An Estimating Model for the Number of Node Accesses in NN Search

Feng, Yaokai
Department of Intelligent Systems, Graduate School of Information Science and Electrical Engineering, Kyushu University : Graduate Student

Makinouchi, Akifumi
Department of Intelligent Systems, Faculty of Information Science and Electrical Engineering, Kyushu University

https://doi.org/10.15017/1525449

出版情報：九州大学大学院システム情報科学紀要. 7 (2), pp.87-92, 2002-09-26. 九州大学大学院システム情報科学研究院

九州大学大学院
システム情報科学紀要
第7巻 第2号 平成14年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.7, No.2, September 2002

An Estimating Model for the Number of Node Accesses in NN Search

Yaokai FENG* , Akifumi MAKINOUCHI**

(Received June 14, 2002)

Abstract: Nearest Neighbor (NN) search has been widely used in spatial databases (e.g., find neighbor cities) and multimedia databases (e.g., similarity search). However, the theoretical analysis on its performance with m (the number of neighbor objects reported finally), n (the cardinality of database) and d (the dimensionality) as parameters has not been done yet. This paper presents an analytical model for estimating performance of the newest NN search algorithm using uniformly distributed objects, focusing on the number of node accesses. The theoretical analysis is verified by experiments.

Keywords: Multidimensional index, Nearest neighbor search, Estimating model

1. Introduction

During the last decade, the increase in the number of computer applications that rely heavily on spatial data and on multimedia data has caused the database community to focus on the management and retrieval of multidimensional data. Nearest Neighbor (NN) search is very important in Geographic Information Systems (GIS) as well as in Multimedia Applications. For example, in GIS applications, NN search can be used to find the neighbor cities, schools or factories to a given location. In multimedia database fields, NN search can be used to perform similarity search, which is a very popular kind of content-based search.

This paper analyzes performance of the newest NN search algorithm for uniformly distributed point data with m (the number of neighbor objects reported finally), n (the cardinality of database) and d (the dimensionality of the space where the points are located) as parameters. The analysis focuses on the number of node accesses, which is a very important factor on performance of NN search. The theoretical analysis is verified by experimental results.

This paper is organized as follows: Related studies and the motivation of our investigation are presented in Section 2. The analytical model for estimating performance of the newest NN search algorithm is presented in Section 3. The model presented in this paper is verified by experiments in Section 4. The conclusion is drawn in Section 5.

* Department of Intelligent Systems, Graduate Student
** Department of Intelligent Systems

2. Related Work

Before giving related work, R-tree and NN search on it are explained briefly.

2.1 R-trees

R-trees are widely used in multi-dimensional databases and they are regarded as being among the best multi-dimensional indexes.

An R-tree is a hierarchy of nested d-dimensional MBRs (Minimum Bounding Rectangles). MBR is a hyper-rectangle that minimally bounds the objects in the corresponding subtree. Each non-leaf node of the R-tree contains an array of entries, each of which consists of a pointer and an MBR (note that each MBR can be indicated by two points). The pointer refers to one child node of this node and the MBR is the minimum bounding rectangle of the child node referred to by the pointer. Each leaf node of the R-tree contains an array of entries, each of which consists of an object identifier and its corresponding point (for point-objects) or its MBR (for extended-objects). The capacity of each node except the root node is usually chosen such that a node fills up one disk page (or a small number of pages).

In an R-tree for point objects, since it needs two points to indicate each MBR in non-leaf nodes, the fanout of leaf nodes is twice as much as that of non-leaf-root nodes. The fanout of the root node should be greater than one. That is, $1 < f_r \leq M$, $m \leq f_i \leq M$ and $2m \leq f_l \leq 2M$, where f_r, f_i and f_l refer to the fanouts of root node, non-leaf-root nodes and leaf nodes, respectively; m and M are the minimum and the maximum limitation on the fanout of its non-leaf-root nodes, respectively.

2.2 NN search on R-trees

The existing NN search algorithms can be classified into two different groups. One group is k-NN search algorithms, where k, the number of neighbor objects to be retrieved, is known and fixed in advance. The other is Incremental NN (INN) search algorithms, which can also be used when the number of neighbor objects to be retrieved is unknown and is not fixed in advance. The INN search algorithms find and report the neighbor objects one by one from the nearest one until the user is satisfied with the search result. The INN search algorithm[1)] has been regarded as the optimal one because of the minimum number of node accesses[6)]. Thus, the INN search is analyzed in this paper.

