A Hashed Schema for Similarity Search in Metric Spaces

Claudio Gennaro
IEI-CNR
Pisa, Italy
gennaro@iei.pi.cnr.it

Pasquale Savino
IEI-CNR
Pisa, Italy
savino@iei.pi.cnr.it

Pavel Zezula
Masaryk University
Brno, Czech Republic
zezula@fi.muni.cz

Abstract

A novel access structure for similarity search in metric data, called *Similarity Hashing* (SH), is proposed. Its multi-level hash structure of separable buckets on each level supports easy insertion and bounded search costs, because at most one bucket needs to be accessed at each level for range queries up to a pre-defined value of search radius. At the same time, the number of distance computations is always significantly reduced by use of pre-computed distances obtained at insertion time. Buckets of static files can be arranged in such a way that the I/O costs never exceed the costs to scan a compressed sequential file. Experimental results demonstrate that the performance of SH is superior to the available tree-based structures. Contrary to tree organizations, the SH structure is suitable for distributed and parallel implementations.

1 Introduction

Similarity searching has become a fundamental computational task in a variety of application areas, including multimedia information retrieval, data mining, pattern recognition, machine learning, computer vision, genome databases, data compression, and statistical data analysis. This problem, originally mostly studied within the theoretical area of computational geometry, is recently attracting more and more attention in the database community, because of the increasingly growing needs to deal with large, often distributed, volume of data. Consequently, high performance has become an important feature of prosperous designs.

In this article, we consider the problem from a brought perspective and assume the data to be objects from a metric space where only pair-wise distances between objects are known. Contrary to traditional storage structure designs where the I/O costs form the dominant component of the insert (delete) and retrieval functions, the CPU costs in generic metric structures can also be quite high, because in some applications the computation of distances is time consuming. So our major objective is to develop a similarity search structure that would minimize both the I/O and the CPU costs.

Notice that this has not been the objective in previous designs, see for example [Ch94, Br95, BO97, CPZ97, BO99, TTS+00], where some of the designs are only main memory structures, and a paged disk environment is not considered at all. What seems to be a good idea – carefully used in the above designs and also adopted in our proposal – is to reuse once computed distances in later stages, i.e. in the retrieval phase. In this way, the number of necessary distance computations to evaluate a query can significantly be reduced and search time decreased.

To the best of our knowledge, all metric data designs are trees, and the reported node utilization is typically poor (often much less than 50%). This implies high space occupancy and random access to read nodes. Notice that sequential files can be allocated on the minimum of necessary disk memory, and the sequential scan of such disk area is very fast. This contradiction have recently been observed and criticized by several researchers, see for example [WSB98] for a comparative evaluation of vector data organizations.

One of the main problems with metric data is that it is difficult to find a good partitioning of objects. The standard approaches typically result in partitions that are confined in bounding regions with high overlaps. Consequently, many partitions have to be searched to solve a query, which negates the original aim of partitioning. To provide a simple illustration, consider a binary division, also called the *split*, of a set of objects into two subsets. If the access to one of the subsets implies access to the other subset for majority of queries, such split is certainly not good, because the after split sets are not clearly separable from the query point of view. When regions are organized in trees, the performance is not constrained by any explicit cost bounds and due to *backtracking* that is necessary to evaluate a query, even all tree nodes can be accessed.

The only exception in this respect is the *excluded middle vantage point forest*, vp-forest, proposed by Yianilos [Yi99]. It is a data structure that uses the idea of the *excluded middle* partitioning strategy that partitions a set of objects into two separable subsets plus a third subset containing objects that, once stored in a separable subset, would violate the separability condition. The forest supports *worst case* sub-linear time searches (in terms of distance computations) for nearest neighbors with a fixed radius of arbitrary queries. The worst-case performance depends on the data-set, but is not affected by the distribution of queries. In fact, such approach was one of the motivations for our design.

Finally, there are two additional arguments that talk against developing metric structures as trees. First, insertion costs are high and node splitting strategies, both the top-down and the bottom-up, require a lot of distance computations – published articles usually do not report on this issue. Second, trees are not convenient for parallel and/or distributed implementations. On the other hand, parallel (distributed) memories are available and proper exploitation of their potentiality can significantly reduce the search costs.

