Performance of Nearest Neighbor Queries in R-trees *

Apostolos Papadopoulos Yannis Manolopoulos

Department of Informatics
Aristotle University
54006 Thessaloniki, Greece
email : {*apapadop,manolopo*}*@athena.auth.gr*

Abstract. Nearest neighbor (NN) queries are posed very frequently in spatial applications. Recently a branch-and-bound algorithm based on R-trees has been developed in order to answer efficiently NN queries. In this paper, we combine techniques that were inherently used for the analysis of range and spatial join queries, in order to derive measures regarding the performance of NN queries. We try to estimate the number of disk accesses introduced due to the processing of an NN query. Lower and upper bounds are defined estimating the performance of NN queries very closely. The theoretical analysis is verified with experimental results, under uniform and non-uniform distributions of queries and data, in the 2-dimensional address space.

1 Introduction

Spatial data management is an active area of research over the past ten years [Same90a, Same90b, Laur92, Guti94]. Research interests focused mainly on the design of robust and efficient spatial data structures [Gutt84, Henr89, Guen89, Beck90, Kame94], the invention of new spatial data models [Laur92], the construction of effective query languages [Egen94] and the query processing and optimization of spatial queries [Oren86, Aref93, Papa95].

A very important research direction is the estimation of the performance and the selectivity of a query. Given a query, the problem is to estimate the response time (performance) and the fraction of the objects that fulfils the query versus the database population (selectivity). Evidently, we want this information available prior to query processing, in order for the query optimizer to determine an efficient access plan.

Nearest Neighbor (NN) queries are very important in Geographic Information Systems [Same90b, Rous95], in Image Databases [Arya93, Nibl93] as weel as in Multimedia Applications [Fagi96]. However, researchers working on spatial accesses methods focused mainly on range queries [Page93, Kame93, Fal94, Theo96] and spatial join queries [Brin93, LoRa94, Belu95]. In the past the problem of NN query processing has been addressed by examining access methods

* Work supported by European Union's TMR program and by national PENED and EPET programs.

based on k-d trees [Frie77] and quadtrees [Same90b]. Only recently a branch-and-bound algorithm based on R-trees has been developed, in order to answer efficiently NN queries [Rous95]. In this paper we combine techniques that were inherently used for the analysis of range and spatial join queries, in order to derive effective measures regarding the performance of NN queries. We give average lower and upper bounds for the number of leaf pages retrieved during NN query processing. Evidently, CPU time is also important for computationally intensive queries, but in general the I/O subsystem overhead dominates, specifically in large spatial databases.

The rest of the article is organized as follows. In the next section we present the appropriate background on the R-tree family of spatial data structures and describe shortly the branch-and-bound algorithm of [Rous95]. Section 3 contains the derivation of the formulae for the upper and lower bounds towards the prediction of NN query performance. In Section 4 we give the experimental results, and finally in Section 5 we conclude the paper, and motivate for future research in the area.

2 Background

2.1 R-trees

The R-tree [Gutt84] is a hierarchical, height balanced data structure (all leaf nodes appear at the same level), designed for use in secondary storage, and it is a generalization of the B^+-tree for multidimensional spaces. A sample 2-d dataspace with a corresponding R-tree is presented in Figure 1 below. The

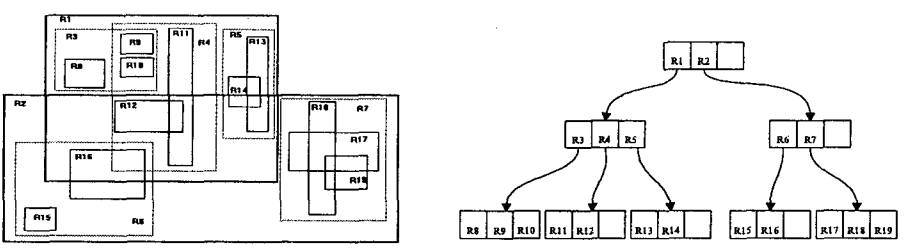


Fig. 1. R-tree example.

structure handles objects by means of their conservative approximation. The most simple and frequently used conservative approximation of an object's shape is the rectilinear Minimum Bounding Rectangle (MBR). Each node of the tree corresponds to exactly one disk page. Internal nodes contain entries of the form *(R,child-ptr)*, where R is the MBR that encloses all the MBRs of its descendants and *child-ptr* is the pointer to the specific child node. Leaf nodes contain entries of the form *(R,object-ptr)* where R is the MBR of the object and *object-ptr* is the pointer to the objects detailed description. Since MBRs of internal nodes are

allowed to overlap, we may have to follow multiple paths from root to leaves when answering a range query. This inefficiency triggered the design of the R^+-tree [Sell87] which does not permit overlapping MBRs of the nodes.

