An Estimating Model for the Length of the Priority Queue in INN 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/10.15017/1515806

出版情報：九州大学大学院システム情報科学紀要. 8 (1), pp.1-6, 2003-03-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第8巻 第1号 平成15年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.8, No.1, March 2003

An Estimating Model for the Length of the Priority Queue in INN Search

Yaokai FENG* and Akifumi MAKINOUCHI**

(Received December 13, 2002)

Abstract: Incremental Nearest Neighbor (INN) search has been widely used in spatial databases and multimedia databases. R*-tree is still regarded as being among the best multi-dimensional indices. This paper presents an analytical model for estimating performance of the INN search algorithm on R*-tree. Our model focuses on the length of the priority queue, the total number of its members. In our model, the particularity on the number of entries in the root node and the possible difference of fanouts between the leaf nodes and the other nodes are taken into account. The theoretical analysis is verified by experiments.

Keywords: Multidimensional index, Nearest neighbor search, Estimating model

1. Introduction

Nearest Neighbor (NN) search is very important in Geographic Information Systems (GIS) as well as in Multimedia Applications. The Incremental NN (INN) search algorithm is very popular because it can be used whether the wanted number of nearest neighbor objects is known in advance or not. R*-tree is still regarded as being among the best multi-dimensional indexes and is widely used in multi-dimensional databases. Our analysis focuses on the length of the priority queue (which still not be analyzed yet), which is important factors on performance of the INN search algorithm. Note that, the length of the priority queue means the total number of its members, which is not related to the implementation of the priority queue.

2. Related Work

Performance estimation of the INN search algorithm is discussed briefly and tested in [1]. However, there exist the following problems in their analysis. (1) The presented model is on the expected number of the accessed leaf nodes and the expected number of objects remaining in the priority queue. (2) The model is for 2-dimensional spaces only. And it can not be simply generalized to high-dimensional spaces. (3) The presented model is not verified by experiments. (4) The analysis is based on the following assumption: "on the average, half of the objects in the leaf nodes that intersect with the search region are inside the search region, while half are outside". We think this assumption is farfetched. Stefan Berchtold et al. present a cost model for NN search[2]. However, as pointed out in the conclusion section of that paper, that 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 reported NN objects.

3. Performance Estimation of the INN Search Algorithm

The priority queue is a very important data structure in the INN search algorithm and it is not long in low-dimensional spaces[1]. However, it becomes very long for large databases in high-dimensional spaces. The long priority queue means a large number of insertions and comparisons. Thus, it is very expensive to maintain, especially for memory-resident indexes.

For simplicity, like some other works[1),3)], 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[4),5),6)], we assume that the data domain is a unit hypercube and we think that, in this case, it is reasonable to assume that the R-tree nodes of the same height have cube-like MBRs roughly of the same size[1),3),4)].

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

Note that the level numbers are counted from the root level whose level number is zero.

Then, let us consider the appearance of the priority queue when the m-th neighbor object is dequeued (or say, obtained), which is shown in **Fig.1**.

The following propositions are proved in our paper[7].

Proposition 1

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

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

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

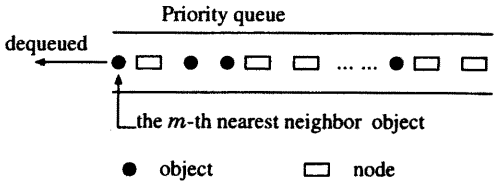

Fig.1 The abstract structure of the priority queue.

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

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

Proposition 2

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

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

Proposition 3

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

Proposition 4

The probability of the search region intersecting with any node of level h, $P_{h,intersect}$, is given by

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

Now, two new propositions and their proofs is given.

Proposition 5

When the m-th neighbor object is dequeued, the expected number of objects that still remain in the priority queue, N_{object}, is given by

$$N_{object} = f_l \cdot N_{leaf,dequeued} - m,$$

where $N_{leaf,dequeued}$ refers to the number of the leaf nodes that have been dequeued.

Proof:

Since $N_{leaf,dequeued}$ leaf nodes have been dequeued, all the $f_l \cdot N_{leaf,dequeued}$ objects in these leaf nodes have been inserted in the queue. Note that m objects have been dequeued. Thus, the number of remaining objects is $f_l \cdot N_{leaf,dequeued} - m$. □

Proposition 6

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

$$N_{h,inter} = n_h \cdot P_{h,intersect}, \quad (1)$$

where $P_{h,intersect}$ is estimated by Proposition 5 and n_h can be given by Equation (3).

