DEIM Forum 2018 H2-2

Balanced Nearest k-Cluster Query

Sang LE†, Yuyang DONG†, Hanxiong CHEN†, Kazutaka FURUSE†, and Hiroyuki KITAGAWA††

† Department of Computer Science, Tsukuba University
Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8577, Japan.
†† Center for Computational Sciences,
University of Tsukuba. Tsukuba, Ibaraki, 305-8577, Japan.
E-mail: †{leanimation,touuyou}@gmail.com, ††chx@cc.tsukuba.ac.jp,
†††{furuse,kitagawa}@cs.tsukuba.ac.jp

Abstract This paper proposes a new search query, named Balanced Nearest k-Cluster query(BNkC). It is inspired by a related work named Nearest Neighborhood(NNH) Query, which is used to find the nearest cluster. We find a problem that NNH query finds an answer with a particular sized circle, and in some specific cases it needs to modify the radius of the circle to get the answer. Therefore we unfix the radius of the circle and evaluate the candidate circles, which has different radius and center, by a formula which combined both conditions. The main algorithm is based on an R-tree structure. We develop a technique to reduce search region by a property of evaluation formula. We also develop it to skip a group of points rather than skip only one point. We do a preliminary experiment to check the effect of weighting parameter, and we also conduct experiments to measure the performance of BNkC, which show that our techniques can improve the performance.

Key words Nearest Neighborhood search, Region of interest, Spatial Index

1. Introduction

Search algorithms play an important role in our life. We use search engines to find the news we want, use global positioning system to find the nearest restaurant, and use search application to find friends near to us.

Nowadays, our search work prefers to find a group with similar features. For example, when we want to buy something from online shopping sites or we want to find some friends in a social network, the searching result may not only provide the answer we desire, but also push other related result to us. In detail, if we want to find a friend in a social network, when we get his/her information, the friends of him/her will also be recommended to us. Therefore, search algorithms have evolved to fulfill the demand, and research single result will not satisfy us.

Assume that we have a location-based social network. In this location-based network, data are stored in a spatial database as two-dimensional vectors, and the distances between positions are measured by the Euclidean distance. We can use the Nearest Neighbor (NN) query to find the nearest user to us. However, NN query does not help to find communities.

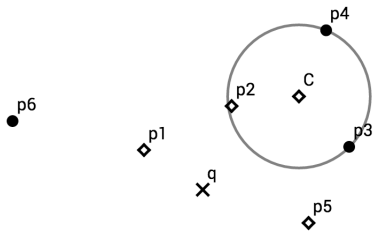

Figure 1 An example of NHH query

Compare to search for only one nearest object, finding a specific community must ensure the relationship between each object within the community. Because of the difference in searching for communities, a group version of NN query must be considered. In fact, the similar search algorithm has been proposed by [1], in which the Nearest

Neighborhood (NNH) search is managed to find a nearest group with k points to query point. Figure 1 shows an example of NNH query.

Figure 1 also shows the difference between NN query and NNH query. Suppose that p1 is the nearest point to q and k=3, 3-NN query will find the 3 nearest points to q, so it will return p1, p2 and p5 as the answer. On the other hand, NNH query will return C, which is the center of ρ-radius circle enclosing three points p2, p3 and p4. Therefore, NNH query is more likely a group version of NN query and combine with k nearest neighbor (k-NN) search.

However, the original NNH query has a problem. NNH query finds k-points cluster by a particular sized circle, and the radius of circle is fixed. It means that if the radius is not large enough, we will not get the answer. If we still want to get an answer we need to modify radius to match enough points. Even though we have enlarged the radius, we can not guarantee the enlarged circle can exactly enclose k points. For example, in Figure 2, there are several points and a defined ρ-radius circle. Obviously, circle c with solid line is not large enough to enclose three or more points. If the user still wants to get answers, they need to enlarge it, then the user will get circle c1 with dot line, but it is too large and enclosed more than three points, so it is not an exact answer. Therefore if the user want to get an exact answer with k points, they should modify the radius again and again.