One of the most important factors affecting the overall structure performance is the node split strategy used. In [Gutt84] three split policies have been reported, namely exponential, quadratic and linear. However, more sophisticated policies reducing the overlap of MBRs have been reported in [Beck90] (the R^*-tree) and in [Kame94] (the Hilbert R-tree).

Finally, some R-tree variants have been reported to support a static or a nearly static database. If the objects composing the dataspace are known in advance, we can apply several packing techniques, with respect to the spatial proximity of the objects, in order to design a more efficient data structure, with increased initial overhead. Packing techniques have been reported in [Rous85, Kame93].

In this paper, we base our work on the packed R-tree of Kamel and Faloutsos [Kame93]. In this variant, the Hilbert value of each data object is calculated and then the whole dataset is sorted. Next, the leaf level of the tree is formulated by taking consecutive objects (with respect to the Hilbert order) and storing them in one data page. The same process is repeated for the upper levels of the structure. The derived R-tree has little overlap and square-like MBRs, both being reasonable properties of a "good" R-tree [Kame93, Fal94, Theo96].

2.2 The Branch-and-Bound Algorithm

In [Rous95] an efficient branch-and-bound R-tree traversal algorithm is reported, that answers NN and k-NN queries. It is a modification of the algorithm reported in [Frie77] for k-d-trees. In order to find the nearest neighbor of a point, the algorithm starts form the root of the R-tree and proceeds downwards. The key idea of the algorithm is that many branches of the tree can be discarded according to some rules. Two basic distances are defined in $n-$d space, between a point P with co-ordinates $(p_1, p_2, ..., p_n)$ and a rectangle R with corners $(s_1, s_2, ..., s_n)$ and $(t_1, t_2, ..., t_n)$ (bottom-left and top-right respectively). Two definitions follow [Rous95]:

Definition 1
The distance $MINDIST(P, R)$ *of a point* P *from a rectangle* R, *is defined as follows:*

$$MINDIST(P, R) = \sqrt{\sum_{j=1}^{n} |p_j - r_j|^2}$$

where:

$$r_j = \begin{cases} s_j, & p_j < s_j \\ t_j, & p_j > t_j \\ p_j, & \text{otherwise} \end{cases}$$

□

Definition 2
The distance $MINMAXDIST(P, R)$ *of a point* P *from a rectangle* R, *is defined as follows:*

$$MINMAXDIST(P,R) = \sqrt{\min_{1 \le k \le n} (|p_k - rm_k|^2 + \sum_{1 \le j \le n, j \neq k} |p_j - rM_j|^2)}$$

where:

$$rm_k = \begin{cases} s_k, & p_k \le \frac{s_k + t_k}{2} \\ t_k, & \text{otherwise} \end{cases}$$

$$rM_j = \begin{cases} s_j, & p_j \ge \frac{s_j + t_j}{2} \\ t_j, & \text{otherwise} \end{cases}$$

□

Clearly the $MINDIST$ is the optimistic metric, since it is the minimum possible distance that the NN of the query point P can reside in the corresponding data page. On the other hand, $MINMAXDIST$ is the pessimistic metric since it is the furthest possible distance where the NN of P can reside in the current data page. Therefore, the latter metric guarantees that the NN of P lies in a distance $\le MINMAXDIST$. The above definitions are shown graphically in Figure 2.

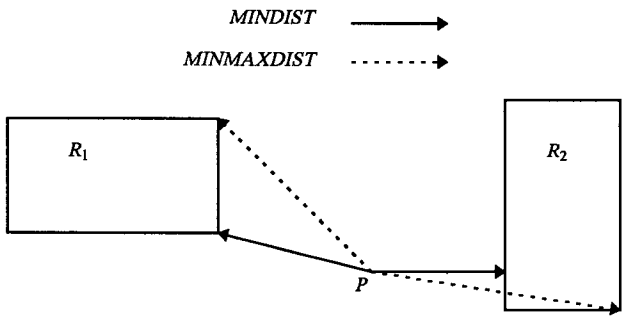


Fig. 2. *MINDIST* and *MINMAXDIST* between a point P and two rectangles R_1 and R_2.

