Pattern Recognition Letters 54 (2015) 69–74

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

A novel indexing scheme for similarity search in metric spaces ☆

Umut Tosun*

Department of Computer Engineering, Baskent University, Baglica Kampusu, Eskisehir Yolu 20. km 06530, Ankara, Turkey

ARTICLE INFO

Article history:
Received 21 June 2014
Available online 23 December 2014

Keywords:
Metric space
Metric access methods
Kvp
Hkvp
M-Tree
Slim-Tree

ABSTRACT

Sparse spatial selection (SSS) allows insertions of new database objects and dynamically promotes some of the new objects as pivots. In this paper, we argue that SSS has fundamental problems that result in poor query performance for clustered or otherwise skewed distributions. Real datasets have often been observed to show such characteristics. We show that SSS has been optimized to work for a symmetrical, balanced distribution and for a specific radius value. Our main contribution is offering a new pivot promotion scheme that can perform robustly for clustered or skewed distributions. We show that our new indexing scheme performs significantly better than tree-based dynamic structures while having lower insertion costs.

1. Introduction

Index structures are used to reduce the number of distance computations. The index is built using the objects in the database. When performing queries, some of the objects are eliminated using triangle inequality without computing the distance to the query object. Static indices are built using the whole collection, whereas dynamic methods allow insertion and deletion operations. Studies show that the actual query times are either dominated by or in direct proportion to the number of distance computations.

The global pivot based methods are static in nature, except for the recent structure, Sparse spatial selection (SSS) [2]. SSS solves two problems at once: how many pivots to keep for a particular database, and which new objects to promote as pivots. We will show that SSS is not designed very robustly in both of its missions under different distribution types and for different radius values. The M-Tree [5] is a dynamic structure with the ability to handle data files that change size dynamically. It handles frequent deletions and insertions successfully while optimizing the I/O performance using the covering radius and node splitting. Maintaining the covering radius allows organizing disk blocks. Node splitting aims to distribute the objects into two subtrees when there is an overflow. A new node is allocated and entries are redistributed. The node split requires two new pivots to be selected. The corresponding covering radii are adjusted to reflect the membership of two new nodes. Slim-Tree [9] is an improvement over the M-Tree by employing a more efficient splitting method. It generates the minimum spanning tree of the objects and removes the longest arc. The remaining two subsets are the new nodes after node splitting. DF-Tree [10] gives the best performance among the tree based methods. It uses the foci concept of Omni [6] over Slim-Tree and defines an adaptive structure by when to update/how to update algorithms to dynamically modify the global representative set.

Selection of more pivots increases the insertion costs and the number of distance computations to all pivots when executing a query. Therefore, we use a calibration parameter, which is called drop rate, to dynamically eliminate inefficient pivots. Our first contribution is to devise a new method of automatically adjusting the drop rate. Thus, when the pivot promotion criteria promotes too much objects as pivots, or the number of pivots is optimal for high radius values but too much for lower radius values; the structure can still avoid computing distances to some of the pivots. The second contribution is to avoid assigning too few pivots. For example, SSS fails when the distribution is skewed toward high distance values. We will use a distribution sensitive method of deciding when to create a new pivot. The proposed method dynamically adapts to changes in the database by selecting new pivots if needed.

2. HKVP as a global pivot based method

A pivot is more effective for objects that are close to or distant from it. Kvp [4] finds such pivots, and keeps only the distances to these promising pivots. It evaluates the distance relations between the pivots and database elements at construction time. Only the most promising distances are stored, and this reduces the CPU overhead while decreasing the space requirements.

The underlying working principle of global pivot based methods is that we have a very large database with a very expensive distance function. The number of pivots is assumed to be very limited with respect to the database size. There are exceptions to this assumption. Database size may be limited or application may require high number

☆ This paper has been recommended for acceptance by M.A. Girolami.

* Tel.: +90 312 246 6661, +903124832357; fax: +90 312 246 6660.
E-mail address: utosun@baskent.edu.tr, tosun@ceng.metu.edu.tr

http://dx.doi.org/10.1016/j.patrec.2014.12.004
0167-8655/

of pivots. The ratio of number pivots to the database size is not always low as assumed in Kvp [4].

A typical pivot based structure begins search process by computing all distances between pivots and query object. With the assumption of a pivot not eliminating an object is f, after processing k pivots, there would still be f^k objects that remain not eliminated. Hence, the total cost of a query is expressed by the equation:

$$\text{Cost}(q, r) = k + nf^k \quad (1)$$

HKvp [4] tries to find an optimum k value for a given query object and radius. Classical pivot based methods including Kvp fail to find the query result with a meaningful number of distance computations when the solution requires fewer number of distance computations than k. After an optimal number of pivots is reached, second part of Eq. (1) is dominated by the first part which is the cost of calculating pivot distances with the query object.

3. Spatial selection of sparse pivots

Even though search algorithms are based on pivots to improve the performance, almost all proximity search algorithms based on pivots choose them randomly [3]. It is a well known fact that search performance is affected from selection of pivots. Heuristics to choose pivots better than random only try to choose objects that are far from each other. In spite of the fact that good pivots are outliers, selecting pivots as outliers do not guarantee the best pivot set [3,11].