Proof:

To calculate $N_{h,inter}$ we have to sum $P_{h,intersect}$ for every node in this level. Because the objects and the query point are uniformly distributed, all the nodes in this level have the same probability of intersecting with the search region. Thus, $N_{h,inter}$ can be given by multiplying $P_{h,intersect}$ with the number of nodes in this level, n_h. □

3.1 Estimating d_m

See **Fig.2**. The shadow hyper-sphere is called *RatioSphere* whose center is q and volume is m/n. Its

radius is denoted as δ_m. Considering the volume of the whole space is 1 and all the objects are uniformly distributed, the expected number of the objects in the RatioSphere should be m. In this study, δ_m is used as d_m.

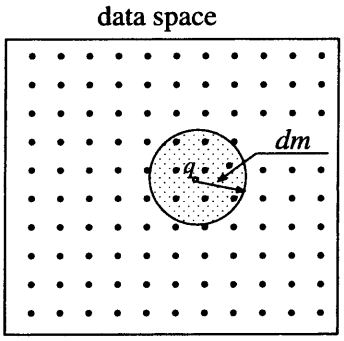

Fig.2 Estimation of d_m.

Let Vol_{region} be the volume of the RatioSphere. According to the knowledge of geometry, the volume of the RatioSphere, Vol_{region}, is given by

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

$$\Gamma(x+1) = x \cdot \Gamma(x), \quad \Gamma(1) = 1, \quad \Gamma(1/2) = \sqrt{\pi}.$$

Considering $Vol_{region} = \frac{m}{n}$ and δ_m is used as d_m in this study, d_m can be estimated by

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

3.2 Side Length of Each Node at Level h (i.e., σ_h)

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

$$n_h = f_r \cdot f^{h-1} (h \geq 1). \tag{3}$$

Let *ShareSpace*(w) refer to the average space share of w NN objects in the whole space. Its volume should be w/n since the objects are uniformly distributed and the volume of the whole space is 1. The number of objects in each node of level h is n/n_h. Thus, the corresponding space share of each node at level h is $ShareSpace(n/n_h)$. Its volume, Vol_{share}, is

$$Vol_{share} = \frac{n}{n_h} \Big/ n = \frac{1}{n_h}.$$

Like the analysis in Section 3.1, Vol_{share} is used to estimate the volume of each node at level h. If Equation (3) is substituted, then σ_h can be given by

$$\sigma_h = \sqrt[d]{\frac{1}{f_r \cdot f^{h-1}}}. \tag{4}$$

Considering the number of the objects in each leaf node is f_l, in the same way, σ_l can be given by

$$\sigma_l = \sqrt[d]{\frac{f_l}{n}}. \tag{5}$$

3.3 Expected Length of the Queue

Using the priority queue is a distinctive feature of the INN search algorithm and the priority queue is not long in low-dimensional spaces. However, according to our investigations, the priority queue may be very long for large databases in high-dimensional spaces. It is clear that maintaining a very-long sorted queue means a large number of comparisons and insertions. Thus, the length of the priority queue is an important factor on performance of the INN search algorithm (see the discussion at the beginning of this section). However, analyzing on the length of the priority queue has not been done yet. Our model for estimating the length of the priority queue is presented in this subsection.

Let us see **Fig.1** again. Obviously, after the m-th neighbor object is dequeued,

$$L(q) = N_{object} + N_{node}, \tag{6}$$

where N_{object} and N_{node} refer to the number of objects and the number of nodes that still remain in the queue when the m-th neighbor object is dequeued, respectively. First, N_{object} is estimated.

According to Proposition 5, the probability of the search region intersecting with any leaf node, $P_{l,intersect}$, can be given by

$$\tau_l = \sum_{i=0}^{d} \binom{d}{i} \cdot \sigma_l^{(d-i)} \cdot \frac{\sqrt{\pi^i}}{\Gamma(i/2+1)} \cdot d_m^i, \qquad (7)$$
$$P_{l,intersect} = min\{\tau_l, 1\},$$

where σ_l can be given by Equation (5).

According to Proposition 1 and Equation (1), the number of leaf nodes that have been dequeued, $N_{leaf,dequeued}$, can be given by

$$\begin{aligned} &N_{leaf,dequeued} \\ &= N_{leaf,inter} \\ &= Num_{leaf} \cdot P_{l,intersect} \\ &= \tfrac{n}{f_l} \cdot P_{l,intersect}, \end{aligned} \qquad (8)$$

where $N_{leaf,inter}$ refers to the number of leaf nodes that intersect with the search region. $P_{l,intersect}$ can be given by Equation (7). Num_{leaf} is the expected number of the leaf nodes.