2 The Idea and Properties of Similarity Hashing

The main challenge of our work is to build a similarity search organization based on hashed partitioning, that is the Similarity Hashing (SH) technique. It is a multi-level hash structure that takes advantage of the excluded middle partitioning. In fact, all buckets on a level are separable so that maximally one bucket must be accessed for any query up to specific value of query radius. Objects that do not conform to such arrangement at the level are excluded from storage on this level and become candidates for storing on the next level. Depending on the number of levels and the data file, some objects can remain excluded, thus stored in a separate *exclusion partition* that must always be accessed. Once computed distances are remembered at practically zero storage costs – a distance between objects is a number. At query time, a computation of simple functions determines for each level maximally one partition to access. But distance computation between the query and accessed object is not always performed since the knowledge about pre-computed distances can infer that the object can not belong to the query response set. A necessary specification of SH can be found in [GSZ00]. In summary, the main features of our approach can be characterized as follows:

- each object is stored through hashing in one bucket of the multilevel structure of separable buckets;

- queries need to access maximally one bucket per level, plus the exclusion partition, and the number of distance computations is significantly reduced through pre-computed distances;
- an upper bound of the number of accessed partitions is the number of hash levels;
- by storing the partitions in a sequential file and using the hashed structure as the main memory directory to its parts, the I/O costs are upper-bounded by the costs of optimized sequential scan, but typically much lower;
- contrary to tree organizations, the SH structure is suitable for parallel and distributed implementations.

In a way, the structure can be seen as a generalization of the sequential scan for similarity search and the hash structure for primary key (exact-match) retrieval, because these organizations form two special cases of SH. More precisely, when a separable partitioning is not possible, e.g. for high-dimensional spaces of uniformly distributed vectors [BGR+99], SH results in one partition, and all objects must be physically accessed. But also in this case, the pre-computed distances can save a lot of distance computations. On the other extreme, exact-match queries always require access to only one partition.

We believe that such approach is completely innovative as far as the generic metric spaces are concerned. The Similarity Search in High Dimensions via Hashing by Gionis, Indyk, and Motwani [GIM99] has completely different aim and is effectively reduced to a special case of vector spaces. Besides, in the Gionis et al. work Hashing structure some imprecision in the results is allowed, so the similarity search is only approximate.

3 Experimental Evaluation

Due to the limited space of this article, we present a very concise evaluation of a prototype SH system and measure its performance in terms of the bucket reads and distance computations to solve range queries. In order to allow comparison with previous designs, we use the same data sets and queries as suggested in [BO99]. Though other possibilities exist, we have used a ρ-split function that is based on the *vantage point* principle [Uh91].

3.1 Data sets and queries

In the this section we present the results of our experimental study for the evaluation of performance of SH structure. We used the synthetic sets of Euclidean vectors with two different distributions presented in [BO99]. The first set of experiments were conducted on uniformly distributed Euclidean vectors. We used 50,000 uniformly distributed vectors in 10-dimensional Euclidean space. For this set, all vectors were chosen randomly from the 10-dimensional unit hypercube. The second set of experiments were conducted on 20-dimensional Euclidean vectors generated in clusters of equal size. The clusters were generated as follows. First, a random vector is generated from the 20–dimensional unit hypercube with each side of size 1. This random vector becomes the seed for the cluster. Then the other vectors in the cluster are generated from this vector, or a previously generated vector in the same cluster, simply by altering each dimension of that vector with the addition of a random value chosen from the interval $[-\epsilon, \epsilon]$ where ϵ is a small constant equal to 0.2. The pair-wise distance distribution density of the vectors of these two sets are shown in Figure 1.

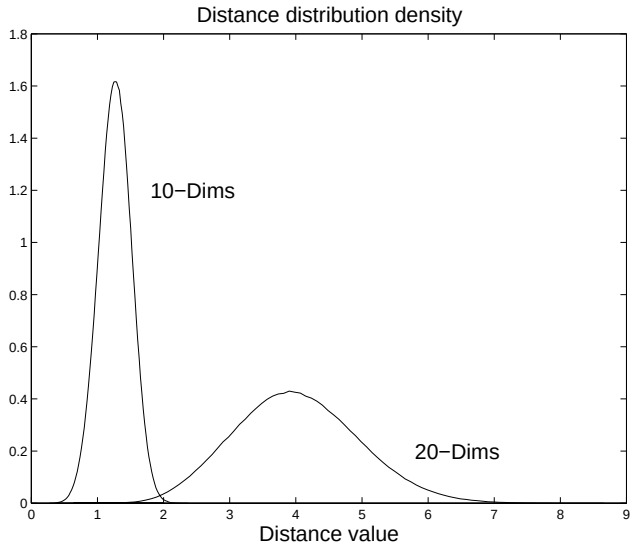

Figure 1: Distance distribution density for uniformly distributed vectors in 10-dimensional Euclidean and for 20-dimensional Euclidean generated in clusters.

