DEIM Forum 2022 J23-2

A Study on Dense Nearest Neighborhood Query

Hina SUZUKI†, Hanxiong CHEN†, Kazutaka FURUSE††, and Toshiyuki AMAGASA†

† University of Tsukuba,
1–1–1 Tennodai, Tsukuba-shi, Ibaraki, Japan
†† University of Hakuoh,
2–2–2 Ekihigashidori, Oyama-shi, Tochigi, Japan
E-mail: †suzuki.hina.ss@alumni.tsukuba.ac.jp, {chx, amagasa}@cs.tsukuba.ac.jp, ††furuse@fc.hakuoh.ac.jp

Abstract Nearest Neighbor (NN) query is principal factor in applications that handle multidimensional vector data, such as location-based services, data mining, and pattern recognition. Meanwhile, Nearest Neighborhood (NNH) query finds neighborhoods which are not only near to the query but also dense. However, it cannot find desired groups owing to strong restrictions such as fixed group size in previous studies. Thus, Dense Nearest Neighborhood (DNNH) query without strong constraints, and three efficient algorithms to solve the queries are proposed in this thesis. The proposed methods can efficiently find a solution by reducing unnecessary processing using a filtering threshold and expansion breaking criteria. Experiments on various datasets confirm the effectiveness.

Key words Nearest neighborhood query, Spatial database, Information retrieval, Grid index

1 Introduction

In multi-dimensional vector data such as spatial databases, a nearest neighbor (NN) query is a fundamental and important query in many fields, such as data mining and information retrieval, and it is widely used in various applications, such as services using pattern recognition, facility information, and map and navigation services using location information.

Many neighbor queries have been developed from the NN query. Sometimes users want to search a "dense group" of neighboring points quickly. Examples are as follows:

- A tourist who wants to visit several stores without moving too much
- A user who wants to find a social networking community whose hobbies and interests match his own
- A user who wants to identify an accident-prone area near a school

Figure 1 shows an example for some neighbor points from a given query q. In (b), four points nearest to the query are searched, which are obtained by repeating the single neighbor point search four times. As indicated in this example, searched points may be scattered in the data space. In (c), a dense group including four points is searched. This paper addresses the latter type of searches.

Among the many studies on the efficiency and extension of the NN query, the most relevant queries for this type are the nearest neighborhood (NNH) query [1] and the balanced nearest neighborhood (BNNH) query [2], which are explained in detail in Section 2. The BNNH query is an extension of the NNH query that solves the query of the empty output owing to the strong constraints. However, the remaining constraints of the BNNH query interfere with users obtaining the desired output.

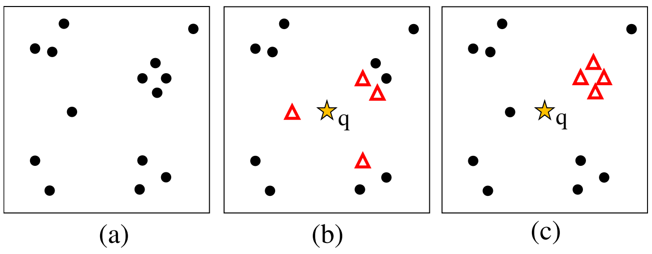

Figure 1: Dense neighborhood. (a) Data Points, (b) 4-NN of query q (data points that are emphasized as triangles), (c) A Dense neighborhood of q

In this paper, a novel flexible query **dense nearest neighborhood (DNNH)**, releasing the above constraints, and three algorithms for solving the queries are proposed. One is an intuitive method that uses clustering techniques, and the other two algorithms provide faster search by exploiting the filtering thresholds and expansion breaking criteria for group retrieval. To verify the usefulness and the efficiency of the proposed methods, experiments on large datasets were conducted and the preformance of the proposed methods compared.

2 Related Works

Nearest Neighbor query starts with the most basic (k-)NN query, which searches for the (k) points closest to the query point, and has been studied in many ways to improve its efficiency and extension [3–8].

The queries that are most relevant to the search for dense neighborhood groups are the NNH query [1] and the BNNH query [2]. When each of these queries is applied to the dataset in Figure 1(a), the candidate groups are as shown in (a), (b) and (c) of Figure 2.

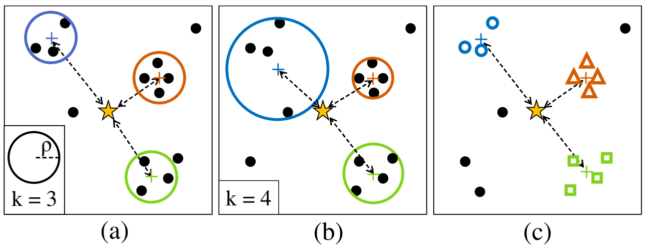

Figure 2: NNH, BNNH and DNNH. (a) NNH, specifying $k = 3$ and a fixed radius ρ. Circles with a smaller ρ may not guarantee 3 points contained. (b) BNNH, specifying $k = 4$. The radius ρ is changeable. May enlarge the circles to guarantee 4 points contained and gives up the density. (c) DNNH.