According to Proposition 3 and Equation (8), N_{object} in Equation (6) can be estimated by

$$\begin{aligned} &N_{object} \\ &= f_l \times N_{leaf,dequeued} - m \\ &= n \cdot P_{l,intersect} - m. \end{aligned} \qquad (9)$$

We can understand Equation (9) as follows. When the m-th neighbor object is dequeued from the queue, $N_{leaf,dequeued}$ leaf nodes have already been dequeued and all of their objects (the total number is $f_l \times N_{leaf,dequeued}$) are inserted in the queue. m means the number of objects that have already dequeued.

Then, let us estimate N_{node} in Equation (6). See **Fig.1** again. Proposition 2 and Equation (1) present the calculating method for $N_{h,left}$. If we sum $N_{h,left}$ for each level except the root level, the expected number of the nodes left in the priority queue when the m-th neighbor object is dequeued, N_{node}, can be given by

$$N_{node} = \sum_{h=1}^{H-1} N_{h,left}, \qquad (10)$$

where $N_{h,left}$ can be given by Proposition 2.

By substituting Equation (10), Equation (9) and the other corresponding equations in Equation (6), the length (the number of members) of the priority queue when the m-th neighbor object is obtained can be estimated by

$$\begin{aligned} &L(q) = \\ &\sum_{h=1}^{H-1} N_{h,left} + \\ &min\left\{\sum_{i=0}^{d} \binom{d}{i} \cdot \frac{m^{\frac{i}{d}} \cdot \Gamma^{\frac{i}{d}}(d/2+1) \cdot f_l^{1-\frac{i}{d}}}{\Gamma(i/2+1)},\ C\right\} \\ &-m, \end{aligned}$$

where

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

$$N_{h,inter} = f_r \cdot f^{h-1} \cdot min\left\{\sum_{i=0}^{d} \binom{d}{i} \cdot \frac{\pi^{\frac{i}{2}} \cdot d_m^i \cdot \sigma_h^{d-i}}{\Gamma(i/2+1)},\ 1\right\};$$

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

$$\sigma_h = \sqrt[d]{\left(\frac{1}{f_r \cdot f^{h-1}}\right)}.$$

3.4 Discussion of f_r, f_i, f_l and H

f_r, f_i, f_l and H in the above estimating equations are discussed here.

Let F_i and F_l denote the maximum number of entries in each non-leaf node and the maximum number of entries in each leaf node, respectively. When one R*-tree is built, there are two possibilities that $F_l = 2 * F_i$ (for point objects) and $F_l = F_i$. Here the two cases are discussed separately.

3.4.1 Case of $F_l = 2 * F_i$

According to the analysis made by C. Faloutsos and I. Kamel[5], the average node utilization of all nodes in R*-tree is 70%. Clearly,

$$\begin{aligned} n &= (0.7 * F_i)^{(H-1)} \cdot f_l \\ &= (0.7 * F_i)^{(H-1)} \cdot (0.7 * F_l) \\ &= 2 * (0.7 * F_i)^H. \end{aligned}$$

$$H = \lceil log_{(0.7*F_i)}(n/2) \rceil. \qquad (11)$$

where F_i is given by user and it decides the size of each node.

The number of entries in the root node has its specific characteristics. Anyway, the number of the non-root nodes far exceed that of the root node (only one root node). Thus, we can think the average

node utilization of the non-root nodes is also 70%. That is,

$$f_i = 0.7 * F_i \quad f_l = 0.7 * F_l = 2 * f_i.$$

It is clear that

$$n = f_r \cdot f_i^{(H-2)} \cdot f_l = 2f_r \cdot f_i^{(H-1)}.$$

$$f_r = \frac{n}{2f_i^{(H-1)}}. \tag{12}$$

Now, we prove that f_r calculated by Equation (12) meets the necessary condition. That is, $1 < f_r \leq F_i$.

According to Equation (11), $log_{f_i}(n/2) \leq H < log_{f_i}(n/2) + 1$. Then it becomes easy to prove $1 < f_r \leq F_i$, which is omitted because of the limitation of space.

3.4.2 Case of $F_l = F_i$ (say F)

In the same way as the former case, f_r, f_i, f_l and H can be given by

$$f_l = f_i = 0.7 * F. \quad H = \lceil log_{f_i}(n) \rceil. \quad f_r = \frac{n}{f_i^{(H-1)}}.$$

Note that the above formulas in this chapter are based on the average case and not absolute ones. However, we are interested in the average case, and exceptional cases do not harm the generality.

4. Experimental Evaluation

Using uniformly distributed points we verified our estimating model.

4.1 Evaluation with Different d

Since the analysis in this chapter is based on the average case, the results are the average values of 100 random trials.