3.2 Results

First we compare the search performance of the mvp-trees with the SH structure for the two different vector sets.

Figure 2 shows the performance results for the two data sets where the query points have been generated in the same way as the data points are; that is, they conform to the same uniform distribution. The result of each experiment is obtained by averaging the results of 100 search queries. As shown in the figure, the SH performs better than mvp-trees that is the structure which exhibits the best performance in [BO99]. The improvement ranges from 6 to 26 times for the 10-dimensional data-set and from 4.5 to about 14 for the 20-dimensional clustered data-set. Note that, from the perspective of the number of distance computations, the Search Naive Algorithm and the Exclusive Search give the same performance. This is due to the fact that we use the pre-computed distance obtained at insertion time. The proof of this behavior will be provided in a next full version of this paper.

The advantage of the Exclusive Search can be seen by analyzing the average number of buckets accessed during the search. As explained above for the Naive Search Algorithm, we access always one bucket for each level plus the exclusion bucket. Instead, the Exclusive Search can save I/O time by avoiding bucket accesses. Figure 3 shows the average number of buckets accessed for the 20-dimensional data-set, for a SH structure with 8 levels ($h = 8$).

4 Concluding Remarks

Similarity Hashing is a new organization for metric data that is able to efficiently support execution of similarity queries with radius up to a pre-defined value ρ. It is a multilevel hash organization of separable buckets, where objects are inserted with one bucket access that is determined by computation of a small number of distances. Such distances are remembered and reused for searching to avoid distance computations. When a query is executed, maximally one bucket is accessed on each level, because the others are sure not to contain qualifying data. Exclusion bucket is accessed

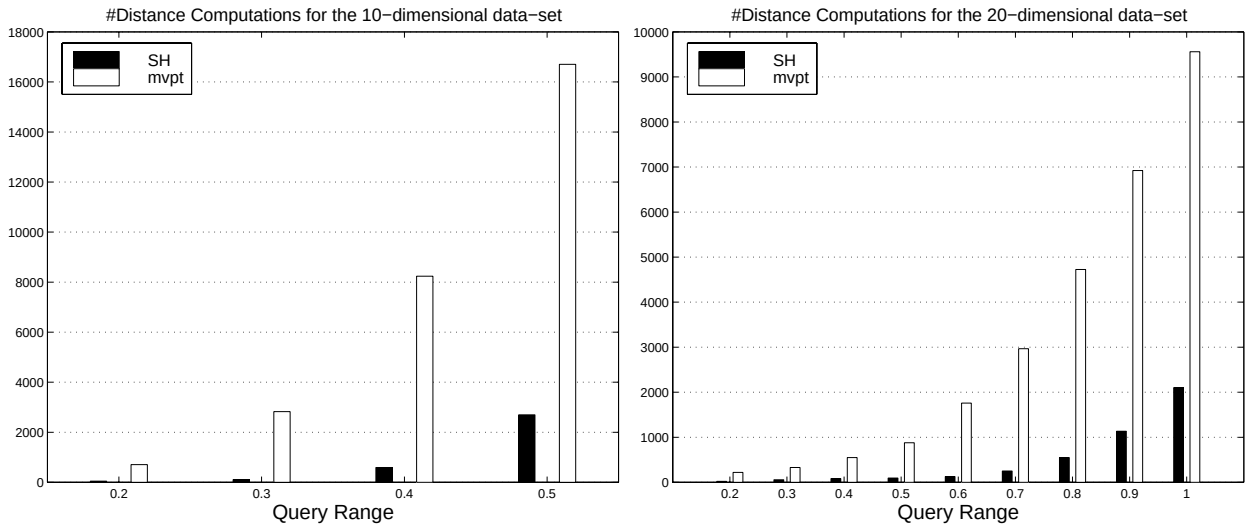

Figure 2: Search performance of mvp-tree and SH structure for 10-dimensional randomly generated Euclidean vectors and for 20-dimensional Euclidean generated in clusters.

for all queries. Experimental results conform expectations and demonstrate excellent performance especially for small query radii.

