A Cost Model for K-Closest-Pair Queries

Antonio Corral[1], Yannis Theodoridis[2], and Michael Vassilakopoulos[3]

[1] Department of Languages and Computation, University of Almeria,
04120 Almeria, Spain. Email: acorral@ual.es

[2] Department of Informatics, University of Pireaus,
18534 Pireaus, Greece. Email: ytheo@unipi.gr

[3] Department of Informatics, Technological Educational Institute of Thessaloniki,
54101 Thessaloniki, Greece. Email: vasilako@it.teithe.gr

Abstract. The K Closest Pairs Query (K-CPQ) discovers the K pairs of objects formed from two different datasets that have the K smallest distances between them. This kind of distance-based query is a new database operation which appears in spatial applications. Recently, branch-and-bound algorithms based on R-trees have been developed in order to answer efficiently K-CPQs. Cost models are needed to estimate the selectivity of a specific query in order to be able to compare execution costs of alternative processing strategies during query optimizations. In this paper, we combine techniques that have been used for the analysis of nearest neighbor and join queries and derive the performance cost (in terms of disk accesses) of K-CPQs using R-trees.

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, rather than simply to store and represent them. The most basic form of such a manipulation is answering queries related to the spatial properties of data. Some typical spatial queries are the following:

- a *Point Location Query* seeks for the spatial objects that fall on a given point.
- a *Range Query* seeks for the spatial objects that are contained within a given region (usually expressed as a rectangle).
- a *Join Query* may take many forms. It involves two or more spatial data sets and discovers pairs (or tuples, in case of more than two data sets) of spatial objects that satisfy a given predicate. For example, a join query that acts on two data sets, may discover all pairs of spatial objects that intersect each other.
- Finally, a very common spatial query is the *Nearest Neighbor Query* that seeks for the spatial objects residing more closely to a given object. In its simplest form, it discovers one such object (the Nearest Neighbor). Its generalization discovers K such objects (K Nearest Neighbors), for a given K.

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). Given two different datasets S_1 and S_2 of N_{S1} and N_{S2} points, respectively, a K-CPQ retrieves the $1 \leq K \leq N_{S1}{*}N_{S2}$ 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 pair search follow branch-and-bound techniques that aim at finding quickly a good set of pairs, in order to prune the search space as early as possible. The first K-CPQ algorithms were proposed in [9, 10] for R-trees [16], although they can be modified for any data-partition index. In such an algorithm, the MINMINDIST metric is applied for pruning the search space effectively: MINMINDIST is a generalization of the minimum distance between points and MBRs. MINMINDIST 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 the K-CPQ. In addition, the distance-based plane-sweep technique is applied for reducing the response time of the K-CPQ algorithms.

A very important and interesting research direction is the query-cost modeling. Cost models are used in practice by the optimizer of a DBMS to rank and select the promising processing strategies, given a spatial query and spatial datasets. Cost models are needed to estimate the selectivity of spatial search and join operations during the 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 [26].

The analysis of query performance in spatial access methods is important for query optimization (during the operation of a DBMS) and for evaluating access method designs (by researchers working in this area). 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 thispaper, we derive formulae to estimate the number of disk accesses for K closest pairs query between two R-trees, for real datasets. These formulae depend on several input parameters, highlighting the capacity dimension (D_0), 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 [21], which assumes that *queries are more probable in high-density areas of the address space.*

The rest of the paper is organized as follows. Section 2 surveys Previous work on cost models for nearest neighbors and join queries over R-trees. In section 3, we review the R-tree family as spatial data structure and describe the branch-and-bound algorithms for K-CPQ. Section 4 presents our cost model and the formulae that estimate the K-

CPQ performance. Finally, section 5 presents the conclusion of this research and gives directions for future work.

2 Related Work

The first attempt to provide an analysis for R-tree index structures appeared in [13], where a model that estimates the performance of R-trees and R^+-trees for selection queries was proposed. Later, [20] and [23], 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 representative research efforts on analytical performance studies for nearest neighbor and spatial intersect join queries using R-trees.