The three basic rules used for pruning the search in the R-tree during traversal follow. Notice that these rules are applied only if one nearest neighbor is required.

Rule 1
If *an MBR* R *has* $MINDIST(P, R)$ *greater than the* $MINMAXDIST(P, R')$ *of another MBR* R', **then** *it is discarded because it cannot enclose the nearest neighbor of* P. □

Rule 2
If *an actual distance d from P to a given object, is greater than the $MINMAXDIST(P,R)$ of P to an MBR R,* **then** *d is replaced with $MINMAXDIST(P,R)$ because R contains an object which is closer to P.* □

Rule 3
If *d is the current minimum distance,* **then** *all MBRs R_j with $MINDIST(P,R_j) > d$ are discarded, because they cannot enclose the nearest neighbor of P.* □

Upon visiting an internal node of the tree, Rules (1) and (2) are used in order to discard irrelevant branches. Then, a branch is selected according to a priority order. Roussopoulos et al. suggest that when the overlap is small, the $MINDIST$ order should be used since it would discard more candidates. This is also verified in the experimental results of their work. Therefore, the branch which correspond to the minimum $MINDIST$ among all node entries is chosen. Upon returning from the processing of the subtree, Rule (3) is applied in order to discard other candidates (if there are any). Due to limited space, we are not going into more details of the branch-and-bound algorithm. The reader is prompted to reference [Rous95].

3 Analytical Considerations

3.1 Preliminaries

Symbol	Description
$\mathcal{S}$	a set of 2-d points
N	population of the indexed dataset
n	space dimensionality
σ	side of the square-like data page MBR
D_0	*Hausdorff* fractal dimension
D_2	*correlation* fractal dimension
C_{max}	maximum number of objects per node
C_{avg}	average number of objects per node
U_{avg}	average space utilization
d_{nn}	distance between a query point and its NN point
d_m	distance from a query point to the $MINMAXDIST$ vertex of the first retrieved data page
q	query window side
$L(q)$	number of leaf accesses for a window query of side q
L_{bound}	lower bound for the number of leaf accesses
U_{bound}	upper bound for the number of leaf accesses

Table 1. Basic notations used throughout the analysis.

In this section, we derive lower and upper bounds for the performance of the branch-and-bound algorithm. We are interested in the estimation of the number of disk accesses to R-tree leaf pages, because in general the upper levels occupy small space in comparison to the leaf level, and therefore can fit in main memory. The basic notations are presented in Table 1.

Assume the dataspace is composed of a set of points $\mathcal{S}$ in the 2-d space. The problem is, given a point $P(p_1, p_2) \in \mathcal{S}$, to find its NN point $Q(q_1, q_2)$. Let d_{nn} be the actual Euclidean distance between the points P and Q. The following propositions hold:

Proposition 1
The minimum number of leaf pages touched is the number of leaf pages intersected by the circle C_1 with center P and radius d_{nn}.

Proof
The distance d_{nn} is not known in advance. Therefore, even if the nearest neighbor of the query point is found, the algorithm does not stop until all candidates are examined. As a consequence, all data pages X_i with $MINDIST(P, X_i) \leq d_{nn}$ must be searched. □

Before stating Proposition 2, we introduce the following basic assumption which is a reasonable property of the algorithm, when the tree nodes have no or very little empty space:

Basic Assumption
The first data page that the algorithm visits, is the data page with the minimum $MINDIST$ among all data pages. □

Proposition 2
The maximum number of leaf pages touched is the number of leaf pages that the circle C_2 with center P and radius d_m intersects, where d_m is the $MINMAXDIST$ between P and the first touched leaf page.

Proof
Let R denote the first visited data page MBR. Clearly, the distance $MINMAXDIST(P, R)$ is the maximum possible "safe" distance where a nearest neighbor can be found in this data page. Moreover, it is possible that all data pages X_i with $MINDIST(P, X_i) \leq MINMAXDIST(P, R)$ will be visited, if a particular visiting sequence occurs. □

In Figure 3a an example is illustrated for Proposition 1. The arrow points to the NN of the query point P. Even if the algorithm reaches this point, it is not known that this is the NN of P, until data pages 1 and 2 are examined. In Figure 3b Proposition 2 is explained. Page 1 is the first visited data page. In the worst case the NN of P, in this page, resides in $MINMAXDIST(P, 1)$ from P.