NNH query. The NNH query [1] outputs a circle having smallest distance between the center of it and a query point among the circles that contain more than a specified number (k) of points within the circle of a specified radius (ρ). As shown in Figure 2(a), it is possible to obtain the desired group (triangle mark points in Figure 1(c)) by specifying the appropriate parameters. However, in reality, it is not always the case that more than k points are found within the circle of fixed ρ, which implies that the query obtains an empty result if the appropriate parameters cannot be specified. Estimating the appropriate parameters in advance is particularly difficult for large datasets.

BNNH query. The BNNH query uses a circle of variable radius to guarantee that it contains the specified number (k) of points. This is implemented by finding the circles minimizing the evaluation $\Delta(C, q)$ expressed by the following equation:

$$\Delta(C, q) = \alpha\|q - c(C)\| + (1 - \alpha)\rho$$

Here, $c(C)$ is the center of C, and ρ is the radius of C. $\|q - c(C)\|$ is the Euclidean distance between q and $c(C)$. α is a value for $0 < \alpha < 1$, where the closer to 0 the value is, the greater the weight of the cohesion is, and the closer to 1 it is, the greater the weight of the distance to q is. Unlike

Table 1: Notations and symbols

Symbols	Description
p, P	point, point set
q	query point
$C, \mathbb{C}$	cluster, cluster set
$c(C)$	the centroid of cluster C
α	sigma rule coefficient
$sd(C)$	standard deviation of cluster C
p_{next}	next expansion point
$pd_{mean}(C)$	mean of pairwise points distance in C
$nd_{mean}(C, p_{next})$	mean of distance between p_{next} and $p \in C$
$\pi_e(C, p_{next})$	the expandability of group C with p_{next}
$\Delta_e(C)$	the evaluation function of expanding C
n	grid size in grid expanding

NNH queries, BNNH queries allow to enlarge the circle to guarantee k data in the answer.

However, there is still the problem that the group circle will not be dense unless an appropriate k is specified. As an example, the candidate groups in the sample dataset (Figure 2(b)) by BNNH with varying parameters are shown in Figure 3. The best dense nearby group is the group C with parameter $k = 4$, as shown in (a). However, in (b), for parameter $k = 5$, it returns the group C', which is neither dense nor close to the query q.

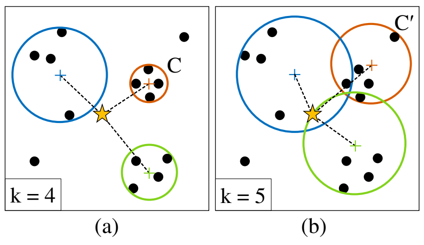

Figure 3: The candidate groups by BNNH with varying parameters. (a) $k = 4$, (b) $k = 5$

Therefore, the BNNH query strongly depends on the parameter k, and there is a high possibility that the desired dense nearby group cannot be obtained unless an appropriate value is given. Nevertheless, it is difficult to estimate the value depending on the dataset and query location in advance. To address the problem that neither NNH nor BNNH guarantees that the answer groups found are dense, the proposed DNNH finds a dense group without specifying the parameters k and ρ. DNNH queries are more flexible than BNNH queries because they compare dense groups that are retrieved regardless of the number of points in the group.

3 Dense Nearest Neighborhood Query

The notations used in this thesis are shown in Table 1.

Definition 1. *(Dense nearest neighborhood query, DNNH). Given a set of points P and a query point q, the DNNH query returns a group C that minimizes the value of $\Delta(C, q)$.*

The total degree of approximation Δ is defined by the following equation(注1):

$$\Delta(C, q) = \|q - c(C)\| + \mathrm{sd}(C) \quad (1)$$

Here, $c(C)$ is the centroid of C, and $\mathrm{sd}(C)$ is the standard deviation of C. Compared with the previous studies that used the center and radius of a circle, this method can accurately represent the variation of points in a group. Moreover, DNNH overcomes the disadvantage of BNNH such that the searched groups are not necessarily dense, because it does not require the number of points in each group.

The most difficult part of solving a DNNH query is to efficiently retrieve dense groups. For example, if considering the case of finding a dense group including a point p, BNNH queries only need to perform a number of NN searches based on specified k for p. Meanwhile, in a DNNH query, the number of data in a group is not specified; thus, it needs to find a dense group from a large number of combinations including p, and it is NP-hard to find the optimal solution. To overcome the difficulty, three approximate for implementing efficient DNNH search solutions by heuristics — the one is by clustering and the others are by expanding — are proposed in this paper.

