An Effective Method for Locally Neighborhood Graphs Updating

Hakim Hacid and Abdelkader Djamel Zighed

Lyon 2 University,
ERIC Laboratory- 5, avenue Pierre Mendès-France
69676 Bron cedex - France
hhacid@eric.univ-lyon2.fr, Abdelkader.Zighed@univ-lyon2.fr

Abstract. Neighborhood graphs are an effective and very widespread technique in several fields. But, in spite of the neighborhood graphs interest, their construction algorithms suffer from a very high complexity what prevents their implementation for great data volumes processing applications. With this high complexity, the update task is also affected. These structures constitute actually a possible representation of the point location problem in a multidimensional space. The point location on an axis can be solved by a binary research. This same problem in the plan can be solved by using a voronoi diagram, but when dimension becomes higher, the location becomes more complex and difficult to manage. We propose in this paper an effective method for point location in a multidimensional space with an aim of effectively and quickly updating neighborhood graphs.

1 Introduction

Nowadays, advances occur in all fields, more particularly in information technologies. The needs of information and data processing are more and more increasing. In addition, information is conveyed by data being in various forms like multimedia data, temporal data, spatial data and the Web. In addition to the data heterogeneity, their significant mass make their processing more difficult.

In front of this situation, store and retrieve multimedia databases is not an easy task. The traditional Database Management Systems (DBMS) are not able to process the new emergent data such as multimedia data. This kind of systems is conceived to manage textual databases. The structure of the data qualified as "complex" is not adequate for these systems. Data structuration enables to center and clean them. Also, this structuration allows solving the performance problem of the data-processing support which can store and process only a limited quantity of information. Starting from this divergence, the concept of complex data was born.

Deal with multidimensional data structuration means deal with the point location problem in a multidimensional space. The point location problem is a key question in the automatic multidimensional data processing. This problem can be described as follows: having a set of data Ω with n items in a multidimensional space $\mathcal{R}^{d}$, the problem is then to find a way to pre-process the data so that if we have a new query item q, we'll able to find its neighbors within as short as possible time.

In this article, we deal with the point location in a multidimensional space, and this, in order to find an efficient way for optimizing the updating task of the neighborhood graphs. We propose a fast algorithm of a point location in a multidimensional space based on some neighborhood graphs properties. Section 2 presents a state of art on the point location problem, on neighborhood graphs as well as the problematic. Our proposition, based on the search of an optimal hyper sphere, is presented in section 3. Next, we present an illustration example of the method as well as some results. Finally, we conclude and give some future works in section 5.

2 State of Art

Neighborhood search is a significant problem in several fields, it is handled in data mining [8], classification [6], machine learning [5], data compression [13], multimedia databases [10], information retrieval [7], etc. Several works in connection with the neighborhood research in databases exist like [3][18][23].

The point location problem in one-dimensional space can be solved by applying a sorting then a binary search which is rather fast and inexpensive in term of resources with a complexity of $O\,(n\ log\ n)$ (n corresponds to the number of items). In a two-dimensional space, this problem can be solved by using voronoi diagram [19]. Unfortunately, when the dimension increases, the problem becomes more complex and more difficult to manage. Several methods of point location in a multidimensional space were proposed. We can quote for example the ones based on the points projection on only one axis [11][14][17], or the work based on partial distances calculation between items [2].

As introduced, the objective of the point location is to find a way to preprocess the data so that if we have a new query item, we'll be able to find its neighbors within as short as possible time. One possible way to represent such data structure is a neighborhood graph.

Neighborhood graphs, or proximity graphs, are geometrical structures which use the concept of neighborhood to determine the closest points to another given a query point. For that, they are based on the distance measures [22]. We will use the following notations throughout this paper.

Let Ω be a set of points in a multidimensional space $\mathcal{R}^d$. A graph $\mathcal{G}(\Omega,\varphi)$ is composed by the set of points Ω and a set of edges φ. Then, for any graph we can associate a binary relation upon Ω, in which two points $(p, q) \in \Omega^2$ are in binary relation if and only if the couple $(p, q) \in \varphi$. In an other manner, (p, q) are in binary relation (R) if and only if they are directly connected in the graph $\mathcal{G}$. From this, the neighborhood $\mathcal{N}(p)$ of a point p in the graph $\mathcal{G}$ can be considered as a sub-graph which contains the point p and all the points which are directly connected to it.