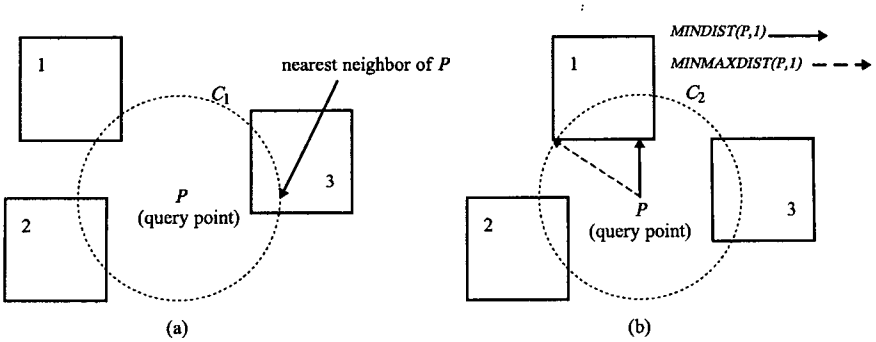


Fig. 3. (a): example of Proposition 1, (b): example of Proposition 2

However, it is not guaranteed that pages 2 and 3 will be visited. This will occur in the worst case only, and depends on the visiting sequence and the location of the "temporary" NN point in each data page.

The above propositions give a lower bound (Proposition 1) and an upper bound (Proposition 2) for the number of leaf nodes touched by the algorithm, on the average. We note the importance of the distance d_{nn}, which is the expected distance from P to its nearest neighbor. Therefore, if we had an estimation for d_{nn}, we could provide estimations for the best and worst performance of NN queries. The following subsection deals with the estimation of d_{nn} and d_m.

3.2 Estimation of d_{nn} and d_m

We are interested in the estimation of d_{nn} for arbitrary object distributions. Real datasets show a clear divergence from the uniformity and independence assumption [Fal94] and, therefore, it is better to consider uniformity as a special case. In [Belu95] a formula has been reported that estimates the average number of neighbors $nb(\epsilon, shape)$ of a point P within distance ϵ from P, using the concept of the *correlation* fractal dimension of the point set:

$$nb(\epsilon, shape) = \left(\frac{volume(\epsilon, shape)}{volume(\epsilon, rect)}\right)^{D_2/n} \cdot (N-1) \cdot 2^{D_2} \cdot \epsilon^{D_2} \qquad (1)$$

where N is the population of the dataset, D_2 is the correlation fractal dimension, n is the dimensionality of the dataspace (2 in our case), and *shape* is the shape that has its center of gravity on a point P of the dataset. Since we are interested in NN queries with respect to the Euclidean distance, it is sufficient to set $shape = circle$. Making the appropriate modifications in Equation (1) we get:

$$nb(\epsilon, circle) = \left(\frac{\pi\epsilon^2}{4\epsilon^2}\right)^{D_2/2} \cdot (N-1) \cdot 2^{D_2} \cdot \epsilon^{D_2}$$

By simplifying we get:

$$nb(\epsilon, circle) = (\sqrt{\pi})^{D_2} \cdot (N-1) \cdot \epsilon^{D_2} \qquad (2)$$

We can use Equation (2) to estimate the average distance (d_{nn}) of a point P to its nearest neighbor. We are searching for an ϵ such that $nb(\epsilon, circle) = 1$. After substitution in Equation (2) and algebraic manipulations we reach:

$$d_{nn} = \epsilon = \frac{1}{\sqrt{\pi} \cdot \sqrt[D_2]{(N-1)}} \tag{3}$$

The above equation holds for an arbitrary dataset, when we allow queries to land only on data points. The uniformity case is derived by setting $D_2 = 2$.

Let us now try to estimate the distance d_m, which is the minimum $MINMAXDIST$ between the query point P and the first visited data page. We assume that the MBRs of the data pages are squares with side σ. The following proposition holds:

Proposition 3
The maximum possible difference between $MINMAXDIST(P, R)$ and $MINDIST(P, R)$ from a query point P to an MBR R is σ.

