Information Systems 92 (2020) 101530

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Efficient processing of reverse nearest neighborhood queries in spatial databases

Md. Saiful Islam [a,*], Bojie Shen [b], Can Wang [a], David Taniar [b], Junhu Wang [a]

[a] School of Information and Communication Technology, Griffith University, Gold Coast, Australia
[b] Faculty of Information Technology, Monash University, Melbourne, Australia

ARTICLE INFO

Article history:
Received 28 June 2019
Received in revised form 31 March 2020
Accepted 2 April 2020
Available online 13 April 2020
Recommended by Philip Korn

MSC:
00-01
99-00

Keywords:
Reverse nearest neighborhood
Nearest enclosing circle
Influence zone
Queries and algorithms

ABSTRACT

This paper presents a novel query for spatial databases, called *reverse nearest neighborhood* (RNH) query, to discover the neighborhoods that find a query facility as their nearest facility among other facilities in the dataset. Unlike a *reverse nearest neighbor* (RNN) query, an RNH query emphasizes on group of users instead of an individual user. More specifically, given a set of user locations U, a set of facility locations F, a query location q, a distance parameter ρ and a positive integer k, an RNH query returns all ρ-radius circles C enclosing at least k users $u \in U$, called *neighborhoods* (NH) such that the distance between q and C is less than the distance between C and any other facility $f \in F$. The RNH queries might have many practical applications including on demand facility placement and smart urban planning. We present an efficient approach for processing RNH queries on location data using R-tree based data indexing. In our approach, first we retrieve candidate RNH users by an efficient *bound, prune and refine* technique. Then, we incrementally discover RNHs of a query facility from these candidate RNH users. We also present the variants of RNH queries in spatial databases and propose solutions for them. We validate our approach by conducting extensive experiments with real datasets.

1. Introduction

These days we are experiencing voluminous spatial user data, e.g., Facebook has more than 600 million active users as of January 2011. At least half of these users login everyday and more than 150 million users actively access Facebook through location service enabled handheld devices [1]. Understanding these spatial users is crucially important for the location-based service providers, e.g., restaurants, supermarkets, gas stations etc., to sustain in the market. In the *reverse nearest neighbor* (RNN) queries, the users for which a given facility is the closest facility are considered to be its potential customers and believed to be influenced by the targeted marketing or special deals. Due to its utmost importance in location-based applications, RNN query and its variants have received significant research attention in the community ([2–5] for survey). However, there are many applications where the facility center needs to retrieve a group of users and the users in the same group need to be geographically close to each other. Unfortunately, RNN queries and their variants are not suitable for this kind of applications.

Reverse nearest neighborhood query. This paper studies a group version of RNN queries and propose a novel query called *reverse nearest neighborhood* (RNH) query for two-dimensional location data. That is, instead of retrieving dispersed users as in RNN queries, we are interested in finding groups of users which find a given facility as their nearest facility among all the existing facilities. However, there should be at least a certain number of users in a group and also, the users in each group should not be dispersed much geographically as the service providers wish to minimize their promotion cost. Like the work proposed by Choi et al. in [6], the group of users that satisfy the given cardinality constraint as well as can be encircled by a given radius constraint is considered as the *neighborhood*. We consider that the center of the neighborhood is always pulled towards the given facility by its special deal or the quality of services it provides to the member users of the neighborhood. Therefore, a neighborhood of a given facility can be modeled by the *nearest enclosing circle* (NEC) and the *closeness* between a neighborhood and a facility can be measured by the distance of a facility to the neighborhood representing NEC center.[1] We formally define a reverse nearest neighborhood (RNH) query in a spatial database as given as follows:

* Corresponding author.
E-mail addresses: saiful.islam@griffith.edu.au (M.S. Islam), bshe21@student.monash.edu (B. Shen), c.wang@griffith.edu.au (C. Wang), David.Taniar@monash.edu (D. Taniar), j.wang@griffith.edu.au (J. Wang).