The key of the INN search algorithm is to use one priority queue to contain objects and nodes of the index. The objects and the nodes in the priority queue are sorted in ascending order of their distance values (for objects) or $MINDIST$s (for nodes) from the given query point, where $MINDIST$ is the minimum distance of a node (i.e., its MBR) from the query point. Initially, the priority queue is empty. This algorithm begins with inserting the root node in the priority queue. The members (nodes or objects) of the priority queue are dequeued one by one. If the dequeued member is a non-leaf node, then all of its child nodes are inserted in the priority queue. If the dequeued member is a leaf node, then all of its objects are inserted. If the dequeued member is an object, this object is reported as the newest neighbor object. The algorithm repeats the "dequeue-insert" process untill user is satisfied with the search result or a wanted number of NN objects have been reported.

2.3 Performance Analysis of Nearest Neighbor Search

Stefan Berchtold et al. present a cost model for NN search[6)]. However, as pointed out in the conclusion section of that paper, the cost model can be used for 1-NN search only. That is, the cost model can be used only in the case that one NN object is reported and that model can not simply be generalized to an arbitrary number of NN objects reported finally. Moreover, it analyzes the number of accessed leaf nodes only.

There are still some performance analytic works for some other search algorithms, including the work[2)] is for k-NN algorithm when k=1 and the work[4)] is for range query algorithm.

With m (the number of neighbor objects reported finally), n (the cardinality of database) and d (the dimensionality of the space where the points are located) as parameters, this paper presents a estimating model for the number of node accesses in INN search. To our knowledge, this work has not been done yet.

3. Estimating the Number of Node Accesses in INN Search

The number of node accesses is an important factor on search performance. For disk-resident indexes, it is directly related to the number of disk I/O operations; for memory-resident indexes, it is directly related to the number of cache misses. In fact, the number of node accesses is often analyzed in the works on the performance analyzing of search algorithms.

For simplicity, like some other works[1),5)], we assume that both data objects and the query points are uniformly distributed in the domain. Without loss of generality, as in other analytical works[2),4)], we assume that the data domain is a unit hypersquare and we think that, in this case, it is reasonable to assume that the R-tree nodes of the same height have square-like MBRs roughly of the same size[1),2),5)].

Some symbols used in the analysis and their description are shown in **Table 1**.

Table 1 some symbols and their descriptions.

Symbol	Description
$Accesses_{node}$	number of node accesses
$L(q)$	length of the priority queue
d	dimensionality of database
n	cardinality of database
q	query point
f_l	average number of entries in each leaf node
f_i	average number of entries in each non-leaf-root node
f_r	number of entries in the root node
m	number of neighbor objects reported finally
d_m	distance of the m-th neighbor object from the query point
σ_h	side of the square-like MBRs in level h of R-tree
σ_l	side of the square-like MBRs in the leaf node level of R-tree
l_h	number of nodes in level h of
n_h	number of nodes in level h
H	Height of the R-tree

Here the definition of search region is given as follows.

Definition (search region):

We wish to analyze the situation up to m neighbor objects have been reported. Let o be the m-th neighbor object of the query point q, and d_m is distance of o from q. The region within distance d_m from q is called the search region.

Then, let us consider the appearance of the priority queue when the m-th neighbor object is dequeued, which is shown in **Fig. 1**. In this figure, the black dots are objects and the rectangles are nodes.

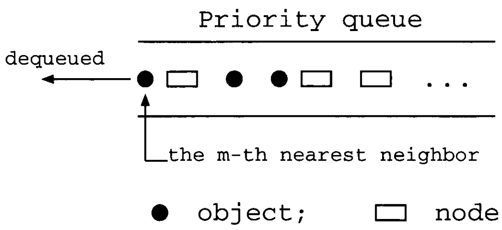

Fig.1 Priority queue when the m-th nearest neighbor is dequeued.

Proposition 1

At the moment when the m-th neighbor object is dequeued, the following equation must be true.

$$\mathrm{N_{h,dequeued}} = \mathrm{N_{h,inter}}$$

where h refers to the level of R-tree and $0 \leq h \leq H-1$. $N_{h,dequeued}$ refers to the number of nodes in level h that have been dequeued from the priority queue. $N_{h,inter}$ refers to the number of nodes in level h that intersect with the search region.