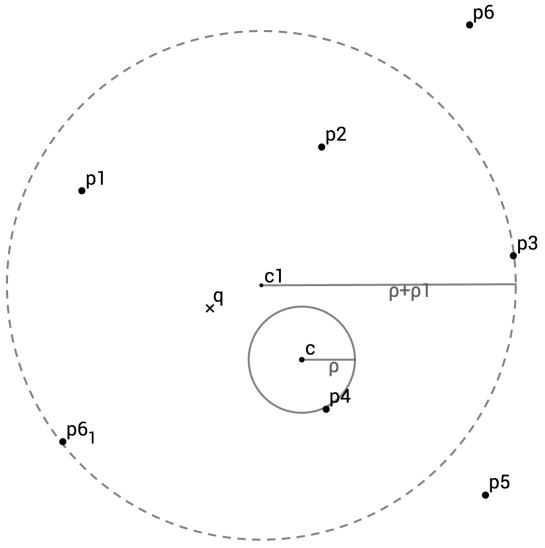

Figure 2 Problem of NNH query(k=3)

Unfixing the circle's radius is a method to resolve this problem. But if the radius of the circle is unfixed, then it is difficult to decide the best answer simply by the distance to the query point. In this paper, for resolving this problem, we propose a new approach, named Balanced Nearest k-Cluster (BNkC) query, and present a new criterion to evaluate which is the best answer among different sized circles. And for extending, the best answer can be changed according to user preference by modifying a weighting parameter.

2. Applications

The BNkC query is improved from NNH query, therefore the application of BNkC query is similar to NNH query. It can play an important role in the applications employing spatial database.

Mobile Social Networks. In a location-based mobile social network [2], the communities are also location-based, therefore we can use BNkC query to find an exact k persons community from the subscribers near to us. We also can get a different answer according to our preference by setting different weighting parameters. For example, by setting a particular weighting parameter, we can get a closer community, even though it has larger size.

Clustered kNN Search. The BNkC can be used to search the nearest k objects group. Assume that we want to build a processing plant, and we need to make sure that the processing plant is not only near to our official store but also near to each material plants(the number of material plants is more than two), Therefore we can use BNkC to find the center of material plants group so that it is the nearest group to official store.

3. Related Works

As described above, there are two related works to our problem, the first one is the Nearest Neighbor search (NN, k-NN) [3]. Another one is the Nearest Neighborhood (NNH) query [1].

Nearest Neighbor Query. The NN query returns the nearest point from the dataset. The most common approach is to store data in a spatial index and Find the nearest point by the branch and bound methodology. For example, if we use R-tree as a spatial index, we just need to choose the nearest minimum bounding rectangle(MBR) repeatedly, then retrieve the nearest data.

Nearest Neighborhood Query. The NNH query returns the center of $\rho-$radius circle that location is the nearest to query point and enclosed at least k points. The outline is preparing some three-point smallest enclosed circles as candidate circles which can enlarge to

eligible circles, then enlarging each of them to a k points enclosing circle but the radius is not greater than ρ . Last we can get the best answer by comparing their distance to query point.

NNH query searches the nearest circle to query point, and there are also two sub problems. The first one is the Smallest Enclosed Circle [4]. Another one is the Nearest Enclosed Circle problem [1]

Smallest Enclosed Circle Problem. The Smallest Enclosed Circle (SEC) problem [5] addresses how to compute the smallest covering circle with given points set. In practice, by using randomized incremental construction (i.e.incremental construction: Add points one by one and maintain the solution so far) this problem can be solved in a linear time. [6] It means that we randomly choose two points and make a SEC first, then add the rest points one by one in a random order. The algorithm will be described in the section 5.

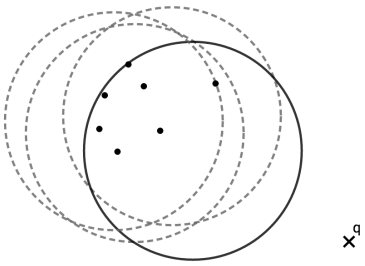

Figure 3 An example of nearest enclosed circle problem

Nearest Enclosed Circle Problem. As Figure 3 shows, the Nearest Enclosed Circle (NEC) problem addresses how to locate the Nearest covering circle when k-points eventually found. For example, in Figure 3, there are lots of circles enclosing k points, but only the circle with solid line is the nearest circle to q, hence, be the answer of the NEC problem. Similar to SEC problem, The NEC problem can be solved in a linear time as well.