3. 1 Clustering-Based Approach

In intuitive methods, datasets are divided into clusters, and the nearest cluster is retrieved as the DNNH answer. The simplest approach for solving the queries is the method calculating Δ after clustering all the points. It is important to note here that the DNNH query is parameter free; therefore, clustering should not take redundant parameters as well.

X-means [9] is an extension of k-means and is characterized by the fact that it can estimate the number of clusters and perform clustering simultaneously without the need to specify the number of clusters in advance, by using the recursive 2-means partitioning and the stopping criterion based on the information criterion BIC. BIC is calculated by the likelihood function and the size of a dataset, intuitively telling whether it is likely to split a set into two. In this paper, the algorithm improved by Ishioka [10] was used. This algorithm differs from the original one by Pellog and Moore in that it considers the possibility that the variance differs among clusters, and it uses approximate computation for some calculations to enhance the efficiency. The pseudocode is shown in Algorithm 1.

(注1): Note that the definition of Δ is not same as the one of BNNH.

Algorithm 1 X-means Clustering-based Algorithm

```
Input: $P$, $q$, $k_0$
Output: $C$
 1: $C, \mathbb{C} \leftarrow \phi$
 2: $C_1, C_2, ..., C_{k_0} \leftarrow k$-means++ $(p, k_0)$   // partition $P$ into $k_0$ clusters
 3: for each $C_i \in \{C_1, C_2, ..., C_{k_0}\}$ do
 4:     splitClusterRecursively($C_i$)
 5: for each $C_i \in \mathbb{C}$ do
 6:     if $\Delta(C_i, q) < \Delta(C, q)$ then
 7:         $C \leftarrow C_i$
 8: return $C$

 9: function splitClusterRecursively($C$)
10:     $C_1, C_2 \leftarrow k$-means++$(C, 2)$
11:     if $BIC(C) > BIC'(C_1, C_2)$ then
12:         for each $C_i \in \{C_1, C_2\}$ do
13:             splitClusterRecursively($C_i$)
14:     else
15:         Insert $C$ into $\mathbb{C}$
```

However, it is inefficient and time consuming to run the clustering algorithm whenever a single datum changes. To compare, an algorithm based on x-means was implemented in this paper, and using other clustering algorithms are believed to make no fundamental difference.

3. 2 Expanding-Based Approach

Our another approach attempts to retrieve groups from nearby the query q. Intuitively, points located near the query are more likely to form the result group. This approach is that retrieving groups from the query's neighborhood and filtering points that cannot be answers using the current degree Δ_e as a threshold. Using this approach, the following is briefly repeated: (1) Extract the query neighbor from the dataset, (2) retrieve the dense group to which the extracted point belongs (Section 3. 3, 3. 5), (3) update the threshold for filtering and remove the points that cannot be answers from the dataset (Section 3. 4).

3. 3 Evaluation Metric for Retrieving a Cluster

This section describes how the retrieval of a dense group C_p to which a point p belongs is performed. In this study, this is achieved by selecting a dense preferred group from among the groups obtained by expansion using an NN search. The problem is to select a group from the enlarged ones based on the criteria, which is addressed in this paper by designing and using the enlargement index Δ_e.

The Δ_e is calculated by

$$\Delta_e(C, p_{next}) = \Delta(C, q) \cdot \pi_e(C, p_{next})$$

where $\pi_e(C, P)$ denotes the expandability of group C in dataset P and is defined by the following equation:

$$\pi_e(C, p_{next}) = \frac{pd_{mean}(C)}{pd_{mean}(C) + nd_{mean}(C, p_{next})}$$

$$pd_{mean}(C) = \frac{1}{\binom{|C|}{2}} \sum_{p_i, p_j \in C} \|p_i - p_j\|$$

$$nd_{mean}(C, p_{next}) = \frac{1}{|C|} \sum_{p \in C} \|p_{next} - p\|$$

$$p_{next} = arg \min_{p \in P - C} \|c(C) - p\|$$

pd_{mean} is the average distance between the samples in the group, whereas nd_{mean} is the average distance between the candidate point p_{next} and the samples in the group. p_{next} is the point that has the smallest distance to the center $c(C)$ among the points not in C.