[1] Interested readers are referred to the work [6] to know about NEC of a given set of points w.r.t. a query point in detail.

https://doi.org/10.1016/j.is.2020.101530
0306-4379/

"Given a set of users U, a set of facilities F, a distance parameter ρ, a query facility q and an integer k, a reverse nearest neighborhood (RNH) query returns the centers of all ρ-radius circles C which cover at least k users and find q as their nearest facility among all facilities in F".

Example 1.1. Consider the users and facilities given in Fig. 1. The reverse nearest neighborhoods of a query facility $q = (12, 9)$ for radius constraint $\rho = 3$ and cardinality constraint $k = 3$ are shown in Fig. 2(b), i.e., C_1 and C_2 are the reverse nearest neighborhoods of q. Here, the neighborhood C_3 finds f_5 nearer than q and therefore, C_3 is not the reverse nearest neighborhood of q.

The center of the same neighborhood (NEC center) may vary for different facilities though it covers the same group of users.[2] However, the nearest enclosing circle (NEC) for a given set of users w.r.t. a given facility f is unique [6]. It should be noted that a subset of these users could be covered by a neighborhood of another facility as long as the set meets the given cardinality and distant constraint for the facility. In our work, two neighborhoods may share users among them as long as they have at least one user which is unique to them. The facilities that attract similar users (subset of another neighborhood users of another facility) can be considered as competitors. The study of neighborhood based competitors analysis demands in depth investigation, which is out of the scope of this paper. However, we propose a few variants of RNH queries in this paper to deal with the neighborhoods that share users among them.

Applications. RNH queries can play a major role in location-based data applications. Some of the important applications are given as follows.

- **Targeted marketing.** Given a dataset of *point of interests* (PoIs) such as restaurants, supermarkets, gas stations etc., and a set of user locations, the RNH queries can be explored to discover the neighborhoods for a given query PoI. The query PoI can then design special promotion plans or deals for the neighborhoods. As the users in a neighborhood are not far from each other geographically, the query PoI could minimize its travel cost to do the promotion and therefore, might increase its profit. For example, consider the case where a query PoI needs to distribute special offer flyers to users' mailboxes as part of the promotion plan. In this case, the query PoI could save money for the salesman by visiting only the collocated users in a neighborhood rather than targeting closest but potentially dispersed users as in RNN queries-with RNN queries the salesman might need to travel more distances to have similar coverage as RNH queries. The query PoI can also make informed decisions on whether the deal would be offered in a neighborhood (distribution of flyers in users' mailboxes) based on its cardinality.
- **Optimizing urban planning.** Consider a facility setup/ replacement problem for the city council. The council can explore the RNH queries for optimizing suburb (neighborhood) design and the setup/ replacement of the facilities it aims to provide such as town-halls, community centers, schools, police stations, train stations, parks, playgrounds etc., so that its neighborhoods find its own facilities as their nearest facilities. This is practically more viable than optimizing the urban planning for individual users as with the traditional RNN queries. With RNH queries, the planner can optimize group influence of a facility based on its neighborhoods. For example, for a given set of potential facility locations, the planner can select the facility that has the maximum number of users covered in its neighborhoods. This is unlike to the min-dist location selection query proposed in [7] where a new facility is chosen if it results in a smaller average distance between a user and her nearest facility for a given set of users, existing facilities and a new set of potential new facility locations.

Facility	x	y
f_1	12	17
f_2	12	18
f_3	6	3
f_4	5	1
f_5	20	1
f_6	30	9

(a) Facilities, F

User	x	y
u_1	4	10
u_2	8	13
u_3	8	8
u_4	13	5
u_5	10	2
u_6	17	10
u_7	20	13
u_8	21	8
u_9	16	17
u_{10}	23	4
u_{11}	26	6
u_{12}	26	3

(b) Users, U

Fig. 1. A dataset of facilities (F) and users (U).

