Available online at www.sciencedirect.com
SCIENCE DIRECT®
Data & Knowledge Engineering 57 (2006) 1–36
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Cost models for distance joins queries using R-trees

Antonio Corral [a], Yannis Manolopoulos [b,*], Yannis Theodoridis [c], Michael Vassilakopoulos [d]

[a] *Department of Languages and Computation, University of Almeria, 04120 Almeria, Spain*
[b] *Department of Informatics, Aristotle University, GR-54124 Thessaloniki, Greece*
[c] *Department of Informatics, University of Piraeus, GR-18534 Piraeus, Greece*
[d] *Department of Informatics, Technological Educational Institute of Thessaloniki, P.O. Box 141, GR-57400 Thessaloniki, Greece*

Received 25 September 2004; received in revised form 3 February 2005; accepted 24 March 2005
Available online 26 April 2005

Abstract

The *K*-Closest-Pairs Query (*K*-CPQ), a type of distance join in spatial databases, discovers the *K* pairs of objects formed from two different datasets with the *K* smallest distances. Recently, branch-and-bound algorithms based on R-trees have been developed in order to answer *K*-CPQs efficiently. For query optimization purposes, analytical models are needed to estimate the processing cost of a specific query in order to evaluate alternative execution plans. In this paper, we combine techniques that have been used for the analysis of nearest neighbor and spatial join queries, and derive the performance cost (in terms of disk accesses) of *K*-CPQs using R-trees. Moreover, we present two interesting extensions of the cost model for *K*-CPQs, one exploiting the buffering management using R-trees and another for a second type of distance join, the so-called buffer queries. The proposed cost models are verified under a variety of distributions in 2-dimensional space on both synthetic and real datasets, shown to achieve accurate estimations of the measured experimental results.

Keywords: Cost models; Distance join queries; R-trees; Buffer model; Performance analysis

* Corresponding author. Tel.: +30 31 991912; fax: +30 31 991913.
E-mail addresses: acorral@ual.es (A. Corral), manolopo@csd.auth.gr (Y. Manolopoulos), ytheod@unipi.gr (Y. Theodoridis), vasilako@it.teithe.gr (M. Vassilakopoulos).

0169-023X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2005.03.004

1. Introduction

The role of spatial databases is continuously increasing in many modern applications during last years. Mapping, urban planning, transportation planning, resource management, geomarketing, archeology and environmental modeling are just some of these applications. The key characteristic that makes a spatial database a powerful tool is its ability to manipulate spatial data, apart from storing and representing them. The most basic form of such a manipulation is answering queries related to the spatial properties of data. Some typical spatial queries include selections with respect to a reference object (point location query; range query; nearest neighbor query) and joins between two spatial datasets (overlap or distance join).

In this paper, the cost of a spatial query that combines join and nearest neighbor queries is studied. It is called *K Closest Pairs Query* (*K*-CPQ), which is a type of distance join query. It is defined as follows: Given two different datasets S_1 and S_2 of N_{S_1} and N_{S_2} points, respectively, a *K*-CPQ retrieves the $1 \leqslant K \leqslant N_{S_1} * N_{S_2}$ different pairs of points from $S_1 \times S_2$ with the *K* smallest distances between all possible pairs of points that can be formed by choosing one point of S_1 and one point of S_2. Like a join query, all pairs of objects are candidates for the result. Like a nearest neighbor query, the *K* nearest neighbor property is the basis for the final ordering. In the degenerate case of $K = 1$, the closest pair of spatial objects is discovered. Consider, for instance, a spatial database where the datasets represent the cultural landmarks and the populated places of North America. A *K*-CPQ will discover the *K* closest pairs of cities and cultural landmarks.

Tree-based algorithms for *K*-closest pairs search follow branch-and-bound techniques that aim at finding quickly a good set of pairs, in order to prune the search space as soon as possible. The earliest *K*-CPQ algorithms were proposed in [11,13] for R-trees [18], although they can be modified for any data-partition index (e.g. LSD^h-tree [19]). These algorithms report the elements of a query result all together, at the end of the algorithm's execution, assuming that the cardinality (*K*) of the result is known in advance. In such algorithms, the R-trees are traversed in a Depth-First (Sorted) or Best-First (Heap) manner, and the *MinDist* distance (MINMINDIST in [11,13]) is applied for pruning the search space effectively, since *MinDist* is a generalization of the minimum distance between points and MBRs (Minimum Bounding Rectangles, which geometrically encloses spatial objects). *MinDist* can be applied to pairs of any kind of elements (i.e. MBRs or points) stored in R-trees during the computation of branch-and-bound algorithms for *K*-CPQ.

In the first algorithm presented in [11,13], the R-tree is traversed in a Depth-First manner. Specifically, starting from the two roots, all possible pairs of MBRs are sorted in ascending order of *MinDist*, and the pair of MBRs with the lowest value is visited first. The process is repeated recursively (internal R-tree nodes) until the leaf level where the first potential set of closest pairs is retrieved. During backtracking to the upper levels, the algorithm only visits entries whose *MinDist* is smaller than or equal to the distance of the *K*th closest pair found so far. Depth-First search, finds many approximate solutions quickly (although, it may take long time to obtain the best solution if it does not traverse the search path in the right direction) and improves their quality along time. In order to overcome these problems, in [11,13] a Best-First *K*-CPQ algorithm was also proposed. This algorithm keeps a minimum binary heap with the references to pairs of nodes (characterized by their MBRs) of the two different R-trees $\langle MinDist, \mathrm{Addr}_{R_1}, \mathrm{Addr}_{R_2} \rangle$ accessed so far, and visits the pair of MBRs with the minimum *MinDist* in the heap. Although, this alternative

can theoretically provide the optimal cost (i.e., it only visits the nodes necessary for obtaining the K closest pairs), its performance in practice can suffer if the available memory is smaller than the required heap. In these situations part of the heap must migrate to disk, which may incur additional disk accesses, affecting the query performance.

An important and interesting research direction is the query-cost modeling. Cost models are used to rank and select the promising processing strategies in spatial databases [30], given a spatial query and spatial datasets. Cost models are needed to estimate the selectivity of spatial search and join operations toward comparison of execution costs of alternative processing strategies for spatial operations during query optimizations. Several cost models have been proposed to estimate, in terms of node accesses, the performance of nearest-neighbors and join queries in the context of the R-trees, but more work is needed [30].

The analysis of query performance in spatial access methods is important for query optimization and for evaluating access method designs. Most I/O cost models *unrealistically* assume uniformity and independence to make the analysis tractable. However, real data overwhelmingly disobey these assumptions; they are typically skewed and often have dependences between dimensions. In this paper, we derive formulae to estimate the number of disk accesses for K-CPQ between two R-trees, for real datasets. These formulae depend on several input parameters, highlighting the correlation exponent (ρ) and number of pairs in the final result (K). Moreover, our analysis provides a cost model of a typical (non-uniform) workload using the so-called biased query model [22], which assumes that *queries are more probable in high-density areas of the address space*. Moreover, we have successfully extended our cost model to other distance join queries as *buffer queries* [9] and to the study of including buffering (LRU buffer model, [6]) in K-CPQ using R-trees, which theoretical results have been very similar to the experimental ones in terms of buffer hit probability and number of R-tree node accesses.

The rest of the paper is organized as follows. In Section 2, we review the R-tree family as spatial access method, describe the branch-and-bound algorithms for K-CPQ and survey previous work on cost models for nearest neighbors and join queries over R-trees. Section 3 presents our cost model and the formulae that estimate the K-CPQ performance, which is a generalization of different cost models as nearest neighbor and spatial join queries using R-trees. In Section 4, two interesting extensions of the cost model are discussed: a cost model for buffer queries (another type of distance join) and a buffer model (LRU) for K-CPQ using R-trees. In Section 5, experimental results of the proposed cost model are presented with respect to R-tree implementations for different data distributions (synthetic (uniform) and real). Finally, Section 6 presents the conclusions of this research paper and gives directions for future work.

2. Related work

2.1. R-trees

An R-tree [18] is a hierarchical, height balanced multidimensional data structure, designed to be used in secondary storage and it is a generalization of B-trees for multidimensional data spaces. It is used for the dynamic organization of a set of d-dimensional objects represented by their d-dimensional MBRs. These MBRs are characterized by *min* and *max* points of hyper-rectangles

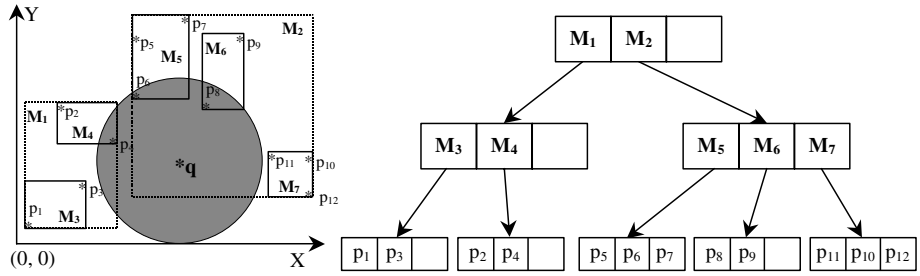

Fig. 1. An example of an R-tree.

with faces parallel to the coordinate axes. Using the MBR instead of the exact geometrical representation of the object, its representational complexity is reduced to two points, where the most important object features (position and extension) are maintained. Consequently, the MBR is an approximation widely employed.

The rules obeyed by an R-tree are as follows: leaves reside on the same level; each leaf contains entries of the form (MBR, Oid), such that MBR is the minimum bounding rectangle that encloses the object determined by the identifier Oid; internal nodes contain entries of the form (MBR, Addr), where Addr is the address of the child node and MBR is the minimum bounding rectangle that encloses MBRs of all entries in that child node; nodes (except possibly for the root) of an R-tree of class $(Cmin, Cmax)$ contain between $Cmin$ and $Cmax$ entries, where $Cmin \leqslant \lceil Cmax/2 \rceil$ ($Cmax$ and $Cmin$ are also called maximum and minimum branching factors or fan-out); the root contains at least two entries, if it is not a leaf. Fig. 1 depicts the 2-dimensional points along with their MBRs, on the left and the corresponding R-tree of class (2, 3), on the right.

Many variations of R-trees have appeared in the literature (an exhaustive survey can be found in [17]). One of the most popular and efficient variations is the R*-tree [2]. The R*-tree added two major enhancements to the R-tree, in case that a node overflows. First, rather than just considering the area, the node-splitting algorithm in the R*-tree also minimized the perimeter and overlap enlargement of the minimum bounding rectangles. It tends to reduce the number of subtrees to follow for search operations. Second, the R*-tree introduced the notion of forced reinsertion to make the tree shape less dependent to the insertion order. When a node overflows, it is not split immediately, but a portion of entries of the node is reinserted from the tree root. With these two enhancements, the R*-tree generally outperforms original R-tree. It is commonly accepted that the R*-tree is one of the most efficient R-tree variants.

2.2. Algorithms for K-CPQ using R-trees

If we assume that the datasets are indexed on any tree-like structure belonging to the R-tree family, then the main objective while answering this type of distance-based query is to reduce the search space. In [11], a generalization of the function that calculates the minimum distance between points and MBRs (*MinDist*) was presented. $MinDist\ (M_1, M_2)$ calculates the minimum distance between two MBRs M_1 and M_2. If any of the two (both) MBRs degenerates (degenerate) to a point (two points), then we obtain the minimum distance between a point and an MBR [29] (between two points).

Definition 1. *MinDist*(M_1, M_2).

Given two MBRs $M_1 = (a, b)$ and $M_2 = (c, d)$, in $E^{(d)}$ (d-dimensional Euclidean space)

$$M_1 = (a, b), \text{ where } a = (a_1, a_2, \ldots, a_d) \text{ and } b = (b_1, b_2, \ldots, b_d) \text{ such that } a_k \leqslant b_k \ \forall 1 \leqslant k \leqslant d$$

$$M_2 = (c, d), \text{ where } c = (c_1, c_2, \ldots, c_d) \text{ and } d = (d_1, d_2, \ldots, d_d) \text{ such that } c_k \leqslant d_k \ \forall 1 \leqslant k \leqslant d$$

we define *MinDist*(M_1, M_2) as follows:

$$MinDist(M_1, M_2) = \sqrt{\sum_{k=1}^{d} l_k^2}, \text{ such that } l_k = \begin{cases} c_k - b_k, & \text{if } c_k > b_k \\ a_k - d_k, & \text{if } a_k > d_k \\ 0, & \text{otherwise} \end{cases}$$

MinDist(M_1, M_2) serves as *lower bound function* of the Euclidean distance from the K closest pairs of objects enclosed by the MBRs M_1 and M_2 (*lower-bounding property*). *MinDist* is monotonically non-decreasing with the R-tree heights [13]. The general pruning mechanism for K-CPQs over R-trees is the following: *if MinDist*$(M_1, M_2) > z$, *then the pair of MBRs* (M_1, M_2) *will be discarded*, where z is the distance value of the Kth closest pair that has been found so far.

In order to design an efficient algorithm that retrieves the $1 \leqslant K \leqslant N_{S_1} * N_{S_2}$ different pairs of points from $S_1 \times S_2$ (where both point datasets are indexed by R-trees), the concept of synchronous tree traversals following a Depth-First or Best-First search can be applied for query processing [13]. Since Best-First search is I/O optimal (in absence of buffers, it only visits the necessary nodes for obtaining the query result [4]) and Depth-First search accesses more partitions than actually necessary, we are going to focus on the first searching strategy for the K-CPQ algorithm that will guide our cost analysis.

The Best-First K-CPQ algorithm needs to keep a minimum binary heap (H) [10] with the references to pairs of internal nodes (characterized by their MBRs) accessed so far from the two different R-trees and their minimum distance ($\langle MinDist, \mathrm{Addr}_{R_1}, \mathrm{Addr}_{R_2} \rangle$). It visits the pair of MBRs (nodes) with the minimum *MinDist* in the H, until it becomes empty or the *MinDist* value of the pair of MBRs located in the root of H is larger than the distance value of the Kth closest pair that has been found so far (z). To keep track of z, we also need an additional data structure that stores the K closest pairs discovered during the processing of the algorithm. This data structure is organized as a maximum binary heap (*K-heap*) [10] and will hold pairs of objects according to their minimum distance (the pair with the largest distance resides in the root). In the implementation of K-CPQ algorithm we must consider the following cases: (1) initially the (K-heap is empty (z is initialized to ∞), (2) the pairs of objects reached at the leaf level are inserted in the K-heap until it gets full (z keeps the value of ∞), (3) if the distance of a new pair of objects discovered at the leaf level is smaller than the distance of the pair residing in the K-heap root, then the root is extracted and the new pair is inserted in the K-heap, updating this data structure and z (distance of the pair of objects residing in the K-heap root). Such an algorithm for two R-trees (R_1, R_2) with the same heights appears in the Fig. 2 (*input*: two R-trees (R_1 and R_2) indexing two point datasets and a maximum binary heap with size K to store the final result (K-heap); *output*: K-heap (maximum binary heap with pairs of points ordered according to their distances)). If the R-trees have different heights, we can use *fix-at-leaves* technique [11].

```
01 Create and initialize H, and z = ∞;
02 for each pair of MBRs <M1i, M2j> in <root_R1, root_R2> do
03     H.insert(MinDist(M1i, M2j), Addr_M1i, Addr_M2j);
04 enddo
05 while (not H.isEmpty()) and (H.MinimumDistance() ≤ z) do
06     minimum = H.deleteMinimum();
07     node_R1 = R1.readNode(minimum.Addr_M1);
08     node_R2 = R2.readNode(minimum.Addr_M2);
09     if (node_R1 and node_R2 are internal nodes) then
10         for each pair of MBRs <M1i, M2j> in <node_R1, node_R2> do
11             if (MinDist(M1i, M2j) ≤ z) then
12                 H.insert(MinDist(M1i, M2j), Addr_M1i, Addr_M2j);
13             endif
14         enddo
15     else
16         for each pair of points <P1i, P2j> in <node_R1, node_R2> do
17             distance = MinDist(P1i, P2j);
18             if (K-heap.isFull()) then
19                 if (distance ≤ z) then
20                     K-heap.deleteMaximum();
21                     K-heap.insert(distance, P1i, P2j);
22                     z = K-heap.getMaximum();
23                 endif
24             else
25                 K-heap.insert(distance, P1i, P2j);
26             endif
27         enddo
28     endif
29 enddo
30 Destroy H;
```

Fig. 2. Best-First *K*-CPQ algorithm using R-trees.

