Construction Strategies on Metric Structures for Similarity Search

Roberto Uribe-Paredes  Claudio Márquez  Roberto Solar
Departamento de Ingeniería en Computación,
Universidad de Magallanes,
Punta Arenas, Chile
Database Group - UART,
Universidad Nacional de la Patagonia Austral,
Río Turbio, Argentina
(roberto.uribeparedes@gmail.com)

Abstract

The List of cluster LC is an effective technique to index high dimension metric spaces. LC is an array-type structure for similarity search based on clustering. Sparse Spatial Selection SSS is a new structure based on pivots for similarity search in metric spaces. This array-type structure has shown good performance during the search as compared to other methods.

This work shows different construction strategies on LC; for instance, the use of SSS as a general selection method of pivots or centers, among others. The article also shows the advantages of the use of other techniques like keeping the distance between the objects and the cluster center, apart from revising the effects of recursive application of such methods. Finally, the influence of the use of Voronoi partitions for the distribution of the objects within the structure will be shown.

Preliminary experimental results show that the new versions of LC have a better performance in terms of distance evaluation than the original data structure and other renowned structures.

Keywords: Databases, Data structures, Algorithms, Metric Spaces, Similarity Search.

1. INTRODUCTION

The search of similar objects in a large array of stored objects in a metric database has become a most interesting problem. For example, one typical query for these applications is the range search which consists of obtaining all the objects that are at a definite distance from the consulted object. From this operation, another can be built, nearest neighbor for example. The application of these techniques can be used in voice and image recognition, data mining, plagiarism and many others.

Similarity is modeled in many interesting cases through metric space and the search of similar objects through range search or nearest neighbor. A metric space is a set $\mathbb{X}$ with a distance function $d : \mathbb{X}^2 \to \mathbb{R}$, so that $\forall x, y, z \in \mathbb{X}$, then there must be properties of positiveness $(d(x, y) \geq 0)$ and $d(x, y) = 0$ ssi $x = y$) symmetry $(d(x, y) = d(y, x))$ and triangular inequality $(d(x, y) + d(y, z) \geq (d(x, z))$.

In a metric space $(\mathbb{X}, d)$, a finite data set $\mathbb{Y} \subseteq \mathbb{X}$, a series of queries can be made. The basic query is the range query, a query being $x \in \mathbb{X}$ and a range $r \in \mathbb{R}$. The range query around $x$ with range $r$ is the set of points $y \in \mathbb{Y}$, as $d(x, y) \leq r$. A second type of query that can be built using the range query is $k$ nearest neighbor, the query being $x \in \mathbb{X}$ and point $k$. Neighbors $k$ nearest to $x$ are
a subset $A$ of objects $\mathbb{Y}$, if $|A| = k$ and an object $y \in A$ does not exist so that $d(y, x)$ is shorter than the distance of any object from $A$ to $x$.

The objective of search algorithms is to minimize the quantity of distance evaluation made to solve the query. Searching methods for metric spaces are mainly based on dividing the space using the distance to one or more selected objects. As they do not work with the particular characteristics of every application, algorithms become more general to work with any type of objects [5].

There exist different structures to search in metric spaces, which can occupy discreet or continuous distance functions, namely: GNAT [1], MTree [6], sSAT [8], S-Lite-Tree [10], EGNAT [11], among others.

Some search structures are focused on pivots and others on clustering. In the first case, data set pivots are selected and the distances are precalculated between the elements and the pivots. When a query is made, the query distance to the pivots is calculated and the triangular inequality is used to discard the candidates.

Clustering-based algorithms divide the space into areas, where every area has a center. Some information is stored in the area to discard all the area by comparing the query with its center. Clustering algorithms are the best for large spaces, which is the biggest problem in practice.

There exist two criteria to define the areas in clustering-based structures: hyper planes and covering radius. The former divides the space in Voronoi partitions and determines the hyper plane the query belongs to according to the corresponding center. The covering radius criterion divides the space in spheres that can be intersected and one query can belong to one or more spheres.

The Voronoi diagram is defined as the plane subdivision in $n$ areas, one per each center $c_i$ of the set \{c_1, c_2, ..., c_n\} (centers) so that $q \in c_i$ area, if and only if the Euclidian distance $d(q, c_i) < d(q, c_j)$ for every $c_j$, with $j \neq i$.

One of the factors that lead to a mistake in the query is the unfortunate choice of centers and pivots. In this respect, this work proposes the use of a new method called Sparse Spatial Selection (SSS) [9], as a general method, the selection of the structure called List of Clusters [4], which is an array-type clustering-structure and the use of covering radius sets of objects during the search.

For the experiments in this article, an area consisting of a Spanish dictionary of 86,061 entries with edition distance objects was selected. The second was an area of 100,000 real coordinate vectors of dimension 10 with Gauss distribution with mean 1 and variance 0.1, the Euclidean distance was used for this space. The third space is a collection of 40,700 images from image files and videos from NASA represented as vectors of dimension 20. The search structure was established with 90% of the data and reserving 10% for queries.