4. The Proposal Solution

NNH query does not guarantee k points enclosed by a given ρ, so we propose a new solution. As stated in introduction, the most straightforward way to resolve this problem is unfixing the radius ρ. Therefore the most important issue is how to determine the best answer from different sized circles.

Like in Figure 4, there are three circles in the figure. We know that the best answer is circle C3 because it has the nearest location and the smallest size. However, which is the best answer between circle C1 and circle C2. We

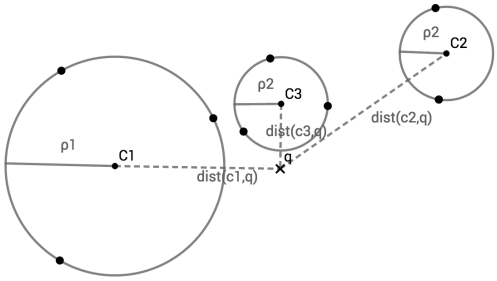

Figure 4 Comparison of different sized circle

cannot compare them simply, because circle C1 is closer than C2, but it has bigger size. On the other hand circle C2 has a smaller size, but it is further than C1. One solution is to compare the sum of distance with radius. By comparing the sum, we evaluate both distance and radius, and the purpose becomes to find the smallest and the nearest circle.

Further more, since we have two evaluation conditions, it hints us to consider about the user preference. In a specific case, a smaller circle will be a better choice, but sometimes a closer circle is preferred. It leads us to add a weighting parameter, rather than sum them simply.

Evaluation Formula . The candidate cirlces will be evaluated by the following expression:

$$\Delta = \alpha dist(c, q) + (1 - \alpha)\rho \quad (0<\alpha<1) \qquad (1)$$

Where dist(c,q) is the distance from q to the center of circle, and the ρ is the radius of the circle."

The formula means that the answer preference is determined by α. For example, if $\alpha > 0.5$ then the circle will be mainly evaluated by the distance from center to query point, and the answer we get will be a circle more closer to query point.

Cluster Definition. In fact, we have many ways to define the center of the cluster, and there are two typical ways. The first one is simply define it as the center of gravity. It means that we just need to calculate the average of k points, and the answer point will be the center of the cluster. As for the radius, it can be calculated by the distance from the center to the furthest point. By this way, we get more high density around the center, and more simple calculation procedure. Another way is to define it as the center of the smallest enclosing circle(SEC) with k points. It means when k points found, we need to calculate its smallest enclosed circle and use its radius as cluster's radius. Since our problem is managed to find a cluster not only near to our query point, but also make sure that the center of cluster to each point is the short-

est, this paper will choose the SEC approach intuitively. And another definition will be a new approach as a future work.

Balanced Nearest k-Cluster Query. With the evaluation method and cluster definition of Balanced Nearest k-Cluster(BNkC) query, our problem is defined formally as follows:

[Definition 1] "Given a set O of points, query point q, a positive number k and a weighting parameter α. The BNkC query returns the Smallest Enclosed Circle C, which has minimum Δ and enclosing k points, where Δ is as defined in 1"

Reduce Search Region. The BNkC problem can be implemented easily by a naive approach. But naive search traverses each point and calculates each SEC enclosing them, which takes huge time to process. Our solution is to enhance the efficiency by reducing the search region.

In fact, the BNkC search area is affected by α. To build a bound for reducing search region, we transform the evaluation expression of Δ to

$$\frac{\Delta}{\alpha} \geqq dist(c,q) + \rho \quad (0 < \alpha \leqq 0.5) \qquad (2)$$

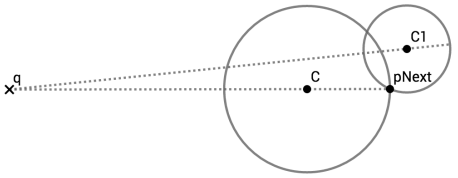

Figure 5 Two circles enclosed pNext

[Lemma 1] When $(0 < \alpha \leqq 0.5)$, given two circles C_1 and C_2, if $dist(C_1,q) + \rho > dist(C_2,q) + \rho$, then $C_1.\Delta > C_2.\Delta$.