As we can observe from Fig. 2, the *K*-CPQ is a combination of spatial join and *K*-nearest neighbor queries following a Best-First search [13]. Like a spatial join query, all pairs of objects are candidates for the final result. Like a *K*-nearest neighbor query, distance functions form the basis for pruning mechanism and the final ordering. Therefore, a combination of both cost models would be reasonable to adopt in order to propose a cost model for the *K*-CPQ using R-trees.

2.3. Cost models using R-trees

The first attempt to provide an analysis for R-tree index structures appeared in [16], where a model that estimates the performance of R-trees and R^+-trees for selection queries was proposed. Later, [21,26], independently, presented a formula that calculates the average number of page accesses in an R-tree accessed by a query window as a function of the average node size and the query window size. Due to the high impact of similarity queries (mainly of the nearest neighbor query), a considerable number of different algorithms using R-trees and the respective cost models for estimating the number of page accesses have already been proposed in the last years. Moreover, most related work on join processing using multidimensional access methods is based on spatial intersect joins using R-trees. In this section, we are going to review the most represen-

tative research efforts on analytical performance studies for nearest neighbor and spatial intersect join queries using R-trees.

2.3.1. Cost models for nearest neighbor queries using R-trees

To the best of our knowledge, [27,22] are the most representative papers for analysis based on fractals of nearest-neighbor queries on R-trees. These models try to account for the non-uniformity of the data by modeling it using a few global parameters, like the fractal dimension. This kind of models estimates the average page geometry assuming square pages and the average query shape in a similar way as the uniform techniques. However, instead of using the embedding dimension, they consider the fractal dimension. In [27] results for estimating data page accesses of R-trees when processing nearest neighbor queries in a Euclidean space were reported. Since it is difficult to determine accesses of pages with rectangular regions for spherical queries, the authors approximate query spheres by minimum bounding and maximum enclosed cubes and thus determine lower- and upper-bounds in average-case formulae for the number of page accesses for 1-nearest-neighbor search. These bounds diverge rapidly with the increase of fractal dimensions. In [22], closed-form formulae for K-nearest-neighbor queries, for arbitrary K were proposed. Moreover, such formulae can be simplified and, thus, lead to fundamental observations that deflate the *dimensionality curse*.

In [4], a cost model for query processing in high-dimensional data spaces was presented. It provides accurate estimations for nearest neighbor queries and range queries using the Euclidean distance, and assumptions of independence are implicit in the formulae. This paper introduces the concept of the Minkowski sum to determine the access probability of a rectangular page for spherical queries (i.e. range queries and nearest neighbor queries). The Minkowski sum can be used to determine the index selectivity of distance-based join operations. Finally, in [7] an excellent study that provides accurate estimations of the number of pages accesses for range queries and nearest neighbor queries under Euclidean and maximum distances was presented. The boundary effects are considered and the concept of fractal dimension is used to take into account the effects of correlated data.

Recently in [33], a cost model for K-NN queries was also proposed for low and medium dimensionalities, using a new technique based on the concept of vicinity rectangles and Minkowski rectangles (instead of the traditional vicinity circles and Minkowski regions, respectively), which simplifies the resulting equations with minimal computational overhead. They confirmed the accuracy of the model through extensive experiments using the R*-tree as the underlying spatial access method (with uniform and real datasets) and, demonstrated its applicability and effectiveness by incorporating it in various query optimization problems related to nearest neighbor search.

2.3.2. Cost models for spatial join queries using R-trees

Considering join queries, in [1], analytical formulae for cost and selectivity, based on the R-tree analysis of [21], were proposed. The basic idea of [1] was the consideration of one of the datasets as the underlying database and the other dataset as a source for query windows in order to estimate the cost of a spatial join query based on the cost of range queries. Experimental results showing the accuracy of the selectivity estimation formula were also presented in that paper.

In [20], a cost model for spatial joins using R-trees was proposed. It was the first attempt to provide an efficient formula for join performance by distinguishing two cases: considering either

zero or nonzero buffer management. Using the analysis of [21] and assuming knowledge of R-tree properties, this paper provides two formulae, one for each of the above cases. The efficiency of the proposed formulae was demonstrated by comparing analytical estimations with experimental results for varying buffer sizes (with the relative error being around 10–20%).

In [32], a model that predicts the performance of R-tree-based structures for selection (point or range) queries and an extension of this model for supporting join queries (overlap operator between spatial objects, although any other spatial operator could be used instead) were presented. The proposed cost formulae are functions of data properties only, namely, the *cardinality* and the *density* in the workspace, and, therefore, can be used without any knowledge of the R-tree index properties. They are applicable to point or non-point datasets and, although they make use of the uniformity assumption, they are also adaptive to non-uniform distributions, which usually appear in real applications, by reducing its effect from global to local level (i.e., maintaining a density surface and assuming uniformity on a small subarea of the workspace). Experimental results on both synthetic and real datasets showed that the proposed analytical model was very accurate, with the relative error being usually around 10–15% when the analytical estimate is compared to cost measurements using the R*-tree. In addition, for join query processing, a path buffer was considered and the analytical formula was adapted to support it. The performance saving due to the existence of such a buffering mechanism was highly affected by the sizes (and height) of the underlying indices and reached up to 50% for two-dimensional datasets. The proposed formulae and guidelines could be useful tools for spatial query processing and optimization purposes, especially when complex spatial queries are involved.

In [8], an analytical model and a performance study of the similarity join operation on indexes were presented. In this context, the optimization conflict between CPU and I/O optimization was studied. To solve this conflict, a complex index architecture (Multipage Index, MuX) and join algorithm (MuX-join), which allows a separate optimization of the CPU time and the I/O time, was proposed. This architecture (MuX), which is based on R-trees (for a fast index construction, the bottom-up algorithm for X-tree construction [5] was adopted), utilized large primary pages, which are subject to I/O processing and optimized for this purpose. The primary pages accommodate a secondary search structure to reduce the computational effort. The experimental evaluation using the join algorithm (MuX-join) over the index architecture (MuX) showed a good performance.

3. A cost model for *K*-CPQ using R-trees

3.1. Preliminaries

According to the analysis of R-tree joins in [32,15,22] and assuming (without loss of generality) that the search space is a normalized d-dimensional unit hypercube $[0,1]^d$, we will consider the symbols shown in Table 1 for the analysis of the cost model.

f_{R_i}, h_{R_i}, N_{R_i,l_i}, and $s_{R_i,l_i,k}$ can be easily estimated under the following two assumptions [32,14]:

- *Squaredness assumption*. We consider square node MBRs (hypercubes), since this is a reasonable property for *good* R-trees [21]. We make this assumption in order to make modeling

Table 1
List of symbols

Symbols	Description
d	Number of dimensions ($1 \leqslant k \leqslant d$), i.e. embedding dimension
ρ	Correlation exponent of two points datasets (*pair-count exponent* [15])
S_i	Point datasets that are indexed in the R-tree R_i with cardinality N_{S_i}
$Cmax_{R_i}$	Maximum number of objects per node (maximum branching factor) in the R-tree R_i
$Uavg_{R_i}$	Average node utilization in the R-tree R_i
f_{R_i}	Effective R-tree node capacity or average R-tree node fan-out in the R-tree R_i
h_{R_i}	Height of the R-tree R_i
N_{R_i}	Number of points indexed in the R-tree R_i (cardinality of S_i, $N_{S_i} = N_{R_i}$)
N_{R_i,l_i}	Average number of R-tree nodes in the R-tree R_i at level l_i ($N_{R_i,\text{root}} = N_{R_i,0} = 1$)
$Nodes_{R_i}$	Total number of R-tree nodes in R-tree R_i
$s_{R_i,l_i,k}$	Average extent of node MBRs of the R-tree R_i at level l_i on dimension k ($1 \leqslant k \leqslant d$). In other words, it is the average side length of node MBRs of R_i at level l_i on dimension k
$dist_{cp}(K)$	Distance of the Kth closest pair
$\sigma(K)$	K-CPQ index selectivity
$NA_{cp}(K)$	Average number of pages (R-tree nodes) retrieved by a K-CPQ

manageable. R-tree-like structures as the R*-tree and the X-tree try to produce such node MBRs.

- *Biased query model*. We assume that queries are more probable in high-density areas of the address space and that the anchor points are allowed to land only on data points. Thus, high-density areas attract more candidates for the query result.

$$f_{R_i} = Cmax_{R_i} * Uavg_{R_i}, \quad h_{R_i} = 1 + \left\lceil \log_{f_{R_i}} \left(\frac{N_{R_i}}{Cmax_{R_i}} \right) \right\rceil$$

$$N_{R_i,l_i} = \frac{N_{R_i}}{f_{R_i}^{h_{R_i}-l_i}}, \quad Nodes_{R_i} = \sum_{l_i=0}^{h_{R_i}-1} N_{R_i,l_i}$$

where $l_i = 0, 1, \ldots, h_{R_i} - 1$ (the root is assumed at level $l_i = 0$ and leaves at $l_i = h_{R_i} - 1$).

Moreover, in the case of *low-dimensionality*, for uniform data and considering the MBR effect, the average extent s_{R_i,l_i} of a node MBR of the R-tree R_i (with height h_{R_i}) at level l_i (assuming that all node MBRs at the same level have similar extents and each node MBR has identical extent on each dimension) is given by the following formula [7]:

$$s_{R_i,l_i,k} \approx s_{R_i,l_i} = \left(1 - \frac{1}{f_{R_i}}\right) * \left(\min \left\{ \left(\frac{f_{R_i}^{h_{R_i}-l_i}}{N_{R_i}} \right), 1 \right\} \right)^{1/d}$$

where $l_i = 0, 1, \ldots, h_{R_i} - 1$ (the root is assumed at level $l_i = 0$ and leaves at $l_i = h_{R_i} - 1$).

In order to obtain $s_{R_i,l_i,k}$ (*real* average extent of node MBRs of the R-tree R_i at level l_i on dimension k ($1 \leqslant k \leqslant d$)) for a given *real* R*-tree R_i (R*-trees tend to generate squared MBRs in its structure [14], i.e. *good* R-trees) and prove the *squaredness assumption*, we have computed the following formula for each level l_i and dimension k. Let $M_{R_i,l_i,m}$ be the mth MBR ($1 \leqslant m \leqslant Number\,of\,Nodes_{R_i,l_i}$) at the l_ith level ($0 \leqslant l_i \leqslant h_{R_i} - 1$) in the R*-tree R_i, where *Number of*

Table 2
Measured *real* average extent of node MBRs ($s_{R_i,l_i,k}$) of the R*-tree R_i at level l_i on dimension k ($1 \leqslant k \leqslant 2$) for uniform datasets (see Section 5), varying the branching factor for $Cmax_{R_i} = 25$ and 50

Level (l_i)	25 ($Cmax_{R_i}$)				50 ($Cmax_{R_i}$)			
	Unif1 (R_i)		Unif2 (R_i)		Unif1 (R_i)		Unif2 (R_i)	
	X ($k = 1$)	Y ($k = 2$)	X ($k = 1$)	Y ($k = 2$)	X ($k = 1$)	Y ($k = 2$)	X ($k = 1$)	Y ($k = 2$)
0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1	0.265303	0.278601	0.275572	0.285508	0.552201	0.551264	0.542527	0.542287
2	0.069035	0.065891	0.071967	0.071837	0.140719	0.142983	0.142961	0.127784
3	0.013189	0.013205	0.013975	0.014378	0.020989	0.021019	0.020858	0.020719

Table 3
Measured *real* average extent of node MBRs ($s_{R_i,l_i,k}$) of the R*-tree R_i at level l_i on dimension k ($1 \leqslant k \leqslant 2$) for real datasets (see Section 5), varying the branching factor for $Cmax_{R_i} = 100$ and 200

Level (l_i)	100 ($Cmax_{R_i}$)				200 ($Cmax_{R_i}$)			
	$CAS(R_i)$		$CAP(R_i)$		$CAS(R_i)$		$CAP(R_i)$	
	X ($k = 1$)	Y ($k = 2$)	X ($k = 1$)	Y ($k = 2$)	X ($k = 1$)	Y ($k = 2$)	X ($k = 1$)	Y ($k = 2$)
0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1	0.202204	0.212209	0.261987	0.249169	0.286131	0.277676	0.483801	0.497636
2	0.016961	0.017515	0.025633	0.024353	0.026881	0.028771	0.034857	0.033521

$Nodes_{R_i,l_i}$ represents the *real* number of R-tree nodes of R_i at level l_i. $M_{R_i,l_i,m} = (a_m, b_m)$, where $a_m = (a_{m1}, a_{m2}, \ldots, a_{md})$ and $b_m = (b_{m1}, b_{m2}, \ldots, b_{md})$ such that $a_{mk} \leqslant b_{mk}$ for any $1 \leqslant k \leqslant d$ (Tables 2 and 3).

$$s_{R_i,l_i,k} = \frac{\sum_{m=1}^{Number\ of\ Nodes_{R_i,l_i}} \left\{ \sum_{k=1}^{d} \{b_{mk} - a_{mk}\} \right\}}{Number\ of\ Nodes_{R_i,l_i}}$$

We can observe from the R*-trees generated for our experiments (with different maximum fan-outs, $Cmax_{R_i}$) that the *squaredness assumption* is reasonable to use in our cost model, since $X(k = 1) \approx Y(k = 2)$ in the normalized 2-dimensional unit hypercube ($[0,1]^2$) for any level l_i, R*-tree R_i and maximum fan-out $Cmax_{R_i}$. Therefore, we can use square-like MBRs with average extent s_{R_i,l_i} ($s_{R_i,l_i,k} \approx s_{R_i,l_i}$) in the cost models using R*-trees.

3.2. Estimating the number of node accesses using R-trees

3.2.1. Number of node accesses for K-NNQ

We assume the normalized d-dimensional unit space $[0,1]^d$ and a spatial dataset of cardinality N_{R_i} with the corresponding MBR approximations of spatial data being indexed in an R-tree R_i. The problem of R-tree cost analysis for K-NNQ (K Nearest Neighbor Query), which finds the K nearest neighbors to a given point query and arbitrary value of K ($1 \leqslant K \leqslant N_{R_i}$), has been studied actively in [4,7,22]. The study in [22] for Euclidean distance, first it estimates how far away

from the query point the Kth nearest neighbor is located in average (in effect transforming a nearest neighbor query into an equivalent range query).

$$dist_{nn}(K) = \frac{(\Gamma((d/2)+1))^{1/d}}{\sqrt{\pi}} * \left(\frac{K}{N_{R_i}}\right)^{1/D_2} \tag{1}$$

where $\Gamma(x)$ is the gamma function, obeying the recursive definition $\Gamma(x+1) = x * \Gamma(x)$, $\Gamma(1) = 1$, $\Gamma(1/2) = \sqrt{\pi}$, which may be approximated by $\Gamma(x+1) \approx (x/e)^x * \sqrt{2 * \pi * x}$.