2. **List of Clusters**

*List of Clusters* [4] is a structure based on clustering or compacted partitions, which is very similar to a linked list, designed for a good performance in large spaces. Center $c$ is selected in the List of Clusters pertaining to database $\mathbb{Y}$ and a radius $r$ which determines the fraction of space that covers the sphere $(c, r)$ defined by the subset of elements $\mathbb{Y}$, which are at a distance not longer than $r$ from center $c$. Then, all the elements within sphere center $c$ are denominated $I$ (this is also called Bucket $Y$), and $E$ to all elements external to sphere center $c$. This process is repeated recursively. As a result, a list it provider containing a center, a radius and a bucket, which is called cluster. (See figure 1(a)) When compared other clustering algorithms, the List of Clusters only uses the covering radius criterion and instead of like the Voronoi-Tree. It is also possible to see the List of Clusters as a particular case of Voronoi-Tree or an (M-tree), considering $I$ and $E$ as right and left subtrees of
root \( c \). The difference lies in the fact that the above mentioned structures, try to build a balanced tree with internal structures, and the List of Clusters, on the contrary, is extremely unbalanced and does not possess any internal structures. The structure of the data is not symmetric. The first chosen center has preference over the subsequent centers which cause overlapping with the other clusters. Look at figure 1(b). All the elements in the cluster of the first center (c) in figure 1(b) are kept in bucket I. Nevertheless, they can also remain within bucket I of subsequent centers (c\(_2\), c\(_3\), etc. figure 1).

![Diagram](image)

**Figure 1:** Space construction and distribution of the List of Clusters.

The algorithm for the search is shown in figure 2(a), applied to any query \( q \) and a search radius \( r \) over the list of cluster \( L \).

An essential characteristic absent in other clustering algorithms, in which the search needs to enter all the clusters that are intersected by the query sphere. In this structure, the search in the remaining clusters can be cancelled as long as the query sphere is completely contained in a cluster. Figure 2(b), shows three types of queries in a cluster. In \( q_1 \), the current bucket and the rest of the clusters must be considered. The query ball for \( q_2 \) is completely contained in the cluster, and thus the search is done only in that bucket. For \( q_3 \), the search is avoided in the current bucket.

As a general characteristic of this structure, the selection of the centers and radius in every point of construction algorithms is not specified, as it is related to efficiency and not to accuracy of the data structure. A good selection of centers may be costly and an adequate selection of radius must be made for every center of the cluster.

### 2.1. Pivot and center selection

Particularly, center and pivot election according to the structure is relevant to obtain a better performance during the search, which is empirically proven in [2]. Different strategies have been suggested as adequate for the election of pivots. In [1], heuristics are followed to select the pivots that are away from each other. In [2], a criterion of comparison of the efficiency between the two sets of pivots is presented. It also provides several set selection strategies where the criterion of prior efficiency is used presented as well.

### 2.2. Sparse Spatial Selection (SSS)

This method is a new technique proposed for the selection of pivots [9], that was originally implemented in an array type structure and uses the triangular inequality to discriminate objects in the
search. The method performed like those proposed in [2] or better in terms of efficiency, with the additional advantage that it chooses a dynamic set of pivots well distributed in the space.

Let (X, d) be a metric space, Y ⊂ X a collection of objects and M the distance between the two farthest objects. At the beginning, the set of pivots is made up of the first object of the collection. Then, every element of the collection is verified be at as equal or longer distance to M + α from the selected pivots. If this is so, set of pivots is added; being a constant whose value is close to 0.4[9]. Figure 3 shows the attainment of pivots in any space.

The construction is similar to structure FQA[3] and others of the same array, but they differ in the way of choosing and searching.

Basically, there is an array where the quantity of rows is the total quantity of objects in the database and the quantity of columns is the number of pivots. For this work, SSS is basically considered a well distributed selected object selection method in space as it can be applied to any structure, regardless of the type and the criteria to limit the areas. It is also possible to build a structure SSS-based, that is to say, a structure that can be completely adjusted to the metric space in which it is implemented.

3. List of Clusters and Sparse Spatial Selection

During the construction, the List of Clusters initially select a center and a radius. Thus, it becomes natural to choose M * α as radius and every center using the SSS algorithm. In other words, the centers are chosen if they are located at a distance M * α from all the prior centers. SSS can be used for fixed radius, as well as for fixed size. However, preliminary tests determined a better performance in List of Clusters with a fixed radius. The calculation of M is done in all the
objects of the database. This process is costly. Nevertheless, it is off line and is not considered as a construction cost in this work.

Experimentally, it was observed that the difference was minimal for the List of Clusters with fixed radius with center selection at random against the alternative of centers using SSS and radius $M \alpha$ in the word space. It is all due to the fact that the function of distance is discreet, and thus the best radius is closely similar to the best $\alpha$. In vectors, there is an improvement when search ranges are in this experiment. The best radius is the one that recover 0.1% of all the data. For both, the best values for fixed radius and for $\alpha$ were used.

3.1. Recursive List of Clusters

In those described above, no improvement is achieved in the structure performance. This occurs because the best methods were selected for both experiments. The best fixed radius and $\alpha$ were obtained experimentally.