3. 4 Bounding the Expanding Group

This section describes the conditions under which a point p is removed using Δ of an already retrieved group C. Let C_p denote the group to which p belongs. If $\min \Delta(C_p, q) > \Delta(\exists C, q)$ holds, the group to which p belongs will not be preferred to the existing groups, and no further processing of p is necessary. Therefore, if $\min \Delta(C_p, q)$ can be derived from the information of p, whether the group is removed can be determined by filtering using $\Delta(C, q)$. However, in a DNNH query where the number of data in a group is not specified, Δ may be as small as possible depending on the distribution of the data, and it is difficult to determine the exact filtering threshold. Evidently, it makes no sense to find dense groups in a uniform distribution; therefore, the points of C_p are assumed to follow a normal distribution.

Let $arg \min \Delta(C_p, q)$ denote C_p^{min}. Based on the assumption that the data of C_p follow a normal distribution, its centroid and standard deviation can be approximated. The size and positional relationship of C_p^{min} can be illustrated as shown in Figure 4 which indicates that its centroid $c(C_p^{min})$ is located at the center of q and p by the assumotion.

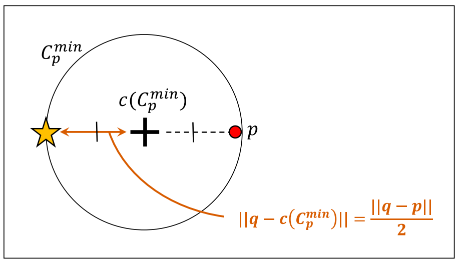

Figure 4: The size and positional relationship of C_p^{min}

In this case, $\min \Delta(C_p, q)$ can be calculated as follows:

$$\begin{aligned}\min \Delta(C_p, q) &= \|q - c_p^{min}\| + sd(C_p^{min}) \\ &= \frac{\|q-p\|}{2} + \frac{\|q-p\|}{2\alpha} \\ &= \frac{\alpha+1}{2\alpha}\|q-p\|\end{aligned}$$

The α indicates the sigma rule coefficient, and all points in group C_p are assumed to be within the radius $\alpha \cdot sd(C_p)$ from the centroid. According to the 68-95-99.7 rule of the normal distribution, about 95% and 99.7% of the points of C_p are within the radius $2sd(C_p)$ and $3sd(C_p)$ from the centroid, respectively. Therefore, $\alpha = 2$ or 3 seems to be effective. Substituting this into $minC_Delta(C_p, q) \leqq \Delta(C, q)$ leads to $lVertq - p\| > \frac{2alpha}{alpha+1} FilterDelta(C, q)$ and the following conclusion.

Theorem 1. *A point p locating further than a bound, that is, p which holds the following inequation:*

$$\|q - p\| > \frac{2\alpha}{\alpha+1} \min_C \Delta(C, q) \tag{2}$$

can be removed in the filtering process.

3. 5 Expansion Breaking Criteria

The bound given above is inefficient because it works only in the second and subsequent retrieval of clusters. For the computation of pd_{mean}, nd_{mean}, the first group retrieval always continues to expand until the entire dataset is included, and its complexity is $\mathcal{O}(|P|^2)$. This is a problem because the DNNH query does not specify the group size, which affects the efficiency.

The definition of Δ_e indicates that the group having the smallest Δ_e is desired, and the enlargement process can be stopped, which reduces the computational cost. Then in this paper, the following determines that Δ_e is small enough. Under the assumption that C to which p belongs follows a normal distribution, and they exist within the radius $\alpha \cdot sd(C)$ from the centroid c, when Eq. 3 holds for the expansion point p_{next} , which can conclude that further expansion is meaningless.

$$\|p_{next} - c\| \geqq \alpha \cdot sd(C) \tag{3}$$

In addition, in the latter half of the cluster retrieval, better answer is less likely to be obtained. Thus, further speed-up of the process can be achieved by terminating the expansion when the cluster is found to be less preferable than the current most preferable cluster C_{best} among the retrieved clusters. However, as mentioned in section 3. 4, because the DNNH query does not specify the group size, Δ may be as small as possible depending on the distribution of the data. For C of a certain size, it is reasonable to assume that $sd(C)$ monotonically increases with each expansion. Let C' be the cluster of C expanded an arbitrary number of times; then, $sd(C') > sd(C)$ holds by the assumption. Here, if

$$sd(C) > \Delta(C_{best}) \quad (4)$$

holds, then by the definition of Δ, $\Delta(C') > sd(C') > sd(C) > \Delta(C_{best})$ holds. This means that Eq. 4 can be used as expansion breaking criteria to stop expanding C.

3.6 Basic Expanding Algorithm

The basic method is shown in Algorithm 2. In this method, the points of dataset P are first sorted in the order of their distance from the query point q. The points are extracted from the sorted dataset in the order from the top (line 3), and groups are retrieved as described in Section 3.3, 3.5 (lines 4, 10-18). After the retrieval is finished, the points in the group are removed from P as already processed (line 5), the threshold *bound* is updated and filtered (lines 6–8), and if there are still unprocessed points, the group is retrieved again (line 2). The process is terminated when there are no more unprocessed points.