Based on this distance, the cost model computes the volume of the Minkowski sum of the query sphere with radius $dist_{nn}(K)$ and the shape of an R-tree index page (average side length of node MBRs of R-tree), which corresponds to the access probability of the R-tree node. Besides, to compute $dist_{nn}(K)$ is used the correlation fractal dimension D_2 [3], which shows how the average number of neighbors of a given point of the dataset grows, as the box size grows (dividing the space into hyper-cube grid boxes with size r). D_2 is computed by a box-computing algorithm based on the concept of sum of squared occupancies of points that are contained within boxes of size r [3]. Therefore, assuming a set of N_{R_i} d-dimensional points with correlation fractal dimension D_2 indexed by an R-tree R_i with effective R-tree node capacity f_{R_i} and height h_{R_i}, the average number of accessed R-tree nodes (NA) to answer a K-NNQ is given by the following formula [22]:

$$NA_{nn}(R_i, K) = \sum_{l_i=0}^{h_{R_i}-1} \left\{ N_{R_i,l_i} * \left(\sum_{k=0}^{d} \left\{ \binom{d}{k} * (s_{R_i,l_i})^{d-k} * \frac{\pi^{k/2} * (dist_{nn}(K))^k}{\Gamma(k/2+1)} \right\} \right)^{D_2/d} \right\} \tag{2}$$

In this case, we can observe that N_{R_i,l_i} is the node density of the R-tree R_i at level l_i, and the term

$$\left(\sum_{k=0}^{d} \left\{ \binom{d}{k} * (s_{R_i,l_i})^{d-k} * \frac{\pi^{k/2} * (dist_{nn}(K))^k}{\Gamma(k/2+1)} \right\} \right)^{D_2/d} \tag{3}$$

corresponds to the *K-NNQ index selectivity*, which is the probability that a node from the R-tree R_i at level l_i is accessed for this distance-based query. When we use indexes, the only gain using algorithms for query processing is the *index selectivity*, i.e. not all nodes must be accessed and not all objects must be compared.

For nearest neighbor queries, assuming smoothly distributed points in the data space, low dimensionality (low D_2 values) and considering the MBR effect, the estimation of the average side length of node MBRs of R-tree R_i at level l_i in presence of the correlation fractal dimension D_2 is as follows [7]:

$$s_{R_i,l_i} = \left(1 - \frac{1}{f_{R_i}}\right) * \left(\min\left\{ \left(\frac{f_{R_i}^{h_{R_i}-l_i}}{N_{R_i}} \right), 1 \right\} \right)^{1/D_2} \tag{4}$$

3.2.2. Number of node accesses for spatial join

Formally, the problem of R-tree cost analysis for spatial join queries is defined as follows [32]: Let d be the dimensionality of the normalized d-dimensional unit space $[0,1]^d$. Let us assume two spatial datasets of cardinality N_{R_1} and N_{R_2}, with the corresponding MBR approximations of spatial data being stored in two R-tree indices R_1 and R_2, respectively. The goal of the cost analysis is

a formula that would efficiently estimate the average number of nodes accessed in order to process a spatial join query between the two datasets indexed by R-trees, based on the knowledge of the data properties and extracting information from the corresponding R-tree structures.

Let the heights of the R-trees R_1 and R_2, be h_{R_1} and h_{R_2}, respectively. At each level l_i, $0 \leqslant l_i \leqslant h_{R_i} - 1$, R_i contains N_{R_i,l_i} nodes of average size $s_{R_i,l_i,k}$, on each dimension k ($1 \leqslant k \leqslant d$). The overall estimation of the total cost in terms of R-tree node accesses for the spatial join operation (when the spatial predicate is *overlap*) is defined by the following formula [32] (without loss of generality, it is assumed that $h_{R_2} \leqslant h_{R_1}$):

$$NA_{sj}(R_1, R_2) = \sum_{l_1=0}^{h_{R_1}-1} \{NA(R_1, R_2, l_1) + NA(R_2, R_1, l_2)\}$$

where

$$l_2 = \begin{cases} l_1, & 0 \leqslant l_1 \leqslant h_{R_2} - 1 \\ h_{R_2} - 1, & h_{R_2} \leqslant l_1 \leqslant h_{R_1} - 1 \end{cases}$$

The cost of the previous formula at each level is the sum of two factors which correspond to the costs for the two R-trees, namely $NA(R_1, R_2, l_1)$ and $NA(R_2, R_1, l_2)$, respectively. For the levels of the two R-trees, where $h_{R_2} \leqslant h_{R_1}$:

$$NA(R_1, R_2, l_1) = NA(R_2, R_1, l_2) = N_{R_1,l_1} * N_{R_2,l_2} * \prod_{k=1}^{d} \left\{s_{R_1,l_1,k} + s_{R_2,l_2,k}\right\}$$

where $h_{R_2} \leqslant l_1 \leqslant h_{R_1} - 1$ and $0 \leqslant l_2 \leqslant h_{R_2} - 1$. Therefore, the estimation of the total I/O cost in terms of R-tree nodes accessed for spatial join query is as follows:

$$NA_{sj}(R_1, R_2) = 2 * \sum_{l_1=0}^{h_{R_1}-1} \left\{ N_{R_1,l_1} * N_{R_2,l_2} * \prod_{k=1}^{d} \left\{s_{R_1,l_1,k} + s_{R_2,l_2,k}\right\} \right\} \tag{5}$$

where

$$l_2 = \begin{cases} l_1, & 0 \leqslant l_1 \leqslant h_{R_2} - 1 \\ h_{R_2} - 1, & h_{R_2} \leqslant l_1 \leqslant h_{R_1} - 1 \end{cases}$$

From $NA_{sj}(R_1, R_2)$ formula, we can see that $N_{R_1,l_1} * N_{R_2,l_2}$ is the R-tree node pairs density of the R-trees R_1 and R_2 at levels l_1 and l_2, and the term

$$\prod_{k=1}^{d} \left\{s_{R_1,l_1,k} + s_{R_2,l_2,k}\right\} \tag{6}$$

corresponds to the *join index selectivity*, and it equals the probability that a random pair of nodes from two R-trees R_1 and R_2 at levels l_1 and l_2 is accessed (i.e. the number of node pairs at levels l_1 and l_2 to be processed divided by the theoretically possible node pairs).

3.2.3. Number of node accesses for K-CPQ

The previous formulae (2) and (5) can be applied for obtaining the I/O cost of the K-CPQ for R-trees, using an appropriate *K-CPQ index selectivity*, (σ). It depends mainly on the characteristics

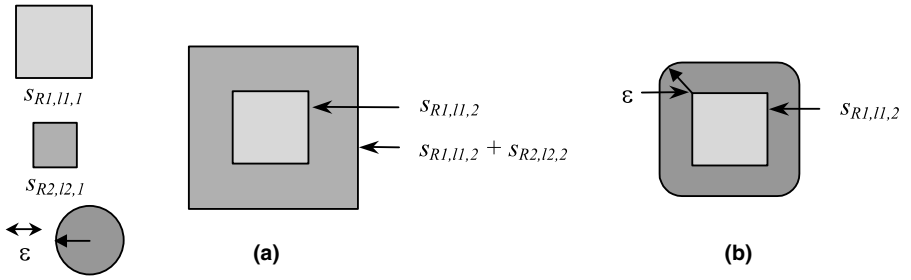

Fig. 3. The 2-dimensional Minkowski sum of two MBRs (a), and an MBR and a ε-sphere (b).

of the index (e.g. average side length of node MBRs) and on the distance parameter $dist_{cp}(K)$, which is the distance of the Kth closest pair for the K-CPQ.

The *K-CPQ index selectivity* (σ) can be also obtained by using the concept of Minkowski sum [4]. The Minkowski sum of two geometric objects A and B, each seen as an infinite number of vectors (points) in the d-dimensional data space (e.g. $A = \{a_1, a_2, \ldots\}$ and $B = \{b_1, b_2, \ldots\}$), is defined as the set of vector sum of all combinations between vectors in A and B: $A \oplus B = \{a + b: a \in A, b \in B\} = \{a_1 + b_1, a_1 + b_2, a_2 + b_1, \ldots\}$. For the cost modeling we are only interested in the volume of the Minkowski sum, $V_{A\oplus B}$, not in its shape. The simplest case is both spatial objects to be d-dimensional MBRs, M_1 and M_2, with side length r_k and s_k ($1 \leqslant k \leqslant d$), respectively. In this case, the volume $V_{M_1\oplus M_2}$ of the Minkowski sum of two MBRs is the MBR with side length t_k, where each t_k corresponds to the sum of r_k and s_k, as in Fig. 3a for $d = 2$. Note the likeness of $V_{M_1\oplus M_2}$ with the expression of *join index selectivity*, see formula (6).

$$V_{M_1\oplus M_2}((r_1, r_2, \ldots, r_d), (s_1, s_2, \ldots, s_d)) = \prod_{k=1}^{d} \{r_k + s_k\}$$

A more complicate case is the volume of the Minkowski sum of an MBR M with side length r_k ($1 \leqslant k \leqslant d$) and a hyper-sphere with radius ε ($V_{M\oplus S_\varepsilon}$). In this case, we enlarge the MBR region so that if the original MBR touched any point of the query sphere, then the enlarged MBR touches the center point of the query. Thus, the MBR region becomes enlarged by a sphere of the same radius ε whose center point is drawn over the surface of the MBR region. The $V_{M\oplus S_\varepsilon}$ equation is given by the following binomial formula and the Fig. 3b shows this transformation for $d = 2$. Again, observe the likeness with the expression of *K-NNQ index selectivity*, see formula (3).

$$V_{M\oplus S_\varepsilon}((r_1, r_2, \ldots, r_d), \varepsilon) = \sum_{k=0}^{d} \left\{ \binom{d}{k} * (r_k)^{d-k} * \frac{\pi^{k/2}}{\Gamma(k/2+1)} * \varepsilon^k \right\}$$

In [8] for the similarity join operation (two datasets of multidimensional points are combined in such a way that the result contains all pairs of points where the distance does not exceed a given distance threshold ε), the volume of the Minkowski sum of three objects (two MBRs and a ε-sphere) was given by the following binomial formula:

$$V_{M_1\oplus M_2\oplus S_\varepsilon}((r_1, r_2, \ldots, r_d), (s_1, s_2, \ldots, s_d), \varepsilon) = \sum_{k=0}^{d} \left\{ \binom{d}{k} * (r_k + s_k)^{d-k} * \frac{\pi^{k/2}}{\Gamma(k/2+1)} * \varepsilon^k \right\}$$

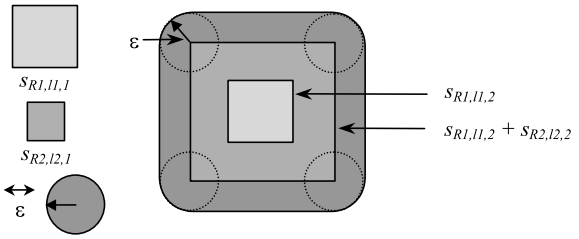

Fig. 4. The 2-dimensional Minkowski sum of two MBRs and a ε-sphere.

In our case (the *K*-CPQ), for the Euclidean distance, a pair of R-tree nodes is processed whenever the minimum distance between their two MBRs does not exceed the $dist_{cp}(K)$ distance value. In order to determine the probability of this event, we enlarge the MBR of one node so that this MBR touches any point of the MBR of the other node (Minkowski sum of two MBRs is an MBR with added side lengths). This new MBR is enlarged even more by a sphere of radius $dist_{cp}(K)$ whose center point is drawn over the perimeter of the enlarged MBR. The 2-dimensional Minkowski sum of two MBRs and a ε-sphere ($\varepsilon \equiv dist_{cp}(K)$) is shown in the Fig. 4.

The volume of the Minkowski sum of two MBRs of two levels l_1 and l_2 R-tree nodes ($0 \leqslant l_i \leqslant h_{R_i} - 1$) from two R-trees R_1 and R_2 with heights $h_{R_2} \leqslant h_{R_1}$ and one sphere of radius $dist_{cp}(K)$, divided by the data space volume (the volume of the d-dimensional unit space $[0,1]^d$ which is equal to 1), expresses the access probability (*K-CPQ index selectivity*, $\sigma(K)$) of the corresponding nodes and it is given by the following binomial formula (note that R_i contains N_{R_i,l_i} nodes of average side lengths s_{R_i,l_i} with minimum distance between them smaller than or equal to $dist_{cp}(K)$):

$$\sigma(K) = \left(\sum_{k=0}^{d} \left\{ \binom{d}{k} * \left(s_{R_1,l_1} + s_{R_2,l_2}\right)^{d-k} * \frac{\pi^{k/2} * \left(dist_{cp}(K)\right)^k}{\Gamma(k/2+1)} \right\} \right)^{\rho/d} \tag{7}$$

where

$$l_2 = \begin{cases} l_1, & 0 \leqslant l_1 \leqslant h_{R_2} - 1 \\ h_{R_2} - 1, & h_{R_2} \leqslant l_1 \leqslant h_{R_1} - 1 \end{cases}$$

For example, if we have low values of dimensionality ($d = 2$ and 3), it is easy to deduce *K-CPQ index selectivity* formula as follows (where $a = (s_{R_1,l_1} + s_{R_2,l_2})$):

$$\sigma(K) = \begin{cases} \left(a^2 + 4 * a * dist_{cp}(K) + \pi * \left(dist_{cp}(K)\right)^2\right)^{\rho/2}, & d = 2 \\ \left(a^3 + 6 * a^2 * dist_{cp}(K) + 3 * \pi * a * \left(dist_{cp}(K)\right)^2 + \frac{4}{3} * \pi * \left(dist_{cp}(K)\right)^3\right)^{\rho/3}, & d = 3 \end{cases}$$

Although, the Minkowski sum is used in [8] for uniform and independent data only, it can also be applied for real data (non-uniform and independent data distributions) to determine $\sigma(K)$, assuming the pairs distribution is *smooth* (in this case, smoothness means that the pairs density (number of pairs inside a given volume) does not vary severely inside the Minkowski enlargement of page pairs), since in this case, the non-uniformity does not have influence on the cost model [7].

Moreover, the use of the exponent ρ/d for the whole sum in the above formula (7) is explained as follows, according to [7]. Under *uniformity and independence* assumption, the number of pairs of points enclosed in a hypercube with side length s is proportional to the volume of the hypercube, $pairs(s) = \tau * s^d = \tau * Vol$, where $\tau = (N_{R_1} * N_{R_2})/Vol_{DS}$ is called *pairs density* ($Vol_{DS} = 1^d = 1$ is the volume of the data space) and $Vol = s^d \Longleftrightarrow s = Vol^{1/d}$. Real datasets (*non-uniform and independence with smooth pairs distribution*) obey a similar power law using the 'correlation exponent' ρ (pair-count exponent) of the datasets [15] $pairs(s) = \tau_p * s^\rho = \tau_p * Vol^{\rho/d}$, where τ_p is called *correlation pairs density*, which is analogous to the *pairs density* τ (and $s = Vol^{1/d}$): for the uniform case, $\rho = d$ and $\tau_p = \tau$. Since our *K-CPQ index selectivity* ($\sigma(K)$) formula is calculated according to the concept of Minkowski sum, really for cost modeling we are interested on its volume (the volume of Minkowski sum corresponds to the volume of intersected pages [4]), and for this reason we raise the volume of the Minkowski sum to ρ/d (for real datasets, obeying a 'similar' power law which depends on ρ and the volume of Minkowski sum), using the correlation exponent ρ and the dimensionality d of the data space.