Proof
This happens when the query point P coincides with a vertex of the MBR R. This is demonstrated in Figure 4. As we can see, when the query point P approaches the bottom-right vertex of the MBR, the difference between $MINMAXDIST$ and $MINDIST$ increases. □

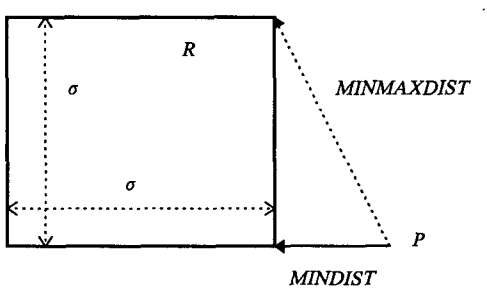


Fig. 4. When the query point P coincides with a vertex of the MBR, then the maximum difference (σ) between $MINDIST$ and $MINMAXDIST$ is obtained.

Assuming that the NN of a query point lies in the half distance (on the average) between the difference of $MINDIST$ and $MINMAXDIST$, we need only to augment d_{nn} by $\sigma/2$ in order to reach the $MINMAXDIST$. Therefore, we conclude that the distance d_m which gives the upper bound of Proposition 2 is calculated by the following Equation:

$$d_m = \frac{1}{\sqrt{\pi} \cdot \sqrt[D]{(N-1)}} + \frac{\sigma}{2} \tag{4}$$

3.3 Performance Estimation

Let $\mathcal{S}$ be a set of N data points distributed in the unit square address space. We are interested in estimating the number of data pages retrieved, when the NN is requested for any point $P \in \mathcal{S}$. Given a query window $q \times q$ the number of leaf nodes $L(q)$ retrieved is given by a formula reported in [Fal94], which assumes that queries are distributed uniformly on the address space i.e. each portion of the dataspace has the same probability to be requested:

$$L(q) = \frac{N}{C_{avg}} \cdot (\sigma + q)^2 \tag{5}$$

$$\sigma = \left(\frac{C_{avg}}{N}\right)^{1/D_0}$$

$$C_{avg} = C_{max} \cdot U_{avg}$$

where N is the population of the dataset, D_0 is the *Hausdorff* (box counting) fractal dimension of the underlying point dataset, C_{max} is the maximum node capacity and U_{avg} is the average space utilization of the R-tree nodes.

However, in our case we cannot use Equation (5). This is because, the queries can land only on (existing) data points and therefore at least one leaf access will occur. In other words, in our case the query model assumes that the query distribution follows the data distribution (i.e. each data object has the same probability of retrieval [Page93]). To the best of the authors' knowledge no closed formula has been reported to estimate the number of leaf accesses in this query model. Therefore, we must derive a formula that obeys the latter query model.

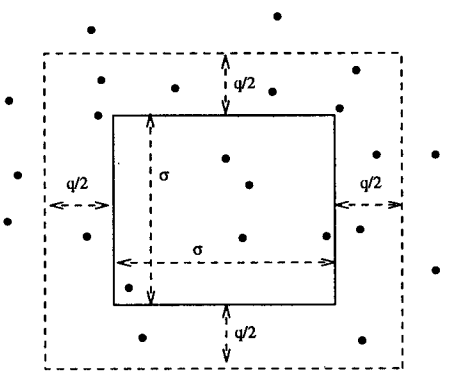


Fig. 5. Example of an enlarged data page.

Assume we have a $q \times q$ window and we have to perform a range query Q over the underlying address space. We know that the average size of each data page MBR is $\sigma \times \sigma$. We are interested in calculating the probability P_{fetch} that a data page will be fetched due to the execution of Q. A data page will be fetched only if the centroid of the window $q \times q$ falls in the area surrounded by the dashed line of Figure 5. Note however, that the centroid of the query window can only coincide with an existing data point (according to the query model considered in this paper). Therefore, the probability P_{fetch} can be defined as:

$$P_{fetch} = \frac{GoodPoints}{AllPoints} \tag{6}$$

where $GoodPoints$ is the number of points enclosed by the enlarged $(\sigma + q) \times (\sigma + q)$ window, and $AllPoints$ is the population, N, of the indexed dataset. However, we have the appropriate mathematical tools to calculate $GoodPoints$. We can use Equation (1) setting[1] $shape = rect$ and $\epsilon = \frac{\sigma+q}{2}$. Therefore, we have:

$$GoodPoints = (N-1) \cdot (\sigma + q)^{D_2} \tag{7}$$

From Equations (6) and (7) we get:

$$P_{fetch} = \frac{N-1}{N} \cdot (\sigma + q)^{D_2} \tag{8}$$

Our next step is to calculate the average number of data page accesses. We know that the total number of data pages is $\frac{N}{C_{avg}}$. Therefore:

$$L(q) = \frac{N}{C_{avg}} \cdot P_{fetch} \Rightarrow L(q) = \frac{N-1}{C_{avg}} \cdot (\sigma + q)^{D_2} \tag{9}$$