Note that only the results of the case that $F_l = 2 * F_i$ (see Section 3.4) is presented in this chapter. F_i is denoted as *Fanout* in this section. Anyway, according to our study in the other case ($F_l = F_i$), the tendency of performance and the error rate of our model do not change much.

Table 2 shows the calculated results and their experimental counterparts. Without loss of generality, we let m be 40 and n be 40,000. From **Table 2**, the following observations can be obtained.

Table 2 Evaluation as d increases (n=40,000, m=40).

d	*Fanout*	L(q)		
		calculated	tested	error rate (%)
2	40	221.99	214.83	3.23
4	20	788.96	743.34	5.78
6	13	3390.57	3147.25	7.18
8	10	15226.00	13530.14	11.14
10	8	39959.90	32540.27	18.57
12	7	39959.90	39012.01	2.37

1. Performance of the INN algorithm degrades exponentially as d increases.

2. As dimensionality increases, the gap between calculated result and tested result gets larger.

3. If dimensionality reaches 10, the calculated result of $L(q)$ roughly reaches its limit, which is $n - m = 39960$.

4. When d increases from 10 to 12, the error rate drops greatly. This is because the INN search tends to access all the objects. Thus, both the calculated result and the tested result tend to the limit. This also mathematically revealed the well-known fact that performance of the NN search algorithm on R-trees may become worse than sequential scan in high-dimensional spaces.

Also, our model is verified as the other two parameters (i.e., m and n) change. The result is not presented in this paper because of the limitation of space. Anyway, the following observations can be obtained.

1. The change of m has not much influence on accuracy of our model when m is relatively very small to n. Another observation is that performance of the INN search algorithm degrades as m increases. We think this is easy to understand.
2. The error rate of our model 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.

4.2 Evaluation with Different m

The results are shown in **Table 3**, where $d = 4$, $n = 40,000$ and M=20.

From **Table 3**, we know that the change of m has not much influence on accuracy of the model when m is relatively very small to n. Another observation is that performance of the INN search algorithm degrades as m increases, which is easy to understand.

4.3 Evaluation with Different n

The database cardinality changes from 200 to 200,000. The results are shown in **Table 4**, where

Table 3 Evaluation as m grows (d=4, n=40,000, M=20).

m	L(q)		
	calculated	tested	error rate (%)
20	583.10	560.08	3.95
40	788.96	743.34	5.78
60	952.08	911.62	4.25
80	1093.12	1038.46	5.00
100	1220.01	1149.91	5.75

$d = 4$, $m = 40$ and M=20.

Table 4 Evaluation as cardinality increases (d=4, m=40, M=20).

C	L(q)		
	calculated	tested	error rate (%)
200	159.88	141.91	11.24
2000	707.68	641.02	9.42
20000	785.32	734.01	6.53
40000	788.96	743.34	5.78
60000	792.61	751.06	5.24
80000	796.25	760.18	4.53
100000	799.90	765.19	4.34
200000	810.57	783.01	3.40

From **Table 4**, we observe that

1. The error rate of the model tends to become smaller as the database becomes larger. I think this is because that larger databases of uniformly distributed points tend to meet well the assumptions in the analysis.
2. If n reaches 20000, the length of the priority queue increase very slowly as n increases. I think this is because that there exist the following two contrary factors that counteract each other to some extent.
   (a) On the one hand, as n grows, the point density increases. Thus, the d_m tends to become shorter and the volume of the search region becomes smaller.
   (b) On the other hand, as the point density increases, the node MBRs become smaller and density of nodes increases.

From all above results, we observe that the tested results are generally close to the calculated results, which means that the actual performance of the INN search for uniformly distributed objects is mathematically verified. In other words, one could use the model presented in this chapter to estimate performance of INN search for uniformly distributed objects.

5. Conclusion

In this paper, we proposed a model for uniformly distributed point data to mathematically analyze performance of the INN search algorithm with m (the number of neighbor objects reported finally), n (database cardinality) and d (dimensionality) as parameters, focusing on the length of the priority queue, which has not been done yet by other works. Using our model, *dimensionality curse* of the INN search was mathematically revealed for an arbitrary number of NN objects to be retrieved.

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) 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.
3) 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.
4) 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.
5) 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.
6) I. Kamel, C. Faloutsos. "On Packing R-trees". In Proceedings of the 2nd International Conference on Information and Knowledge Management, pages 490-499, 1993.
7) Y. Feng and A. Makinouchi. "An Estimating Model for the Number of Node Accesses in NN Search". Research Reports on Information Science and Electrical Engineering of Kyushu University, Vol.7, No.2, pages 87-92, 2002.