Proof:

Remember that all the members of the priority queue are sorted in ascending order of their distances (for objects) or $MINDIST$s (for nodes) from the query point. And the $MINDIST$ value of any child node of each node, obviously, must be greater than or equal to the $MINDIST$ value of this node. Thus,

(1) the $MINDIST$ value of any node dequeued from the queue must be less than d_m. That is, they must intersect with or be contained in the search region, and

(2) on the other hand, it is impossible for the nodes whose $MINDIST$s are less than d_m still to stay in the queue or to be contained in some node that still stays in the queue. In other words, all the nodes that intersect with or that are contained in the search region must have been dequeued. □

Proposition 1 means that the number of nodes in any level that have been dequeued from the queue must be the same as the number of nodes in this level that intersect with the search region.

Proposition 2

At the time when the m-th neighbor object is dequeued, the expected number of nodes in level h that have been inserted and that still remain in the priority queue, $N_{h,left}$, is given by

$$N_{h,left} = \begin{cases} f_r - N_{h,inter} & (if\ \ h = 1) \\ f_i \cdot N_{h-1,inter} - N_{h,inter} & \\ & (if\ \ h > 1) \end{cases}$$

Proof:

(1) When $h > 1$, at the parent level of level h (i.e., level $h-1$), there are $N_{h-1,dequeued}$ nodes have been dequeued and all their $f_i \cdot N_{h-1,dequeued}$ child nodes in level h have been inserted in the priority queue. Of these nodes in level h that have been inserted in the queue, $N_{h,dequeued}$ nodes have been dequeued. Therefore,

$$\begin{aligned} & N_{h,left} \\ & = f_i \cdot N_{h-1,dequeued} - N_{h,dequeued} \\ & = f_i \cdot N_{h-1,inter} - N_{h,inter} \end{aligned}$$

(2) When $h = 1$, then the parent of this level is the root node and all the f_r nodes in this level have been inserted. Of these f_r nodes, $N_{h,inter}$ nodes have been dequeued. Thus, in this case, $N_{h,left}$ is the difference of f_r and $N_{h,inter}$ □

Proposition 3

For uniformly distributed query point, the probability of the query point being contained in any node is the volume of this node.

Proof:

Obviously, if the query point is uniformly located in the whole data space, the probability of the query point being contained in one node, P_{node}, is given by:

$$P_{node} = \frac{Volume_{node}}{Volume_{space}}$$

According to our assumptions given at the beginning of this section, the volume of the whole space is one. Thus, P_{node} is the volume of this node. □

3.1 Expected Distance From Query Point to m-th NN Object

It is clear that the object density in the search region is the same as that in the whole space since the objects are distributed uniformly in the whole space. That is,

$$\frac{m}{Vol_{region}} = \frac{n}{Vol_{whole}} \quad (1)$$

where Vol_{whole} refers to the volume of the whole space and Vol_{region} is the volume of the search region, a d-dimensional hyper sphere with d_m as its radius. Vol_{whole} is one according to our assumptions and, according to the knowledge of geometry, Vol_{region} is given by

$$Vol_{region} = \frac{\sqrt{\pi^d}}{\Gamma(d/2+1)} \cdot d_m^d$$

$\Gamma(x+1) = x \cdot \Gamma(x)$
$\Gamma(1) = 1$
$\Gamma(1/2) = \sqrt{\pi}$
Therefore, d_m can be estimated as follows:

$$d_m = \sqrt[d]{\frac{m}{n} \cdot \frac{\Gamma(d/2+1)}{\sqrt{\pi^d}}} \quad (2)$$

3.2 Expected Side of Each Node

It is clear that the expected number of nodes in level h, n_h, can be given by

$$n_h = f_r \cdot f_i^{h-1} (h \geq 1) \quad (3)$$

Therefore, the expected number of objects in each node of level h is n/n_h. Being same as the analyzing in Section 3.1, the following equation is true.

$$\frac{Vol_{node}}{Vol_{whole}} = \frac{1}{n_h}$$

Thus,

$$Vol_{node} = \sigma_h^d = \frac{1}{n_h}$$

That is, the expected side of each node in level h, σ_h, can be given by

$$\sigma_h = n_h^{-\frac{1}{d}} = f_r^{-\frac{1}{d}} \cdot f_i^{-\frac{h-1}{d}} \quad (4)$$

Considering the number of leaf nodes is $\frac{n}{f_l}$, then the expected side of each leaf node, σ_l, can be given by

$$\sigma_l = \left(\frac{n}{f_l}\right)^{-\frac{1}{d}} \quad (5)$$

3.3 Estimating Model For the Number of Node Accesses

Proposition 4

For uniformly distributed query point, the probability of the search region intersecting with or being contained in any node of level h, $P_{h,intersect}$, is given by:

$$\tau = \sum_{i=0}^{d} \binom{d}{i} \cdot \sigma_h^{(d-i)} \cdot \frac{\sqrt{\pi^i}}{\Gamma(i/2+1)} \cdot d_m^i$$

$P_{h,intersect} = min\{\tau, 1\}$
where d_m is estimated by Equation (2) and σ_h can given by Equation (4).

