A Comparative Study of Different Metric Structures: Thinking on GPU Implementations

Roberto Uribe-Paredes1, Diego Cazorla2, José L. Sánchez2 and Enrique Arias2

Abstract—Nowadays, similarity search is becoming a field of increasing interest because these kinds of methods can be applied to different areas in science and engineering, for instance, pattern recognition, information retrieval, etc. This search is carried out over metric indexes decreasing the number of distance evaluations during the search process, improving the efficiency of this process. However, for real applications, when processing large volumes of data, query response time can be quite high. In this case, it is necessary to apply mechanisms in order to significantly reduce the average query response time. In this sense, the parallelization of the metric structures processing is an interesting field of research. For that purpose, modern GPU/Multi-GPU systems offer a very impressive cost/performance ratio. In this paper, the authors make a comparative study of the most popular metric structures and pivot selection methods in order to establish a set of attractive features from the point of view of future GPU implementations. Therefore, this work represents a state-of-the-art paper setting up the starting point for future parallel implementations of similarity search techniques.

Index Terms—Comparative study, range queries, similarity search, metric spaces.

I. INTRODUCTION

In the last decade, the search of similar objects in a large collection of stored objects in a metric database has become a most interesting problem. This kind of search can be found in different applications such as voice and image recognition, data mining, plagiarism detection and many others. A typical query for these applications is the range search which consists in obtaining all the objects that are at some given distance from the consulted object.

A. Similarity Search in Metric Spaces

Similarity is modeled in many interesting cases through metric spaces, and the search of similar objects through range search or nearest neighbors. A metric space \((X, d)\) is a set \(X\) and a distance function \(d : X^2 \rightarrow \mathbb{R}\), such that \(\forall x, y, z \in X\) fulfills the properties of positivity \([d(x, y) \geq 0, \text{ and } d(x, y) = 0 \iff x = y]\), symmetry \([d(x, y) = d(y, x)]\) and triangle inequality \([d(x, y) + d(y, z) \geq d(x, z)]\).

In a given metric space \((X, d)\) and a finite data set \(Y \subseteq X\), a series of queries can be made. The basic query is the range query \((x, r)\), a query being \(x \in X\) and a range \(r \in \mathbb{R}\). The range query around \(x\) with range \(r\) (or radius \(r\)) is the set of objects \(y \in Y\) such that \(d(x, y) \leq r\). A second type of query that can be built using the range query is \(k\) nearest neighbors \((kNN)\), the query being \(x \in X\) and object \(k\). The set of objects \(x\) are a subset of objects \(Y\), such that if \(|A| = k\) and an object \(y \in A\), there is no object \(z \notin A\) such that \(d(z, x) \leq d(y, x)\).

Metric access methods, metric space indexes or metric data structures are different names for data structures built over a set of objects. The objective of these methods is to minimize the amount of distance evaluations 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.

Metric space data structures can be grouped into two classes [1], clustering-based and pivots-based methods. The clustering-based structures divide the space into areas, where each area has a so-called centre. Some data is stored in each area, which allows easy discarding the whole area by just comparing the query with its centre. Algorithms based on clustering are better suited for high-dimensional metric spaces. Some clustering-based indexes are BST [2], GHT [3], M-Tree [4], GNAT [5], EGNAT [6] and many others.

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

In the pivots-based methods, a set of pivots is selected and the distances between the pivots and database elements are precalculated. When a query is made, the query distance to each pivot is calculated and the triangle inequality is used to discard the candidates. Its objective is to filter objects during a request through the use of a triangular inequality, without really measuring the distance between the object under request and the discarded object. Mathematically, these construction and searching processes can be expressed as follows:

- Let \(\{p_1, p_2, \ldots, p_k\}\) a set of pivots, \(p_i \in X\). For each element \(y\) of the database \(Y\), the distance to the \(k\) pivots \((d(y, p_1), \ldots, d(y, p_k))\) is stored. Given a query \(q\) and a range \(r\), the distance \((d(q, p_1), \ldots, d(q, p_k))\) to the \(k\) pivots is calculated.
- If for some pivot \(p_i\) the expression \(|d(q, p_i) - d(y, p_i)| > r\) is holding, then for triangle inequality \(d(q, y) > r\), and therefore it is unnecessary to explicitly evaluate \(d(q, y)\). All the objects not discarded by this rule have to be directly compared to the query \(q\).