Several possibilities were proposed for building neighborhood graphs. Among them we can quote the Delaunay triangulation [19], the relative neighborhood graphs [21], the Gabriel graph [12], and the minimum spanning tree [19]. For illustration, we describe hereafter two examples of neighborhood graphs, relative neighborhood graph (RNG) and Gabriel graph (GG).

2.1 Relative Neighborhood Graphs

In a relative neighborhood graph $\mathcal{G}_{RNG}(\Omega,\varphi)$, two points $(p, q) \in \Omega^2$ are neighbors if they check the relative neighborhood property defined hereafter.

Let $\mathcal{H}(p, q)$ be the hyper-sphere of radius $d\,(p, q)$ and centered on p, and let $\mathcal{H}(p, q)$ be the hyper-sphere of radius $d(q, p)$ and centered on q.

$d\,(p, q)$ and $d\,(q, p)$ are the dissimilarity measures between the two points p and q. $d\,(p, q) = d\,(q, p)$.

Then, p and q are neighbors if and only if the lune $\mathcal{A}(p, q)$ formed by the intersection of the two hyper-spheres $\mathcal{H}(p, q)$ and $\mathcal{H}(q, p)$ is empty [21]. Formally:

$$\mathcal{A}(p, q) = \mathrm{H}(p, q) \cap \mathcal{H}(q, p) \;\; Then\ (p, q) \in \varphi\ iff\ \mathrm{A}(p, q) \cap \Omega = \phi$$

Figure 1 illustrates the relative neighborhood graph.

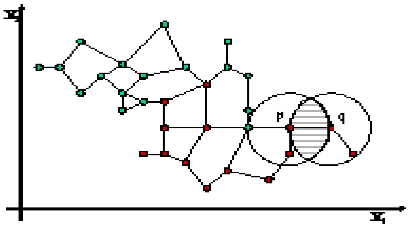

Fig. 1. Relative neighbourhood graph

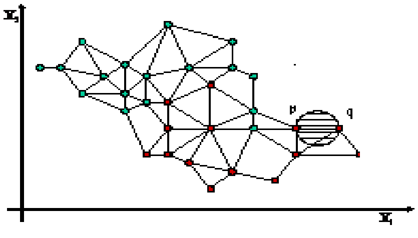

Fig. 2. Gabriel graph

2.2 Gabriel Graph

This graph is introduced by Gabriel and Sokal [12] into a geographical variations measurement concept. Let $\mathcal{H}(p,q)$ be the hyper-sphere of diameter $d\,(p,q)$ (cf. Figure 2). Then, p is the neighbour of q if and only if the hyper-sphere $\mathcal{H}(p,q)$ is empty. Formally :

$$(p,q) \in \varphi \; iff \;\; \mathcal{H}\;(p,q) \cap \Omega = \phi$$

2.3 Neighborhood Graphs Construction Algorithms

Several algorithms for neighborhood graphs construction were proposed. The algorithms which we quote hereafter relate to the construction of the relative neighborhood graph.

One of the common approaches to the various neighborhood graphs construction algorithms is the use of the refinement techniques. In this approach, the graph is built by steps. Each graph is built starting from the previous graph, containing all connections, by eliminating some edges which do not check the neighborhood property of the graph to be built. Pruning (edges elimination) is generally done by taking into account the construction function of the graph or through geometrical properties.

The construction principle of the neighborhood graphs consists in seeking for each point if the other points in the space are in its proximity. The cost of this operation is of complexity $O\left(n^3\right)$ (n is the number of points in the space). Toussaint [22] proposed an algorithm of complexity $O\left(n^2\right)$. He deduces the RNG starting from a Delaunay triangulation [19]. Using the Octant neighbors, Katajainen [16] also proposed an algorithm of complexity. Smith [20] proposed an algorithm of complexity $O\left(n^{23/12}\right)$ which is less significant than $O\left(n^3\right)$.

2.4 Problems Involved in the Neighbourhood Graphs

Neighbourhood graphs are very much used in various systems. Their popularity is due to the fact that the neighbourhood is determined by coherent functions which reflect, in some point of view, the mechanism of the human intuition. Their use is varied from information retrieval systems to geographical information systems.

However, several problems concerning the neighbourhood graphs are still of topicality and require detailed work to solve them. These problems are primarily related to their high cost of construction and on their update difficulties. For this reasons, optimizations are necessary for their construction and update.