Although a tree-based structure, GNAT [1] employs a pivot selection algorithm to decide which subset of the objects covered by a node as pivots. The algorithm works on a sample subset of the objects, but there is nothing that would prevent applying the same algorithm on the whole set except for efficiency considerations. GNAT algorithm first selects a random object and works incrementally. At each step, it chooses the object that maximizes the minimum distance to the current set of pivots. This guarantees that the new pivot is as far away from the pivots as possible.

Sparse Spatial Selection (SSS) [2] is a pivot selection method that is based on the GNAT pivot selection algorithm. According to the experiments in [2], SSS is more efficient than previously defined methods. It adapts itself to the dimensionality of the database and the number of pivots to be used is defined by the method itself. It allows object insertions and deletions unlike the previously defined techniques. It is suitable for secondary storage with these properties.

The pivot set starts with first inserted object of the database. Let (X, d) be a metric space, $S \subseteq X$ an object collection, and $M = \max d(s, s^*)$; $s, s^* \in S$. Then an object is selected as a pivot if and only if its distance to any pivot in the current set of pivots is equal to or greater than $M\alpha$ where α is a constant around 0.4. This constant is obtained experimentally [2]. An object in the database is chosen as a new pivot if it is located at more than a fraction of maximum distance with respect to all current pivots. The pivots in SSS are not too far from the rest of the objects in the collection and this means that they are well distributed in space. They are not too far from each other and rest of the objects in the database. This is a desirable property good pivots must have [2,3].

The set of pivots adopt itself to the growth of the database. When an object s_{new} is inserted to the database, it is compared against the pivots already selected. The method is efficient, dynamic and adaptive. However, it selects a large amount of objects as pivots or an inadequate amount of objects as pivots when distribution is clustered or not normally distributed.

4. Dynamic HKvp

Kvp based methods outperform other pivot based methods in terms of space complexity while not causing a significant increase on query costs. Furthermore, they reduce the CPU overhead significantly. Pivot based methods like AESA [12] and LAESA [8] store all the distances between pivots and objects. This fact limits the usage of these methods in large databases. Even though pivot-based methods outperform tree-based methods in terms of performance, they use a static pivot set except SSS [2] and this causes several problems when database size grows. First of all, the number of pivots should be increased to perform at a sublinear complexity. Second, initial pivots may perform poorly in eliminating newly inserted objects coming from possibly different regions.

SSS [2] is a dynamic method introduced as a LAESA variant that allows insertions of new database objects. It promotes some of the new objects as pivots dynamically. However, it has serious drawbacks that result in poor query performance for clustered or otherwise skewed distributions. SSS is optimized for a symmetrical, balanced distribution and for a specific radius value.

Tree based methods are dynamic in nature. They select new pivots when there are node splits. The split algorithm promotes some objects as representatives. However, tree based methods are more complex to implement and they are more expensive in terms of number of distance computations than pivot based methods since they use less pivots. To the best of our knowledge, the best performing tree based algorithm is DF-Tree [10] which uses extra global representatives in addition to the pivots of the tree structure. It uses the Omni-HF algorithm [6] to dynamically select these global representatives when there are objects inserted into the database. We can think of the DF-Tree concept as the Slim-Tree [9] implementation of Omni [6] with global representatives. HKvp structure has the advantage of being able to increase the number of pivots to improve the query performance. Using more pivots is not problematic in some cases like Kvp, since HKvp optimizes the number of pivots to be used. This is performed by a parameter of HKvp called *drop rate* [4]. Even though HKvp has the drop rate parameter, currently it does not have an optimization scheme to determine the parameter effectively. Current implementation takes the drop rate value as a parameter.

From these discussions, we end up with two problems to make HKvp dynamic: drop rate optimization should be satisfied and a dynamic pivot selection algorithm should be adapted to HKvp. In this section, we propose methods to estimate drop rate parameter at the query time. We implement the SSS pivot selection algorithm, which is stated to be the best performing among the others, to HKvp. As stated in [2], we know that SSS pivot selection technique has a drawback in clustered distributions, and in distributions which are not normally distributed. We propose an alternative to SSS which is called distribution based pivot promotion (DBPP) to overcome this difficulty. SSS promotes new objects based on a fixed ratio of the maximum possible distance. The suggested method takes the distribution of distances into account, not just the maximum possible distance. In this way, we may select the pivots in well distributed amounts even for a clustered distribution or a distribution that is skewed.

Rather than calculating the distances of k pivots to the query object, HKvp eliminates pivots as well as objects using inter-pivot distances and computes only the distances for promising pivots. Dynamic HKvp stores the inter-pivot distances in a matrix of size $k \times (k-1)/2$ and it uses the closest and furthest distances for each object in a storage of $(n-k) \times 2$. Thus, the space complexity for HKvp is $O(k \times (k-1)/2 + (n-k) \times 2)$ which is $O(k^2/2 + 2n - 5k/2)$. Algorithmic complexity is equal to Eq. (1) in the worst case.

4.1. Pivot selection

HKvp [4] computes distances to promising pivots. However, determining how valuable a pivot is a difficult task. It has been shown that a good performance is achieved by selecting the next pivot to process as the one with the lowest lower bound for the distance to the query object [8]. After the chosen pivot has its distance evaluated with the