2.1 Cost Models for Nearest Neighbor Queries Using R-trees

To the best of our knowledge, [22] and [21] are the most representative papers for analysis based on fractals of nearest-neighbor queries on R-trees. In [22] 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-bound 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 [21], 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 [2], 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. In [11] the cost model of [2] has been adapted to estimate the page accesses of different access methods (like the M-tree, an index structure for data spaces which are metric spaces but not vector spaces, i.e. only the distances between the objects are known, but not their explicit positions). 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 metrics was presented. The boundary effects are considered and the concept of fractal dimension is used to take into account the effects of correlated data.

2.2 Cost Models for Join Queries Using R-trees

Considering join queries, in [1], analytical formulae for cost and selectivity, based on the R-tree analysis of [20], 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 [17], 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 [20] 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 percent).

In [28], 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 percent 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 percent 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.

Recently, 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 discovered. 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 presented. This architecture (MuX) 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 Algorithms for K-CPQ Using R-trees

3.1 R-trees

An R-tree [16] 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 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 en-closes MBRs of all entries in that child node; nodes (except possibly for the root) of an R-tree of class (C_{min}, C_{max}) contain between C_{min} and C_{max} entries, where $C_{min} \leq \lceil C_{max}/2 \rceil$ (C_{max} and C_{min} are also called maximum and minimum branching factors or fan-out); the root contains at least two entries, if it is not a leaf.

Many variations of R-trees have appeared in the literature (an exhaustive survey can be found in [15]). One of the most popular and efficient variations is the R*-tree [5]. 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.

3.2 Algorithms for K-CPQ

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 [9], a generalization of the function that calculates the minimum distance between points and MBRs (MINMINDIST) was presented. MINMINDIST(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 [24] (between two points).

Definition. MINMINDIST(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), where a = (a_1, a_2, ..., a_d) and b = (b_1, b_2, ..., b_d) such that $a_i \leq b_i$, $\forall$ $1 \leq i \leq d$
M_2 = (c, d), where c = (c_1, c_2, ..., c_d) and d = (d_1, d_2, ..., d_d) such that $c_i \leq d_i$, $\forall$ $1 \leq i \leq d$
we define *MINMINDIST(M_1, M_2)* as follows:

$$MINMINDIST(M_1, M_2) = \sqrt{\sum_{i=1}^{d} y_i^2}, \text{ such that } y_i = \begin{cases} c_i - b_i, & \text{if } c_i > b_i \\ a_i - d_i, & \text{if } a_i > d_i \\ 0, & \text{otherwise} \end{cases}$$

MINMINDIST(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). An important property that relates MINMINDIST(M_1, M_2) and the minimum distance between two objects O_1 and O_2 in $E^{(d)}$ ($||(O_1, O_2)||$) and serves as the basis for the pruning heuristic is the following. Given two MBRs M_1 and M_2 in $E^{(d)}$, enclosing two set of objects $SO_1 = \{O_{1i}: 1 \leq i \leq N_1\}$ and $SO_2 = \{O_{2j}: 1 \leq j \leq N_2\}$, respectively, for all pairs of objects (O_{1i}, O_{2j}) belonging to $SO_1 \times SO_2$: MINMINDIST(M_1, M_2) $\leq$ $||(O_{1i}, O_{2j})||$.

The general pruning heuristic for K-CPQs over R-trees is the following: *if MINMINDIST(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 K-th closest pair that has been found so far.

In order to design an efficient algorithm that retrieves the $1 \leq K \leq N_{S1} * N_{S2}$ 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 [9]. Since Best-First search is I/O optimal (in absence of buffers, it only visits the necessary nodes for obtaining the query result [2]) and Depth-First search accesses more partitions than actually necessary, we are going to choose 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 (Main-heap) with the references to pairs of nodes (characterized by their MBRs) accessed so far from the two different R-trees and their minimum distance (<MINMINDIST, $Addr_{R1}$, $Addr_{R2}$ >). It visits the pair of MBRs (nodes) with the minimum MINMINDIST in the Main-heap, until it becomes empty or the MINMINDIST value of the pair of MBRs located in the root of Main-heap is larger than the distance value of the K-th 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) 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).