We can consider two possible cases for the neighborhood graphs optimization. The first one is the case where the graph is not built yet, there is a necessity to find an optimal algorithm to build the structure or approximate it in order to have a graph as close as possible to the basic one. The second case is the one where the graph is already built, its rebuilding (with an approximation) can cause an information loss, therefore, it is a question of finding an effective way to update the graph without rebuilding it. In this paper, we are concerned by the second case and we propose an effective method of locally updating the neighborhood graph without having to rebuild it.

3 Neighborhood Graph Local Update

We consider that the neighborhood graph local update task passes by the location of the inserted point (or removed), as well as the points which can be affected by the update. To achieve this, we proceed in two main stages : initially, we determine an optimal space area which can contain a maximum number of potentially closest points to the query point (the point to locate in the multidimensional space). The second stage is done in the aim of filtering the items found beforehand. This is done in order to determine the real closest points to the query point by applying an adequate neighborhood property. This last stage causes the effective updating of the neighborhood relations between the concerned points. The steps of this method are summarized in the followings:

– Lay out the point q in the multidimensional space $\mathcal{R}^d$;
– Seek the first nearest neighbor of q (say p);
– Fix a search area starting from p ;
– Scan the concerned area and update the connections between the items which are there;
– Take the truths neighbors close to q and see if there are exclusions.

The main stage in this method is the search area determination. This can be considered as a question of determining an hyper sphere which maximizes the probability of containing the neighbors of the query item while minimizing the number of items that it contains.

We exploit the general structure of the neighborhood graphs in order to determine the ray of the hyper sphere. We are focusing, especially, on the *nearest neighbor* and the *farther neighbor* concepts. So, two observations in connection with these two concepts seem to us interesting :

– The neighbors of the nearest neighbor are potential candidates neighbors for the query point q.
  From there, we can deduce that:
– All the neighbors of a point are also candidates to the neighborhood of a point to which it is a neighbor.

In order to establish these sets, we determine initially an hyper sphere which maximizes the probability of containing the neighbors of the inserted point, after this, filtering the recovered items to find the partially truths neighbors.

With regard to the first step, determine the ray of an hyper sphere which maximizes the probability of containing the neighbors of the inserted point, this ray is the ray of the sphere including all neighbors of the first nearest neighbor to the query item. We consider this ray as the one formed by the sum of the distances between the inserted point and its nearest neighbor, and the one between the nearest neighbor and its further neighbor. That is, let consider q be the query point and p the nearest neighbor of q with a distance d_1. Let consider X be the further neighbor of p with a distance d_2. The ray SR of the hyper sphere can be expressed as:

$$SR = d_1 + d_2 + \epsilon$$

ϵ is a relaxation parameter, it can be fixed according to the state of the data (their dispersion for example) or by the domain knowledge. In order to avoid the high dimension effects, we fixed experimentally this parameter to 1.

The content of the hyper sphere is processed in order to see whether there are some neighbors (or all the neighbors). The second step constitutes a reinforcement step and aims to eliminate the risk of losing neighbors or including bed ones. This step proceeds so as to exploit the second observation: we take all truths neighbors of the inserted point recovered beforehand (those returned in the first step) as well as their neighbors and update the neighborhood relations between these points. The algorithm hereafter summarizes the various steps of this method.

```
**LocaleUpdate**($\mathcal{G}(\Omega,\varphi)$,$p$)

– $p_x = \{p_i \in \Omega, i = 1, ..., n / ArgMax\ d(p, p_i)\}$
– $N(p_x) = \{p_i \in \Omega, p\ R\ p_i\}$
– $p_y = \{p_i \in N(p_x), p_x \neq p_y / p_y\ R\ p_x \wedge\ ArgMax\ d(p_x, p_y)\}$
– $SR = d(p, p_x) + d(p_x, p_y) + \epsilon$
– $\Omega_1' = \{p_i \in \Omega, i = 1, ..., n / ArgMax\ d(p, p_i)\}$
– **For each** $p_i \in \Omega_1'$ **Do** *Check if* $p\ R\ p_i$ **End For**
– $\Omega_2' = \{p_j \in \Omega_1', j = 1, ..., |\Omega_1'| / d(p, p_j) \leq SR\}$
– $\Omega_3' = \{p_k \in \Omega, p_t \in \Omega_2', / p\ R\ p_k\ \vee\ p_k\ R\ p_t\}$
– **For each** $p_i \in \Omega_3'$ **Do** *Check if* $p\ R\ p_i$ **End For**
– $\Omega_4' = \{p_l \in \Omega_3' / p\ R\ p_j\}$
– Update the neighborhood in $\Omega_4'$
– Return ($\mathcal{G}(\Omega + 1$,$\varphi$'))
```

φ': represents the number of new connections after the addition of the query point.