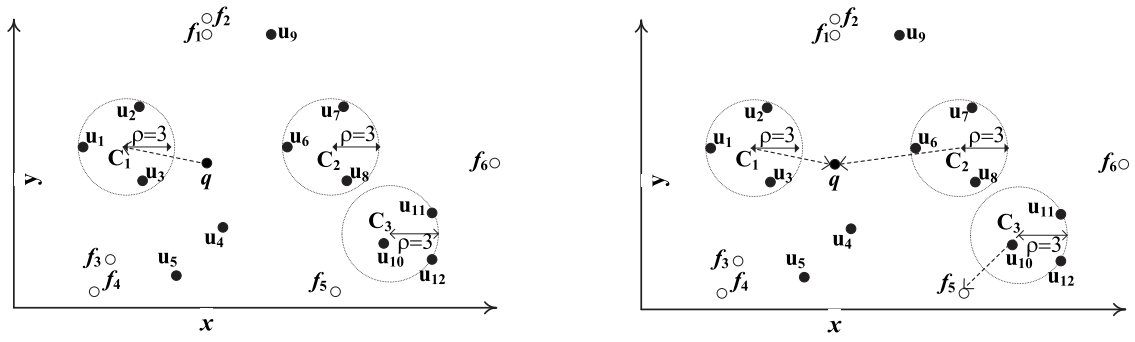
Though the spirit of the RNH queries is somewhat similar to some of the existing works such as aggregate nearest neighbor (ANN) queries [8], there is a clear distinction between an RNH query and an ANN query. an RNH query discovers all groups of points that find a given query point closest among all other competitor points, not the closest point to a group of points as in an ANN query. The RNH query is a reverse version of the recently proposed *nearest neighborhood* (NNH) query [6] where a group of points closest to a given query point is returned. However, the solution for processing NNH queries [6] is not applicable to RNH queries as there are two datasets and the query facility must compete with other facilities in an RNH query.

Our idea. In our proposed approach of RNH query processing, firstly we discover the *candidate region of RNH (NEC) centers* of a query facility q, denoted by $CRC(q)$. We propose a novel *influence-zone* [9] based pruning technique to discover a minimal set of existing facilities that can bound $CRC(q)$. Then, we extend $CRC(q)$ by rolling a ρ-radius half-circle along its edges to bound the *search region of RNH users* of q, denoted by $SRU(q)$. To speed up the discovery of candidate RNH users, we exploit the minimum-bounding-rectangle (MBR) based technique to approximate the $SRU(q)$, which is an upper bound of $SRU(q)$ and is denoted by $\overline{SRU}(q)$. We prune a large number of non-RNH users by executing a range query with $\overline{SRU}(q)$ on the user dataset U. The range query results are then refined to retrieve the *candidate RNH users*, denoted by $CRU(q)$. We call our approach *bound, prune* and *refine* technique. Our approach significantly outperforms the baseline approach. Then, we propose an algorithm to incrementally construct the reverse nearest neighborhoods of q from the candidate RNH users $CRU(q)$. Finally, we present the variants of RNH queries and propose a number of gain-based greedy solutions for them with an approximation guarantee.

Contributions. Our main contributions are given below.

1. We present a group version of the *reverse nearest neighbor* query on location data, called *reverse nearest neighborhood* (RNH) query.
2. We present efficient pruning ideas for the candidate RNH centers as well as the candidate RNH users. The correctness of our ideas is demonstrated theoretically.

[2] Assume that every facility pulls its neighborhoods towards itself.



(a) NNH Query (b) RNH Query

Fig. 2. Examples of (a) NNH and (b) RNH queries.

Table 1
Symbols and their meanings.

Symbol	Meaning
q	Query facility
U	The user dataset
u	A user in the user dataset
F	The facility dataset
f	A facility in the facility dataset
ρ	Radius constraint
k	Cardinality constraint
$NH(\rho, k)$	A ρ-radius neighborhood consisting of k users
$d(q, u)$	Euclidean distance between q and u
C	The nearest enclosing circle of $NH(\rho, k)$
$NF(u)$	The nearest facility of user u
$CRU(q)$	Set of candidate RNH users of q
$CRC(q)$	Candidate Region of RNH Centers of q
$SRU(q)$	Search Region of RNH Users of q
D_l	A data partition in GIndex
$\mu(NH(\rho, k))$	Returns the users in $NH(\rho, k)$

3. We propose an efficient incremental approach for processing the RNH queries in location data based on the bounded candidate RNH centers and users.
4. We present the variants of RNH queries and algorithms to process them in spatial databases.
5. We also conduct extensive experiments with real datasets to demonstrate the efficiency of our approach.