4 A Cost Model for K-CPQ using R-trees

The K-CPQ is a combination of spatial join and K-nearest neighbor queries. Like a spatial join query, all pairs of objects are candidates for the final result. Like a K-nearest neighbor query, proximity metrics form the basis for pruning heuristic 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.

4.1 Preliminaries

According to the analysis of R-tree joins in [21, 28] 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.

Symbol	Definition
d	number of dimensions ($1 \leq$ k $\leq$ d), i.e. embedding dimension
$D_{0,Ri}$	capacity dimension (Hausdorff fractal dimension) of points indexed in the R-tree R_i
ρ	correlation exponent of two points datasets (pair-count exponent)
S_i	point datasets that are indexed in the R-tree R_i with cardinality N_{Si}
$C_{max,Ri}$	maximum number of objects per node (maximum branching factor, M) in the R-tree R_i
$U_{avg,Ri}$	average node utilization in the R-tree R_i
f_{Ri}	effective R-tree node capacity or average R-tree node fan-out in the R-tree R_i
h_{Ri}	height of the R-tree R_i
N_{Ri}	number of points indexed in the R-tree R_i (cardinality of S_i, $N_{Si} = N_{Ri}$)
$N_{Ri,l}$	average number of R-tree nodes in the R-tree R_i at level l ($N_{Ri,root} = N_{Ri,0} = 1$)
$s_{Ri,l,k}$	average extent of node MBRs of the R-tree R_i at level l on dimension k ($1 \leq k \leq d$). In other words, it is the average side length of node MBRs of R_i at level l on dimension k
$dist_{cp}(K)$	distance of the K-th closest pair
$\sigma(K)$	K-CPQ index selectivity
$NA_{cp}(K)$	average number of pages retrieved by a K-CPQ query

Table 1. List of symbols

Under the following two assumptions:

– *Squaredness assumption.* We consider square node MBRs (hypercubes), since this is a reasonable property for "good" R-trees [20]. We make this assumption in order

to make modeling 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_{Ri}, h_{Ri}, $N_{Ri,l}$ and $s_{Ri,l,k}$ can be estimated as follows [12, 28]:

$$f_{R_i} = C_{\max,R_i} * U_{avg,R_i},\ \ h_{R_i} = 1 + \lceil \log_{f_{R_i}}(N_{R_i}/f_{R_i}) \rceil,\ \ N_{R_i,l} = N_{R_i}/\ (f_{R_i})^{h_{R_i}-l},$$

Moreover, taking into account the squaredness assumption in the biased model, we have

$$s_{R_i,l,k} = s_{R_i,l} = \left(\frac{(f_{R_i})^{h_{R_i}-l}}{N_{R_i}} \right)^{1/D_{0,R_i}},$$

$$\text{where } l = 0, 1, \ldots, h_{Ri} - 1$$

(the root is assumed at level $l = 0$ and leaves at $l = h_{Ri} - 1$).

4.2 Estimating the Number of Node Accesses

Formally, the problem of R-tree cost analysis for spatial join queries is defined as follows [28]: Let d be the dimensionality of the normalized d-dimensional unit space $[0, 1]^d$. Let us assume two spatial datasets of cardinality N_{R1} and N_{R2}, 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 (NA) in order to process a join query between the two datasets, 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_{R1} and h_{R2}, respectively. Assume that both R-tree root nodes are stored in main memory. At each level l_i, $0 \leq l_i \leq h_{Ri} - 1$, R_i contains $N_{Ri,li}$ nodes of average size $s_{Ri,li,k}$, on each dimension k ($1 \leq \mathrm{k} \leq \mathrm{d}$). 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 [28] (without loss of generality, it is assumed that $h_{R2} \leq h_{R1}$):

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

$$\text{where, } l_2 = \begin{cases} l_1 - (h_{R_1} - h_{R_2}), & h_{R_1} - h_{R_2} \leq l_1 \leq h_{R_1} - 1 \\ 0, & 0 \leq l_1 \leq h_{R_1} - h_{R_2} \end{cases}$$