In order to get the lower and upper bounds for the number of leaf accesses, we must substitute q in Equation (9), with $2 \cdot d_{nn}$ from Equation (3), and $2 \cdot d_m$ from Equation (4), respectively. Therefore, we have:

$$L_{bound} = \frac{N-1}{C_{avg}} \cdot (\sigma + 2 \cdot d_{nn})^{D_2} \tag{10}$$

$$U_{bound} = \frac{N-1}{C_{avg}} \cdot (\sigma + 2 \cdot d_m)^{D_2} \tag{11}$$

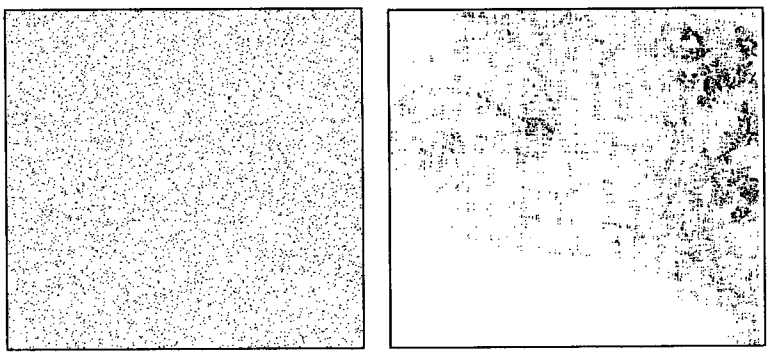
Equations (10) and (11) include uniformity as a special case. Clearly, for uniform point sets $D_0 \approx 2$ and $D_2 \approx 2$, so we can use the above equations for any kind of point set. Also, we note that L_{bound} and U_{bound} are bounds on the average case and not absolute ones. This means that during NN query processing, the lower bound may be higher than the leaf pages touched. However, we are interested on the average case, and exceptional cases do not harm the generality.

[1] This requires an optimistic assumption that we can always find a data point on the centroid of the data page MBR.

4 Experimental Results

4.1 Preliminaries

We implemented the branch-and-bound algorithm [Rous95] and the Hilbert-packed R-tree [Kame93] in the C programming language under UNIX, and ran the experiments on a DEC Alpha 3000 workstation. We used randomly (almost uniform) generated as well as real-life points, in order to verify the theoretical aspects. The datasets used are depicted in Figure 6. The real-life points are 9,552 road intersections of the Montgomery County, Maryland. For uniform point sets we have $D_0 \approx 2$ and $D_2 \approx 2$, whereas for the MG points $D_0 \approx 1.719$ and $D_2 \approx 1.518$ [Belu95].



Random points MG points

Fig. 6. Datasets used in the experiments.

4.2 Experimentation

In all experimental series, for each dataset, the average number of leaf accesses was determined by issuing an NN query for each existing data point. Also, the lower and upper bounds for the average number of leaf accesses were calculated. The measured average number of leaf accesses is shown in the last column of each subsequent table.

Experiment 1
The dataset is composed of a number of uniformly distributed points. The maximum R-tree node capacity was set to 50 objects. In Table 2 we present the results for uniform data of various populations.

Experiment 2
The dataset is composed of uniformly distributed points. Here we keep the population of the dataset constant at 50,000 and vary the maximum fanout of the tree from 10 to 200. The results are shown in Table 3.

Experiment 3
The dataset is composed of the $\approx$ 9,000 MG points. Again, we vary the fanout of the tree from 10 to 200 as in Experiment 2. The results are presented in Table 4.

Population	Lower	Upper	Measured
1,000	1.34	4.66	1.63
2,000	1.34	4.66	1.58
10,000	1.34	4.66	1.70
20,000	1.34	4.66	1.80
50,000	1.34	4.66	2.04
100,000	1.34	4.66	1.88
200,000	1.34	4.66	2.28
500,000	1.34	4.66	1.97

Table 2. Leaf accesses versus data population. Data=Uniform, Fanout=50.