Proof:

See **Fig. 2**. The rectangle is a node MBR in level h whose side length is σ_h. The dotted circle has the same size as the search region and touches the side of the node MBR. The round-corner rectangle (called Minkowski-sum) is the trace of the center of the dotted circle after the dotted circle makes a circuit, keeping the touching state, along the sides of the node MBR.

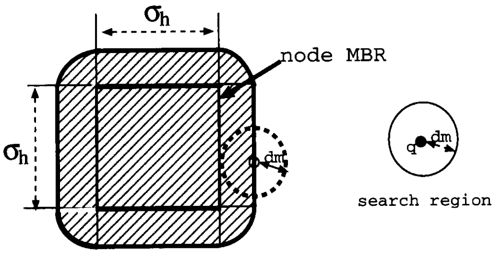

Fig.2 Example of Minkowski-sum in 2-dimensional space.

Obviously, if the search region intersects with the MBR of this node, then the center of the search region, q, is located in the Minkowski-sum of this node and vice versa. That is,

$$P_{h,intersect} = P_{q,mink} = Vol_{mink}$$

where $P_{q,mink}$ refers to the the probability that q is contained in the Minkowski-sum; Vol_{mink} is the volume of Minkowski-sum. Thus, calculating the volume of Minkowski-sum in d-dimensional space is necessary for calculating $P_{h,intersect}$.

A calculating method the volume of Minkowski-sum in d-dimensional space has been proposed and mathematically proved by ChangZhou Wang and X. Sean Wang[7] as follows.

$$Vol_{mink} = \sum_{i=0}^{d} Q_i \cdot \binom{d}{i} \sigma_h^{d-i} \tag{6}$$

As mentioned in Section 3.1, Q_i can be given by

$$Q_i = \frac{\sqrt{\pi^i}}{\Gamma(i/2+1)} \cdot d_m^i \tag{7}$$

If Equation (7) are substituted into Equation (6) and $P_{h,intersect} = Vol_{mink} \leq 1$ is considered, Proposition 4 can be proved. □

Lemma 1 *The expected number of nodes in level h that intersect with the search region, $N_{h,inter}$, can be given by:*

$$N_{h,inter} = n_h \cdot P_{h,intersect} \tag{8}$$

where $P_{h,intersect}$ is estimated by Proposition 4 and n_h can be given by Equation (3).

Proof: Since f is the average number of entries in each node, it is clear that the expected number of nodes in level h is n_h. Thus, to calculate the expected number of nodes in level h that intersect with the search region, we have to sum the probabilities of each node in this level. Since the probabilities of nodes are the same each other, the number of nodes in this level that intersect with the search region can be given by multiplying the probability of one node with the number of nodes in this level, n_h. □

According to Proposition 1 and considering that the root node must be accessed, by the moment when the m-th neighbor object is reported, the expected number of node accesses (i.e., the number of nodes that have been dequeued from the queue), $Accesses_{node}$, is given by

$$Accesses_{node} = 1 + \sum_{h=1}^{H-1} N_{h,inter} \tag{9}$$

where H is the height of the R-tree, $N_{h,inter}$ is given by Equation (8).

Note that f_r, f_i, f_l and H will be discussed in Section 3.4.

3.4 Discussion of f_r, f_i, f_l and H

The estimating methods of f_r, f_i, f_l and H are still not presented in the above equations.