Algorithm 2 Basic Expanding Algorithm

```
Input: $P$, $q$, $\alpha$, $k_{min}$, $k_{max}$
Output: $C_{best}$
1: $bound \leftarrow \infty$
2: while $P$ is not empty do
3:     $p \leftarrow$ nearest point $\in P$ from $q$ that is nearer than $bound$ in Eq. 2
4:     $C \leftarrow$ RetrieveCluster($p$, $k_{min}$, $k_{max}$)
5:     $P \leftarrow P - C$
6:     if $\Delta(C) < \Delta(C_{best})$ then
7:         update $bound$ of Eq. 2
8:         $C_{best} \leftarrow C$
9: return $C_{best}$

10: function RetrieveCluster($p$, $k_{min}$, $k_{max}$)
11:     $C \leftarrow \{p\}$, $C_{best} \leftarrow C$
12:     while $P$ is not empty do
13:         $p_{next} \leftarrow$ nearest point $\in P$ from $c$
14:         if $|C_{best}| = 1 \lor \Delta_e(C, p_{next}) < \Delta_e(C_{best}, p^{best}_{next})$ then
15:             $C_{best} \leftarrow C$
16:         $C \leftarrow C \cup p_{next}$
17:         if Eq. 4 is satisfied $\lor$ $|C| \geqq k_{max}$ $\lor$ ( $|C| \geqq k_{min}$ $\land$ Eq. 3 is satisfied ) then break
18:     return $C_{best}$
```

3.7 Grid Expanding Algorithm

The problem of the basic method is that the entire process, from indexing to filtering of the dataset, is point-based, which is inefficient. A grid-based method can overcome this by prioritising the retrieval from the neighbours of the query and further reducing the search space for NN queries. The outline of this method is illustrated in Figure 5. The figure shows an example of a grid that divides the space into 4 × 4 cells. The grid structure allows us to directly refer to the points in the cells, thereby enabling us to achieve a more efficient refinement of the search space in the NN search and coherent filtering process for each cell.

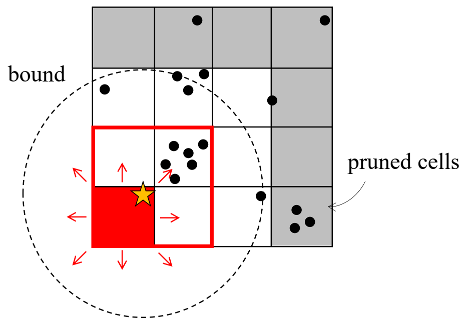

Figure 5: Grid-based algorithm

4 Experiments

Various experiments were conducted for verifying the efficiency and the accuracy of the proposed methods. All algorithms for the solutions presented were implemented in C++ using GNU C++ Compiler 10.3.0. The experiments were conducted with the following specifications: Windows 10 Home, with a 3.4 GHz 4-Core Inter Core i7 processor and memory of 16GB 2133 MHz DDR4.

4.1 Settings

The real data used in the experiments were NE (123,593 data), RR (257,942 data), and CAS (196,902 data) were provided by the U.S. Census Bureau's TIGER project(注2). In addition, uniform random data UN (10,000–200,000 pts) and a cluster dataset RN was prepared to measure the correspondence to the datasets with various distributions. RN is a composite dataset of randomly generated points that follow the standard normal distribution and are scaled and arranged equally in space as clusters. In order to measure the accuracy effectively, S1, S2, S3, S4 [11], R15, D31 [12] and Aggregation [13] were also used, which are common datasets in clustering researches. Some of the datasets to evaluating accuracy are shown in Figure 6. All these datasets were two-dimensional and normalized to $[0, 1]$ (however, since the x-means clustering-based algorithm depends on the data scale, the scale is adjusted to $[0, 100]$ to get the best possible result).