Fanout	Lower	Upper	Measured
5	2.26	6.27	3.02
10	1.84	5.55	2.68
20	1.56	5.07	2.19
50	1.34	4.66	2.03
100	1.23	4.46	1.90
200	1.16	4.32	1.82

Table 3. Leaf accesses versus fanout. Data=Uniform, Population=50,000.

Fanout	Lower	Upper	Measured
5	3.22	7.99	4.13
10	2.70	7.01	3.06
20	2.33	6.24	2.36
50	1.98	5.44	2.27
100	1.77	4.94	1.89
200	1.61	4.52	1.81

Table 4. Leaf accesses versus fanout. Data=MG points, Population $\approx$ 9,000.

4.3 Result Interpretation

From these tables it is evident that the lower and upper bounds enclose very well the measured average number of leaf accesses. Therefore, one could use the simple Formulae (10) and (11) in order to estimate the performance of an NN query. We observe that the measured number of leaf accesses is generally closer to the lower bound than the upper bound. This gives us a strong indication that the branch-and-bound algorithm with the $MINDIST$ criterion exploits the "goodness" property of the packed R-tree very effectively. Lower bound gives an optimistic metric and upper bound a pessimistic metric and are both valuable in query processing and optimization. Another observation is that when the data (and hence the query) distribution is uniform, the bounds do not depend on the

population of the dataset. This can be verified by substituting the appropriate values for σ, d_{nn} and d_m in Equations (10) and (11), and is illustrated in Table 2.

5 Conclusions and Future Work

In this paper we have focused on the performance estimation of NN queries in spatial data structures and particularly in R-trees. The only known algorithm for NN queries in R-trees is the branch-and-bound algorithm of Roussopoulos et al. [Rous95], to the best of the authors' knowledge. We have shown that the actual distance between a point and its NN plays a very important role for the performance estimation of NN queries. Experiments based on synthetic and real-life data have shown that the derived bounds enclose very closely the number of leaf accesses introduced during the processing of an NN query. In fact, the performance of the branch-and-bound algorithm is closer to the lower bound, and therefore is very efficient. This estimation could be exploited by a query optimizer, to derive an efficient query processing plan. However, more work is needed, since this field is yet unexplored. Future work in the area may include:

- modification of the Formulae (10) and (11), in order to estimate the performance of arbitrary k-NN queries,
- derivation of a formula for the exact performance prediction of NN query processing (not just lower and upper bounds),
- the relaxation of the basic assumption (Section 3),
- generalization for non-point objects,
- consideration of complex queries with several constraints (e.g. find the NN of the point P, such that the distance is $\geq d$).
- consideration of the case where we request the NN for a point P that does not belong to the data set.
- examination of the case where the R-tree is not that "good" as the packed R-tree (e.g. Guttman's R-tree).

We are currently working on the performance estimation of general k-NN queries ($k > 1$) in high dimensional address spaces ($d > 2$). Also, we consider more real datasets in order to justify the usefulness of the analytical results in the prediction of NN query performance.

Acknowledgements

The authors would like to thank Alberto Belussi (at Politecnico di Milano) for providing the MG point dataset and also, the anonymous referees for their helpful comments and suggestions regarding this article.

References

[Aref93] W. Aref: "Query Processing and Optimization in Spatial Databases", *Technical Report CS-TR-3097, Department of Computer Science*, University of Maryland at College Park, MD, 1993.

[Arya93] M. Arya, W. Cody, C. Faloutsos, J. Richardson and A. Toga: "QBISM: a Prototype 3-d Medical Image Database System", *IEEE Data Engineering Bulletin*, 16(1), pp.38-42, March 1993.

[Beck90] N. Beckmann, H.P. Kriegel and B. Seeger: "The R*-tree: an Efficient and Robust Method for Points and Rectangles", *Proceedings of the 1990 ACM SIGMOD Conference*, pp.322-331, Atlantic City, NJ, 1990.

[Belu95] A. Belussi and C. Faloutsos: "Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension", *Proceedings of the 21th VLDB Conference*, pp.299-310, Zurich, Switzerland, 1995.

[Brin93] T. Brinkhoff, H.P. Kriegel and B. Seeger: "Efficient Processing of Spatial Join Using R-trees", *Proceedings of the 1990 ACM SIGMOD Conference*, pp.237-246, Washington DC, 1993.