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 upper $\mathrm{h}_{R1} - 1$ levels of the two R-trees:

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

where $h_{R1} - h_{R2} \leq l_1 \leq h_{R1} - 1$ and $0 \leq l_2 \leq h_{R1} - 1$. We can see that the term

$$\prod_{k=1}^{d} (s_{R_1,l_1,k} + s_{R_2,l_2,k})$$

corresponds to the *join index selectivity*, and it equals the probability that a random pair of nodes from two R-trees 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). When we use indexes, the only gain using algorithms for query processing is the *index selectivity*, i.e. not all pairs of nodes must be accessed and not all pairs of objects must be compared.

The above formula can be applied for obtaining the cost of the K-CPQ, using an appropriately defined *K-CPQ index selectivity* (σ) (it depends mainly on the characteristics of the index and on the distance parameter $dist_{cp}(K)$, the distance of the K-th closest pair of the K-CPQ).

The *K-CPQ index selectivity* (σ) can be obtained by using the concept of Minkowski sum [2]. 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, ...} and B = {b_1, b_2, ...}), 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, ...}. For the cost modeling we are only interested in the volume of the Minkowski sum, $V_{A \oplus B}$. 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 ($1 \leq k \leq d$), respectively. In this case, the volume $V_{M1 \oplus M2}$ 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 (note the likeness with the expression of *join index selectivity*):

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

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 ε), the volume of the Minkowski sum of three objects (two MBRs and a ε-sphere) was given by the following binomial formula:

$$V_{M_1 \oplus M_2 \oplus S_\varepsilon}((r_1, r_2, ..., r_d), (s_1, s_2, ..., s_d), \varepsilon) =$$

$$\sum_{k=0}^{d} \left(\binom{d}{k} * (r_k + s_k)^{d-k} * \frac{\sqrt{\pi^k}}{\Gamma(k/2+1)} \varepsilon^k \right)$$

where Γ(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 $\Gamma(x+1) \approx (x/e)^x * \sqrt{2 * \pi * x}$.

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 z-sphere (z = $dist_{cp}(K)$) is shown in the Figure 1.

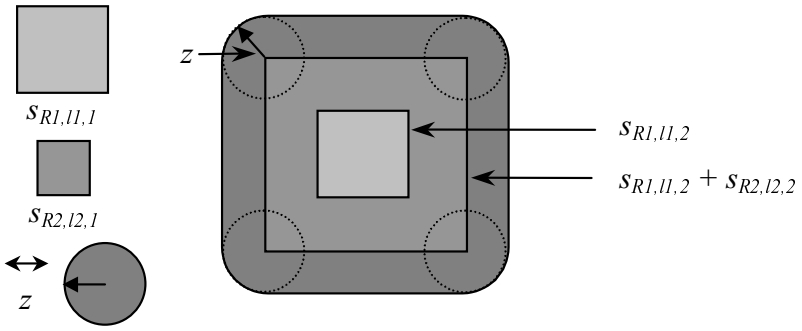

Fig. 1. The 2-dimensional Minkowski sum of two MBRs and a z-sphere (z = dist_{cp}(K)).

The volume of the Minkowski sum of two MBRs of two level l_i R-tree nodes (1 $\leq l_i \leq \mathrm{h}_{Ri} - 1$) from two R-trees R_1 and R_2 with heights $\mathrm{h}_{R2} \leq \mathrm{h}_{R1}$ 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 $N_{Ri,li}$ nodes of average side lengths $s_{Ri,li,k}$ on each dimension k, 1 $\leq$ k $\leq$ d, with minimum distance between them smaller than or equal to $dist_{cp}(K)$) :

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

$$\text{where, } l_2 = \begin{cases} l_1 - (h_{R_1} - h_{R_2}), & h_{R_1} - h_{R_2} \leq l_1 \leq h_{R_1} - 1 \\ 0, & 0 \leq l_1 \leq h_{R_1} - h_{R_2} \end{cases}$$

Although, the Minkowski sum is used in [8] for uniform and independent data only, the interested reader may study [7] (especially Section 6) to see why we can use this method for the calculation of $\sigma(K)$ for large classes of real data. Moreover, the use of the exponent ρ/d for the whole sum in the above formula is explained in [3, 4].