Finally, the overall estimation of the total I/O cost in terms of R-tree node accesses for K-CPQ is given by the following formula (without loss of generality, it is assumed that $h_{R_2} \leqslant h_{R_1}$), in a similar way to $\mathrm{NA}_{nn}(R_i, K)$ and $NA_{sj}(R_1, R_2)$. Where the multiplier (i.e. the factor 2) corresponds to the cost of two R-trees (R_1 and R_2), $N_{R_1,l_1} * N_{R_2,l_2}$ is the R-tree node pairs density of the R-trees R_1 and R_2 at levels l_1 and l_2, and $\sigma(K)$ represents the percentage of pairs of R-tree nodes that are accessed at these levels for the K-CPQ algorithm.

$$NA_{cp}(R_1, R_2, K) = 2 * \sum_{l_1=0}^{h_{R_1}-1} \left\{ N_{R_1,l_1} * N_{R_2,l_2} * \sigma(K) \right\} \tag{8}$$

where

$$l_2 = \begin{cases} l_1, & 0 \leqslant l_1 \leqslant h_{R_2} - 1 \\ h_{R_2} - 1, & h_{R_2} \leqslant l_1 \leqslant h_{R_1} - 1 \end{cases}$$

3.3. Estimation of the distance of the Kth closest pair, $dist_{cp}(K)$

We are interested in the estimation of $dist_{cp}(K)$ under the Euclidean distance for arbitrary object distributions. Real datasets (non-uniform and independent) show a clear divergence from the uniformity and independence assumptions [14] and, hence, it is better to consider the uniformity as a special case. In [15] was proposed a power law to predict the selectivity of spatial join and to estimate the distance of the Kth closest pair ($PC(r) = \beta * r^\rho$). Here, we will use the *Box-Occupancy-Product-Sum* (BOPS) method (box-counting approach) as proposed in [15] to determine the exponent (ρ) of the power law.

For the estimation of $dist_{cp}(K)$, given two point datasets with finite cardinalities N_{R_1} and N_{R_2} embedded in a unit hypercube of dimension d, the average number of pairs within a region of regular shape characterized by a distance $r \geqslant 0$, obeying a power law based on the 'correlation exponent' ρ [15,3], is given by: $pairs(r, \text{'}d\text{-}shape\text{'}) = \tau_p * Vol(r, \text{'}d\text{-}shape\text{'})^{\rho/d}$

$$pairs(r, \text{'}d\text{-}shape\text{'}) = N_{R_1} * N_{R_2} * \left(\frac{(\Gamma(1/2))^d}{(\Gamma(d/2+1))} * r^d \right)^{\rho/d}$$

where $N_{R_1} * N_{R_2}$ is the correlation pairs density (τ_p) in $[0,1]^d$ and $((\Gamma(1/2))^d/\Gamma(d/2+1)) * r^d$ is the volume of that part of an r-hypersphere centered on a point that is inside the d-dimensional hypercube $[0,1]^d$. Therefore, we can use the previous formula to estimate the average distance of Kth closest pair under the Euclidean distance, satisfying that pairs(r, 'd-shape') = K and $r = dist_{cp}(K)$:

$$dist_{cp}(K) = \frac{(\Gamma((d/2)+1))^{1/d}}{\sqrt{\pi}} * \left(\frac{K}{N_{R_1} * N_{R_2}}\right)^{1/\rho} \tag{9}$$

3.4. Estimation of the correlation exponent, ρ

The *pair-count law* [15] governs the distribution of pair-wise distance between two real, d-dimensional point datasets. The exponent of the pair count law, so-called *pair-count exponent* ρ (correlation exponent), can be calculated using the box-counting approach based on the concept of *box-occupancy-product-sum* (BOPS) [15]. The most important properties of ρ are the following: (1) ρ includes the *correlation fractal dimension* (D_2) as special case; (2) ρ is invariant to affine transformations (translation, rotation and uniform scaling); (3) ρ is invariant to sampling; and (4) ρ is invariant to L_p metric distance used [15]. The algorithm for computing BOPS is an interesting extension of the algorithm proposed in [3] for computing D_q (generalized fractal dimension).

Definition 2. Pair-count function, PC(r) [15].

For two point datasets, S_1 and S_2, and a given distance $r \geqslant 0$, the *pair-count function*, PC(r), counts the number of pairs within a Euclidean distance smaller than or equal to r. $PC(r) = \{(a_i, b_j)$: distance$(a_i, b_j) \leqslant r$, $a_i \in S_1$ and $b_j \in S_2\}$.

Law 1. Power law for pair-count function (follows from Law 1 of [15]).

The pair-count function, PC(r), follows a *power law* $PC(r) \propto r^{\rho}$, where ρ is the correlation exponent and $\propto$ stands for 'proportional', i.e. $PC(r) = \beta * r^{\rho}$, where β is a proportionality constant.

Definition 3. Pair-count exponent, ρ [15].

The exponent of the power law, called *pair-count exponent* ρ in [15], is defined as

$$\rho = \frac{\partial(\log(PC(r)))}{\partial(\log(r))} = \lim_{r \to 0} \frac{\log(PC(r))}{\log(r)}$$

In order to obtain PC(r), we can use the box-counting algorithm (it is linear O(N+M) over the total number of points in both datasets) based on the concept of *Box-Occupancy-Product-Sum* (BOPS) [15]. Intuitively, BOPS(r) counts the number of pairs of points (from two different datasets) that lie within boxes of size r. Usually, to calculate the correlation exponent ρ (pair-count exponent), it is useful to plot log(BOPS(r)) as a function of log(r) and measure the slope of the linear part of the obtained curve by performing a linear interpolation (this slope corresponds to the correlation exponent (ρ) and the proportionality constant to β); hence $\rho = \delta(\log(\text{BOPS}(r))/\delta(\log(r))$. Intuitively, ρ shows how the average number of pairs from two different points datasets grows, as the distance r increases.

Definition 4. Box-Occupancy-Product-Sum, BOPS(r) [15].

The BOPS of a grid with box size r is defined as the sum of products of occupancies as

$$\mathrm{BOPS}(r) = \sum_{i=1}^{N(r)} p_{i,S_1} * p_{i,S_2}$$

where $N(r)$ is the number of boxes with side r that are occupied by the two point datasets; p_{i,S_1} and p_{i,S_2} are the percentage of points of the datasets S_1 and S_2, respectively, which fall inside the ith box with side r.

A consequence of this definition is that $\mathrm{BOPS}(r) \propto r^{\rho}$ (BOPS follows a power law with its exponent equal to the correlation exponent, ρ) [15]. BOPS plays a similar role that the sum of squared occupancies ($S_2(r) = \sum p_i^2$), which is proportional to the correlation fractal dimension $D_2(S_2(r) \propto r^{D2})$ [3]. That is, the correlation exponent ρ includes the correlation fractal dimension D_2 as a special case [15].

Therefore, we have to estimate how the access probability of an R-tree page changes in presence of the correlation exponent described by ρ. Let us assume that the point density of each dataset is constant throughout the populated part of the page region and its Minkowski enlargement, and taking into account the MBR effect [7], we can estimate the average extent s_{R_i,l_i} of a node MBR of the R-tree R_i at level l_i according to the power law as follows:

$$s_{R_i,l_i} = \left(1 - \frac{1}{f_{R_i}}\right) * \left(\min\left\{\left(\frac{f_{R_i}^{h_{R_i}-l_i}}{N_{R_i}}\right), 1\right\}\right)^{1/\rho} \tag{10}$$

where $l_i = 0, 1, \ldots, h_{R_i} - 1$ (the root is assumed at level $l_i = 0$ and leaves at $l_i = h_{R_i} - 1$).

Having a method for the calculation of ρ from the involved datasets, we have completed our cost model: we are able to evaluate the total cost of the K-CPQ for various values of K, dimensions and ρ using formula (8), and the other dependent formulae (7), (9) and (10).

Finally, the formula (8) is a generalization of the cost models for K nearest neighbor and spatial join queries (formulae (2) and (5), respectively). For spatial join query, we have $dist_{cp}(K) = 0$, and the formula (7) is equivalent to (6); hence the formula (8) corresponds to (5). For the case of K-NNQ, we have only one R-tree (R_1, and R_2 does not need to be considered), and we have to remove from the formula (9) N_{R_2} and replace ρ by D_2 (ρ includes D_2 as a special case for one dataset [15]), obtaining the formula (1), in the formula (10) we also replace ρ by D_2 to obtain (4), and removing s_{R_2,l_2} from (7) we get (3); hence the formula (8) corresponds to (2), because we have removed the R-tree R_2 and the multiplier in (8) is now 1 instead of 2.

4. Extensions of the cost model

4.1. Buffer queries

We can adapt our cost model to another type of distance join query as *buffer query* [9]. This query involves two spatial datasets and a distance threshold δ. The answer is a set of pairs of spatial objects (in our case, pairs of points) from the two input datasets that are within distance

δ from each other (it is related with the similarity join [23], where the problem of deciding if two objects are similar is reduced to the problem of determining if two high-dimensional points are within a certain distance of each other). Formally: Given two different datasets S_1 and S_2 of N_{S_1} and N_{S_2} spatial objects, respectively, a *buffer query* retrieves all different pairs of objects from $S_1 \times S_2$ with distance less than or equal to δ. The cardinality of the answer set is in the range $[0 \ldots N_{S_1} * N_{S_2}]$. An example of such a query is to find pairs of cities and cultural landmarks that are within 3 km of each other. In [9], this problem is solved for non-point (lines and regions) spatial datasets, where efficient algorithms for computing the minimum distance for lines and regions, pruning techniques for filtering in a Depth-First algorithm, and extensive experimental results are presented.

Therefore, we can adapt the Best-First K-CPQ algorithm ($z \equiv \delta$ and K does not need to be considered) as is the Fig. 5 (*input*: two R-trees (R_1 and R_2) indexing two point datasets, a distance threshold (δ), and a file for the final result (resultSet); *output*: a file for the final result (resultSet)).

Of course, if the R-trees have different heights, we can use the *fix-at-leaves* technique, although in [9] are executed window queries (*δ-distance range query*, which involves one point dataset, a query point and a distance threshold δ; and the answer is a set of points from the input dataset that are within distance δ from the query point) from the leaves of the smallest R-tree on the largest one.

Therefore, a simple derivation our cost model (formulae (7) and (8)) can estimate the number of node accesses for *buffer queries* (where points are indexed by R-trees) as follows:

```
01 Create and initialize H;
02 for each pair of MBRs <M1i, M2j> in <root_R1, root_R2> do
03     H.insert(MinDist(M1i, M2j), Addr_M1i, Addr_M2j);
04 enddo
05 while (not H.isEmpty()) and (H.MinimumDistance() ≤ δ) do
06     minimum = H.deleteMinimum();
07     node_R1 = R1.readNode(minimum.Addr_M1);
08     node_R2 = R2.readNode(minimum.Addr_M2);
09     if (node_R1 and node_R2 are internal nodes) then
10         for each pair of MBRs <M1i, M2j> in <node_R1, node_R2> do
11             if (MinDist(M1i, M2j) ≤ δ) then
12                 H.insert(MinDist(M1i, M2j), Addr_M1i, Addr_M2j);
13             endif
14         enddo
15     else
16         for each pair of points <P1i, P2j> in <node_R1, node_R2> do
17             distance = MinDist(P1i, P2j);
18             if (distance ≤ δ) then
19                 resultSet.append(distance, P1i, P2j);
20             endif
21         enddo
22     endif
23 enddo
24 Destroy H;
```

Fig. 5. Best-First algorithm for *buffer query* using R-trees.

$$NA_{bq}(R_1, R_2, \delta) = 2 * \sum_{l_1=0}^{h_{R_1}-1} \left\{ N_{R_1,l_1} * N_{R_2,l_2} * \sigma(\delta) \right\} \tag{11}$$

$$\sigma(\delta) = \left(\sum_{k=0}^{d} \left\{ \binom{d}{k} * \left(s_{R_1,l_1} + s_{R_2,l_2}\right)^{d-k} * \frac{\pi^{k/2} * \delta^k}{\Gamma(k/2+1)} \right\} \right)^{\rho/d} \text{ and } l_2 = \begin{cases} l_1, & 0 \leqslant l_1 \leqslant h_{R_2} - 1 \\ h_{R_2} - 1, & h_{R_2} \leqslant l_1 \leqslant h_{R_1} - 1 \end{cases}$$

4.2. The effect of buffering

For the buffering study, on the one hand, we can extend our formula when a *path buffer* is activated in each R-tree, in the same way as in [32] for a Depth-First algorithm for spatial joins. Each R-tree makes use of a *path buffer* accommodating all nodes of the *active path*, which were accessed last. For this reason, this buffer has an important impact for Depth-First algorithms, where they put higher priority to the nodes in the largest path, and the *path buffer* always maintains the *active path* of R-tree nodes from root to the last accessed leaf. In our case, we have Best-First algorithms, and they put higher priority to the nodes (pairs of nodes) with the smallest lower bound (*MinDist* value) and the impact of *path buffers* (one for each R-tree) can be smaller than the effect of *path buffers* on Depth-First algorithms. Moreover, the total size of the *path buffer* for *K*-CPQ is limited by $h_{R_1} + h_{R_2}$ R-tree nodes and we cannot enlarge its size in order to get better performance results, although small buffer sizes can notably affect the cost of join queries using Depth-First algorithms. Therefore, here, we omit this study.

On the other hand, we can study the buffering impact when a *global LRU buffer* is incorporated over a distance join query (*K*-CPQ or *buffer query*). The only study of a cost model for predicting the performance of spatial join query using R-trees when a global buffer is present was proposed in [20]. In such analysis, a probabilistic analysis of paging pattern assuming that the inter-access page faults are exponentially distributed was studied. They did not consider the LRU buffer management over this kind of indexes (R-trees). Another related work was proposed in [24], when a buffer model was presented to estimate the expected number of distinct R-tree node accesses in N consecutive range queries. In this buffer model, a *uniform access pattern* in the context of buffering of R-tree nodes was considered. Here, we will base on the buffer model proposed in [6], which can model buffer performance with *non-uniform distributions* (*access patterns*).

In our analysis, we need to estimate the buffer hit probability in an LRU buffer, and we are going to use the buffer model proposed in [6]. Here, we will only briefly review this buffer model, and for the interested reader, the derivation and details behind the equations can be found in [6]. A database, in this buffer model, has of size N data pages (granules in [6]), partitioned into p partitions. Each partition contains β_j of the data pages (as a fraction of the total database size), and α_j of the accesses (access probability) is done to each partition. The distributions within each of the partitions are assumed to be uniform. All accesses are assumed to be independent of all previous requests (Independent Reference Model). We also denote an *access pattern* (partition set) for a set of data pages as AP, where AP has p partitions, each j ($0 \leqslant j \leqslant p-1$) partition is characterizes by β_j and α_j ($AP = \{\beta_0, \beta_1, \ldots, \beta_{p-1}, \alpha_0, \alpha_1, \ldots, \alpha_{p-1}\}$); and the following is satisfied:

$$\sum_{j=0}^{p-1} \beta_j = 1.0 \quad \text{and} \quad \sum_{j=0}^{p-1} \alpha_j = 1.0$$

After n accesses with access pattern AP (p is the number of partitions) to the database containing N data pages, the number of distinct data pages from partition j ($0 \leqslant j \leqslant p-1$) that have been accessed can be approximated by:

$$B_j(n, N, AP) = \beta_j * N * \left(1 - \left(1 - \frac{1}{\beta_j * N}\right)^{\alpha_j * n}\right)$$

When the number of accesses n is such that the number of distinct data pages accessed is less than or equal to the buffer size $B, B(n, N, AP) = \sum_{j=0}^{p-1} B_j(n, N, AP) \leqslant B$, the buffer hit probability for partition j ($0 \leqslant j \leqslant p-1$) can be approximated by:

$$H_j(n, N, AP) = 1 - \left(1 - \frac{1}{\beta_j * N}\right)^{\alpha_j * n}$$

($1/(\beta_j * N)$ represents the probability that a data page is accessed in the partition j) and the average buffer hit probability can be estimated as:

$$H(n, N, AP) = \sum_{j=0}^{p-1} \left\{\alpha_j * H_j(n, N, AP)\right\}$$

The steady state average buffer hit probability in an LRU-managed buffer given a certain access pattern can be approximated to the buffer hit probability when the buffer first becomes full (i.e., n is chosen as the largest n that satisfies $B(n, N, AP) = \sum_j B_j(n, N, AP) = B$, where B is the number of data pages that fits in the buffer: $H_{LRU}(B, n, N, AP) = H(n, N, AP)$.