Organization. The rest of the paper is organized as follows: Section 2 presents the preliminaries and defines the problem studied in this paper in detail; Section 3 presents the solution sketch; Section 4.1 presents our approach of discovering the candidate RNH users; Section 4.2 presents our approach of RNH query processing; Section 4.3 discusses the variants; Section 5 demonstrates the efficiency of our approach; Section 6 discusses the related work; and finally, Section 7 concludes the paper.

2. Problem definition

This section presents the preliminaries and states the reverse nearest neighborhood queries in detail. Table 1 shows the symbols used in the paper.

2.1. Background

Assume that U is a set of user locations and F is a set of facility locations in a 2D space, e.g., x-y plane. The values of a user location $u \in U$ (the facility location $f \in F$) in the xth dimension and the yth dimension are denoted by u^x (f^x) and u^y (f^y), respectively. We use the user and user location as well as the facility and facility location interchangeably in this paper. We use D to denote the data space. Given a set of users U, a user $u \in U$ is called a nearest neighbor of a query facility q iff $\nexists u' \in U \setminus u$ such that $d(q, u') < d(q, u)$, where $d(q, u)$ denotes the distance between q and u. The nearest neighbor of q is denoted by $NN(q)$. We use $NF(u)$ to denote the nearest facility of u in F. Given a set of users U and a set of facilities F, a user $u \in U$ is called a reverse nearest neighbor of a query facility q iff $\nexists f \in F$ such that $d(f, u) < d(q, u)$. The reverse nearest neighbors of q is denoted by $RNN(q)$.

Definition 2.1 (*Neighborhood (NH)*)**.** Given a set of users U, a distance parameter ρ and a positive integer k, a neighborhood w.r.t. ρ and k, denoted by $NH(\rho, k)$, is a ρ-radius circle C enclosing at least k users in U.

Example 2.1. Consider the dataset of users $U = \{u_1, u_2, \ldots, u_{12}\}$ as given in Fig. 1(b). Some of the neighborhoods for $\rho = 3$ and $k = 3$ are C_1, C_2 and C_3 as shown in Fig. 2. The users u_4, u_5 and u_9 cannot form any neighborhood as they do not have sufficient number of users around them that can be enclosed by a 3-radius circle.

Lemma 2.1. *A neighborhood $NH(\rho, 1)$ consisting of an arbitrary user $u \in U$ is nearer to an arbitrary query facility q than any other neighborhood $NH'(\rho, 1)$ consisting of u iff (i) u, c and q are co-linear, (ii) c lies on the line segment between u and q, and (iii) c is ρ distance away from u, where c is the center of $NH(\rho, 1)$.*[3]

Definition 2.2 (*Nearest Neighborhood Query (NNH) [6]*)**.** Given a set of users U, a distance parameter ρ, a query location q and a positive integer k, a nearest neighborhood query finds the center of the nearest neighborhood $NH(\rho, k)$ to q.

Example 2.2. Consider the dataset of users $U = \{u_1, u_2, \ldots, u_{12}\}$ as given in Fig. 1 and the query location $q = (12, 9)$. The NNH query of q for $\rho = 3$ and $k = 3$ returns C_1 as it is the nearest neighborhood to q as shown in Fig. 2(a). The other two neighborhoods C_2 and C_3 are not returned by the NNH query of q as they are not nearer to q than C_1.

A NNH query [6] of a given query facility always returns a single neighborhood unless there is a tie.

2.2. Reverse nearest neighborhood queries

Here, we present our reverse nearest neighborhood queries.

[3] If $d(q, u) < \rho$, we can always consider q as the center of $NH(\rho, 1)$ and we get $d(q, NH(\rho, 1)) = d(q, c) = d(q, q) = 0$.