Efficiency experiments compare the performance of the three proposed methods (x-means clustering-based, basic expanding, and grid expanding), measuring the processing time

(注2) : http://chorochronos.datastories.org/?q=user/15/track

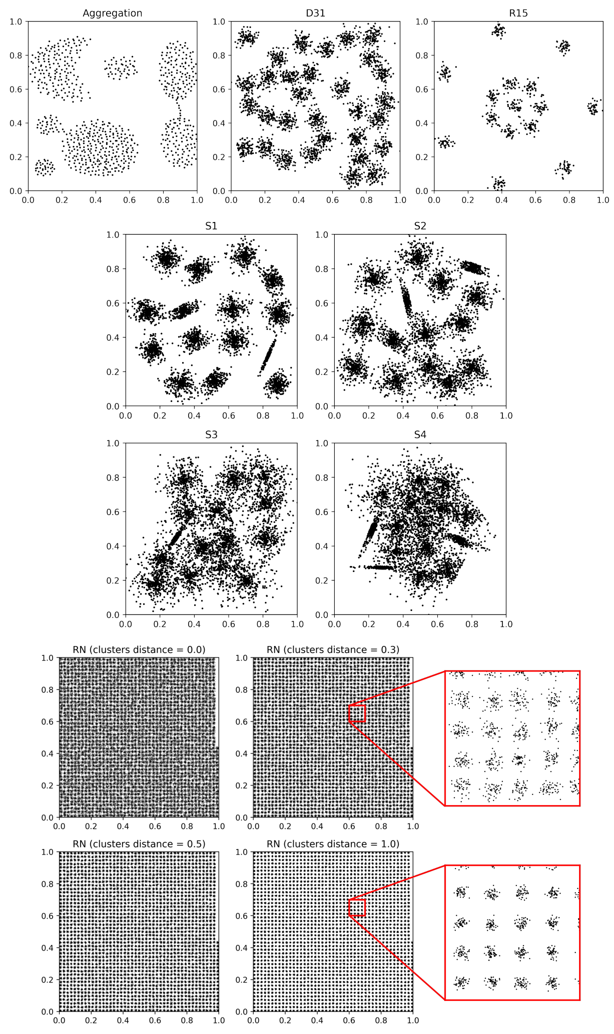

Figure 6: The datasets for evaluating accuracy

for real data, scalability, changes in cluster size $|C|$, α, and distance between clusters. In the grid expanding algorithm, the grid size n was varied and the appropriate value was investigated. Unless otherwise stated, $\alpha = 2$, $n = 50$, the distance between clusters $= 1.0$, and the cluster size lower limit $k_{min} = 10$. Accuracy experiments compare the precision, recall and f-measure on the x-means clustering-based and the grid expanding algorithm, assuming that the cluster having the smallest Δ is the correct answer. q was selected randomly from the dataset. All results are reported as the average for conducting DNNH queries 100 times.

4. 2 Results of Experiments on Efficiency

Performance for the real datasets. The results are presented in Figure 7. The results of the x-means clustering-based algorithm are labeled as "Xmeans"; the basic expanding and the grid expanding algorithm are "Basic" and "Grid," respectively; and the upper limit cluster size k_{max} is set to 30 and 100. As shown in this figure, the grid expanding algorithm is the fastest for all datasets. This is especially obvious with $k_{max} = 30$, which is up to 32 times faster than the x-means clustering-based. The basic expanding algorithm ($k_{max} = 30$) also shows higher performance, and it is only in some cases of $k_{max} = 100$ that the x-means clustering-based algorithm is faster.

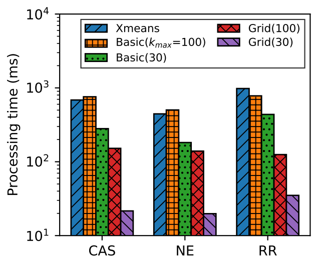

Figure 7: Performance for the real datasets

Effect of dataset size. The results are shown in Figures 8 and 9. As shown in this figure, the experimental results for the UN datasets (Figure 8) show that the processing time of the grid expanding algorithm increases more slowly than the other methods. In particular, it is clearly faster and the difference of the time between 150,000 and 200,000 points is almost negligible in $k_{max} = 30$, which indicates that the grid expanding algorithm is less sensitive to dataset size. The experimental results for the RN dataset (Figure 9) show that the basic expanding and the grid expanding algorithms are faster than the x-means clustering-based algorithm when the cluster size $|C| = 50$. Especially the grid expanding algorithm performs nearly 100 times faster than that, and the processing time also increases more slowly.

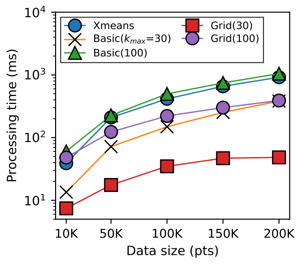

Figure 8: Effect of data size (UN)

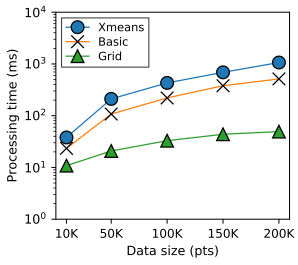

Figure 9: Effect of data size (RN) ($k_{max} = 200$; $|C| = 50$)

Effect of cluster size. The results are shown in Figure 10. From 10 to 100, the basic expanding and the grid expanding algorithms are from 10 to 1,000 times faster than the x-means clustering-based algorithm. However, the processing time of the basic expanding and the grid expanding algorithms increased with an increase in the cluster size and completely reversed when the cluster size was 500. For the x-means clustering-based algorithm, the decrease in the processing time as the cluster size increases can be attributed to the fact that the number of clusters in the entire dataset decreases owing to the fixed data size, which reduces the number of divisions by k-means and the amount of BIC computations.