The complexity of this method is very low and meets perfectly our starting aims (locating the neighborhood points in an as short as possible time). It is expressed by:

$$O(2n + n'^2)$$

With

- n : the number of items in the database.
- n' : the number of items in the hyper sphere ($<< n$).

This complexity includes the two stages described previously, namely, the search of the ray of the hyper sphere and the seek of the truths neighbors which are in it and corresponds to the term $O\,(2n)$. The second term corresponds to the necessary time for the effective update of the neighborhood relations between the real neighbors which is very weak taking into account the number of candidates turned over. This complexity constitutes the maximum complexity and can be optimized by several ways. The most obvious way is to use a fast nearest neighbor algorithm.

4 Illustrations and Experiments

In the followings, we'll illustrate the various steps of the proposed method. Let us consider a relative neighborhood graph for an item set as well as a query point as illustrated on Figure 3. Once the point q situated, it is then necessary to seek its nearest neighbor as well as the further neighbour of this last one. This step is illustrated in Figure 4.

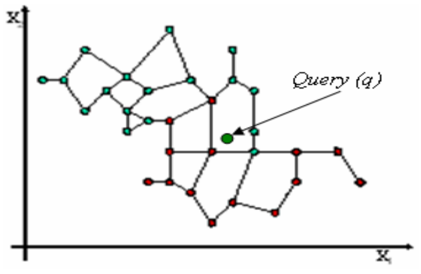

Fig. 3. Query point Position

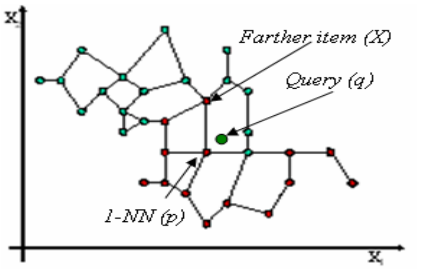

Fig. 4. Parameters determination

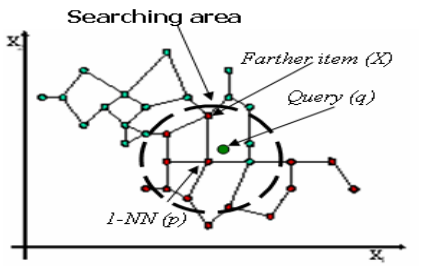

Fig. 5. Seeking of the candidates points

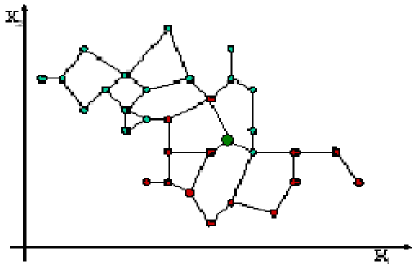

Fig. 6. Effective update

The parameters determined, we can then build the (hyper) sphere which contains all the potentially candidate points and update the relations within the (hyper) sphere like illustrated in Figure 5 and Figure 6 respectively.

To check the utility and the interest of the suggested method, we carried out some tests on various data sets. The principle of these experiments is as follows: We have m data sets $\{\mathcal{S}_1,\mathcal{S}_2,...,\text{S}_m\}$ with different items count $\{n_1, n_2, ..., n_m\}$, we build a neighborhood graph on each data set and we save the corresponding graph structure of each set which will serve like reference.

Once the reference graphs are built, we take each data set and we again build new graphs using $n - 1$ items. We use then our method in order to insert the remaining item and we calculate the recall on the various built graphs. This operation is repeated on several items on the m data sets.

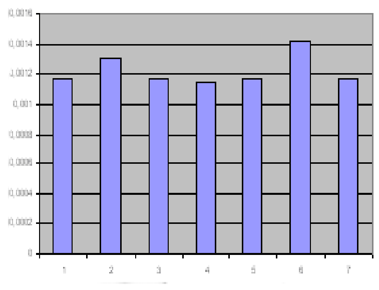

Fig. 7. Recall variations on different data sets

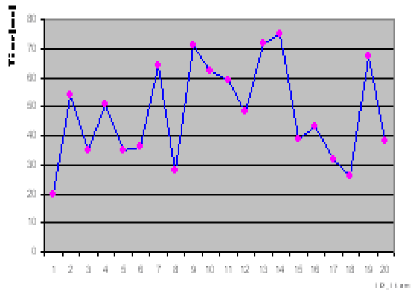

Fig. 8. Time variations on 20 items insertions

We used several data sets for these experiments, for example, Iris [1][9], UCI Irvine [15] and the Breiman's waves [4]. The graphic of Figure 7 illustrates the variation of the recall on various data sets by taking three situations : the recall is calculated on the reference graph, then on the graph with n-1 items and finally after the insertion of the remaining point. The first experiment constitutes our reference. The experiments carrying numbers 2,4,6 were carried out with an item in less and experiments 3,5,7 after the insertion of the remaining item in the corresponding previous experiment. That is, after the insertion of the remaining item, we always find the recall of the reference graph, this means that the method finds the good neighbors of the inserted item into each experiment.