Definition 2.3 (*Reverse Nearest Neighborhood Query (RNH)*)**.** Given a set of users U, a set of facilities F, a distance parameter ρ, a query location q and a positive integer k, a reverse nearest neighborhood query discovers the centers of all neighborhoods $NH(\rho, k)$ that finds q as the nearest facility among all facilities $f \in F$. Mathematically, $RNH(q, \rho, k, U, F) = \{NH(\rho, k) | d(q, NH(\rho, k)) \leq d(f, NH(\rho, k)), \forall f \in F\}$, where $d(q, NH(\rho, k))$, denotes the distance between q and the center of the nearest enclosing circle (NEC) representing the neighborhood $NH(\rho, k)$. We use $RNH(q)$ to denote RNH query of q.

Example 2.3. Consider the users $U = \{u_1, u_2, \ldots, u_{12}\}$ and the facilities $F = \{f_1, f_2, \ldots, f_6\}$ as given in Fig. 1 and the query location $q = (12, 9)$. The RNH query of q for $\rho = 3$ and $k = 3$ returns C_1 and C_2 as shown in Fig. 2(b) as these neighborhoods find q as their nearest facility among all facilities in F. The RNH query of q does not return C_3 as $d(q, C_3) > d(f_5, C_3)$, i.e., C_3 finds f_5 as its nearest facility instead of q.

Definition 2.4 (*Reverse Nearest Neighborhood User*)**.** A user $u \in U$ is said to be a reverse nearest neighborhood user of a query facility q iff $\exists NH(\rho, k)$ such that $u \in NH(\rho, k)$ and $d(q, NH(\rho, k)) < d(f, NH(\rho, k)), \forall f \in F$.

Example 2.4. Consider the users $U = \{u_1, u_2, \ldots, u_{12}\}$ and the facilities $F = \{f_1, f_2, \ldots, f_6\}$ as given in Fig. 1 and the query location $q = (12, 9)$. The users u_1, u_2, u_3, u_6, u_7, u_8, u_{10}, u_{11} and u_{12} are the reverse nearest neighborhood users of q for $\rho = 3$ and $k = 3$.

Definition 2.5 (*Candidate Reverse Nearest Neighborhood User*)**.** A user $u \in U$ is called a candidate reverse nearest neighborhood user of q iff $k = 1$ and $\exists NH(\rho, k)$ such that $u \in NH(\rho, k)$ and $d(q, NH(\rho, k)) < d(f, NH(\rho, k)), \forall f \in F$. The set of candidate reverse nearest neighborhood users of a query facility q is denoted by $CRU(q)$.

Example 2.5. Consider the users $U = \{u_1, u_2, \ldots, u_{12}\}$ and the facilities $F = \{f_1, f_2, \ldots, f_6\}$ as given in Fig. 1 and the query location $q = (12, 9)$. The users u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_{10}, u_{11} and u_{12} are the candidate reverse nearest neighborhood users of q for $\rho = 3$.

Lemma 2.2. *A user $u \in U$ is a candidate reverse nearest neighborhood user of a query facility q iff $\nexists f \in F$ such that $d(f, c) < d(q, c)$, where $c = ((d(q, u) - \rho) \times cos\theta, (d(q, u) - \rho) \times sin\theta)$ in the transformed data space D' considering q as the origin, where θ is the angle formed by the line segment qu to the x-axis.*

Lemma 2.3. *A user $u \in U$ is a candidate reverse nearest neighborhood user of a query facility q if $d(q, c) < \rho$, where c is the center of the neighborhood $NH(\rho, k)$ consisting of u.*

Lemma 2.4. *If a user $u \in U$ is a reverse nearest neighbor of an arbitrary query facility q, then u is also a candidate reverse nearest neighborhood user of q.*

Lemma 2.5. *The reverse nearest neighbors of an arbitrary query facility q is a subset of its candidate reverse nearest neighborhood users, i.e., $RNN(q) \subseteq CRU(q)$.*

Lemma 2.6. *The reverse nearest neighborhood users of an arbitrary query point q is a subset of its candidate reverse nearest neighborhood users $CRU(q)$.*

Definition 2.6 (*Candidate Region of RNH Centers (CRC)*)**.** A region in a spatial database is called the candidate region of RNH centers of a query facility q, denoted by $CRC(q)$, iff we get $c \in CRC(q)$ where c is the center of the NEC representing each reverse nearest neighborhood $NH(q, k)$ of q, i.e., $d(q, c) \leq d(f, c), \forall f \in F$. The $CRC(q)$ is a subset of the data space D, i.e., $CRC(q) \subseteq D$.

