threshold can be chosen for a specific situation. The hierarchical clustering groups contiguous entities in some defined way, and thus can be used for statistical analysis as well.

The bases for clustering are logical or spatial distances or both (for example close cities in the same country). The distance has to be defined for any two entities and describes the relationship between them.

The basic algorithm can be described following way [8]:

INPUT: m objects with $\frac{m(m-1)}{2}$ pairs of distances, $\{d_{i,j} \mid 1 \le i \le j \le m\}$. It is assumed that self-distances are all 0 and $d_{i,j} = d_{j,i}$.

STEP 1: Find the smallest distance $d_{i,j}$, $i < j$.

STEP 2: Agglomerate objects i and j. That is, i and j are replaced by a new object, denoted by $i \cup j$. The distance of this new object from the remaining m-2 objects is defined to be $d_{i \cup j,k} = f(d_{i,j}, d_{i,k}, d_{j,k})$, where f is a function described bellow. Then distances $d_{i,k}$ and $d_{j,k}$ for all possible k are deleted.

STEP 3: The new object $i \cup j$ is renamed to i, and j disappears. The use of the smaller index is arbitrary. The new object, i, is called a cluster, and we denote by $|i|$ the number of primary objects in this cluster. We also allow single objects to be called clusters (of size 1).

STEP 4: If 2 or more clusters remain, repeat steps 1-3.

Steps 1 through 3 are applied m-1 times before the algorithm terminates.

The function f can be defined one of these ways:

- Single link – nearest neighbour: $d_{i\cup j,k} = \min\{d_{i,k}, d_{j,k}\}$
- Complete link – diameter: $d_{i\cup j,k} = \max\{d_{i,k}, d_{j,k}\}$
- Average link – nearest neighbour: $d_{i\cup j,k} = \frac{|i|d_{i,k}+|j|d_{j,k}}{|i|+|j|}$
- McQuitty's method: $d_{i\cup j,k} = \frac{d_{i,k}+d_{j,k}}{2}$

The methods of distance calculation differ in their meaning. *Single link* defines the distance from cluster as the distance from its closest entity. In contrast, *Complete link* counts it as the distance from the furthest entity of the cluster. The distance of the *average link* method is based on the centre of gravity of the created cluster. Finally, McQuitty's method just averages the distances from both parts of the new cluster.

The *average link* method is most suitable for the purposes of the designed system, because it uses the centre of gravity of the cluster, into which the created cluster is then displayed.

Position lists

The basic algorithm mentioned in the preceding section looks for the two closest clusters using a brute force approach. It always counts all distances and compares all of them.

Since this seemed to be unnecessary, the development of a new algorithm was started. The new algorithm is based on a heuristic for reduction of the number of tested clusters. Since elimination of distant entities is needed, it is suitable to sort entities according to their positions. The heuristic uses one sorted list for each dimension (e.g. two lists for a 2D map).

The algorithm works in several steps. At first, it creates one sorted list for each dimension. In the second step, the 'close neighbours lists' for each cluster is initialised. This initialisation is the most important part of the algorithm and is discussed in more detail below.

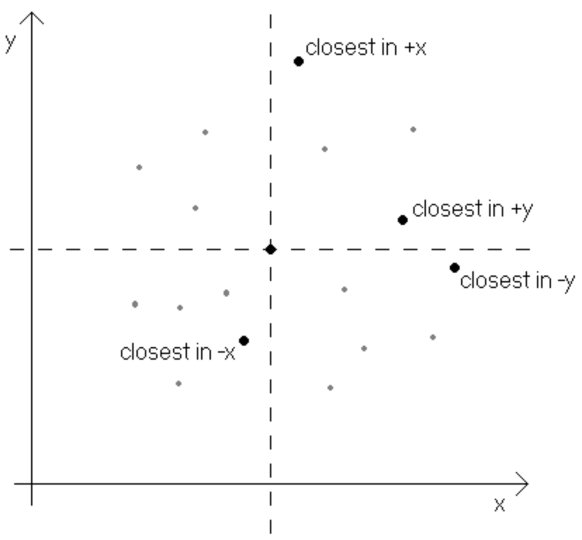

Fig. 1: Closest neighbours in lists

At first, the closest neighbours of the cluster in the lists are taken (there are 4 such clusters for 2D - see Fig. 1).

