CHAPTER 4

ESTIMATING THE NUMBER OF NODE ACCESSES IN R*-TREE FOR k-NEAREST NEIGHBOR QUERY

Efficient storage and retrieval of multidimensional data in large volumes has become one of the key issues in the design and implementation of commercial and application software. The kind of queries posted on such data is also multifarious. Today’s research has turned towards the development of powerful analytical method to predict the performance of indexing structures for various categories of queries such as range, NN and join. This work focuses on performance analysis of R*-tree for kNNQs. General approaches are available in literature that works better for larger values of \( k \). But these models seldom give the same better performance for small values of \( k \). A proposal has been made for improved performance analysis model for kNNQ for small values of \( k \). The results are tabulated and compared with existing models; the proposed model outperforms the existing models in a significant way for small values of \( k \).

4.1 Preamble

Given a collection of \( N \) objects in a \( d \)-dimensional space, a classical NNQ returns the object with the lowest distance to the query point among all the objects of the space as result. A variation of the NNQ is the kNNQ. If the query does not only want one closest object as answer, but rather a natural number \( k \) of closest points, it is called a kNNQ. The kNNQ selects \( k \) points from the space such that no point among the remaining points in the space is closer to the query point than any of the selected points. Problem of ties are solved by non-determinism.

Analysis of kNNQs aims at predicting:

a) the nearest distance \( D_k \), that is, the distance between the query and the \( k^{th} \) NN and
b) the query cost in terms of the number of index nodes accessed or equivalently, the numbers of nodes, whose MBR intersect the search region $\Theta(q, D_k)$.

$\Theta(q, D_k)$ is called the vicinity circle or the search region of the query. Vicinity circle is pictorially depicted in Figure 4.1. In the figure the vicinity circle determines that the MBRs $M1$ and $M4$ are the potential candidates to contain the kNNs of query $q$.

![Figure 4.1 Vicinity circle](image1)

![Figure 4.2 Vicinity circle with boundary effect](image2)

The first approach to estimate the cost of NNQ [243] suffered in terms of the convergence of $N$ and boundary effects. Boundary effect is due to the vicinity circle including the search regions beyond the boundaries of the data space that do not
contribute any data to the computations or poorly approximate the data that is included in the vicinity region. This is pictorially explained in Figure 4.2. The vicinity circle centered at \( q \) in the figure includes space beyond the area that actually contains the data.

Further investigations [240, 241] and subsequent investigations [239] were done by allowing non-rectangular-bounded pages, but still do not handle boundary effects satisfactorily. It was shown that the number of data points have to be exponential in number of dimensions for the models to provide accurate estimates [238]. The condition for the models for more accurate predictions was also established. The issue of boundary effects were further analyzed [242] and subsequently the lower and upper bounds of the NNQ performance on R*-trees for the \( L_2 \) norm was provided [244]. But these bounds become excessively loose when dimensionality increases and as a result their usage becomes more restricted [245]. The average distance \( D_l \) from a query point \( q \) to its NN was derived by utilizing the fact that for uniform data and query distributions in unit data space \( U \), the probability \( P(q, r) \) that a point falls in the vicinity circle \( \Theta(q, r) \) corresponds to its volume \( Vol(\Theta(q,r)) \) [202, 245]. Part of \( \Theta(q, r) \), however, may fall outside the data space and should not be taken into account in computing \( P(q, r) \). To capture this boundary effect, \( P(q, r) \) should be calculated as the expected volume of the intersection of \( \Theta(q, r) \) and \( U \). The above methods used the concepts of vicinity circles and \textit{Minkowski regions}. Figure 4.3 explains the formation of Minkowski region for a given query point.

![Figure 4.3 The Minkowski region](image)

Figure 4.3 The Minkowski region
A very important observation in all the above said methods is that, they give the same estimate for any query given irrespective of location of the query.

An alternative method that captures the performance of the NNQs using approximation was proposed based on the concept of vicinity rectangles and Minkowski rectangles instead of vicinity circles and Minkowski regions [248]. This method is referred as YT Model in this thesis after its proposer. The approximation of vicinity circle to vicinity rectangle is shown in Figure 4.4. The approximation of Minkowski region to Minkowski rectangle is shown in Figure 4.5.