Example 2.6. Consider the datasets given in Fig. 1 and the query $q = (12, 9)$. The candidate region of RNH centers of the query $q = (12, 9)$ is illustrated in Fig. 5. Any neighborhood of arbitrary shape (i.e., for any setting of ρ and k) that has its center within this region will find the query $q = (12, 9)$ nearer than any other facility in the dataset.

The discovery of candidate region of RNH centers of a query q is important to bound the search space of its candidate RNH users. The discovery process of this region is explained in Section 4.1.2 in detail.

Definition 2.7 (*Search Region of RNH Users (SRU)*)**.** A region in a spatial database is called the search region of RNH users of a query facility q, denoted by $SRU(q)$, iff every user $u \in CRU(q)$ appears in $SRU(q)$. The $SRU(q)$ is a subset of the data space D, i.e., $SRU(q) \subseteq D$.

Example 2.7. Consider the datasets given in Fig. 1 and the query $q = (12, 9)$. The search region of the candidate RNH users of the query $q = (12, 9)$ is illustrated in Fig. 6. Any user within this region satisfies the Lemma 2.2.

The discovery of the search region of RNH users of a query q is important to prune users from the dataset to expedite the processing of RNH queries. The construction of this search region is explained in Section 4.1.2 in detail.

2.3. Hardness of the RNH query problem

Processing RNH queries is more complex than NNH queries as we need to consider the competitor facilities for RNH queries in addition to user points. There are $\mathcal{O}(N^3)$ possible combinations of the users U as it suffices to consider only the circular convex sets [6], where $N = |U|$. We need to discover these combinations offline and index them as smallest enclosing circles (SECs).[4] Then, we need to compute the NECs of those SECs that find the query facility as their nearest facility at run time. Computing NEC at run time is needed as its realization is query dependent and cannot be precomputed.

Unfortunately, the above naïve approach is impractical as $\mathcal{O}(N^3)$ number of SECs is too large to be precomputed and thereafter, processed at run time w.r.t the given query and facility dataset.

3. Solution sketch

The solution approach of finding reverse nearest neighborhoods can be divided into two major steps as follows.

- **Discovering candidate RNH users.** Given the datasets of existing facilities and users, this step discovers all candidate reverse nearest neighborhood (RNH) users of the query facility q, i.e., $CRU(q)$.
- **Incrementally constructing RNHs.** Given the sets of existing facilities and a query facility q, this step incrementally constructs all reverse nearest neighborhoods $NH(\rho, k)$ of q under the radius constraint ρ and cardinality constraint k from the candidate RNH users.

[4] Given a set of points U_C, the SEC finds the center of the smallest circle C_{min} such that all points in U_C are enclosed by C_{min} [10].

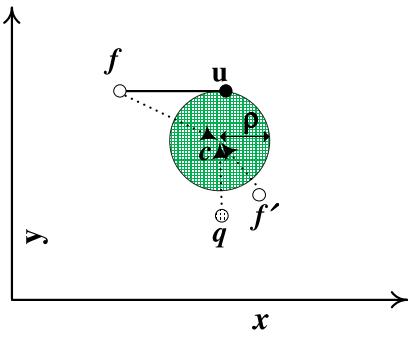


Fig. 3. Checking closest facility of a user $u \in U$ for Lemma 2.2 is not enough for deciding whether u is an RNH user of q.

One may argue that an extended approach of processing nearest neighborhood queries (NNH) [6] would be sufficient to replace the above steps and the approach could be as follows: (i) construct a neighborhood $NH(\rho, k)$ after repeatedly retrieving the nearest neighbor (nearest user) of q, i.e., $NN(q)$ from the user dataset U as per [6]; (ii) check $d(q, NH(\rho, k)) \leq d(f, NH(\rho, k))$, $\forall f \in F$; and (iii) repeat steps (i) and (ii) until we finish all users in U. However, these steps are either flawed or inefficient.