The structure is obviously suitable for parallel (distributed) implementation. The simplest possibility is to allocate each level on different parallel site (parallel independent disk or a computer) and uniformly divide the exclusion bucket among these sites. In this paper, we have only considered range queries, however the implementation of an algorithm for the *nearest neighbors* queries is quite straight forward, provided the distance to the last required neighbor is not more than ρ. There is also a possibility to run approximate queries by either considering radii greater than ρ or accessing only some of the query relevant partitions - such strategies can also be combined. Though only static organization was considered, growing data files can be managed through techniques known from dynamic hashing, for example linear hashing. We plan to investigate all this aspects in the near future.

However, this research also opens new ways of investigation. For example, can we use better split functions that the vantage point strategy? We have already tried to apply several other split strategies, but more research is needed to understand why some strategies perform better in certain situations while they are not so good in the others. Such questions can certainly be generalized into the SH structure design problem. In particular, what is the most suitable structure given the file, maximum search radius, and the relationship between the disk read and distance computation costs? It would be good to find the optimum number of levels and the number buckets on each of the levels. Notice that in our experiments, we did not use any optimization techniques in this respect.

References

[BGR+99] K.S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is Nearest Neighbor Meaningful? *ICDT'99*, pp. 217-235, Jerusalem, Israel, January 1999.

[BO97] T. Bozkaya and Ozsoyoglu. Distance-Based Indexing for High-Dimensional Metric Spaces. *Proceedings of the 1997 ACM SIGMOD Conference*, Tucson, pp. 357-368, 1997.

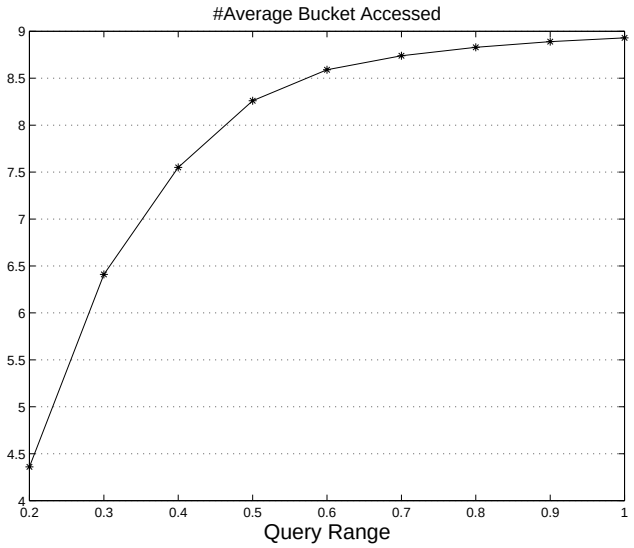

Figure 3: Average number of buckets accessed for 20-dimensional Euclidean vectors generated in clusters, for SH structure with $h = 8$.

[BO99] T. Bozkaya and Ozsoyoglu. Indexing Large Metric Spaces for Similarity Search Queries. *ACM TODS*, 24(3):361-404, 1999.

[Br95] S. Brin. Near Neighbor Search in Large Metric Spaces. *Proceedings of the 21st VLDB Conference*, pp. 574-584, 1995.

[GSZ00] C. Gennaro, P. Savino, and P. Zezula. Similarity Hashing for Metric Data. Sobmitted for publication.

[Ch94] T. Chiueh. Content-Based Image Indexing. *Proceedings of the 20th VLDB Conference*, pp. 582-593, 1994.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. *Proceedings of the 23rd VLDB Conference*, pp. 426-435, 1997.

[GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. Proceedings of *25th International Conference on Very Large Data Bases*, September 7-10, 1999, Edinburgh, Scotland, UK.pp. 518-529.

[TTS+00] C. Traina Jr, A. Traina, B. Seeger, and C. Faloutsos. Slim-trees: High Performance Metric Trees Minimizing Overlap Between Nodes. In *Proceedings of the 7th EDBT International Conference*, Konstanz, Germany, March 2000, pp. 51-56.

[Uh91] J.K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179, November 1991.

[Yi99] P.N. Yianilos. Excluded Middle Vantage Point Forests for Nearest Neighbor Search. Tech. rep., NEC Research Institute, 1999, Presented at Sixth DIMACS Implementation Challenge: Nearest Neighbor Searches workshop, January 15, 1999.

[WSB98] R. Weber, H.-J. Schek, and S. Blott. A Quantitative Analysis and Performance Study for Similarity Search Methods in High-Dimensional Spaces. *VLDB'98*, pp. 194-205, New York, NY, August 1998.