The lemma implies that for two circles, if $dist(c,q) + \rho$ of one circle is greater than that of another, then the Δ of it is also definitely greater. By this lemma, we can reduce the search region by calculating the Δ of a circle then use it as the upper bound. Any data point with distance to q is larger than the value will be discard because it cannot be included in any circle better than the current one.

Assume that the next nearest point we need to retrieve as pNext, before we retrieve its top k-1 nearest points, we cannot know the circle enclosing it, but we can know the minimum $dist(c,q) + \rho$ of the circle enclosing it. As showed in Figure 5, the minimum $dist(c,q) + \rho$. It equals to the distance from pNext to q. Because of lemma 1, we know that c1's Δ is greater than c, even though c1's radius is shorter than c. It means that when we retrieve pNext, if its distance to query point is greater than $\frac{\Delta}{\alpha}$ (Δ is from the current circle) the Δ of the circle enclosing pNext will be greater than before.

Obviously, the bound is determined by α. When α =0.5 The formula 2 has minimum solution, therefore we can obtain the most efficient bound, and if α infinitely close to 0, the bound will not work no longer. In fact, it is natural that when α is close to 0, then we only evaluate circle by radius of circle, therefore the algorithm must search all of region to get answer.

In above, we discuss the reducing of search region when $\alpha \leqq 0.5$, when $\alpha > 0.5$ we choose the minimum of α and $1 - \alpha$, so that we can reduce search area when $\alpha > 0.5$ same with $\alpha \leqq 0.5$.

MBR with query bound. We retrieve each top-k nearest point one by one as pNext, and skip the point that is greater than query bound. However retrieving only one point is not an efficient way. Therefore we design a way to retrieve a group of points and skip them

We know that the R-tree stores the data points in minimum bounding rectangle(MBR), and we can retrieve the data points from these. When we want to find the nearest data from the R-tree [7], we just need to find the nearest MBR repeatedly and retrieve the nearest point from each MBR. If the minimum distance(the minimum distance from query point to MBR) of one MBR is greater than a specific value, the distance from the data points in MBR is also definitely greater than it. By using this property of the R-tree, we can skip the MBR before we retrieve the data in it.

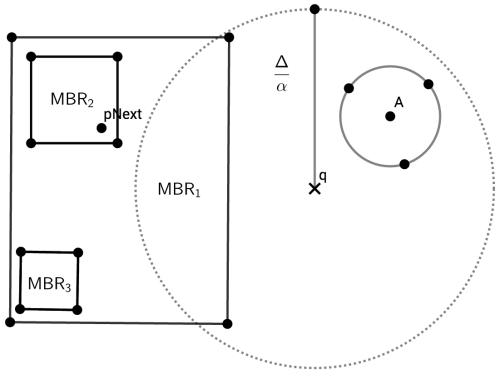

Figure 6 Filter MBR with query bound

Back to our problem, just like figure 6 shows that there are several MBRs, and a query bound we found before. If we want to retrieve the pNext, we will find MBR_1 first,

then we will get MBR_2 and MBR_3 in MBR_1. For the last, we can get pNext from MBR_2 by choosing the nearest MBR from MBR_2 and MBR_3. However, we find that MBR_2 and MBR_1 is out of query bound. According to Lemma 1 and the property of the R-tree, the points in MBRs will not be included by better circles than the current result. Therefore we can skip them directly rather than check the points one by one.

By this approach, we can save the processing time from extending the MBRs which are out of query bound. However we still need to extend the MBRs which are in the query bound or overlap with query bound. Therefore this approach can improve the performance of BNkC significantly in some specific cases.

5. The BNkC Algorithm

The first algorithm(algorithm 1) is designed for calculating SEC of k points set, it can be solved in expected linear time by using randomized algorithm.

Algorithm 1 Smallest Enclosing Circle(SEC)

Input: P=$p_1, ..., p_n$ take the points in a random order
Output: c=smallest enclosing circle

```
c ← SEC(p1, p2)
for i←3 to n do
    if pi in or on C_{i-1} then
        C_i ← C_{i-1}
    else
        calculate smallest enclosing circle for p1,...,p_{i-1}
        with one point known (p_i)
return c
```

The main algorithm(algorithm 2) is used to solve BNkC query, it improves efficiency by avoiding naive search by a bound to filtering no-answer data. The time complexity is $O(knlogn)$.