---

**Figure 4.4** Approximation of vicinity circle to vicinity rectangle

**Figure 4.5** Approximation of Minkowski region to Minkowski rectangle

YT extended the approach to non-uniform data sets with the help of histograms. The rationale behind their approach is that data within a sufficiently small region can be regarded as uniform, even though the global distribution may deviate significantly. The

75
4.2 Estimation of the Number of Node Accesses for a Nearest Neighbor Query

An overall perspective of the computations involved in the estimation of the number of node accesses for a NNQ is given below which is based on various models in the literature, but predominantly YT, which gives better results than other models. The algorithm to find $D_k$ is given in Figure 4.6. In the algorithm, $L_r$ is replaced with $L_D$ for non-uniform data that produces result that are more specific to a given query. $L_D$ is given by

$$L_D = \left[ \sum_{i=0}^{d} \binom{d}{i} s_{M}^{d-i} \frac{\sqrt{\pi}}{\Gamma(i/2+1)} D_k^i \right]^{1/2} \tag{4.1}$$

where $D_k$ is the distance from query point to the $k^{th}$ NN, and $\Gamma\left(\frac{i}{2}+1\right)$ is the function used in calculating the volume of the vicinity circle and $C_v$ is a constant.

Once $D_k$ is computed, the total number of nodes accessed [242] is given by

$$N_A = \sum_{i=0}^{\log_{f}(N/f)} \left( \frac{N}{f^{i+1}} \right)^d \left( \frac{L_i - (L_i/2 + S_i/2)^2}{1-S_i} \right)^d \tag{4.2}$$

where $N$ is the cardinality of the data set; $f$ is the average fanout of the R*-tree; $L_i$ is the side length of Minkowski rectangle; $S_i$ is the average extent of a level $i$ node; and $d$ is the dimensionality of the data space. $S_i$ is given by

$$S_i = \left( 1 - \frac{1}{f} \right)^{1/2} \min \left( \frac{L_i}{N}, 1 \right)^{1/2}, \quad 0 \leq i \leq h-1 \tag{4.3}$$

where $h$ is the height of the tree.
Algorithm Estimate $D_k(q)$

/* Existing algorithm to estimate $D_k$ for NNQ $q$, given input histogram*/
1. begin
2. for each dimension $i=1$ to $d$ do
3. begin
4. initialize $l_i-$ = distance $q$ travels along the negative direction to reach the cell boundary that contains $q$;
5. initialize $l_i+$ = distance $q$ travels along the positive direction to reach the cell boundary that contains $q$;
6. compute heap HP by inserting $l_i+$ and $l_i-$;
7. end;
8. initialize $E_{\text{old}}=0$; initialize $L_{\text{old}}=0$;
9. do
10. compute $l=$deheap HP; set $L=2.1$; compute $E_n =$ Expected number of points in the vicinity rectangle;
11. if ($k \leq E_n$) then
12. begin
13. compute $L_r=\left(\frac{L_{\text{old}}^d (k-E_n)-L^d(k-E_{\text{old}})}{E_n-E_{\text{old}}}\right)^{\frac{1}{d}}$;
14. compute $D_k=L_r/C_v$; return;
15. end;
16. else
17. begin
18. compute $L_{\text{old}}=L$; compute $E_{\text{old}}=E_n$ ;
19. compute HP by inserting updating of $l$;
20. end;
21. while(true);
22. end;

Figure 4.6 Existing algorithm to estimate $D_k$
4.3 **Reason for the Inefficiency of the Existing Model**

*YT* model is suitable for query optimization of *kNNQ* with the larger values of *k*. But it fails to produce accurate results for larger values of *k*. The reason is given below.

![Histogram, query cell and query point](image)

Figure 4.7 Histogram, query cell and query point

In *YT* model, initially, the searching space, which is the vicinity rectangle, is constructed with the minimum distance onto the cell boundaries from the query point. In a R*-tree the number of nodes at $i^{th}$ level is $N / f^i$, where $N$ is the cardinality of the data set and $f$ is the fan out of the node. The number of leaf level MBRs in query cell is $(N / f) / H^d$. Here the histogram resolution $H$ refers to the number of partitions along each dimension and $d$ is the dimensionality of the data set. Figure 4.7 gives an example for $H = 3$. The query point falls in one of these cells. The accuracy of the *YT* model depends on where the query falls within the cell.