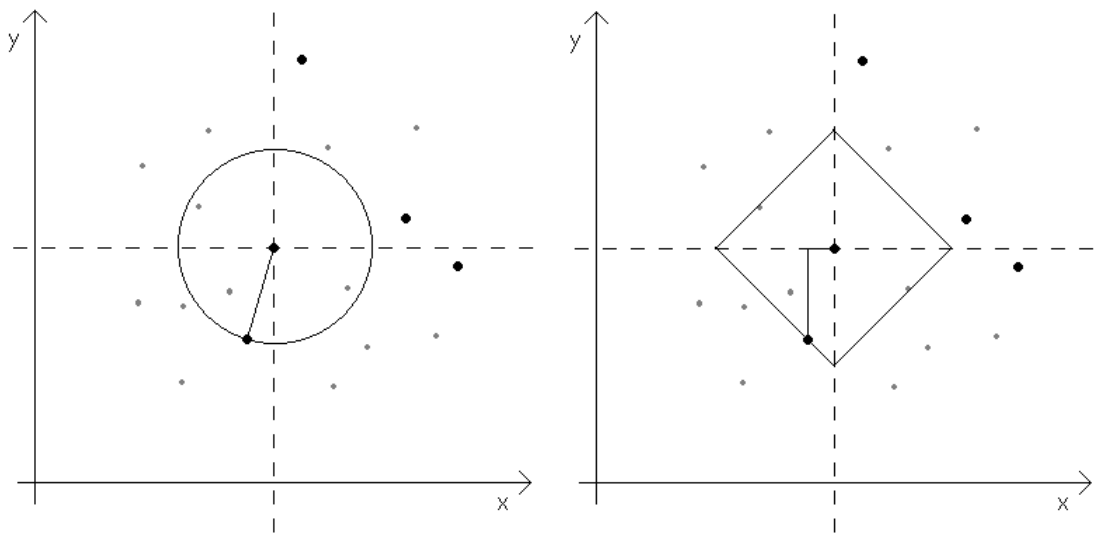

Fig. 2: Closest from neighbours for different distance definitions

Then, distances of the neighbours are computed and the distance of the closest one is taken as a threshold (see Fig. 2, which shows examples using Euclidean distance and ‘city block’ distance). At this point, a sub-optimally closest cluster has been found, and the maximal distance for searching for the true closest cluster has been determined.

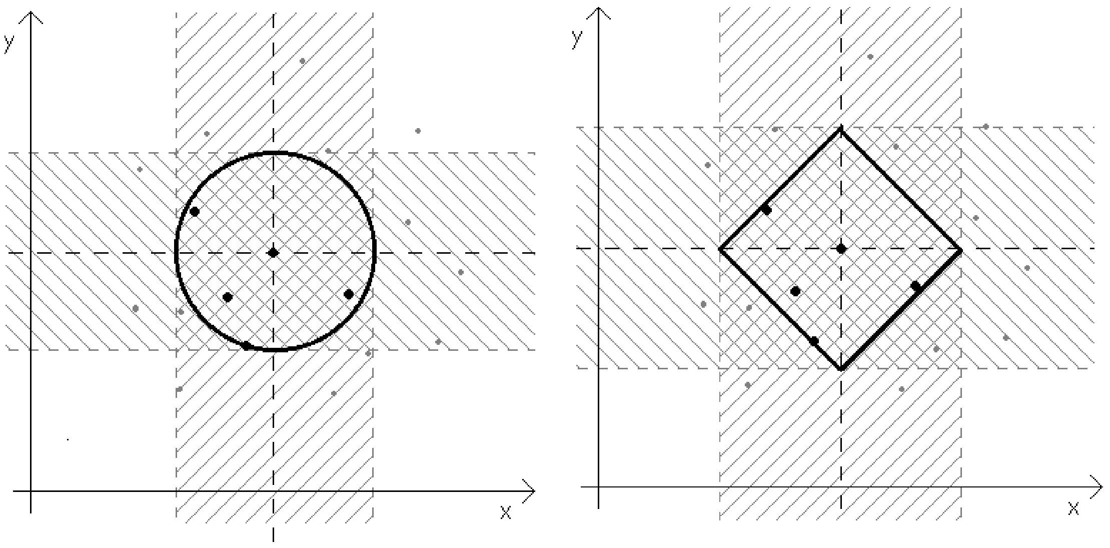

Fig. 3: Searched intervals and threshold distances for different distance definitions

When the maximal distance is known, the algorithm goes through the lists, starting with the initialised cluster, and looks for clusters whose distance is smaller than the threshold (the bold ones in Fig. 3). Such (successful) clusters are added into the list of close neighbours. The algorithm stops when the distance in list dimension is bigger then the threshold. Thus the algorithm searches only in strips given by maximal distance (see Fig. 3) and adds only several (the bold ones in Fig. 3) clusters into the close neighbours list of the initialised cluster.

All clusters are initialised this way. In the next step, the algorithm looks into the close neighbours lists of clusters and chooses the closest pair. Since the lists are sorted according to distance, no additional search is needed.