In term of execution time, this last is variable and depends on several parameters among which we can quote primarily the data density. This parameter influences the number of points to take into account in the hyper sphere. A summary is presented in Figure 8 which is a result of locally insertion of 20 points taken arbitrary in a data set containing 5000 items represented in 21 dimensions. The obtained results are interesting (the time is expressed in milliseconds) and answer perfectly to the objectives laid down at the beginning.

5 Conclusion and Future Works

The direct use of the neighborhood graphs is not suitable, because their complexity does not make it reasonable to build them starting from great databases. We proposed in this article an effective method for updating neighborhood graphs locally. This method is based on an intelligent function of point location in a multidimensional space, and on the exploitation of some neighborhood graphs characteristics. The experiments carried out with this method showed its effectiveness. The use of this method can be very beneficial in particular in on line applications.

Like future works, we plan to fix the problem of the relaxation parameter determination by setting up an automatic determination function. This can be done by taking into account some statistical indicators on the data like the dispersion. Also we plan to extend this algorithm in order to make an incremental algorithm for the construction of the neighborhood graphs.

References

1. E. Anderson. The irises of the gaspé peninsula. *Bulletin of the American Iris Society 59*, pages 2–5, 1935.
2. C.-D. Bei and R. M. Gray. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Transactions on Communications 33*, pages 1132–1133, 1985.
3. S. Berchtold, C. Böhm, D. A. Keim, and H.-P. Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *PODS*, pages 78–86, 1997.
4. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. *Wadsworth International Group: Belmont, California*, pages 43–49, 1984.
5. R. S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.
6. T. M. Cover and P. E. Hart. Nearest neighbor pattern classication. *IEEE Trans.Inform. Theory*, 13:57–67, 1967.
7. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.
8. U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1–34. 1996.
9. R. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.
10. M. Flickner, H. S. Sawhney, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *IEEE Computer*, 28(9):23–32, 1995.
11. J. H. Friedman, F. Baskett, and L. J. Shustek. An algorithm for finding nearest neighbors. *IEEE Trans. Computers*, 24(10):1000–1006, 1975.
12. K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic zoology*, 18:259–278, 1969.
13. A. Gersho and R. M. Gray. Vector quantization and signal compression. *Kluwer Academic, Boston, MA*, 1991.

14. L. Guan and M. Kamel. Equal-average hyperplane partitioning method for vector quantization of image data. *Pattern Recognition Letters*, 13(10):693–699, 1992.
15. S. Hettich, C. Blake, and C. Merz. Uci repository of machine learning databases, 1998.
16. J. Katajainen. The region approach for computing relative neighborhood graphs in the lp metric. *Computing*, 40:147–161, 1988.
17. C.-H. Lee and L. H. Chen. Fast closest codeword search algorithm for vector quantisation. *IEE Proc.-Vis. Image Signal Process*, 141:143–148, 1994.
18. K.-I. Lin, H. V. Jagadish, and C. Faloutsos. The tv-tree: An index structure for high-dimensional data. *VLDB J.*, 3(4):517–542, 1994.
19. F. Preparata and M. I. Shamos. *Computationnal Geometry-Introduction.* Springer-Verlag, New-York, 1985.
20. W. D. Smith. Studies in computational geometry motivated by mesh generation. *PhD thesis, Princeton University*, 1989.
21. G. T. Toussaint. The relative neighborhood graphs in a finite planar set. *Pattern recognition*, 12:261–268, 1980.
22. G. T. Toussaint. Some insolved problems on proximity graphs. *D. W Dearholt and F. Harrary, editors, proceeding of the first workshop on proximity graphs. Memoranda in computer and cognitive science MCCS-91-224. Computing research laboratory. New Mexico state university Las Cruces*, 1991.
23. D. A. White and R. Jain. Similarity indexing: Algorithms and performance. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 62–73, 1996.