Effect of the cluster distance. The results are shown in Figure 11. If the distance between clusters is x, there is an interval of x clusters between the clusters. Consequently, while the basic expanding and the grid expanding algorithms are from 2 to 10 times faster than the x-means clustering-based algorithm in many cases, the performance of the basic expanding and the grid expanding is deteriorated when the distance of the clusters under 0.3. This is because the expansion breaking criteria can no longer function due to the loss of distance between clusters, but it does function over that, indicating that the proposed expansion breaking criteria is effective even for small intervals.

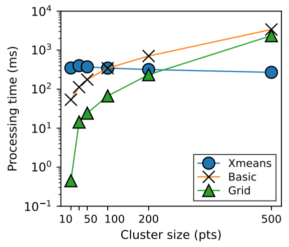

Figure 10: Effect of cluster size ($|P|$ = 100,000; $k_{min}=5$; $k_{max}=500$)

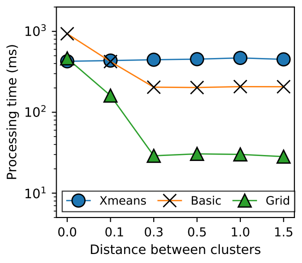

Figure 11: Effect of distance between clusters ($|P|$ = 100,000; $|C|$ = 50; $k_{max}=200$)

Effect of sigma rule coefficient α. The data size was 100,000, and the cluster size of the RN dataset was fixed at 50. The results are shown in Figure 12. Consequently, it is the fastest when $\alpha = 2$ or 3 especially in the grid expanding algorithm. The slowest speed is obtained when $\alpha = 1$ or 5 because it is quite small or large value that the expansion breaking criteria or filtering no longer works.

Effect of grid size n. For example, because the datasets are two-dimensional, when $n = 100$, the number of cells is $n^2 = 100,000$. The results are shown in Figure 13. The data size of UN and RN is 100,000 and the cluster size of RN is fixed at 50 (RN-50P) and 100 (RN-100P). The fastest processing time was obtained when n is from 10 to 30, and the time increased slowly. This is because, when the grid size becomes quite large, the cells become smaller than necessary, and the amount of search and the expansion processing of each cell increases.

4.3 Results of Experiments on Accuracy

The results are shown in Table 2. "RN-xS" denotes that the distance between clusters is x for the RN datasets. All values are rounded off to 4 decimal places. The overall average shows that the grid expanding outperforms the x-means clustering-based by about 23% in precision, and in particular, recall is about 55% more accurate. This is obvious especially in the S2, D31, Aggregation, and RN datasets when the distance between clusters is between 0.0 and 0.3, which is up to 4.7 times higher. This is because the x-means clustering-based tends to over-segment while the grid expanding is able to detect a suitable expansion stage. The performance of the grid expanding declines only when the x-means clustering-based also results badly for such as S3 and S4, which indicates that the grid expanding can realize more stable search.

4.4 Evaluation of the Proposed Methods

Overall, the grid expanding algorithm is the fastest. In particular, in comparison between the basic expanding and the grid expanding algorithms, the grid expanding algorithm is faster in all results, unless the data size is small or inefficient parameter settings (such as $\alpha = 5$) are used . Therefore, the grid expanding algorithm should be chosen unless there is concern about memory usage or the small overhead of building the cellular data. Depending on the distribution of the data, a grid size of approximately 10–50 is considered the most suitable for achieving a good balance between memory usage and processing efficiency.

However, depending on the distribution of the dataset and the purpose of the search, the x-means clustering-based algorithm may be a better choice. For example, the dataset may be sparsely distributed or the cluster size may be larger than 200. However, DNNH query considers finding a set of nearby facilities in a location-based service using a spatial database, or finding a set of objects with attributes similar to those of a particular user in a social network service (SNS). In these situations, the basic expanding or the grid expanding algorithm is preferable because it can rapidly detect small- to medium-sized neighborhood clusters.

In addition, the grid expanding algorithm is obviously superior to the x-means clustering-based algorithm in terms of accuracy. The grid expanding algorithm has about 55% higher recalls on average than the x-means clustering-based, and can cover more of the desired dense neighborhood. It is also highly versatile, as it can detect clusters with relatively stable accuracy even for datasets that the x-means clustering-based algorithm is not good at.