According to the analysis made by C. Faloutsos and I. Kamel[4],

$$f_i = Fanout \times u \tag{10}$$

where *Fanout* is the maximum numbers of entries in each non-leaf node. u is the average node utilization (typically, 70% for the R*-tree[4]). Note that (1) *Fanout* is given by user and it decides the size of each node. (2) All experiments in this study is performed with R*-tree. Thus, f_i can be given by

$$f_i = 0.7 * Fanout$$

As mentioned in Section 2.1,

$$f_l = 2f_i \quad 1 < f_r \leq f_i \tag{11}$$

It is clear that

$$\begin{aligned} n &= f_r \cdot f_i^{(H-2)} \cdot f_l \\ &= 2f_r \cdot f_i^{(H-1)} \end{aligned} \tag{12}$$

Thus,

$$2f_i^{(H-1)} \leq n \leq 2f_i^H$$

That is,

$$log_{f_i}(n/2) \leq H \leq log_{f_i}(n/2) + 1$$

That means

$$H = \lceil log_{f_i}(n/2) \rceil$$

Considering Equation (12), then f_r can be given by

$$f_r = \frac{n}{2f_i^{(H-1)}}$$

4. Experimental Evaluation

Using uniformly distributed points we verified our estimation formulas as the three parameters (i.e., d, n and m) change.

The tested results are shown in **Table 2**, **Table 3** and **Table 4** along with the calculated results.

From these results, we can observe that

1. as dimensionality increases, the gap between calculated result and test result gets large. We think

Table 2 Verification of the estimation formulas as d increases (n=40000, m=40).

d	$Fanout$	$Access_{node}$	
		Calculated	Tested
2	40	6.01	5.91
4	20	37.24	35.96
6	13	243.28	235.32
8	10	1412.42	1302.41
10	8	4348.04	3151.74

Table 3 Verification of the estimation formulas as m grows (d=4, n=40,000, $Fanout$=20).

m	$Access_{node}$	
	calculated	tested
20	27.72	26.71
40	37.24	35.96
60	44.90	42.75
80	51.60	49.10
100	57.69	54.34

Table 4 Verification of the estimation formulas as n increases (d=4, m=40, $Fanout$=20).

n	$Access_{node}$	
	calculated	tested
2000	31.15	29.02
20000	37.14	35.51
40000	37.24	35.96
60000	37.25	36.13
80000	38.28	37.21
100000	38.54	37.75

this is because in high-dimensional spaces, the objects become very sparse and it seems that some other factor(s) should be taken into account in very-high-dimensional spaces. Anyway, according to our experiments, the error rate can be reduced if we use bigger databases. Note that, R*-tree can not be used efficiently for very-high-dimensional spaces.

2. the change of m has not much influence on the degree of accuracy of our model when m is relatively very small to the cardinality of the database. Another observation is that performance of the INN search algorithm degrades as m increases.

3. the gap between the calculated result and the test result tends to become smaller as the database becomes larger. We think this is because that larger databases of uniformly distributed points tend to meet well the assumptions in our analysis.

From all above results, we observe that the test results are generally close to the calculated results, which means that the performance of the INN search algorithm for uniformly distributed objects is mathematically verified.

5. Conclusion

In this paper we proposed a model for uniformly distributed point data to mathematically analyze performance of the newest NN search algorithm with m (the number of neighbor objects reported finally), n (the cardinality of database) and d (the dimensionality) as parameters. The experimental results show that our model is efficient for the objects with the dimensionality less than 10. Although the model is presented for uniformly distributed data, we think, as the parameters (m, n, d) change the performance tendency of uniformly distributed points revealed by our model is roughly similar to that with actual databases. We believe that the analyzing method can be applied to other performance analysis, too.

References

1) G.R. Hjaltason, H. Samet. "Distance Browsing in Spatial Database". ACM Transactions on Database Systems, Vol. 24, No. 2, pages 265-318, June 1999.
2) A. Papadopoulos, Y. Manolopoulos. "Performance of Nearest Neighbor Queries in R-trees". In Proceedings of International Conference on Database Theory, pages 394-408, Delphi, Greece, January 1997.
3) Y. FENG, M. KUBO et al: A New SOM-based R*-tree: Building and Retrieving. Research Reports on Information Science and Electrical Engineering of Kyushu University. Vol.6, No.2, 2001: 209-214.
4) C. Faloutsos, I. Kamel. "Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension". In Proceedings of ACM PODS Symposium, pages 4-13, 1994.
5) K. Kim, S. K. Cha, K. Kwon. "Optimizing Multidimensional Index Trees for Main Memory Access". In Proceedings of ACM SIGMOD International Conference on Management of Data, pages 139-150, Santa Barbara, California, USA, 2001.
6) S. Berchtold, C. Bohm, D. A. Keim, HP. Kriegel. "A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space". in Proceedings of PODS, pages 78-86, Tucson, Arizona, 1997.
7) Changzhou Wang, Xiaoyang Sean Wang: Indexing very high-dimensional sparse and quasi-sparse vectors for similarity searches. VLDB Journal 2001(9): 344-361.