Considering that the space is divided once in $N$ parts, it is possible to apply this division in the generated subspaces; that is to say, every cluster of the structure can be a List of Clusters at the same time. Then, a second alternative of construction is to apply recursively the process of construction in every cluster formed in the original structure. The SSS method is used for the construction of this structure.

Finally, a tree structure is obtained where the sparsely disperse centers are selected using a radius of $M \alpha$.

Every cluster of the original structure represents a subspace whose characteristics differ from the original. In fact, the sizes of such space are much smaller. If this tree is built using the original $M$, it could be disadvantageous due to the fact that the space would be over dimensioned, causing a low performance in the methods efficiency. Now, it is possible to calculate $M$ again for the new subspace, but this would imply an elevated cost during the construction. However, it is possible to use the same covering radius of subspace to calculate an approximate $M$ without paying additional costs.

The covering radius is the distance from the center to its farthest element therefore, $M$ would be always smaller than or the same as $2r_c$ (twice the covering radius or diameter of cluster). This could be used every time the process of construction is done recursively.

The recursive use of SSS in every cluster, modifying value $M$, forces the structure to adapt to the new form of space. This implies that the quantity of centers in every node of the tree will be dynamic, that is to say, the nodes will not usually have the same quantity of objects.

Finally, the process finishes when the cluster has sufficient data for a quota, for example a disk page.

Similar works to lists of recursive clusters have been carried out [7], where one of the proposed selection methods is to choose the farthest away center from the others.

3.2. List of Clusters and other techniques

Initially, the application of center selection using SSS did not yield the expected results. However, when combining the recursive application of LC with SSS, the results improve the performance of the original version.

In this respect, this work analyses experimentally the structure of diverse renowned techniques\(^1\).

\(^1\)DF: Distance to Father, R: Recursive, VD: Voronoi Diagrams
One of the added techniques was to keep the distance of an father object (or cluster center) so that the data can be used during the search. This technique is applied in the SSS version (LC+SSS+DF) and with recursive SSS (LC+SSS+R+DF). Figure 4 shows the behavior of different versions against the original version with SSS (LC+SSS). In all the experiments, a calculated α was used experimentally for every space, using the one with the best performance in each. The value used as α appears between brackets. Finally, the latest applied technique was to modify the LC+SSS but distributing the objects in the clusters using Voronoi diagrams and keeping the distance to the father (LC+SSS+VD+DF). The results presented in this article are preliminary but promising. It is important to note that all versions improve the performance of the original List of Clusters. From the graphics, it can be inferred that the choice of keeping the distance to the father results in a good idea, enhancing the results obtained in all spaces. In general, the recursive versions are the second best choice in all spaces, except in the vectors of dimension 20. From the graphs, it is surprising that the version with data distribution using Voronoi diagrams outperforms all versions with notable differences in the Gauss space. It is considered that the advantage difference in favor can be found in the space distribution, which is favored by this data distribution.

![Cost Average Search](36630 vectors, dim 20)

Figure 4: Average search costs for the different versions of LC.

3.3. **List of Clusters and other structures**

Figures 5(a) and 5(b) show the decrease of distance evaluations of the new versions of LC against data structures known as M-Tree, GNAT and EGNAT. In these figures, the best three versions of LC are shown. It can be noted that LC has a better performance in the words space. EGNAT is the only competitive structure in the space with Gauss distribution. This structure is similar to all other versions of LC and is only beaten by the version with Voronoi diagrams.
Figure 5: Searches, different structures against the different LC versions.

Figure 5(c) shows a comparative graph for the space of dimension 20 between the three prior structures, and the three versions of LC. For this space, differences are smaller, however LC outperforms EGNAT.

4. Conclusions

A good choice of pivots or centers during the construction of metric structures will be always relevant for query processes. Considering that the best centers are dependent on space, it is ideal to boast mechanisms that allow the recollection of best centers regardless of the form of space. In this respect, the authors consider that SSS obtains an adequate number of centers, which is clearly proven throughout the paper.

The main contribution of this work is to develop different versions of LC structures, where all the new proposals have a better performance than the original version. The new versions are all based on clustering and use covering radius to discriminate during the search.

It is shown that the use of distance to father (cluster center) diminishes distance evaluations, which is a usual technique based on pivots. It is worth mentioning that the recursive version of the structure adjust itself to the form of space if SSS is used. This is also possible due to the estimate of value $M$ during the construction, which does not have any additional costs. It is also possible to affirm that preliminarily in certain spaces the use of Voronoi diagrams allows for a better distribution of space and thus enhances search efficiency.
The authors highly recommend that during the design stage of new structures the following should be considered: the effects that some techniques may have on adequate center or pivot selection, the use of distance to the cluster center and the use of Voronoi partitions. The experimental results provide a vision of the enormous advantages of the new versions of the List of Clusters against other promising structures.

Acknowledgements

Partially funded by research programs PR-F1-02IC-08, Universidad de Magallanes, Chile and 29/C035-1, Unidad Académica de Rio Turbio (UART), Universidad Nacional de la Patagonia Austral, Argentina.

References