Then, the closest pair is grouped. The new cluster is created and inserted into dimension lists, while the old clusters are deleted. After this, the close neighbours lists are updated for clusters, which contain the changed clusters. Only clusters with neighbourhoods containing these clusters are checked. When deleting grouped cluster, the close neighbours list can get empty. In such cases, the initialisation mentioned above is made. The new cluster is initialised as well.

If there is still more then one cluster, the algorithm starts the next cycle by searching for the closest pair.

Position hierarchy

Since the position determination in the underlying database system is based on known postcodes for each student, the hierarchy partitioning follows the hierarchical structure of postcode groupings. Its hierarchical structure can be shown on the postcode used in the UK, which represents position quite exactly.

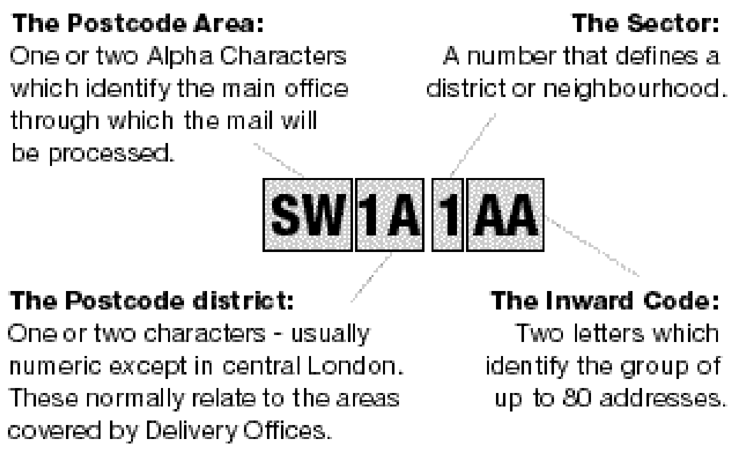
The Postcode Area:
One or two Alpha Characters which identify the main office through which the mail will be processed.
The Sector:
A number that defines a district or neighbourhood.
SW 1A 1 AA
The Postcode district:
One or two characters - usually numeric except in central London. These normally relate to the areas covered by Delivery Offices.
The Inward Code:
Two letters which identify the group of up to 80 addresses.

Fig. 4: UK postcode hierarchy

The figure Fig. 4 (taken from [9]) shows meaning of each part of the UK postcode. The construction of postal codes is similar in other countries or at least is based on some division of a state into administrative regions.

We take advantage of this existing administrative hierarchy to divide entities into groups and speed up the algorithm. The partitioning can be done at several levels following the hierarchy of postcodes. Thus, the clustering starts inside the smallest areas described by postcode and ends by grouping countries or continents. Areas are clustered on each hierarchy level using the same technique (the closest first).

Other attributes related to the deployed co-ordinate system can be used for partitioning as well, for example building, floor, and room for geographic space or branch, and specialisation for space of courses or interests.

Implementation

The data structures used in implementation of the new clustering algorithm are shown in Fig. 5.

Each cluster has its own main record containing its attributes (courses, names, and so on). It also manages a one-way sorted list of close neighbours (starting with the closest). Aside from these structures, a bidirectional sorted list of pointers to clusters is used for each dimension.

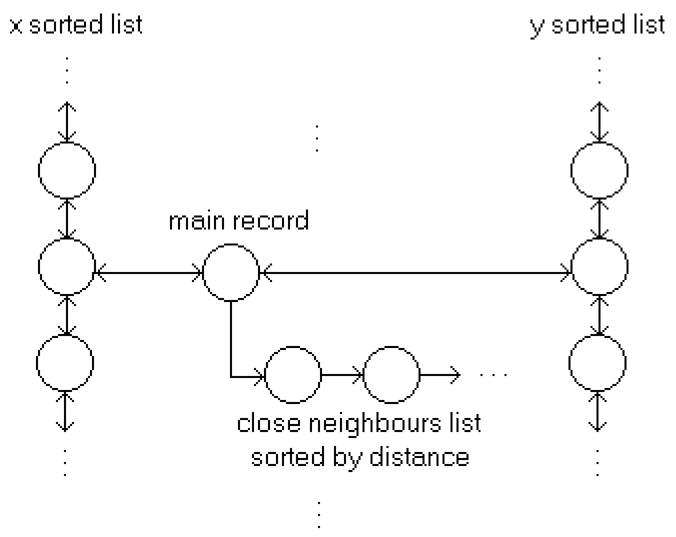

Fig. 5: Data structures used for new clustering

Algorithm complexity

Let us discuss the time complexities of the algorithms step by step.

Basic algorithm:

- distances initialisation O(n*n)

n-1 times n-1 times

  - the closest pair search O(n*n)
  - grouping O(1)
  - update of distances O(n)
  - ⇨ O(n*n*n)

- ⇨ O(n*n*n)

It follows, that the basic algorithm complexity is O(n*n*n).