When the buffer size is varied, the expected number of accesses (n) needed to fill the buffer (i.e. to satisfied the condition $B(n, N, AP) \leqslant B$) can be determined by a binary search (because $B_j(n, N, AP)$ is monotonically increasing function with respect to n). Alternatively, we could compute $B(n, N, AP) = \sum_j B_j(n, N, AP) = B$ using the equation of $B_j(n, N, AP)$ for all partitions (p), increasing n with some stride, and plot $B(n, N, AP)$ versus $H_j(n, N, AP)$ ($0 \leqslant j \leqslant p-1$), in order to obtain the same characteristic [6].

We have reviewed the buffer model for independent and non-hierarchical access [6]. Modeling buffer for hierarchical access methods, as R-trees, is not difficult to adapt [25]. Even searches to the leaves can be considered to be random and independent, R-tree nodes (index pages) accessed during traversal of the R-tree are not completely independent. For this reason, we are going to modify the previous buffer model to take into account a hierarchical index structure (R-tree), in order to approximate the average buffer hit probability assuming that each R-tree level is accessed with the same probability, obtaining an R-tree buffer model different to the proposed in [24], where an *uniform access pattern* within each partition in the context of buffering of R-tree nodes was considered.

Initially, for a given R-tree R_i, we have $p_{R_i} = h_{R_i}$ partitions (h_{R_i} is the height of R_i) with access pattern AP_{R_i,l_i} at each l_i level, and each of these partitions has N_{R_i,l_i} R-tree nodes. The access probability of each partition is $1/h_{R_i}$. According to [6,25], if we want to account for *hot spots*, we divide

the partition of leaf nodes of R_i into $p_{R_i,h_{R_i}-1}$ partitions with access pattern $AP_{R_i,h_{R_i}-1}$ (in total, we have $p_{R_i} = (h_{R_i} - 1) + p_{R_i,h_{R_i}-1}$ partitions), each with a fraction of $\beta_{R_i,h_{R_i}-1,j}$ of the R-tree (R_i) leaf nodes ($R_i, h_{R_i} - 1, j$ represents the j partition at leaf level ($h_{R_i} - 1$) of the R-tree R_i), and access probability $\alpha_{R_i,h_{R_i}-1,j}$ relative to the same partition at leaf level. Thus, each of these partitions have size $\beta_{R_i,h_{R_i}-1,j} * N_{R_i,h_{R_i}-1}$ (where $h_{R_i} - 1$ is the level of the leaf nodes in the R-tree R_i) and access probability $\alpha_{R_i,h_{R_i}-1,j}/h_{R_i}$. The *hot spots* at the leaf level make access to R-tree nodes on upper R-tree levels non-uniform, but as long as the average R-tree node fan-out (f_{R_i}) is sufficiently large, and the *hot spot* areas are not too narrow, we can treat accesses to R-tree nodes on upper levels (non-leaf levels) as uniformly distributed within each R-tree level (partition) [25]. With these modifications, an R-tree R_i of height h_{R_i}, and $p_{R_i,h_{R_i}-1}$ partitions at leaf level with access pattern $AP_{R_i,h_{R_i}-1}$, the equations for $\beta_{R_i,l_i,j}$ (fraction of R-tree nodes in each partition in the R-tree R_i at level l_i) and $\alpha_{R_i,l_i,j}$ (access probability of each partition in the R-tree R_i at level l_i) become:

$$\beta_{R_i,l_i,j} = \begin{cases} \dfrac{N_{R_i,l_i}}{Nodes_{R_i}}, & 0 \leqslant l_i, j \leqslant h_{R_i} - 2 \\ \dfrac{\beta_{R_i,l_i,j} * N_{R_i,l_i}}{Nodes_{R_i}}, & l_i = h_{R_i} - 1 \text{ and } h_{R_i} - 1 \leqslant j \leqslant p_{R_i} - 1 \end{cases}$$

$$\alpha_{R_i,l_i,j} = \begin{cases} \dfrac{1}{h_{R_i}}, & 0 \leqslant l_i, j \leqslant h_{R_i} - 2 \\ \dfrac{\alpha_{R_i,l_i,j}}{h_{R_i}}, & l_i = h_{R_i} - 1 \text{ and } h_{R_i} - 1 \leqslant j \leqslant p_{R_i} - 1 \end{cases}$$

where $Nodes_{R_i}$ represents the total number of R-tree nodes in the R-tree R_i

$$Nodes_{R_i} = \sum_{l_i=0}^{h_{R_i}-1} N_{R_i,l_i} \text{ (see Table 1) and } \sum_{j=h_{R_i}-1}^{p_{R_i}-1} \beta_{R_i,h_{R_i}-1,j} = 1.0 \text{ and } \sum_{j=h_{R_i}-1}^{p_{R_i}-1} \alpha_{R_i,h_{R_i}-1,j} = 1.0 \text{ are held.}$$

Hence, under buffer model [6], after n R-tree node accesses with access pattern AP_{R_i,l_i} (p_{R_i} is the number of partitions) to the R-tree R_i containing $Nodes_{R_i}$ R-tree nodes, the number of distinct R-tree nodes of R-tree R_i at level l_i from partition j ($0 \leqslant j \leqslant p_{R_i} - 1$) that have been accessed can be approximated by:

$$B_{R_i,l_i,j}(n, Nodes_{R_i}, AP_{R_i,l_i}) = \beta_{R_i,l_i,j} * Nodes_{R_i} * \left(1 - \left(1 - \frac{1}{\beta_{R_i,l_i,j} * Nodes_{R_i}}\right)^{\alpha_{R_i,l_i,j}*n}\right)$$

where $1/(\beta_{R_i,l_i,j} * Nodes_{R_i})$ represents the probability that an R-tree node of the R-tree R_i at level l_i is accessed in the partition j. From this equation, $1 - (1 - 1/(\beta_{R_i,l_i,j} * Nodes_{R_i}))^{\alpha_{R_i,l_i,j}*n}$ represents the probability that an R-tree node is accessed after n accesses from the R-trees R_i at level l_i in the partition j.

The average number of distinct R-tree nodes of R-tree R_i, can be approximated by

$$B_{R_i}(n, Nodes_{R_i}, AP_{R_i,l_i}) = \sum_{j=0}^{p_{R_i}-1} \left\{B_{R_i,l_i,j}(n, Nodes_{R_i}, AP_{R_i,l_i})\right\}$$

where

$$l_i = \begin{cases} j, & 0 \leqslant j \leqslant h_{R_i} - 2 \\ h_{R_i} - 1, & h_{R_i} - 1 \leqslant j \leqslant p_{R_i} - 1 \end{cases}$$

Finally, we can estimate the average buffer hit probability in an LRU-managed buffer given a certain access pattern, by using the buffer hit probability equation ($H(n, N, AP) = \sum_j \{\alpha_j * H_j(n, N, AP)\}$) such that $B_{R_i}(n, Nodes_{R_i}, AP_{R_i,l_i}) = \sum_j B_{R_i,l_i,j}(n, Nodes_{R_i}, AP_{R_i,l_i}) = B (0 \leqslant j \leqslant p_{R_i} - 1)$, with $B_{R_i,l_i,j}(n, Nodes_{R_i}, AP_{R_i,l_i})$, $\beta_{R_i,l_i,j}$ and $\alpha_{R_i,l_i,j}$ defined above as: $H_{LRU_{R_i}}(B, n, Nodes_{R_i}, AP_{R_i,l_i}) = H(n, Nodes_{R_i}, AP_{R_i,l_i}) = \sum_j \{\alpha_{R_i,l_i,j} * H_{R_i,l_i,j}(n, Nodes_{R_i}, AP_{R_i,l_i})\}$,

$$H_{LRU\_R_i}(B, n, Nodes_{R_i}, AP_{R_i,l_i}) = \sum_{j=0}^{p_{R_i}-1} \left\{\alpha_{R_i,l_i,j} * H_{R_i,l_i,j}(n, Nodes_{R_i}, AP_{R_i,l_i})\right\}$$

where

$$H_{R_i,l_i,j}(n, Nodes_{R_i}, AP_{R_i,l_i}) = 1 - \left(1 - \frac{1}{\beta_{R_i,l_i,j} * Nodes_{R_i}}\right)^{\alpha_{R_i,l_i,j} * n}$$

and

$$l_i = \begin{cases} j, & 0 \leqslant j \leqslant h_{R_i} - 2 \\ h_{R_i} - 1, & h_{R_i} - 1 \leqslant j \leqslant p_{R_i} - 1 \end{cases}$$

In our case (the *K*-CPQ), we have two R-trees (R_1 and R_2, i.e. the total number of R-tree nodes is $Nodes_{R_1} + Nodes_{R_2}$), a global LRU buffer with size B R-tree nodes as in Fig. 6; and $\sigma(K)$ represents the probability that a pair of MBRs (one from R_1 at level l_1 and another from R_2 at level l_2) are accessed whenever the minimum distance between the two MBR regions is smaller than or equal to $dist_{cp}(K)$ for *K*-CPQ (similarly, $\sigma(\delta)$ is for *buffer queries*, where δ is the distance threshold).

Assuming $h_{R_2} \leqslant h_{R_1}$ and $p_{cp} = h_{R_1} + 1$ $(0 \leqslant j \leqslant h_{R_1})$ partitions with access pattern $AP_{cp,l_1}(l_2 \leqslant l_1)$. That is, there are two additional partitions at leaf level following an 80/20 access

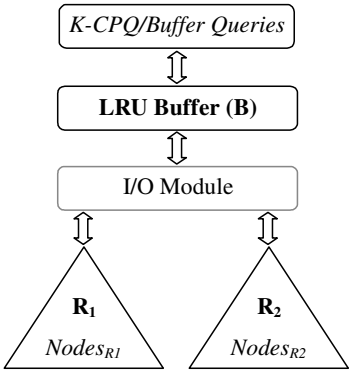

Fig. 6. Global LRU buffer scheme.

pattern ($AP_{cp,h_{R_1}-1} = \{\beta_{cp,h_{R_1}-1,h_{R_1}-1} = 0.2,\ \beta_{cp,h_{R_1}-1,h_{R_1}} = 0.8,\ \alpha_{cp,h_{R_1}-1,h_{R_1}-1} = 0.8,\ \alpha_{cp,h_{R_1}-1,h_{R_1}} = 0.2\}$), since this access pattern has been widely employed in many analysis and simulations [6,25],

$$\beta_{cp,l_1,j} = \frac{N_{R_1,l_1} + N_{R_2,l_2}}{Nodes_{R_1} + Nodes_{R_2}} \quad \text{and } \alpha_{cp,l_1,j} = \frac{1}{h_{R_1}}$$

$$\beta_{cp,l_1,h_{R_1}-1} = \frac{0.2 * (N_{R_1,l_1} + N_{R_2,l_2})}{Nodes_{R_1} + Nodes_{R_2}} \quad \text{and } \alpha_{cp,l_1,h_{R_1}-1} = \frac{0.8}{h_{R_1}}$$

$$\beta_{cp,l_1,h_{R_1}} = \frac{0.8 * (N_{R_1,l_1} + N_{R_2,l_2})}{Nodes_{R_1} + Nodes_{R_2}} \quad \text{and } \alpha_{cp,l_1,h_{R_1}} = \frac{0.2}{h_{R_1}}$$

where, $l_1 = \begin{cases} j, & 0 \leqslant j \leqslant h_{R_1} - 2 \\ h_{R_1} - 1, & h_{R_1} - 1 \leqslant j \leqslant p_{cp} - 1 \end{cases}$ and $l_2 = \begin{cases} l_1, & 0 \leqslant l_1 \leqslant h_{R_2} - 1 \\ h_{R_2} - 1, & h_{R_2} \leqslant l_1 \leqslant h_{R_1} - 1 \end{cases}$.

According to [6], we know that $(1 - \sigma(K))^{\alpha_{cp,l_i,j} * n}$ is the probability that a pair of MBRs separated by a distance smaller than or equal to $dist_{cp}(K)$ is not accessed after n accesses from each level of the R-trees R_1 and R_2 (l_i, such that $l_2 \leqslant l_1$) in the partition j. Thus, $1 - (1 - \sigma(K))^{\alpha_{cp,l_i,j} * n}$ is the probability that a pair of MBRs separated by a distance smaller than or equal to $dist_{cp}(K)$ is accessed after n accesses from each level of the R-trees R_1 and R_2 (l_i, such that $l_2 \leqslant l_1$) in the partition j. Therefore, after n accesses to R-trees R_1 and R_2 with access pattern AP_{cp,l_1} ($p_{cp} = h_{R_1} + 1$ ($0 \leqslant j \leqslant h_{R_1}$) partitions) and containing $Nodes_{R_1} + Nodes_{R_2}$ R-tree nodes, the number of distinct R-tree nodes of R-trees R_1 and R_2 at levels l_1 and l_2 that have been accessed for K-CPQ can be approximated by:

$$B_{cp,l_1,j}\left(n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1}\right) = \left(\beta_{cp,l_1,j} * \left(Nodes_{R_1} + Nodes_{R_2}\right)\right) * \left(1 - (1 - \sigma(K))^{\alpha_{cp,l_1,j} * n}\right)$$

and the average number of distinct R-tree nodes can be also estimated by:

$$B_{cp}\left(n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1}\right) = \sum_{j=0}^{p_{cp}-1} \left\{B_{cp,l_1,j}\left(n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1}\right)\right\}$$

where

$$l_1 = \begin{cases} j, & 0 \leqslant j \leqslant h_{R_1} - 2 \\ h_{R_1} - 1, & h_{R_1} - 1 \leqslant j \leqslant p_{cp} - 1 \end{cases} \quad \text{and } l_2 = \begin{cases} l_1, & 0 \leqslant l_1 \leqslant h_{R_2} - 1 \\ h_{R_2} - 1, & h_{R_2} \leqslant l_1 \leqslant h_{R_1} - 1 \end{cases}$$

Finally, we can estimate the average buffer hit probability for K-CPQ in an LRU-managed buffer given a certain access pattern, by using the buffer hit probability equation ($H(n,N,AP) = \sum_j \{\alpha_j * H_j(n,N,AP)\}$), with $B_{cp}(n, Nodes_{R_1} + Nodes_{R_2}, \mathrm{AP}_{cp,l_1}) = \mathrm{B}$ defined above as: $H_{LRU\_cp}(B,n,Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1}) = H(n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1}) = \sum_j \{\alpha_{cp,l_1,j} * H_{cp,l_1,j}(n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1})\}$. That is,

$$H_{LRU\_cp}\left(B, n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1}\right) = \sum_{j=0}^{p_{cp}-1} \left\{\alpha_{cp,l_1,j} * H_{cp,l_1,j}\left(n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1}\right)\right\} \tag{12}$$

where, $H_{cp,l_1,j}(n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1}) = 1 - (1 - \sigma(K))^{\alpha_{cp,l_1,j} * n}$, and

$$l_1 = \begin{cases} j, & 0 \leqslant j \leqslant h_{R_1} - 2 \\ h_{R_1} - 1, & h_{R_1} - 1 \leqslant j \leqslant p_{cp} - 1 \end{cases}, \quad l_2 = \begin{cases} l_1, & 0 \leqslant l_1 \leqslant h_{R_2} - 1 \\ h_{R_2} - 1, & h_{R_2} \leqslant l_1 \leqslant h_{R_1} - 1 \end{cases}$$