- **Inefficiency:** This approach needs to access all users in the dataset as we cannot guarantee the inclusion of the next NN user of q in the next RNH of q (Lemma 2.2).
- **Flaw:** The steps (i) and (ii) are flawed as we may miss some of the users from these steps which we could potentially include in the next RNHs of q.
- **Cumulative inefficiency:** One may argue that we can retain all users from these steps for the next rounds whether they are included in some of the previously found RNHs of q. However, it would make the next round(s) of constructing RNH(s) inefficient cumulatively.

We argue that dividing the approach of discovering RNHs of an arbitrary query facility into two major steps is efficient and practically viable. The advantage of the two step approach is that the candidate RNH users can also be discovered efficiently and can be managed better while incrementally constructing the RNHs for the query facility in the second step.

4. Our approach

This section presents the theories, pruning ideas and algorithms of our two steps approach of discovering RNHs of a query facility in a spatial database.

4.1. Discovering candidate RNH users

Here, we present our approach of discovering candidate RNH users of an arbitrary query facility q, i.e., $CRU(q)$. First, we present our GIndex based baseline algorithm and then an efficient *influence-zone* [9] based *bound, prune and refine* approach.

4.1.1. GIndex based algorithm

One might be tempted to compute the candidate RNH users of an arbitrary query facility q by checking the condition given in Lemma 2.2 for each user $u \in U$ with its nearest facility $f \in F$ only. If it would work, we could precompute the nearest facility of each user $u \in U$ in the dataset. At runtime, we could only scan the user dataset once and exploit this precomputed nearest facility information to determine the candidate RNH users of q. Unfortunately, this tempting idea might generate too many false positives as illustrated in Fig. 3, where c is the center of the neighborhood $NH(\rho, 1)$ consisting of u and $d(f, u) < d(f', u)$, $\forall f' \in F \setminus f$, but $d(f', c) < d(q, c) < d(f, c)$. Therefore, checking the condition given in Lemma 2.2 is unavoidable for *non-reverse nearest neighborhood user* of q, i.e., checking the distance of neighborhood center c to its nearest facility $f \in F$ (not the nearest facility of u in F) with the distance of c to the query facility q. However, precomputed information about the distribution of existing facilities in the dataset could prune lots of non-reverse nearest neighborhood users for which we could avoid checking the condition given in Lemma 2.2. This section presents our baseline approach of computing the candidate RNH users of an arbitrary query facility based on precomputed information of existing facilities in the dataset.

Data indexing. To precompute the information about the distribution of existing facilities in the data space D, we use query-independent *pivot-based grid partitioning scheme* [11]. The scheme applies an $n \times n$ grid to partition the whole data space such that each dimension is divided into n parts. That is, there are n^2 partitions in total for a 2-dimensional location data. A partition, denoted by D_l, is qualified by a 2-dimensional positional vector pos_l, which is used to identify the corresponding partition in the data space. The range of values covered in the ith dimension of a partition are: $(pos_l^i \times \delta^i, (pos_l^i + 1) \times \delta^i]$,where $\delta^i = \frac{max(ob^i)}{n}$, $\forall i \in \{x, y\}$, $pos^i \in \{0, 1, \ldots, n-1\}$ and ob denotes any object from the facility and user dataset. The positional vector pos_l of the partition in which the data objects belong to are: $pos_l^i = \frac{ob^i}{n}$, $\forall i \in \{x, y\}$. In location data, each partition D_l is a rectangular box. To index the data, we scan the facility dataset once and retain a number of facilities for each partition as *pivots*. The indexing is performed as a preprocessing step and termed as *GIndex*. The complexity of computing the partition of an arbitrary data point in GIndex is $\mathcal{O}(1)$ and the overall complexity of indexing the whole dataset is $\mathcal{O}(|D|)$.

Example 4.1. Consider the users $U = \{u_1, u_2, \ldots, u_{12}\}$ and the facilities $F = \{f_1, f_2, \ldots, f_6\}$ as given in Fig. 1. The pivot based grid partitioning and indexing of the dataset is given in Fig. 4(a). As the example dataset is very small, we retain all of the facilities as *pivots* in our *GIndex*.