5 Conclusion and Future Work

In this paper, DNNH query and the efficient methods for solving the queries "the x-means clustering-based", "the ba-

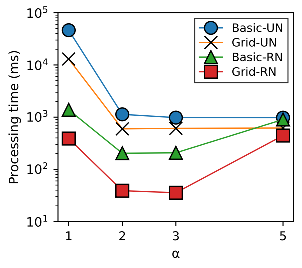

Figure 12: Effect of α ($|P|=100{,}000$; $|C|=50$; $k_{max}=200$)

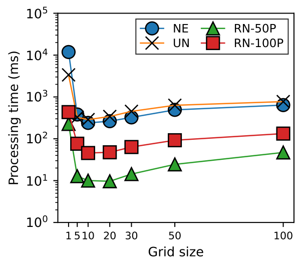

Figure 13: Effect of grid size n ($|P|=100{,}000$ (UN, RN); $k_{max}=200$)

Table 2: Accuracy of proposed methods ($k_{max}=400$)

	X-means Clustering-based			Grid Expanding		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Aggregation	0.886	0.182	0.253	0.871	0.863	0.862
D31	0.937	0.677	0.745	0.857	0.897	0.845
R15	0.984	0.957	0.967	0.988	0.9	0.939
S1	0.988	0.448	0.526	0.999	0.798	0.858
S2	0.944	0.299	0.393	0.959	0.736	0.809
S3	0.838	0.195	0.291	0.861	0.497	0.572
S4	0.747	0.278	0.35	0.788	0.386	0.458
RN-0.0S	0.512	0.564	0.5	0.895	0.83	0.843
RN-0.1S	0.51	0.57	0.475	0.965	0.898	0.921
RN-0.2S	0.62	0.594	0.565	0.996	0.935	0.962
RN-0.3S	0.726	0.677	0.671	1	0.942	0.97
RN-0.4S	0.689	0.645	0.631	1	0.939	0.966
RN-0.5S	0.653	0.667	0.62	1	0.925	0.958
RN-1.0S	0.691	0.674	0.661	1	0.947	0.972
Overall	0.766	0.531	0.546	0.941	0.821	0.853

sic expanding" and "the grid expanding" algorithm were proposed. DNNH query can flexibly find more desirable dense neighborhood without severe constraints, which cannot be achieved by the existing queries with the strong constraints. Among the proposed methods, the grid expanding algorithm is the fastest and the most accurate, and it can contribute to many applications that deal with large datasets.

For future work, the grid based method can be extended for diverse datasets such as high dimensional, non-normal distribution and over-lapping clusters. For example, an approach using density-based clustering methods such as DBSCAN [14] is possible to be used for group retrieval. It is also under consideration to parallel our algorithms and apply multi-threaded solution.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 19K12114.

References

[1] D.-W. Choi and C.-W. Chung, "Nearest neighborhood search in spatial databases," in *2015 IEEE 31st International Conference on Data Engineering*, pp. 699–710, April 2015.

[2] S. Le, Y. Dong, H. Chen, and K. Furuse, "Balanced nearest neighborhood query in spatial database," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 1–4, Feb 2019.

[3] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020.

[4] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, "Aggregate nearest neighbor queries in spatial databases," *ACM Trans. Database Syst.*, vol. 30, no. 2, pp. 529–576, 2005.

[5] K. Deng, S. Sadiq, X. Zhou, H. Xu, G. P. C. Fung, and Y. Lu, "On group nearest group query processing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 2, pp. 295–308, 2012.

[6] H.-J. Jang, K.-S. Hyun, J. Chung, and S.-Y. Jung, "Nearest base-neighbor search on spatial datasets," *Knowl Inf Syst*, vol. 62, pp. 867–897, 2020.

[7] I. Stanoi, D. Agrawal, and A. E. Abbadi, "Reverse nearest neighbor queries for dynamic databases," in *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pp. 44–53, 2000.

[8] X. Guo and X. Yang, "Direction-aware nearest neighbor query," *IEEE Access*, vol. 7, pp. 30285–30301, 2019.

[9] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *In Proceedings of the 17th International Conf. on Machine Learning*, pp. 727–734, 2000.

[10] T. Ishioka, "Extended k-means with an efficient estimation of the number of clusters," *Data Mining, Financial Engineering, and Intelligent Agents*, 2000.

[11] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognition*, vol. 39, no. 5, pp. 761–765, 2006.

[12] C. Veenman, M. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1273–1280, 2002.

[13] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *21st International Conference on Data Engineering (ICDE'05)*, pp. 341–352, 2005.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96(34): 226-231*, 1996.