The estimated average buffer hit probability ($H_{LRU\_cp}(B, n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1})$) for K-CPQ represents the number of hits (avoided disk accesses) in an LRU-managed buffer with size B with respect to the total disk accesses without buffer ($NA_{cp}(R_1, R_2, K)$) for this kind of query.

$$H_{LRU\_cp}(B, n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1}) = \frac{NA_{avoided\_LRU\_cp}(B, R_1, R_2, K)}{NA_{cp}(R_1, R_2, K)}$$

On the other hand, the estimated average buffer no-hit probability ($1.0 - H_{LRU\_cp}(B, n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1})$) for K-CPQ represents the number of failures (disk accesses or node faults) in an LRU-managed buffer with size B with respect to the total disk accesses without the presence of buffer.

$$NA_{LRU\_cp}(B, R_1, R_2, K) = \left(1.0 - H_{LRU\_cp}(B, n, Nodes_{R_1} + Nodes_{R_2}, AP_{cp,l_1})\right) * NA_{cp}(R_1, R_2, K) \tag{13}$$

Obviously, B must be smaller than or equal to $Nodes_{R_1} + Nodes_{R_2}$ to have impact in the query processing, i.e. larger buffer sizes will not have any positive effect in the overall buffer performance. Moreover, $NA_{LRU\_cp}(B, R_1, R_2, K)$ can be bounded by $Nodes_{R_1} + Nodes_{R_2}$ as lower bound, although in real experiments this value can be slightly smaller than $Nodes_{R_1} + Nodes_{R_2}$, due to some R-tree nodes cannot be accessed during the processing of the query algorithm. Therefore, the estimated average buffer hit probability can be bounded as follows:

$$H_{LRU\_cp}(\cdots) = \begin{cases} 1.0 - \dfrac{NA_{LRU\_cp}(\cdots)}{NA_{cp}(\cdots)}, & NA_{LRU\_cp}(\cdots) > Nodes_{R_1} + Nodes_{R_2} \\ 1.0 - \dfrac{Nodes_{R_1} + Nodes_{R_2}}{NA_{cp}(\cdots)}, & \text{otherwise} \end{cases} \tag{14}$$

To extend the previous formulae (12), (13) and (14) from K-CPQ to *buffer queries* (bq) is not difficult. First, $\beta_{bq,l_i,j}$ and $\alpha_{bq,l_i,j}$ are the same as $\beta_{cp,l_i,j}$ and $\alpha_{cp,l_i,j}$. Second, to compute $B_{bq,l_i,j}$, we have to consider $\sigma(\delta)$ instead of $\sigma(K)$. And finally, for computing $NA_{LRU\_bq}(B, R_1, R_2, \delta)$ and $H_{LRU\_bq}(B, n, Nodes_{R_1} + Nodes_{R_2}, AP_{bq,l_1})$, we need to use $NA_{bq}(R_1, R_2, \delta)$ instead of $NA_{cp}(R_1, R_2, K)$.

5. Experimental results

This section experimentally evaluates the proposed model, using R*-tree [2] as the underlying spatial access method. If the R*-trees have different heights, we will use the *fix-at-leaves* technique [11]. We have used synthetic datasets (uniform distributions, UN1 and UN2) that contain 100,000 2-dimensional points and real-life datasets (the 2-dimensional data space is normalized to have unit length). Real 2-dimensional datasets are data of California: (1) from [34] that contains 98,451 points (MBRs of streams (line segments), which have been transformed to points by taking the middle point of each segment, CAS); and (2) from Sequoia benchmark [31] that consists of 62,556 points (populated places, CAP). The used point datasets are depicted in the Fig. 7: (a) uni-

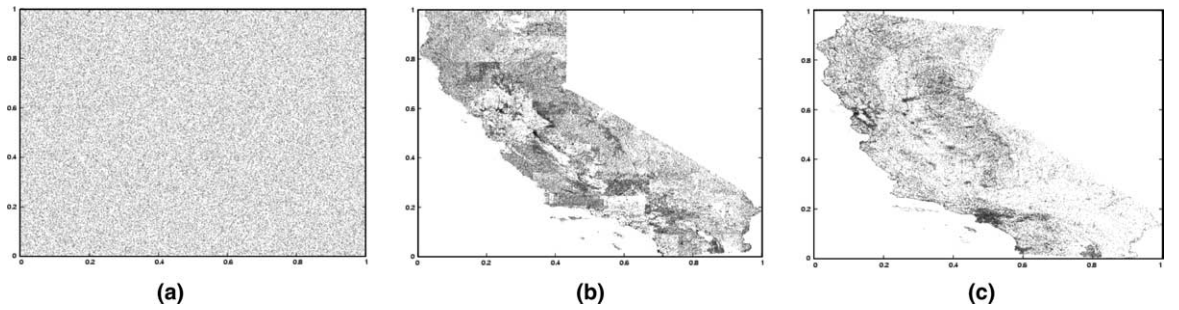

Fig. 7. Datasets used in the experiments: (a) uniform, (b) CAS, (c) CAP.

Table 4
Estimated $dist_{cp}(K)$ and measured distance value Kth closest pair at the end of the execution of K-CPQ algorithm (z) for uniform and real datasets

K	(UN1, UN2)		(CAS, CAP)	
	z	$dist_{cp}(K)$	z	$dist_{cp}(K)$
1	0.0000007123	0.0000056419	0.0000041164	0.0000054578
10	0.0000299811	0.0000178412	0.0000131463	0.0000177561
100	0.0000679477	0.0000564189	0.0000530100	0.0000577669
1000	0.0001854371	0.0001784124	0.0001649575	0.0001879355
10,000	0.0005614171	0.0005641895	0.0005294667	0.0006114188
100,000	0.0017900597	0.0017841241	0.0016640083	0.0019891558

form, (b) CAS and (c) CAP. For uniform datasets, we have the correlation exponent value ρ(UN1, UN2) = 2.0; and for real datasets ρ(CAS, CAP) = 1.951866859. Moreover, the average node utilization is set to 70% ($Uavg_{R_i} = 0.7$), typical value according to [14] for the R*-tress.

Our cost model is based on the estimation of the distance of the Kth closest pair ($dist_{cp}(K)$). Table 4 shows the estimated $dist_{cp}(K)$ and z (measured distance value of the Kth closest pair found at the end of the execution of the K-CPQ algorithm), varying the cardinality of the result (K) from 1 to 100,000, for uniform and real datasets. We can observe the small difference between them, even for large K values. Therefore, our formula to estimate the distance of the Kth closest pair ($dist_{cp}(K)$) is quite accurate and appropriate to use in the computation on *K-CPQ index selectivity* ($\sigma(K)$), which is the basis of the $NA_{cp}(R_1, R_2, K)$ formula.

To show the accuracy of our cost model, the next experiment studies the behavior of the formula (8) that estimates the total I/O cost in terms of R-tree node accesses ($NA_{cp}(R_1, R_2, K)$) for K-CPQ with respect to the increase of K (number of pairs in the final result) values, varying from 1 to 100,000. Fig. 8 illustrates the measured and the estimated disk accesses for uniform (Fig. 8a) and real (Fig. 8b) datasets where the maximum R-tree node capacity ($Cmax_{R_i}$) was set to 50 (1 Kbyte). We can notice from both charts that they follow very similar trends (the lines for measured values is slightly larger than the estimated one), which average relative error is around 0–4% for uniform and 1–2% for real datasets.

To study the behavior of our cost model with respect to the variation in the maximum branching factor of the R*-trees ($Cmax_{R_i} = 25$, 50, 100 and 200) as in [22] for K-NNQ, and K (from 1 to

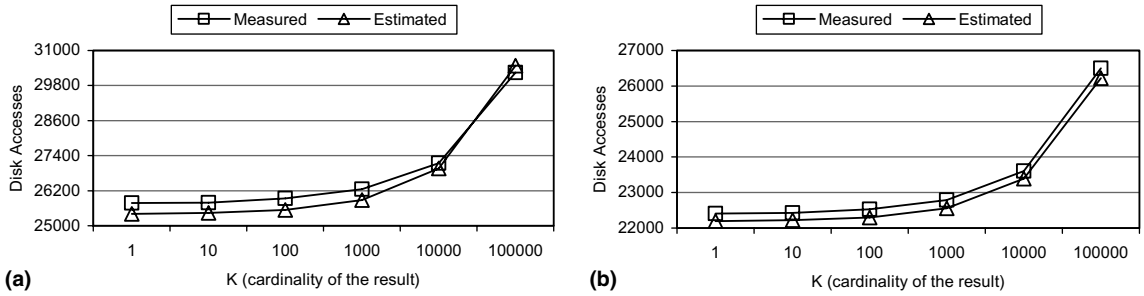

Fig. 8. Estimated and measured number of R-tree node accesses for K-CPQ, varying K and using uniform (a) and real (b) datasets.

Table 5
Estimated $NA_{cp}(R_1, R_2, K)$ and measured R-tree node accesses for K-CPQ for real datasets (CAS, CAP), varying the branching factor ($Cmax_{R_i}$) and cardinality of the final query result (K)

K	25		50		100		200	
	Measured	Estimated	Measured	Estimated	Measured	Estimated	Measured	Estimated
1	33,232	33,179	22,410	22,194	9142	9031	4330	4355
10	33,292	33,238	22,430	22,218	9148	9039	4332	4358
100	33,488	33,432	22,528	22,297	9176	9065	4338	4366
1000	34,046	34,066	22,784	22,553	9270	9151	4358	4394
10,000	36,256	36,159	23,602	23,395	9550	9431	4430	4487
100,000	43,486	43,312	26,502	26,225	10,458	10,365	4726	4791

100,000), keeping the population of the read datasets constant (CAS, CAP, and $\rho = 1.951866859$). Table 5 shows the estimated number of R-tree node accesses from the formula $NA_{cp}(R_1, R_2, K)$ and the measured disk accesses using the Best-First K-CPQ algorithm. From these results we can experimentally prove the accuracy of our cost model for this kind of distance-based join query (K-CPQ). For example, we have considered very large K values ($K = 100{,}000$) and the difference between measured and estimated R-tree node accesses is very low (around 2% in average for all $Cmax_{R_i}$ values).

The same experiment was run for uniform datasets and the Table 6 shows the measured and estimated number of R-tree node accesses. As in Table 5, the estimated values are close to the measured ones (e.g. $Cmax_{R_i} = 50$ the average relative error is around 1.5% for all K values), although for $Cmax_{R_i} = 200$ the relative error is increased around 5%.

Another alternative to measure the accuracy of our cost model is to show the *relative error* (RE) of the measured results (R-tree node accesses) compared to the predictions.

$$RE = \sum_{i=1}^{K} \frac{abs(\text{EstimatedNodeAccesses} - \text{MeasuredNodeAccesses})}{\text{MeasuredNodeAccesses}}$$

Fig. 9 illustrates the relative error of the measured R-tree node accesses with respect to the estimated ones for uniform (Fig. 9a) and real (Fig. 9b) datasets, where the K and $Cmax_{R_i}$ are varied. For uniform distributions the relative error is below 6% (for the highest $Cmax_{R_i} = 200$, we can no-

Table 6
Estimated $NA_{cp}(R_1, R_2, K)$ and measured R-tree node accesses for K-CPQ for uniform datasets, varying the branching factor ($Cmax_{R_i}$) and cardinality of the final query result (K)

K	25		50		100		200	
	Measured	Estimated	Measured	Estimated	Measured	Estimated	Measured	Estimated
1	47,576	46,237	25,780	25,405	12,148	12,124	5406	5674
10	47,622	46,326	25,788	25,438	12,156	12,136	5412	5678
100	48,008	46,611	25,934	25,545	12,190	12,171	5422	5690
1000	48,802	47,516	26,254	25,882	12,336	12,285	5458	5727
10,000	51,292	50,424	27,148	26,958	12,638	12,645	5558	5845
100,000	59,772	60,088	30,240	30,477	13,634	13,812	5914	6224

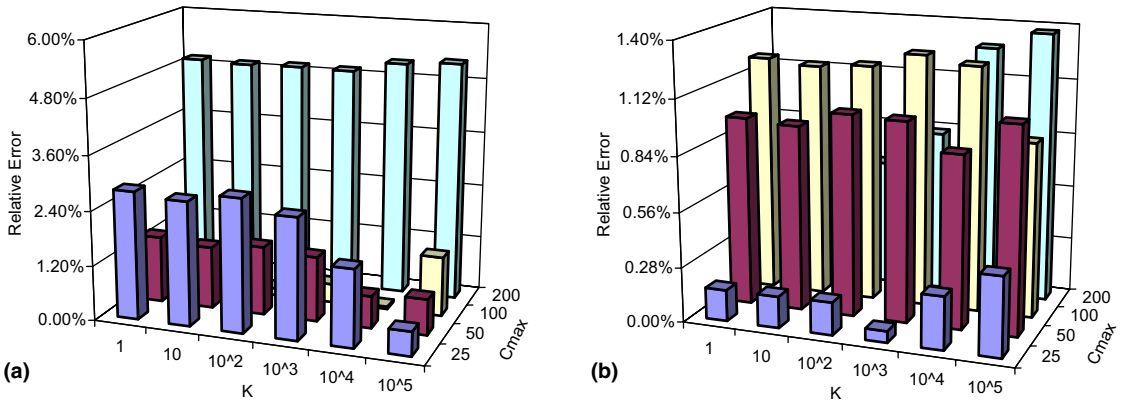

Fig. 9. Relative error for the estimation cost of K-CPQ, using uniform (a) and real (b) datasets.

tice the largest relative error). For real datasets, the relative error is lower than the uniform (less than 1.4%), highlighting the values for the smallest fan-out ($Cmax_{R_i} = 25$), which are smaller than 0.28%. In general, the relative error is very small for all our experiments. It means that our cost model for K-CPQ is very accurate.

In the next experiment, we evaluate the analytical formula for K-CPQ estimation consists of varying the cardinality of one of the real dataset (CAS), keeping constant the other one (CAP). We have produced three different datasets from CAS, choosing randomly 73,838 (CAST), 49,226 (CASH) and 24,613 (CASC). The values of the correlation exponent for these datasets with respect to CAP are the following: ρ(CASC, CAP) = 1.925609934, ρ(CASH, CAP) = 1.901360599, ρ(CAST, CAP) = 1.978895579, and ρ(CAS, CAP) = 1.951866859. Also, we have to highlight that fixing $Cmax_{R_i} = 50$ the heights of the R*-trees are $h_{CAP} = 3$, $h_{CAS} = 4$, $h_{CAST} = 4$, $h_{CASH} = 3$ and $h_{CASC} = 3$. Fig. 10 illustrates estimated R-tree node accesses using $NA_{cp}(R_1, R_2, K)$ (Fig. 10a) and the relative error (Fig. 10b) of the measured node accesses with respect to the estimated ones for real datasets varying the cardinality of CAS. From the left chart, we can observe trends very similar to the measured disk accesses, even when the R-tree heights are different. But the most interesting conclusion can be obtained from the right chart, the relative error (of the measured results compared to the predictions of the proposed model) is usually around 0–2%, which confirm the

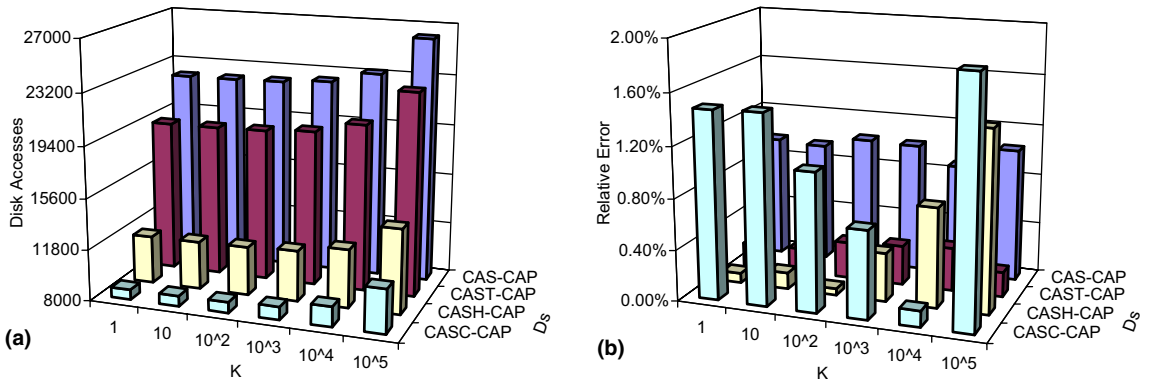

Fig. 10. (a) Estimated R-tree node accesses for K-CPQ for real datasets (CAS*, CAP); (b) relative error (right) with respect to the measured results.