Finally, the overall estimation of the total 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 $h_{R2} \leq h_{R1}$, and that both root R-tree nodes are stored in main memory):

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

$$\text{where, } l_2 = \begin{cases} l_1 - (h_{R_1} - h_{R_2}), & h_{R_1} - h_{R_2} \leq l_1 \leq h_{R_1} - 1 \\ 0, & 0 \leq l_1 \leq h_{R_1} - h_{R_2} \end{cases}$$

4.3 Estimation of the Distance of the K-th 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 show a clear divergence from the uniformity and independence assumptions [12] and, hence, it is better to consider the uniformity as a special case. Our analysis uses the following result from [3]: Given a set of points S_1 with finite cardinality N_{R1} embedded in a unit hypercube of dimension d and its correlation dimension D_2, the average number of neighbors $nb(\varepsilon$, 'd-shape') of a point within a region of regular shape and radius ε is given by *nb(ε, 'd-shape') = (N_{R1} − 1) * Vol(ε, 'd-shape')*$^{D2/d}$, where Vol(ε, 'd-shape') is the volume of a shape (e.g. cube, sphere) of radius ε.

If we extend this result to two points datasets (S_1 and S_2) with finite cardinalities N_{R1} and N_{R2} (in a similar way used for spatial join selectivity $Sel_{join}(\varepsilon)$ in [3]), embedded in a unit hypercube of dimension d (embedding dimension) and consider their correlation exponent ρ, then we can obtain the average number of closest pairs within regular shape characterized by a distance r, using the following formula: *cp(r, 'd-shape') = ($N_{R1} * N_{R2}$) * Vol(r, 'd-shape')*$^{\rho/d}$, where Vol(r, 'd-shape') is the volume of a d-dimensional regular shape characterized by a distance $r \geq 0$.

We can use the previous formula to estimate the average distance of K-th closest pair under the Euclidean distance, using a similar procedure used in [21] to get the K-th nearest neighbor for K-NNQ:

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

4.4 Estimation of the Correlation Exponent (ρ)

The *pair-count law* [14] 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 a box-counting algorithm based on the concept of *box-occupancy-product-sum* (BOPS) [14]. This is an interesting extension of the algorithm proposed in [4] for computing D_q (generalized fractal dimension).

Definition. Pair-count function, PC(r) [14]
For two point datasets, S_1 and S_2, and a given distance $r \geq 0$, the *pair-count function*, PC(r), counts the number of pairs within a Euclidean distance smaller than or equal to r. PC(r) = count((a_i, b_j): dist(a_i, b_j) $\leq r$), such that $a_i \in S_1$ and $b_j \in S_2$.

Definition. Correlation function, C(r).
For two point datasets, S_1 and S_2, with finite cardinalities N_{R1} and N_{R2}, and a given

distance $r \geq 0$, the *correlation function*, C(r), represents the probability that the Euclidean distance between two different d-dimensional points from two different point datasets is less than or equal to r.

$$C(r) = \frac{Number\ of\ Pairs\ of\ Points\ Separated\ by\ less\ than\ r}{Total\ Number\ of\ Pairs\ of\ Points} = \frac{PC(r)}{N_{R_1} * N_{R_2}}$$

$$C(r) = \frac{1}{N_{R_1} * N_{R_2}} \sum_{i=1}^{N_{R_1}} \sum_{j=1}^{N_{R_2}} \Theta(r - dist(a_i, b_j))$$

where $a_i \in S_1$, $b_j \in S_2$, the closest pairs are counted through an indicator function: $\Theta(x) = 1$ for $x \geq 0$ and $\Theta(x) = 0$ for $x < 0$, and *dist* is the Euclidean distance.