Some pivots-based indexes are LAESA [7], FQT and its variants [8], Spaghettis and its variants [9], FQA [10], SSS-Index [11] and others.

Array-type structures implement these concepts directly. The difference among the array-type structures lies on extra
structures used to reduce the computational cost to obtain the number of candidates keeping invariable the evaluation of distances.

Many indexes are trees and the children of each node define areas of space. Range queries traverse the tree, entering into all the children whose areas cannot be proved to be disjoint with the query region.

The increased size of databases and the emergence of new data types create the need to process a large volume of data. Then, new research topics appear such as efficient use of computational resources (storage and its hierarchy, processors, network, etc) that allows us to reduce the execution time and to save energy. In this sense, recent appearance of GPUs for general purpose computing platforms offers powerful parallel processing capabilities at a low price and energy cost. However, this kind of platforms has some constraints related to the memory hierarchy.

The present work analyses, by means of a set of experiments, the results obtained for several metric structures in order to obtain those attractive features from the point of view of these new GPU-based implementation: selection of pivots and centres techniques, needed storage and simplicity of the data structure.

The present work analyses, by means of a set of experiments, the results obtained for several metric structures in order to obtain those attractive features from the point of view of a future GPU-based implementation: selection of pivots and centres techniques, needed storage and simplicity of the data structure.

The paper is structured as follows. In Section II the metric structures considered in this paper are described. In Section III the features to be evaluated are presented. Section IV outlines the experimental results and discussion. Finally, the conclusions and future work are commented in Section V.

II. METRIC STRUCTURES

The metric structures considered in this comparative study are:

- **a) Generic Metric Structure (GMS):** This structure represents the most basic structure: it is an array-type structure based on pivots, which are obtained randomly. From this generic structure could be derived the rest of structures based on arrays and the choice of the pivots could be carried out according to SSS-Index or MSD methods. These pivot selection techniques will be introduced later.

- **b) Spaghetti:** It is an array-type structure based on pivots and does not assume any pivot selection method. However, each entry in the array, that represents distances between an element in the database and the pivots, is sorted with respect to this distance, obtaining a reduction on the execution time of a binary search. In this work, the array is sorted considering only the first pivot.

- **c) SSS-Index: SSS-Index (Sparse Spatial Selection)** [11] is basically the generic structure varying the way in which the pivots are selected. The selection methods will be introduced later.

- **d) LAESA:** Like SSS-Index, it is a structure similar to the generic one, but the selection of pivots is carried out by a method called Maximum Sum of Distances (MSD).

- **e) EGNAT:** Evolutionary GNAT [6] is a clustering tree-type structure derived from GNAS structure. This method pretends to exploit the secondary memory hierarchy (see Figure 1). This structure is far from the array-type of the generic structure.

The choice of these metric structures is motivated because they are representative of this field of knowledge, and we have considered structures based on pivots and on clustering, array-type and tree-type.

With respect to the choice of pivot selection, we have considered the following:

- **a) Randomly:** As the name suggests, this method consists in selecting randomly the set of pivots of the database.

- **b) Sparse Spatial Selection (SSS):** Sparse Spatial Selection [11] is a method to select a dynamic set of pivots or centres distributed in the space. Let \( (X, d) \) be a metric space, \( \mathbb{U} \subset X \) and \( M \) the largest distance between all pairs of objects, i.e., \( M = \max \{d(x, y) | x, y \in \mathbb{U}\} \). Initially, the set of pivots contains the first element of the collection. After that, an element \( x_i \in \mathbb{U} \) is selected as a pivot if and only if the distance between it and the rest of selected pivots is greater than or equal to \( M + \alpha \), being \( \alpha \) a constant whose optimum values are close to 0.4 [11] (see Figure 2).

III. METRIC STRUCTURES FEATURES TO BE EVALUATED

In the literature it is possible to find a wide range of metric structures for similarity searching [1], [13].

In this work a set of representative metric structures have been considered based on pivots, clustering, array-type or tree-type. We have considered this variety of structures in order to determine, experimentally, if the cost in the searching...
process compensates the complexity of the implementation, taking into account that the decision taken here will condition the future implementation on a GPU-based platform.