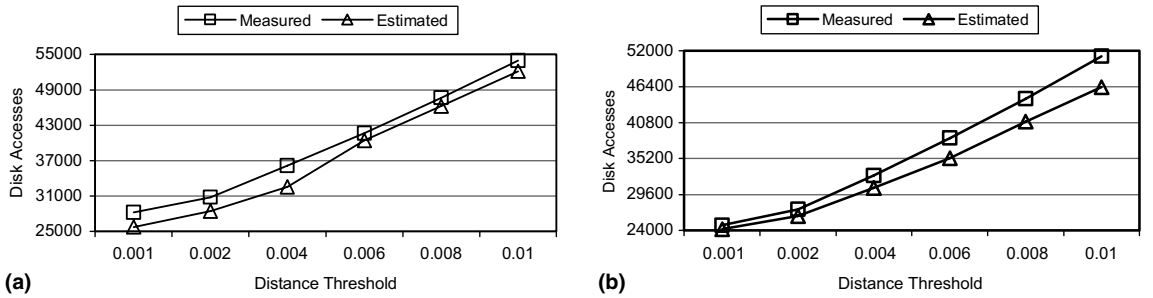

Fig. 11. Estimated and measured number of R-tree node accesses for *buffer queries*, varying δ and using uniform (a) and real (b) datasets.

accuracy of the formula of our cost model for K-CPQ (mainly based on the correlation exponent ρ) when the cardinality of the datasets involved on the distance-based join query is varied.

The same experiments have been performed for *buffer queries* (another kind of distance join), where the distance threshold δ can be given by the user and it is not necessary its estimation. The Fig. 11 shows the estimated and measured number of R-tree node accesses for such a query from the formula (11), varying δ for the values {0.001, 0.002, 0.004, 0.006, 0.008 and 0.01} and using uniform (Fig. 11a) and real (Fig. 11b) datasets ($Cmax_{R_i} = 50$). Again, both charts follow the same tendency, although the relative error between them is slightly larger than for K-CPQ, it is around 3–9% for uniform and 2–10% for real datasets. It is interesting to observe the trend of the curves in the right chart, where the larger the threshold distance is the larger the gap becomes.

The behavior of the cost model with respect to the variation of $Cmax_{R_i}$ of the R*-trees and the distance threshold δ is shown in Table 7. The estimation of the number of R-tree node accesses using the formula $NA_{bq}(R_1, R_2, \delta)$ is quite accurate, e.g. for large δ values ($\delta = 0.01$) the difference between the measured and estimated R-tree node accesses is around 10% in average for all $Cmax_{R_i}$ values.

Table 7
Estimated $NA_{bq}(R_1, R_2, \delta)$ and measured R-tree node accesses for *buffer queries* for real datasets (CAS, CAP), varying the branching factor ($Cmax_{R_i}$) and the distance threshold (δ)

δ	25		50		100		200	
	Measured	Estimated	Measured	Estimated	Measured	Estimated	Measured	Estimated
0.001	39,308	38,684	24,776	24,179	9918	9404	4554	4572
0.002	45,764	44,743	27,302	26,248	10,726	10,056	4832	4866
0.004	59,562	56,582	32,560	30,607	12,394	11,229	5356	5333
0.006	75,572	70,890	38,392	35,258	14,284	13,051	6000	5816
0.008	93,486	85,395	44,502	40,960	16,288	14,321	6650	6316
0.01	113,064	101,089	51,170	46,326	18,314	15,879	7304	6832

Table 8
Estimated $NA_{bq}(R_1, R_2, \delta)$ and measured R-tree node accesses for *buffer queries* for real datasets (CAS, CAP), varying the branching factor ($Cmax_{R_i}$) and the distance threshold (δ)

δ	25		50		100		200	
	Measured	Estimated	Measured	Estimated	Measured	Estimated	Measured	Estimated
0.001	54,144	52,171	28,246	25,762	12,994	12,685	5686	5980
0.002	61,318	59,889	30,804	28,475	13,814	13,637	5968	6293
0.004	76,282	72,817	36,166	32,559	15,656	15,370	6588	7048
0.006	92,598	90,147	41,646	40,447	17,574	16,844	7148	7610
0.008	110,892	109,124	47,646	46,229	19,548	19,241	7816	8306
0.01	129,870	129,747	53,952	52,157	21,608	21,077	8510	9029

The same experiment was run for uniform datasets and the Table 8 shows the measured (experimental results using the algorithm of Fig. 5) and estimated number of R-tree node accesses ($NA_{bq}(R_1, R_2, \delta)$. As in Table 7, the estimated values are close to the measured ones (e.g. $Cmax_{R_i} = 25$ the average relative error is around 3% for all δ values), although for $Cmax_{R_i} = 50$ and $Cmax_{R_i} = 200$ the relative errors are increased around 6% in average.

Fig. 12 illustrates the relative error of the measured R-tree node accesses with respect to the estimated ones ($NA_{bq}(R_1,R_2, \delta)$) of *buffer queries* for uniform (Fig. 12a) and real (Fig. 12b) datasets, where δ and $Cmax_{R_i}$ are varied. For uniform distributions the relative error is below 9% (for a medium $Cmax_{R_i}$ (=50) and low δ values, we can detect the largest relative error, 8.79%). For real datasets, the relative error is larger than the uniform (less than 14%), highlighting the relative error values for the largest $Cmax_{R_i}$ (=200), which are smaller than 4% in average for all δ values.

As conclusion of the previous results of our cost model for R-tree distance join queries (*K*-CPQ and buffer queries) is that it achieves an average relative error for R-tree node accesses with a maximum value of 6% (*K*-CPQ) and 14% (buffer queries). We believe these error values are within an acceptable confidence level of cost prediction for distance join queries, demonstrating that our cost model can be considered as an effective tool in the cost estimations of R-tree distance joins during the distance join query optimization step.

On the other hand, to validate experimentally our LRU buffer model for distance join queries (*K*-CPQ), we have also compared the estimated result with the measured ones for the number of

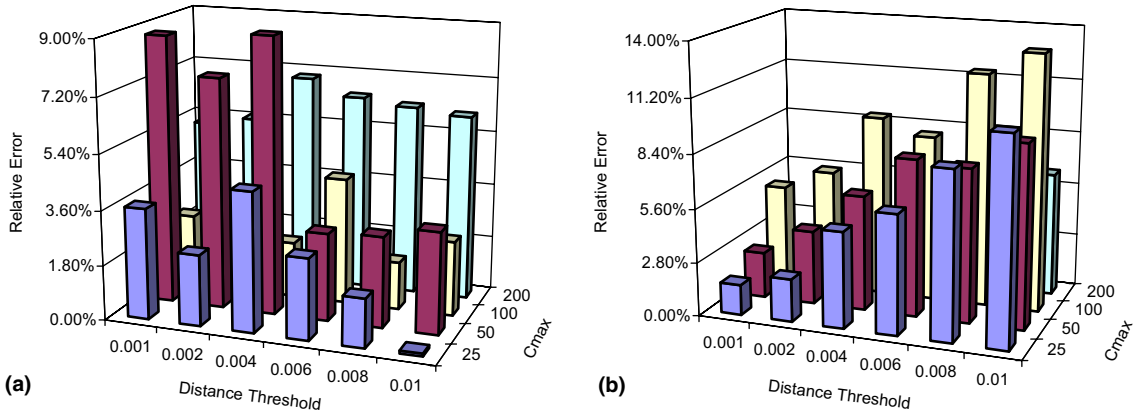

Fig. 12. Relative error for the estimation cost of *buffer queries*, using uniform (a) and real (b) datasets.

disk (R-tree node) accesses and the average buffer hit probability according to the formulae (13) and (14), respectively. The formula (13) for $\mathrm{NA}_{LRU\_cp}(B, R_1, R_2, K)$ is bounded by $Nodes_{R_1} + Nodes_{R_2}$. The estimated results have been performed for the 80/20 access pattern for partitions at leaf level, $K = 100$, with different maximum R-tree node capacity ($Cmax_{R_i} = 50$ and 200, although for the other $Cmax_{R_i}$ values we have got similar qualitative results), buffer sizes ($B = 0$, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 and 8192) and data distributions (uniform and real datasets). In our experiments, we maintain an LRU-managed buffer with size B and an access to an R-tree node resident in the buffer, makes the R-tree node move to the front of the LRU buffer according to this page replacement policy. The buffer model formulae assume a *hot buffer*, i.e. we warm up the buffer by doing a large number of requests, before we start to measure the buffer hit probability. Finally, we will compute the measured average buffer hit probability as follows (similar to Eq. (14)):

$$H_{LRU\_cp\_measured}(B, R_1, R_2, K) = 1.0 - \frac{NA_{LRU\_cp\_measured}(B, R_1, R_2, K)}{NA_{cp\_measured}(R_1, R_2, K)}$$

Fig. 13 illustrates the overall buffer hit probability for uniform datasets, different maximum fanouts and buffer sizes. We can see clearly how the average buffer hit probability increases when an LRU buffer is present, even with small size ($B = 8$ R-tree nodes). After that point, it grows more slowly (in the range between 0.4 and 0.8), until it reaches the point where the buffer size is equal to $Nodes_{R_1} + Nodes_{R_2}$. We can also observe how close the estimated values are to the measured results, and the small difference between $Cmax_{R_i} = 50$ and $Cmax_{R_i} = 200$ (the larger $Cmax_{R_i}$ the smaller buffer hit probability).

For number of R-tree nodes, the Fig. 14 shows the same experimental configurations. The measured results are very close to the estimated ones, which confirm the accuracy of our buffer model for this kind of R-tree distance join query (K-CPQ). For example, the average relative error between the measured and estimated number of R-tree node accesses for $Cmax_{R_i} = 50$ is around 7.5% and for $Cmax_{R_i} = 200$ is 9.3%, which are acceptable results to demonstrate the accuracy of our R-tree buffer model for K-CPQ.

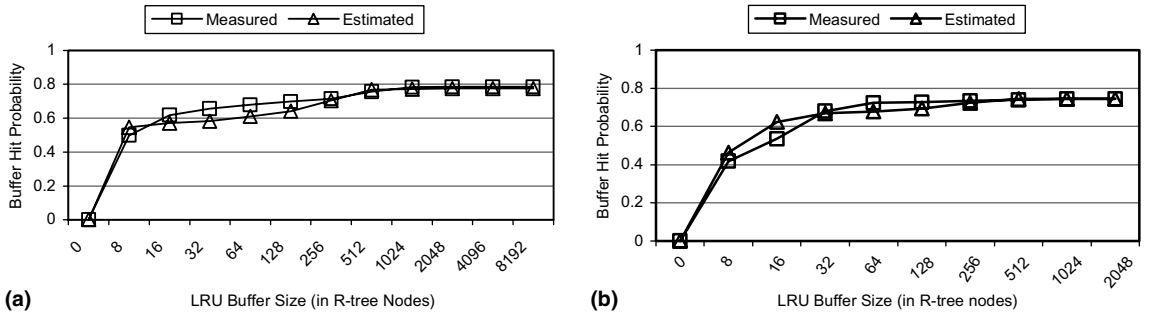

Fig. 13. Estimated and measured average buffer hit probability for *K*-CPQ, varying the LRU buffer size B for the values {0, 8, 16, 64, 128, 256, 512, 1024, 2048, 4096 and 8192}, $K = 100$, $Cmax_{R_i} = 50$ (a) and $Cmax_{R_i} = 200$; (b) for both uniform datasets.

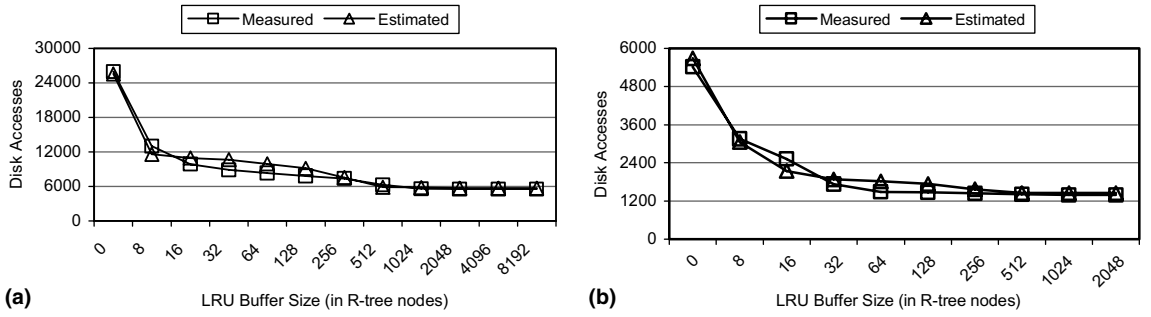

Fig. 14. Estimated and measured R-tree node accesses for *K*-CPQ, varying the LRU buffer size B for the values {0, 8, 16, 64, 128, 256, 512, 1024, 2048, 4096 and 8192}, $K = 100$, $Cmax_{R_i} = 50$ (a) and $Cmax_{R_i} = 200$ (b) for both uniform datasets.

For real datasets, we have observed very similar behaviors for the buffer hit probability and for the I/O activity needed for *K*-CPQ, using similar configurations (80/20 access pattern, $K = 100$, different $Cmax_{R_i}$ values (50 and 200), buffer sizes, etc.). For instance, in the Fig. 15, the trends of the curves for measured and estimated buffer hit probability are similar to the uniform case. And in the

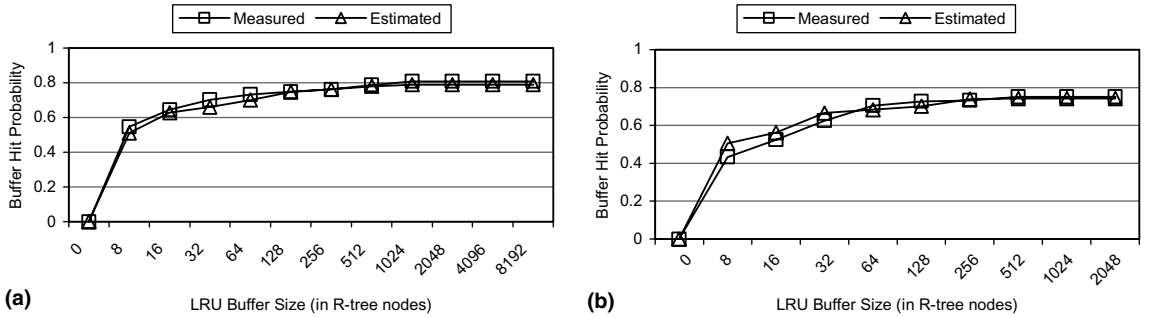

Fig. 15. Estimated and measured average buffer hit probability for *K*-CPQ, varying the LRU buffer size B for the values (0, 8, 16, 64, 128, 256, 512, 1024, 2048, 4096 and 8192), $K = 100$, $Cmax_{R_i} = 50$ (a) and $Cmax_{R_i} = 200$ (b) for both real datasets (CAS, CAP).