The new algorithm:

- 2 times insert into sorted list O(log n), (in our implementation O(n))
- distances initialisation n times
    - closest neighbour search in lists O(1)
    - search for potentially closest neighbours not worse than O(n)
    - ⇨ O(n*n)

n times

- the closest pair search: O(n)
- update of distances – the number of clusters in neighbourhood is not dependent on n O(1)
- initialisation of distances of the new cluster O(n)
- ⇨ O(n*n)

⇨ O(n*n)

The time complexity of the designed algorithm is thus O(n*n), an improvement on the complexity of the basic one.

The hierarchical partitioning is able to improve the asymptotic time complexity only in cases where the number of areas depends on n. For example, for two level clustering, complexity O(n*n) is improved to O(f(n)^2*n/f(n) + (n/f(n))^2), where f(n) is number of clusters in the region and n/f(n) is the number of regions – this means that for sqrt(n) regions (square root of n), the complexity would be n*sqrt(n) + n. For partitioning involving more levels, the complexity improvement is even more significant.

Table Tab.1 shows experimental results measured for a random distribution of entities without partitioning into areas.

number of entities	basic alg. Euclidean	new alg. Euclidean	new alg. block
1000	8	4	2
2000	65	15	9
3000	216	31	18
4000		82	39
5000		130	71

Tab. 1: Experimental results - times in seconds

The tests were made for the basic algorithm with Euclidean distance definition and for the new algorithm with Euclidean and also block distance definition. The results are in seconds and were obtained on a Pentium III PC.

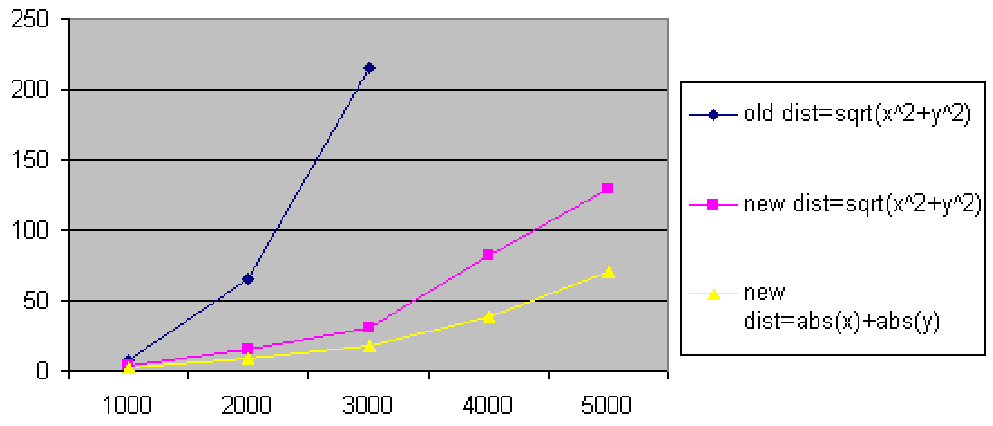

Fig. 6: Experimental results - times in seconds

Conclusion

A new hierarchical clustering algorithm was designed. It is based on a reduction of the number of potentially closest neighbours. The time complexity is demonstrably improved compared with the basic algorithm. Further speed up can be achieved by hierarchical partitioning of the space into regions and by separate clustering inside these regions.

References

[1] Daniel, J. *Mega-Universities and Knowledge Media: Technology Strategies for Higher Education.* London: Kogan-Page, 1998.

[2] Eisenstadt, M. and Vincent, T. (Eds). *The Knowledge Web: Learning and Collaborating On the Net.* London: Kogan-Page, 1998.

[3] Salmon, G. *E-Moderating: The Key to Teaching and Learning Online.* London: Kogan-Page, 2000.

[4] Pulver, J. Presence and Instant Messaging conferences: http://www.pulver.com/pim/

[5] SAS. http://www.sas.com/

[6] MapInfo. http://www.mapinfo.co.uk/

[7] MapPoint: http://www.microsoft.com/mappoint/

[8] Zucker, M. Hierarchical Clustering Algorithm. Center for Computational Biology, Washington University in St. Louis, 2000-02-25. http://www.ibc.wustl.edu/~zuker/Bio-5495/multalign-html/node7.html

[9] RoyalMail: Postcodes online, http://www.royalmail.co.uk/quick_tools/postcodes/default.htm

Acknowledgement

The Knowledge Media Institute at The Open University (UK) provided support for this work. Marc Eisenstadt provided editorial comments on the paper and feedback on the original implementation. Matt Koss and Zdenek Zdrahal provided feedback and advice on the algorithm design.

The research has been performed in the frame of PhD study at the Department of Computer Science and Engineering (Czech Technical University in Prague – Czech Republic) under supervision of Pavel Slavik.