Law. Power law for correlation function (follows from Law 1 of [14])
The correlation function, C(r), follows a *power law* $C(r) \propto r^{\rho}$, where ρ is the correlation exponent and $\propto$ stands for 'proportional', i.e. $C(r) = \beta * r^{\rho}$, where β is a proportionality constant.

Definition. Correlation exponent, ρ.
The exponent of the power law, called *correlation exponent* ρ, is defined as

$$\rho = \frac{\partial(\log(C(r)))}{\partial(\log(r))}$$

In order to obtain C(r), we can use the box-counting algorithm based on the concept of *box-occupancy-product-sum* (BOPS) and $C(r) \propto BOPS(r)$ [14]. In order to calculate the correlation exponent ρ, 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); hence $\rho = \delta(\log(BOPS(r))\ /\ \delta(\log(r))$. Intuitively, ρ shows how the average number of closest pairs from two different pints datasets grows, as the distance r increases.

Definition. Box-Occupancy-Produce-Sum, BOPS(r).
The BOPS of a grid with cell size r is defined as the sum of products of occupancies as

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

where, $N(r)$ is the number of grid cells with side r that are occupied by the two point datasets; $p_{i,S1}$ and $p_{i,S2}$ are the percentage of points of the datasets S_1 and S_2, respectively, which fall inside the i-th cell with side r.

A consequence of this definition is that $BOPS(r) \propto r^{\rho}$ [14], in a similar sense that the sum of squared occupancies, $S_2(r) = \Sigma p_i^2$ is proportional to the correlation fractal dimension D_2 ($S_2(r) \propto r^{D2}$) [4].

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, d and D_0 using formula (1).

5 Conclusions and Ideas for Future Extensions

In this paper, we presented an I/O cost model for the *K Closest Pairs Query* (K-CPQ) using R-trees. The development of our model was based on previous analytical results for nearest neighbor and spatial intersect join queries. We did not follow the *unrealistic* assumptions of uniformity and independence. On the contrary, the formulae we developed correspond to real data and depend mainly on the capacity dimension (D_0), 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 cost model we developed can be used for the estimation of selectivity of a specific query which is needed during query optimizations (during the comparison of execution costs of alternative processing strategies that is performed by the optimizer of a DBMS) and/or for the evaluation of access method designs by the research community.

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 [21].
- Study of the effect of buffering, extending the previous analysis if we use a *path-buffer* for each underlying R-tree [28] and a *global LRU buffer* to both R-trees [17].
- A theoretic proof of the fact that the correlation exponent (ρ) has several interesting properties [14]: it is invariant to the used L_t-distance and it includes the *correlation fractal dimension*, D_2, as a special case.
- Generalization of our study for non-points spatial objects.

References

1. W.G. Aref and H. Samet; "A Cost Model for Query Optimization using R-trees", *Proceedings ACM-GIS Symposium*, (1994)
2. S. Berchtold, C. Böhm, D.A. Keim and H.P. Kriegel; "A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space", *Proceedings ACM PODS Symposium*, (1997), 78-86
3. A. Belussi and C. Faloutsos; "Estimating the Selectivity of Spatial Queries using the 'Correlation' Fractal Dimension", *Proceedings VLDB Conference*, (1995) 299-310
4. A. Belussi and C. Faloutsos: "Self-Spatial Join Selectivity Estimation Using Fractal Concepts", *ACM Transactions on Information Systems*, 16(2):161-201, (1998)
5. N. Beckmann, H.P. Kriegel, R. Schneider and B. Seeger; "The R*-tree: an Efficient and Robust Access Method for Points and Rectangles", *Proceedings ACM SIGMOD Conference*, (1990) 322-331
6. T. Brinkhoff, H.P. Kriegel and B. Seeger; "Efficient Processing of Spatial Joins Using R-trees", *Proceedings ACM SIGMOD Conference*, (1993) 237-246
7. C. Böhm; "A Cost Model for Query Processing in High-Dimensional Data Spaces", *ACM Transactions on Database Systems*, 25(2):129-178 (2000)
8. C. Bohm and H.P. Kriegel; "A Cost Model and Index Architecture for the Similarity Join", *Proceedings IEEE ICDE Conference*, (2001) 411-420
9. A. Corral, Y. Manolopoulos, Y. Theodoridis and M. Vassilakopoulos; "Closest Pair Queries in Spatial Databases", *Proceedings ACM SIGMOD Conference*, (2000) 189-200