[Egen94] M. Egenhofer: "Spatial SQL: a Query and Presentation Language", *IEEE Transactions on Knowledge and Data Engineering*, vol.6, no.1, pp.86-95, 1994.

[Fagi96] R. Fagin: "Combining Fuzzy Information from Multiple Systems", *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '96)*, pp.216-226, Montreal, Canada, 1996.

[Fal94] C. Faloutsos and I. Kamel: "Beyond Uniformity and Independence, Analysis of R-trees Using the Concept of Fractal Dimension", *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '94)*, pp.4-13, Minneapolis, MN, 1994.

[Frie77] J.H. Friedman, J.L. Bentley and R.A. Finkel: "An Algorithm for Finding the Best Matches in Logarithmic Expected Time", *ACM Transactions on Math. Software*, vol.3, pp.209-226, 1977.

[Guen89] O. Guenther: "The Design of the Cell Tree: an Object-Oriented Index Structure for Geometric Databases", *Proceedings of the 5th IEEE Conference on Data Engineering*, pp.598-615, Los Angeles, CA, 1989.

[Guti94] R.H. Guting: "An Introduction to Spatial Database Systems", *The VLDB Journal*, vol.3, no.4, pp.357-399, 1994.

[Gutt84] A. Guttman: "R-trees: a Dynamic Index Structure for Spatial Searching", *Proceedings of the 1984 ACM SIGMOD Conference*, pp.47-57, Boston, MA, 1984.

[Henr89] A. Henrich, H.W. Six and P. Widmayer: "The LSD-tree: Spatial Access to Multidimensional Point and non-Point Objects", *Proceedings of the 15th VLDB Conference*, pp.45-53, Amsterdam, Netherlands, 1989.

[Kame93] I. Kamel and C. Faloutsos: "On Packing R-trees", *Proceedings of the 2nd Conference on Information and Knowledge Management (CIKM)*, Washington DC, 1993.

[Kame94] I. Kamel and C. Faloutsos: "Hilbert R-tree: an Improved R-tree Using Fractals", *Proceedings of the 20th VLDB Conference*, pp.500-509, Santiago, Chile, 1994.

[Laur92] R. Laurini and D. Thompson: *"Fundamentals of Spatial Information Systems"*, Academic Press, London, 1992.

[LoRa94] M.L. Lo and C.V. Ravishankar: "Spatial Joins Using Seeded Trees", *Proceedings of the 1994 ACM SIGMOD Conference*, pp.209-220, Minneapolis, MN, 1994.

[Nibl93] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic and P. Yanker: "The QBIC Project: Querying Images by Content Using Color, Texture and Shape", *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases*, vol.1908, pp.173-187, 1993.

[Oren86] J. Orenstein: "Spatial Query Processing in an Object-Oriented Database System", *Proceedings of the 1986 ACM SIGMOD Conference*, pp.326-336, Washington DC, 1986.

[Page93] B.U. Pagel, H.W. Six, H. Toben and P. Widmayer: "Towards an Analysis of Range Query Performance in Spatial Data Structures", *Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '93)*, pp.214-221, Washington DC, 1993.

[Papa95] A. Papadopoulos and Y. Manolopoulos: "Multiple Range Query Optimization in Spatial Databases", *Information Systems*, submitted.

[Rous85] N. Roussopoulos and D. Leifker: "Direct Spatial Search on Pictorial Databases Using Packed R-trees", *Proceedings of the 1985 ACM SIGMOD Conference*, pp.17-31, Austin, TX, 1985.

[Rous95] N. Roussopoulos, S. Kelley and F. Vincent: "Nearest Neighbor Queries", *Proceedings of the 1995 ACM SIGMOD Conference*, pp.71-79, San Jose, CA, 1995.

[Same90a] H. Samet: *"The Design and Analysis of Spatial Data Structures"*, Addison-Wesley, MA, 1990.

[Same90b] H. Samet: *"Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS"*, Addison-Wesley, MA, 1990.

[Sell87] T. Sellis, N. Roussopoulos and C. Faloutsos: "The R^+-tree: a Dynamic Index for Multidimensional Objects", *Proceedings of the 13th VLDB Conference*, pp.507-518, Brighton, UK, 1987.

[Theo96] Y. Theodoridis and T. Sellis: "A Model for the Prediction of R-tree Performance", *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '96)*, Montreal, Canada, 1996.