The main idea of algorithm 2 is searching pNext from near to far, then searching the k-1 points to pNext. The SEC of k-1 points with pNext is calculated by the previous algorithm, and the Δ are compared. The process will be executed until the distance to pNext is greater than bound. As for the bound, it is equal to $\frac{\Delta}{\alpha}$, where the Δ is equal to the circle retrieved at the first time, and will be updated when we have a better one.

The algorithm 3 is improved by algorithm 2 with a MBR skipping. As above stated, we do not need to extend the MBR whose minimum distance is greater than query bound.

The main idea of algorithm 3 is skipping the MBR whose minimum distance is greater than query bound, so

Algorithm 2 Balanced Nearest k-Cluster Query

Input: O= a set of points, q= the query points, k=a positive number, alpha←a real number between 0 and 1
Output: C=the circle of minimum Δ

```
minimumΔ ← ∞, bound← ∞, minialpha←mini( α, 1−α)
pNext← a nearest point to q
pSet ← top k-1 nearest point to pNext with pNext
C← SEC(pSet)
minimumΔ ← C.Δ
bound← c.Δ/α
while bound>dist(pNext,q) do
    pNext←the next nearest point
    pSet ← top k-1 nearest point to pNext with pNext
    Ctempt← SEC(pSet)
    if Ctempt.delta<minimumΔ then
        minimumΔ ← C.Δ
        C←Ctempt
return C
```

Algorithm 3 Balanced Nearest k-Cluster Query skip with MBR

Input: O= a set of points, q= the query points, k=a positive number, alpha←a real number between 0 and 1
Output: C=the circle of minimum Δ

```
minimumΔ ← ∞, bound← ∞, minialpha←mini( α, 1−α)
pNext← a nearest point to q
pSet ← top k-1 nearest point to pNext with pNext
C← SEC(pSet)
minimumΔ ← C.Δ
bound← c.Δ/α
while bound>minidist(MBRnext,q) do
    pNext←the next nearest point from MBRnext
    pSet ← top k-1 nearest point to pNext with pNext
    Ctempt← SEC(pSet)
    if Ctempt.delta<minimumΔ then
        minimumΔ ← C.Δ
        C←Ctempt
return C
```

that we can skip the MBRs out of query bound instead of skip the points one by one.

Preliminary Experiments. For preliminary experiments, we implement the algorithm with C++ programing language.

Experimental Results. We generate 100,000 data points randomly following uniform distribution. We compare the processing with same $k = 8$ and different weighting parameter α.

From Figure 6 we know that the algorithm still takes lots of time to process, depending on α. The processing time is depend on how much data the algorithm can skip. As described above, data with the dist(pNext, q)

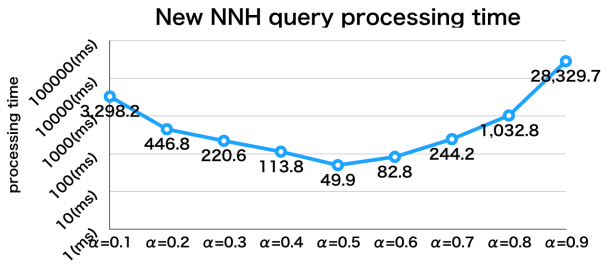

Figure 7 Processing time(ms) with different weighting parameter

greater than the current bound are skipped. Currently the tightness of the bound depends on α. Therefore, a more efficient and reasonable bound estimation good for various α is requested.

6. Experiments

In this section, we introduce formal experiments. Since our algorithm do not have related works to compare with, we show that the effect of some basic parameters, and the effectiveness of our improvement technique.

First of all, we want to introduce the algorithms which used to compare. There are three algorithms the algorithm of naive approach, the algorithm of region reduce with points, and the algorithm of region reduce with MBRs. Since naive approach(brute force) process in such a huge time, we only compare it with other two approach in a small dataset, then comparing the processing time of latter two algorithms with different weighting parameter α and different parameter k.