The relevant features considered in this work are:

a) Execution time: The execution time is a key factor in order to determine the best implementation. In the literature lot of papers are found talking about evaluation of distances [6], [11], but they do not consider execution time (floating point operations and I/O operations), memory accesses, etc.

b) Distance evaluations: In general, the reduction on evaluation of distances has been considered as the main goal of the new structures design, and evidently, it has a direct impact on the execution time. However, the high processing capacity of current computational platforms implies that distance evaluation is not always the operation with a higher computational cost. For instance, in GPU-based platforms, sorting operation affects to the execution time more than the evaluation of distances.

c) Storage requirements: A very interesting feature to evaluate is the memory needed to store a structure, even more if memory constraints are considered as is the case of GPU platforms. We have only addressed main memory, being secondary memory out of the scope of this paper. The point is, "how much storage I am willing to sacrifice versus performance?".

IV. EXPERIMENTAL EVALUATION

A. Case of studies and platform

In this section, the case studies used as benchmarks and the testbed considered in this paper are described.

We considered two datasets: a subset of the Spanish dictionary and a color histograms database, obtained from the Metric Spaces Library1. The Spanish dictionary we used is composed of 86,061 words. The edit distance was used. Given two words, this distance is defined as the minimum number of insertions, deletions or substitutions of characters needed to make one of the words equal to the other. The second space is a color histogram. It is a set of 112,682 color histograms (112-dimensional vectors) from an image database. Any quadratic form can be used as a distance, so we chose Euclidean distance as the simplest meaningful alternative.

The results presented in this section belong to a set of experiments with the following features:

- For all data structures considered in this paper, a set of tests were carried out using pivots from 1 to 1362 for word space and from 1 to 244 for color histograms (see Figure 3). From all the results, only the best results have been plotting.
- For words space, each experiment has 8,606 queries over a Spaghettis with 77,455 objects. For vectors space, we have used a dataset of 101,414 objects and 11,268 queries.
- For each query, a range search between 1 and 4 was considered for the first space, and for vectors space we have chosen ranges which allow to retrieve 0.01%, 0.1% and 1% from the dataset.

We have chosen this experimental environment because is the most usual environment to evaluate this kind of algorithms. Also, these datasets are representative of discrete and continuous searching, respectively.

The hardware platform used is called Marte and belongs to the Albecete Research Institute of Informatics (I3A: http://www.i3a.uclm.es.). Marte is a 2 Quadcore Xeon E5530 at 2.4GHz and 48GB of main memory, and Ubuntu 9.10 (64 bits) Linux Operating System. The compilation has been done using gcc 4.3.4 compiler.

B. Experimental results and discussion

Although results are usually shown considering the search ranges in the X axis, in this paper we have considered a different approach. In order to compare the behaviour of different pivot-based structures, in our opinion, it is more interesting to show the results against the number of pivots, typically 4, 8, 16 and 32, but also 1 and a number of pivots bigger than 32, especially when we need to compare with SSS-Index. This structure does not allow to choose the number of pivots (they are calculated depending on several parameters such as the value of α and the kind of search space) and usually uses a big number of pivots.

Figure 3 shows an overview of the behaviour of the generic structure based on pivots for both datasets.

Usually in metric structures, the performance of a structure increases with the number of pivots. Nevertheless, as can be seen in Figure 3 for the generic structure, the performance increases till a point (that depends on the range considered, e.g. 32 pivots for range 1) from where the performance remains the same or decreases. This behaviour is common to all the structures as shown in Figure 4. In this figure only the results close to the best one are shown.

Notice that when using the SSS structure we cannot select a priori an exact number of pivots. This is the reason why the minimum number of pivots is 44 (for word space) and 35 (for color histograms). For word space, as the distance is discrete, there are not values between 328 and 665, so the value 500 does not exist in SSS-Index. The value 500 neither is shown in EGNAT because the needed structure is bigger than the RAM memory, swapping is needed and consequently performance is poor.

Analysing the results in Figure 4 we can conclude that for small ranges Spaghettis has the overall best performance considering both datasets. The reason is that the use of binary search allows a quick search of the first element in the database inside the range. GMS is very close to Spaghettis performance, and the other 3 structures have a bad behaviour in one of the two datasets, color histograms in MSD and SSS, and the Spanish dictionary in EGNAT. Nevertheless when we consider bigger search ranges, the advantage of Spaghettis is lost; in this case the price in time of the binary search is not worthy because less elements are discarded.