The side length of MBR [239] in an R*-tree at level $i$ is $S_{i-1,k} = \left( \frac{1}{f^d} - 1 \right) t_{lk} + S_{lk}$, where $t_{lk} = \frac{1}{(N_l)^{\frac{1}{d}}}$ and $S_{lk}$ is the side length of the node at level 1. $t_{lk}$ is the distance in between centers of the rectangles projections. The
side length of MBR at leaf level is $S_1 = \left( f_1^d - 1 \right) \frac{1}{\sqrt{N}}$. Hence, the distance in between two MBRs along the boundary of the query cell is given by

$$\text{dist} = \frac{1}{H} - \sqrt{\frac{N}{f}} \left( f \frac{1}{d} - 1 \right) t_0$$

(4.4)

In the above equation, in the numerator part $N/f$ is the number of leaf level MBRs and so $\sqrt{(N/f)/H^d} >> (1/H)$. Therefore $\sqrt{(N/f)/H^d} \left( f^d - 1 \right) t_0$ is the positive fraction less than 1. Hence, in the formula the numerator is almost equal to denominator. Therefore, it is concluded that the probability of failure is very near to 1. The distance is pictorially depicted in Figure 4.8. Thorough analysis and the experimental results show that if the query point falls within this distance from the cell boundary, the estimations of $D_k$ are more accurate, and hence the cost of the query in terms of the numbers of nodes accessed. The region covered by the rectangle that is
formed at a distance of \( \text{dist} \) inside the cell is termed as *failure region* and the region between the failure region and the cell boundary is termed as the *success region*. The failure probability is the ratio of the area of failure region to the area of the query cell.

4.4 Improved Model

Due to the drawbacks discussed above, the YT model does not provide accurate estimations of \( D_k \). The inherent drawbacks in the concepts and assumptions lead to poor initialization of iterating variables. An improved model has been proposed that eliminates the drawbacks of the existing method and provide accurate estimations of \( D_k \). The improved model is referred as *NNsk* model.

4.4.1 Estimating the Nearest Distance \( D_k \)

The improved algorithm to estimate \( D_k \) finds the distances onto the boundaries of cell from query point. The vicinity rectangle is formed with one-fourth of the minimum boundary distance and the expected number of points \( E_n \) in \( R \) is calculated. If \( E_n = k \), then algorithm terminates, otherwise if \( E_n > k \), the vicinity rectangle needs to be decreased with \( L = L / 2 \), if \( E_n < k \) then the vicinity rectangle needs to be enlarged with \( L = 2 * L \). But before the vicinity rectangle is altered \( E_n \) and \( L \) are preserved in \( E_{old} \) and \( L_{old} \) respectively. The algorithm will be continued up to area of the vicinity rectangle reaches the area of unit data space. The algorithm is given in Figure 4.9.

4.4.2 Estimating the Number of Node Accesses

The number of nodes accessed can be calculated by replacing cardinality of the data set, \( N \) by average density, \( D.H^2 \) of the intersecting cells with the vicinity rectangle of the uniform data model. Thus the number of nodes accessed can be given by

\[
NA(k) = \sum_{i=0}^{\log_e(D.H^2)} \left[ \left( \frac{D.H^2}{f^{i+1}} \right) \left( \frac{L_i - (L_i / 2 + S_i / 2)^2}{1 - S_i} \right)^d \right]
\]  

(4.5)

where \( S_i \) is the node extension and \( L_i \) is the side length of the Minkowski rectangle.
Algorithm \textit{Estimate-D}_k(q)