Experiment Environment . The algorithm is implemented on a computer system of 2.3GHz Intel Core i7 processor, with memory 16GB Mhz DDR3, running on MacOS 10.

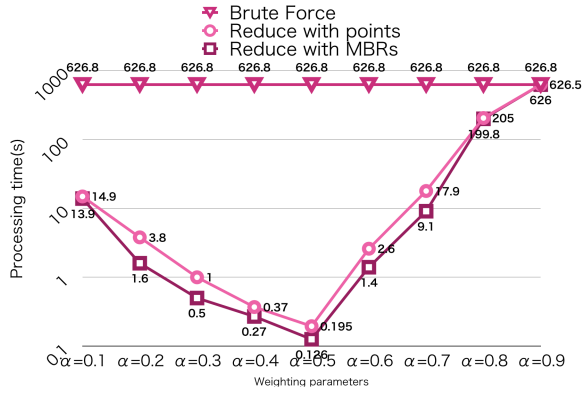

Figure 8 Effectiveness of region reducing

Effectiveness of region reduce. The experiment in figure 8 shows the effectiveness of region reducing method, we generate 3,000 random points between 1000 and 2000. The first will execute with different α and same $k = 30$.

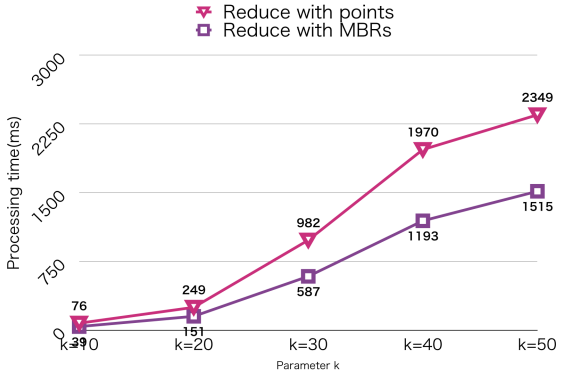

Figure 9 Effect of parameter k

Effect of parameter k. The experiment in figure 9 shows the parameter k how to effect the processing time of BNkC, and the experiments only compare the algorithm region reduce with points and region reduce with MBRs. Because brute force algorithm will execute in more 40minute, we only check the effect of region reduce approach. It is processed with 100,000 random points and weighting parameter is equal to 0,5 which have the most efficient query bound, the graph is figure 9:

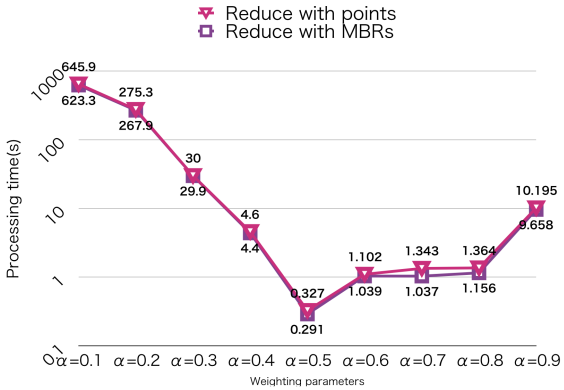

Figure 10 Effect of Weighting parameter

Effect of Weighting parameter. The experiment in 10 is similar to preliminary experiment, but it is based on a dataset from [8] which is named Letter [8], and it has a clustering distribution. It processes with 160,000 points of interest, and it compares the processing time with different weighting parameters and parameter $k = 10$, the

graph is showed in figure 10:

Effect of Query point. The experiment in figure 11 is also based a dataset from [8], which is named birch3 [8], and it is also a clustering dataset. But different from before, this experiment is checking a potential effective parameter, that is the location of query point, we check the distance τ, which is the distance from query point to answer eventually found, how to effect the processing time. The volume of dataset is 100,000, and we compare the processing time with same weighting parameter, same parameter $k = 100$ and different query point, from near to far. the graph shows as figure 11