10. A. Corral, Y. Manolopoulos, Y. Theodoridis and M. Vassilakopoulos; "Algorithms for Processing K Closest Pair Queries in Spatial Databases", Tech Report, Department of Informatics, Aristotle University of Thessaloniki, (2001)
11. P. Ciaccia, M. Patella and P. Zezula; "A Cost Model for Similarity Queries in Metric Spaces", *Proceedings ACM PODS Symposium*, (1998) 59-68
12. C. Faloutsos and I. Kamel; "Beyond Uniformity and Independence: Analysis of R-trees using the Concept of Fractal Dimension", *Proceedings ACM PODS Symposium*, (1994) 4-13
13. C. Faloutsos, T. Sellis, and N. Roussopoulos; "Analysis of Object Oriented Spatial Access Methods", *Proceedings ACM SIGMOD Conference*, (1987) 426-439
14. C. Faloutsos, B. Seeger, A. Traina and C. Traina; "Spatial Join Selectivity using Power Law", *Proceedings ACM SIGMOD Conference*, (2000) 177-188, 2000
15. V. Gaede and O. Günther; "Multidimensional Access Methods", *ACM Computing Surveys*, 30(2):170-231, (1998)
16. A. Guttman; *"R-trees: a Dynamic Index Structure for Spatial Searching"*, *Proceedings ACM SIGMOD Conference*, (1984) 47-57
17. Y.W. Huang, N. Jing and E.A. Rundensteiner; "A Cost Model for Estimating the Performance of Spatial Joins Using R-trees", *Proceedings SSDBM Conference*, (1997) 30-38
18. G.R. Hjaltason and H. Samet; "Incremental Distance Join Algorithms for Spatial Databases", *Proceedings ACM SIGMOD Conference*, (1998) 237-248, 1998
19. G.R. Hjaltason and H. Samet; "Distance Browsing in Spatial Databases", *ACM Transactions on Database Systems*, 24(2):265-318, (1999)
20. I. Kamel and C. Faloutsos, "On Packing R-trees", *Proceedings CIKM Conference*, (1993) 490-499
21. F. Korn, B.U. Pagel and C. Faloutsos; "On the 'Dimensionality Curse' and the 'Self-Similarity Blessing"', *IEEE Transactions on Knowledge and Data Engineering*, 13(1):96-111, (2001)
22. A. Papadopoulos and Y. Manolopoulos; "Performance of Nearest Neighbor Queries in R-Trees", *Proceedings ICDT Conference*, (1997) 394-408
23. B.U. Pagel, H.W. Six, H. Toben, and P. Widmayer; "Towards an Analysis of Range Query Performance", *Proceedings ACM PODS Symposium*, (1993) 214-221
24. N. Roussopoulos, S. Kelley and F. Vincent; "Nearest Neighbor Queries", *Proceedings ACM SIGMOD Conference*, (1995) 71-79
25. L.A. Smith; "Intrinsic Limits on Dimension Calculations", *Physics Letters A*, 133:283-288, (1988)
26. S. Shekhar, S. Chawla, S. Rivada, A. Fetterer, X. Lui and C. Lu; "Spatial Databases, Accomplishments and Research Needs", *IEEE Transactions on Knowledge and Data Engineering*, 11(1):45-55, (1999)
27. H. Shin, B. Moon and S. Lee; "Adaptive Multi-Stage Distance Join Processing", *Proceedings ACM SIGMOD Conference*, (2000) 343-354
28. Y. Theodoridis, E. Stefanakis and T. Sellis; "Efficient Cost Models for Spatial Queries Using R-Trees", *IEEE Transactions on Knowledge and Data Engineering*, 12(1):19-32, (2000)