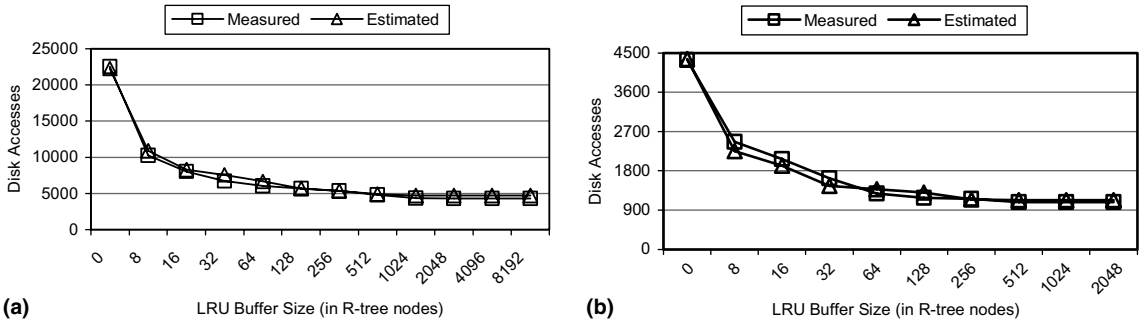

Fig. 16. Estimated and measured R-tree node accesses for *K*-CPQ, varying the LRU buffer size B for the values (0, 8, 16, 64, 128, 256, 512, 1024, 2048, 4096 and 8192), $K = 100$, $Cmax_{R_i} = 50$ (a) and $Cmax_{R_i} = 200$ (b) for both real datasets (CAS, CAP).

Fig. 16, the average relative error between the measured and estimated number of R-tree node accesses for $Cmax_{R_i} = 50$ is around 6.2% and for $Cmax_{R_i} = 200$ is 6% (these errors are acceptable).

Again, as conclusion of these results related to LRU buffer model for R-tree distance join queries (*K*-CPQ) is that, it achieves an average relative error for R-tree node accesses with a maximum value of 9% and the estimated average buffer hit probability obtained results very close to the measured ones. Thus, we believe these error values are within an acceptable confidence level of cost prediction for distance join queries, demonstrating that our buffer model is quite precise.

6. Conclusions and ideas for future extensions

In this paper, we have presented an I/O cost model for the *K Closest Pairs Query* (*K*-CPQ) and others R-tree distance join queries. The development of our model was based on previous analytical results for nearest neighbor and spatial intersect join queries, which can be considered a generalization of them. We avoided following the *unrealistic* assumptions of uniformity and independence. On the contrary, the formulae we developed correspond to real data and depend mainly on the correlation exponent (ρ) and number of pairs in the final result (K). Our analysis assumes a typical (non-uniform) workload where *queries are more probable in high-density areas of the address space*. The proposed cost model (formulae and guideline) could be used in a query optimizer for spatial query processing and optimization purposes, where distance join queries are involved. Moreover, we have successfully extended our cost model to *buffer queries* and to the analytical study of including buffering (LRU buffer model) in *K*-CPQ using R-trees.

Experimental results on both synthetic (uniform) and real datasets showed that the proposed analytical model is very accurate (varying several parameters as cardinality of the query result (K), maximum branching factor ($Cmax_{R_i}$), different dataset cardinalities and data distributions, etc.), with the relative error being usually around 0–6% (*K*-CPQ) and 0–14% (buffer queries) when the estimated results are compared to the measured costs using R*-trees. We believe these error values are within an acceptable confidence level of cost prediction for distance join queries, demonstrating that our cost model can be considered as an effective tool in the cost estimations of R-tree distance joins during the distance join query optimization step. Finally, the experiments

of our cost model, when an LRU-managed buffer is added, have shown that theoretical results have been very similar to the measured ones in terms of buffer hit probability and number of R-tree node accesses, which confirm the accuracy of the proposed extensions of our cost model.

Future work may include:

- Experimentation with high-dimensional points datasets, taking into account an upper and lower bound of the average number of query sensitive anchors (based on Euclidean and maximum metrics) rather than the exact values as in [22].
- Use of new approximation techniques for K-CPQ, in a similar way that in [33] for K-NNQ.
- Extend the proposed LRU buffer model to distance joins with multiple inputs indexed in R-trees [12], not only when we have two R-trees.
- Experimentation with different access pattern for the LRU buffer model (70/30, 90/10 and 95/05) with two partitions at leaf level and more partitions with appropriate access patterns.
- Generalization of our study for non-points spatial objects [28].

Acknowledgement

Research partially supported by INDALOG TIC2002-03968 project "A Database Language Based on Functional Logic Programming" of the Spanish Ministry of Science and Technology under FEDER funds, EPEAEK II/ARCHIMEDES 2.2.14 project "Management of Moving Objects and the WWW" of the Greek Ministry of National Education and Religious Affairs co-funded by the European Union and EPEAEK II/PYTHAGORAS 2.2.3 project "Spatio-temporal Data and Knowledge Management in Expert Virtual Environments" of the Greek Ministry of National Education and Religious Affairs co-funded by the European Union.

References

[1] W.G. Aref, H. Samet, A cost model for query optimization using R-trees, in: Proceedings 2nd ACM Conference on Geographical Informaton Systems (GIS), 1994.
[2] N. Beckmann, H.P. Kriegel, R. Schneider, B. Seeger, The R*-tree: an efficient and robust access method for points and rectangles, in: Proceedings ACM SIGMOD Conference, 1990, pp. 322–331.
[3] A. Belussi, C. Faloutsos, Self-spatial join selectivity estimation using fractal concepts, ACM Transactions on Information Systems 16 (2) (1998) 161–201.
[4] S. Berchtold, C. Böhm, D.A. Keim, H.P. Kriegel, A cost model for nearest neighbor search in high-dimensional data space, in: Proceedings 16th ACM Conference on Principles of Database Systems (PODS), 1997, pp.78–86.
[5] S. Berchtold, D. Kiem, H.P. Kriegel, The X-tree: an index structure for high-dimensional data, in: Proceedings 22nd International Conference on Very Large Data Base Conference (VLDB), 1996, pp. 28–39.
[6] A.K. Bhide, A. Dan, D.M. Dias, A simple analysis of the LRU buffer policy and its relationship to buffer warm-up transient, in: Proceedings 9th IEEE International Conference on Data Engineering (ICDE), 1993, pp. 125–133.
[7] C. Bohm, A cost model for query processing in high-dimensional data spaces, ACM Transactions on Database Systems 25 (2) (2000) 129–178.
[8] C. Bohm, H.P. Kriegel, A cost model and index architecture for the similarity join, in: Proceedings 17th IEEE International Conference on Data Engineering (ICDE), 2001, pp. 411–420.
[9] E.P.F. Chan, Buffer queries, IEEE Transactions Knowledge Data Engineering 15 (4) (2003) 895–910.
[10] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2003.

[11] A. Corral, Y. Manolopoulos, Y. Theodoridis, M. Vassilakopoulos, Closest pair queries in spatial databases, in: Proceedings ACM SIGMOD Conference, 2000, pp. 189–200.
[12] A. Corral, Y. Manolopoulos, Y. Theodoridis, M. Vassilakopoulos, Distance join queries of multiple inputs in spatial databases, in: Proceedings 7th East-European Conference on Advances in Databases and Information Systems (ADBIS), 2003, pp. 323–338.
[13] A. Corral, Y. Manolopoulos, Y. Theodoridis, M. Vassilakopoulos, Algorithms for processing K closest pair queries in spatial databases, Data and Knowledge Engineering 49 (1) (2004) 67–104.
[14] C. Faloutsos, I. Kamel, Beyond uniformity and independence: analysis of R-trees using the concept of fractal dimension, in: Proceedings 13th ACM Conference on Principles of Database Systems (PODS), 1994, pp. 4–13.
[15] C. Faloutsos, B. Seeger, A. Traina, C. Traina, Spatial join selectivity using power law, in: Proceedings ACM SIGMOD Conference, 2000, pp. 177–188.
[16] C. Faloutsos, T. Sellis, N. Roussopoulos, Analysis of object oriented spatial access methods, in: Proceedings ACM SIGMOD Conference, 1987, pp. 426–439.
[17] V. Gaede, O. Günther, Multidimensional access methods, Computing Surveys 30 (2) (1998) 170–231.
[18] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: Proceedings ACM SIGMOD Conference, 1984, pp. 47–57.
[19] A. Henrich, The LSD^h-Tree: an access structure for feature vectors, in: Proceedings 14th IEEE International Conference on Data Engineering (ICDE), 1998, pp. 362–369.
[20] Y.W. Huang, N. Jing, E.A. Rundensteiner, A cost model for estimating the performance of spatial joins using R-trees, in: Proceedings 9th International Conference on Scientific and Statistical Database Management (SSDBM), 1997, pp. 30–38.
[21] I. Kamel, C. Faloutsos, On packing R-trees, in: Proceedings 2nd International Conference on Information and Knowledge Management (CIKM), 1993, pp. 490–499.
[22] F. Korn, B.U. Pagel, C. Faloutsos, On the 'Dimensionality Curse' and the 'Self-Similarity Blessing', IEEE Transactions on Knowledge and Data Engineering 13 (1) (2001) 96–111.
[23] N. Koudas, K.C. Sevcik, High dimensional similarity joins: Algorithms and performance evaluation, IEEE Transactions Knowledge Data Engineering 12 (1) (2000) 3–18.
[24] S.T. Leutenegger, M.A. Lopez, The effect of buffering on the performance of R-trees, IEEE Transactions Knowledge Data Engineering 12 (1) (2000) 33–44.
[25] K. Nørvåg, K. Bratbergsengen, An analytical study of object identifier indexing, in: Proceedings 9th International Conference on Database and Expert Systems Applications (DEXA), 1998, pp. 38–49.
[26] B.U. Pagel, H.W. Six, H. Toben, P. Widmayer, Towards an analysis of range query performance, in: Proceedings 12th ACM Conference on Principles of Database Systems (PODS), 1993, pp. 214–221.
[27] A. Papadopoulos, Y. Manolopoulos, Performance of nearest neighbor queries in R-Trees, in: Proceedings 6th International Conference on Database Theory (ICDT), 1997, pp. 394–408.
[28] G. Proietti, C. Faloutsos, Accurate modeling of region data, IEEE Transactions on Knowledge Data Engineering 13 (6) (2001) 874–883.
[29] N. Roussopoulos, S. Kelley, F. Vincent, Nearest neighbor queries, in: Proceedings ACM SIGMOD Conference, 1995, pp. 71–79.
[30] S. Shekhar, S. Chawla, S. Rivada, A. Fetterer, X. Lui, C. Lu, Spatial databases, accomplishments and research needs, IEEE Transactions on Knowledge and Data Engineering 11 (1) (1999) 45–55.
[31] M. Stonebraker, J. Frew, K. Gardels, J. Meredith, The Sequoia 2000 Benchmark, in: Proceedings ACM SIGMOD Conference, 1993, pp. 2–11.
[32] Y. Theodoridis, E. Stefanakis, T. Sellis, Efficient cost models for spatial queries using R-trees, IEEE Transactions on Knowledge and Data Engineering 12 (1) (2000) 19–32.
[33] Y. Tao, J. Zhang, D. Papadias, N. Mamoulis, An efficient cost model for optimization of nearest neighbor search in low and medium dimensional spaces, IEEE Transactions on Knowledge and Data Engineering 16 (10) (2004) 1169–1184.
[34] Y. Manolopoulos, A. Nanopoulos, A.N. Papadopoulos, Y. Theodoridis, R-Trees: Theory and Applications, Advanced Information and Knowledge Processing Series, Springer, 2005.

Antonio Corral was born in Almeria, Spain in 1970. He received the B.Sc. degree (1993) from the University of Granada (Spain), and the Ph.D. degree (2002) in Computer Science (European Doctorate) from the University of Almeria (Spain). Since July 2003, he is an Assistant Professor at the Department of Languages and Computation, University of Almeria, Spain. Apart from the above, he has received several research scholarships in Spain (FIAPA and University of Almeria) and Greece (CHOROCHRONOS Project) and has published in International Journals (DKE, GeoInformatica, The Computer Journal) and conferences (SIGMOD, SSD, ADBIS, PADL). His main research interests include query processing in spatial databases and geographical information systems. Further information can be found at http://www.ual.es/~acorral/.

Yannis Manolopoulos was born in Thessaloniki, Greece in 1957. He received a 5-year Diploma Degree (1981) in Electrical Eng. and a Ph.D. (1986) in Computer Eng., both from the Aristotle Univ. of Thessaloniki. Currently, he is Professor at the Department of Informatics of the latter university. He has been with the Computer Science Departments of the Univ. of Toronto, the Univ. of Maryland at College Park and the University of Cyprus. He has published over 150 papers in refereed scientific journals and conference proceedings and received over 900 citations. He is co-author of the following books "Advanced Database Indexing" and "Advanced Signature Indexing for Multimedia and Web Applications" by Kluwer, as well as "Nearest Neighbor Search: a Database Perspective" and "R-trees: Theory and Applications" by Springer. He served/serves as PC Co-chair of the 8th National Computer Conference (2001), the 6th ADBIS Conference (2002) the 5th WDAS Workshop (2003), the 8th SSTD Symposium (2003), the 1st Balkan Conference in Informatics (2003) and the 16th SSDBM Conference (2004). Also, currently he is Vice-chairman of the Greek Computer Society. His research interests include databases, data mining, Web and Geographical Information Systems, Performance evaluation of storage subsystems. Further information can be found at http://delab.csd.auth.gr.

Yannis Theodoridis was born in 1967, received his 5-year Diploma Degree (1990) and Ph.D. (1996) in Electrical and Computer Engineering, both from the National Technical University of Athens, Greece. Since January 2002, he is Assistant Professor at the Department of Informatics, University of Piraeus, and has also a joint research position with the Computer Technology Institute (CTI). His research interests include spatial and spatiotemporal databases, location-based data management, data mining and geographical information systems. He is co-author of the book "Advanced Database Indexing" (1999, Kluwer Academic Publishers) and has published over 30 articles in scientific journals such as Algorithmica, ACM Multimedia, IEEE Transactions in Knowledge and Data Engineering, and in conferences such as ACM SIGMOD, PODS, ICDE, VLDB. His work has over 300 citations in scientific journals and conference proceedings. He has served in the program committee for several conferences (SIGMOD, ICDE, SSTD etc.) and in the editorial board of International Journal of Data Warehousing and Mining (IJDWM). He was general chair for the 8th International Symposium on Spatial and Temporal Databases (SSTD'03). He is member of ACM and IEEE.

Michael Vassilakopoulos was born in Thessaloniki, Greece in 1967. He received his 5-year Diploma Degree (1990) in Computer Eng. and Informatics from the Univ. of Patras and his Ph.D. degree (1995) from the Aristotle Univ. of Thessaloniki. He followed post-doctoral studies at the Lab of Data Engineering of the Dept. of Informatics of the Aristotle University of Thessaloniki. He has been awarded scholarships for undergraduate, graduate and postdoctoral studies from the Scholarship Foundation of the Greek State. His Diploma thesis deals with Algorithms of Computational Geometry, his doctoral thesis deals with Spatial Databases and his post-doctoral research deals with Query Processing in Spatial Databases. Apart from the above, his research interests include Spatio-temporal Databases, GIS, WWW, Information Systems Integration and Multimedia. He has been with the Dept. of Applied Informatics of the Univ. of Macedonia, the Dept. of Informatics of the Aristotle Univ. of Thessaloniki and the Dept. of Informatics of the Higher Technological Educational Institute of Thessaloniki. For three years he also served the Greek Public Administration as an Informatics Engineer where he participated in the evaluation and supervision of several information technology projects. Currently, he is an Associate Professor with the Dept. of Informatics of the Higher Technological Educational Institute of Thessaloniki. Further information can be found at http://www.it.teithe.gr/~vasilako/.