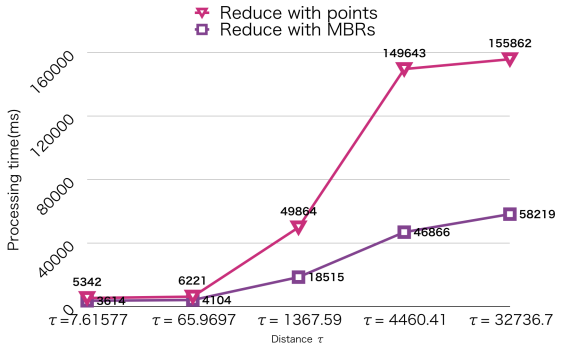

Figure 11 Effect of Query point

The experiment of figure 8 addresses the effectiveness of region reduce. As above stated, when weighting parameter equals to 0.5, we can have the most efficient query bound, and it has been proved in figure 8. However it still has a short coming, that is when weighting parameter close to 1 or 0, the query bound can not filter so efficiently.

The experiment of figure 9 shows the effect of parameter k, and k leads a long processing time. However when k is greater, MBR skipping approach can work efficiently.

The experiment of figure 10 measures the effect of weighting parameter. From figure 10, we can find that same with before we can have the most efficient query bound, and when it close to 0 or 1, the processing speed will slow down. But different before, in this experiments the processing speed of $\alpha = 0.1$ is slower than $\alpha = 0.9$. It is because wether the query bound can filter efficiently is depend on the distribution of dataset.

The experiment of 11 is conducted by a potential parameter which also effect the performance of BNkC. We show the result in figure 11 .Farther query points has a slower speed. However MBR skipping approach has more efficient performance than points skipping approach.

Conclusion for the experiments we find that region reduce is necessary for our algorithm, it can improve the performance with different weighting parameters. We can also find that MBR skipping approach can improve the performance in a large k case or the case of a far query point.

7. Conclusion and Future Works

We address that NNH query needs to match answer by a user defined radius circle, and in some specific cases, it needs to modify the radius to get answer, therefore we unfix the radius of circle .We also resolve the problem of answer evaluation among different sized circles. We propose a new evaluation corresponding to the new BNkC problem, then designed and implemented an algorithm which reduces search region and enhanced the efficiency. We develop a new way to skip points, that is skip points by MBR, by this method we can skip the MBR whose minimum distance is greater than query bound, and discard all points in it. By doing some experiments, we find MBR approach can improve the performance of BNkC in some specific cases, such as high cardinality cases or far query point cases.

As a future work, there are some problems need to be solved. We need to implement the BNkC query with different center. Though the SEC approach has a good property that the cluster center to each data point is the nearest, it takes time to calculate the circle. As for the simple center of points (calculate the average of points) can compare the density of each cluster. Recently, we cannot determined which one is better way .

For the search region reducing, we only find the property when $\alpha \leqq 0.5$ of evaluation formula. As for the case of $\alpha > 0.5$ we want to find a new method to optimize it.

Developing a brand new index to optimize our algorithm is also necessary for us. Since our algorithm is based on a spatial index R-tree, then calculate the circles one by one. This is not an efficient way. In Lemma 1, we have figured a bound which is dependent to α. Therefore, if we can develop a new index to retrieve the circle directly we can improve the performance of the case in which query bound cannot work so efficiently, such as $\alpha = 0.9$ or $\alpha = 0.1$

References

[1] Dong-Wan Choi and Chin-Wan Chung. Nearest neighborhood search in spatial databases. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 699–710. IEEE, 2015.

[2] Robert Lübke, Daniel Schuster, and Alexander Schill. Mobilisgroups: Location-based group formation in mo-

bile social networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 502–507. IEEE, 2011.

[3] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *ACM sigmod record*, pages 71–79. ACM, 1995.

[4] Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry: Introduction*. Springer, 2008.

[5] Alon Efrat, Micha Sharir, and Alon Ziv. Computing the smallest k-enclosing circle and related problems. *Computational Geometry*, 4(3):119–136, 1994.

[6] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). *New results and new trends in computer science*, pages 359–370, 1991.

[7] King Lum Cheung and Ada Wai-Chee Fu. Enhanced nearest neighbour search on the r-tree. *ACM SIGMOD Record*, 27(3):16–21, 1998.

[8] Pasi Fränti et al. Clustering datasets, 2015.