/* Algorithm to estimate \(D_k\) for a NNQ \(q\) given an input histogram*/

1. \begin
2. \hspace{1em} \textit{compute} the distances onto the cell boundaries from \(q\), in each dimension;
3. \hspace{1em} \textit{initialize} \(dd\) with the minimum boundary distance;
4. \hspace{1em} \textit{compute} a searching space \(R\), whose centroid at \(q\) and with extend \(L = 2*(dd/4)\) along each dimension;
5. \hspace{1em} \textit{initialize} \(E_{n,old}=0; \text{initialize } L_{old}=0;\)
6. \hspace{1em} \textit{do}
7. \hspace{2em} \textit{compute} \(E_n = \text{expected number of points in vicinity rectangle;}
8. \hspace{2em} \textit{if} (k==E_n) \textit{then}
9. \hspace{3em} \begin
10. \hspace{4em} \textit{compute} \(L_r = \left(\frac{L_{old}^d (k - E_n) - L^d (k - E_{n,old})}{E_{n,old} - E_n}\right)^{\frac{1}{d}};\)
11. \hspace{4em} \textit{compute} \(D_k = L_r/C_v; \text{return;}\)
12. \hspace{3em} \end
13. \hspace{2em} \textit{else if} (\(E_n>k\)) \textit{then compute } L=L/2;
14. \hspace{2em} \textit{else}
15. \hspace{3em} \begin
16. \hspace{4em} \textit{compute} \(L_{old}=L;\)
17. \hspace{4em} \textit{compute} \(E_{n,old}=E_n;\)
18. \hspace{4em} \textit{compute} \(L=L*2;\)
19. \hspace{4em} \end
20. \hspace{2em} \textit{while} (area of vicinity rectangle < \text{unit data space});
21. \hspace{1em} \end

Figure 4.9 Revised algorithm to compute \(D_k\)
4.5 Experimental Results and Discussion

In the pursuit of establishing the claimed improvement due to the proposed method, YT model was taken as the base for comparisons. This is due to the fact of the superiority of YT model is already well established in the literature.

The experiments were conducted over a unit space $[0, 1]^d$ and with sets of 100K point data. The first set of data contained 2-dimensional points with random distribution. The second, third, fourth and fifth sets of data contained 3, 5, 10 and 20 dimensional points respectively with random distribution. The node size and the minimum fill percentage were fixed 32 MBRs and 50% respectively. $float$ data type was used to represent the data. R*-tree for each set of data was constructed separately.

For each set, 500 query points were generated separately of which 250 belonged to success region and 250 belonged to the failure region shown in Figure 4.8. The average relative error in answering a workload of 500 queries for each $k = 1, 10, 100, 500, 1000, 2000$ was measured. If $est_i$ and $act_i$ are the estimated and actual costs of the $i^{th}$ query ($1 \leq i \leq 500$), respectively, then the average relative error is computed as

$$\text{AvgRE} = \left( \frac{1}{500} \right) \sum_{i=1}^{500} \frac{|est_i - act_i|}{act_i}$$

(4.6)

The results for 2-dimensional points are given in Figure 4.10. The results for 3-dimensional points are given in Figure 4.11. The results for 5-dimensional points are given in Figure 4.12. The results for 10-dimensional points are given in Figure 4.13. The results for 20-dimensional points are given in Figure 4.14.

Closely observing the graphs, it is evident that NN$sk$ model improves the prediction for lower values of $k$. As the value of $k$ increases, the performance is on par with YT model. The reduction in the average relative error is both experiments for smaller values of $k$, say 1 to 20 is around 50%. This reduction in the average relative error approaches around 2% for larger values of $k$, say 2000.
Figure 4.10 Comparison of average relative error obtained by applying the estimation models on 2-dimensional data

Figure 4.11 Comparison of average relative error obtained by applying the estimation models on 3-dimensional data
Figure 4.12 Comparison of average relative error obtained by applying the estimation models on 5-dimensional data

Figure 4.13 Comparison of average relative error obtained by applying the estimation models on 10-dimensional data
4.6 Summary

NNQs are one of the most often posted queries on multidimensional data. To answer such queries within short time, databases employ multidimensional indexing structures such as R*-trees and its ramifications. Even though quite a few models are available in the literature to predict the performance of R*-trees, they do not work efficiently for smaller values of $k$. This work attempts to improve the performance of $YT$ model for smaller $k$ values. The results obtained are graphically presented and show a significant improvement over the previous models. The approach discussed in this chapter would facilitate better query optimization.