Figure 5 shows the same scenario for distance evaluations. The number of evaluations decreases when the number of pivots increases. This means that, comparing Figures 4 and 5, at some point is not worthy to increase the number of pivots because the time consumed in their management is bigger than the time consumed in the distance evaluations we save. We can also conclude that using more pivots is better for big ranges, and it has little influence for small ranges.
Search Costs for Generic Structure (n=86,061 words)

(a) General result for Spanish dictionary.

Search Cost for Implemented Methods (n=86,061 words)

(a) General result for Spanish dictionary.

Search Cost for Implemented Methods (n=112,682 vectors)

(b) General result for color histograms.

Search Cost for Generic Structure (n=112,682 vectors)

(b) General result for color histograms.

Fig. 3. Execution time for the implementation considering a generic metric structure (GMS).

Fig. 4. Execution time for the implementation considering all structures.
Tables I and II show, in detail, the execution time (in seconds) of the best cases depending on the range or on the data retrieved percentage, respectively. In these tables several modifications of the generic structure are considered. In these modifications the pivots were not selected randomly but following the pivots selection methods used by the other structures. Thus, first we get a subset of pivots from the database randomly or using \(SSS\) and then \(MSD\) is applied to get the number of pivots for the best performance case (32 or 44 depending on the range). Only modifications of the structure with a good performance are considered in the tables (e.g. “\(MSD\) x on \(SSS\) y” cases are not included in color histograms because they have a poor performance).

The results obtained for the modified generic structures are good. For small ranges \(Spaghetti\) is still better, but when the range increases the new structures have better performance. The advantage of using \(MSD\) over a big number of pivots randomly chosen is that it allows to choose the best pivots and the exact number of pivots desired and, consequently, it allows to determine the size of the structure which is an important factor to consider when we need to fit the structure in a virtual page or in GPU memory.

Looking forward to the GPU implementation, the size of the structure is a very important factor. A structure that does not fit into GPU memory will not have a good performance. Figure 6 shows that \(EGNAT\) structure is much bigger than pivot-based structures. As expected, the size of the structure in pivot-based structures is directly proportional to the number of pivots. In order to have a more detailed view, the bigger values were removed from the table (e.g. in color histograms \(EGNAT\) with 119 centres needs 2 GBytes, in word space \(EGNAT\) with 328 centres needs 6 GBytes).

Tree-type structures have a good performance when the radius increases, and they are very stable with respect to the number of pivots or centres. This means that we can get a good performance even selecting a small number of centres. The problem with this kind of structure is that when a new node in the tree is created, there is no guarantee that it will be completed, leading to a situation in which the size of the structure can grow a lot depending on how objects are distributed in subtrees. In the tree-based structure used in this paper we obtained that less than 20% of the nodes were completed.

V. CONCLUSIONS AND FUTURE WORKS

In this work a comparative study of different metric structures has been carried out. Different types of metric structures and pivot selection methods have been considered in order to make a good comparison. The comparison has been made according to three criteria: execution time, evaluation of distances and storage requirements.

According to the experimental results, it is not possible to select a metric structure as the best one, because it depends on the space distribution of the database. Three structures are candidates to be eligible as the best: \(Spaghetti, Generic\)
structure + MSD and EGNAT. However from the point of view of a future GPU implementation the best one is Generic + MSD due to:

1) By using a generic structure it is not necessary to apply a binary search like Spaghettis. Binary search operation is very expensive in a GPU-based platform in comparison with the evaluation of distances.
2) Using a generic structure the storage requirements are lower than using an EGNAT structure.
3) Thanks to the combination of generic structure and MSD pivot selection, it is possible to reduce the number of pivots till satisfying the memory constraints inherent to the GPU-based platforms.

To sum up, using the generic structure we will take benefits in terms of execution time, storage and, in addition, the code is more simple.

As we said in the introduction, the work presented in this paper allows us to choose the best option from the point of view of a parallel implementation of the similarity search method based on metric structures on a GPU platform, representing that the future work.

ACKNOWLEDGMENT

This work has been supported by the Ministerio de Ciencia e Innovación, project SATSIM (Ref: CGL2010-20787-C02-02), Spain and Research Center, University of Magallanes, Chile.

REFERENCES