Retrieving candidate RNH users. To retrieve the candidate RNH users from the user dataset, we first read the precomputed grid information and the corresponding *pivots* from *GIndex*. Then, we label a partition D_l as *pruned* as per the following lemma.

Lemma 4.1. *A partition D_l is labeled as* pruned *if there exists a pivot facility f in GIndex such that $d(q, c_{D_l}) > d(f, c_{D_l})$, where c_{D_l} is the center of the neighborhood $NH(\rho, 1)$ consisting of the closest user u of q in D_l and c_{D_l} is computed as per Lemma 2.2.*

Example 4.2. Consider the pivot based grid partitioning and indexing as shown in Fig. 4(a) of the example dataset given in Fig. 1 and the query facility $q = (12, 9)$. The gray-patterned partitions are the *pruned* partitions of q as shown in Fig. 4(b) as per Lemma 4.1. Here, the purple points represent the center c_{D_l} of the hypothetical neighborhood $NH(\rho, 1)$ consisting of the MIN point of the corresponding partition D_l w.r.t. the query facility q and these points are ρ distance away from the MIN point towards the query facility q. For the gray-patterned partitions, we get $d(q, c_{D_l}) > d(f, c_{D_l})$ for some pivot facilities f in GIndex as marked with arrows. Therefore, these partitions are pruned as per Lemma 4.1.

Complexity analysis. We can always use the MIN point of D_l as the closest (possible) user location for the query facility q in

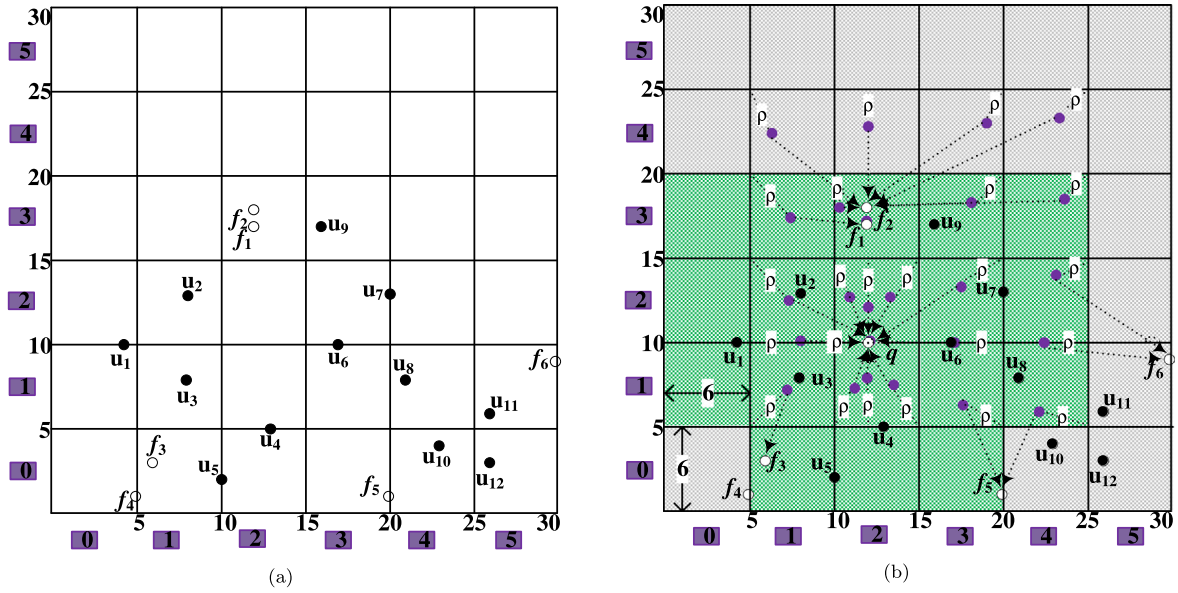


Fig. 4. Pivot-based grid partitioning: (a) data indexing and (b) green-patterned region: reduced search space of CRU(q).

Algorithm 1: GIndex based Algorithm for Computing $CRU(q)$

```
Input  : GIndex: grid info, F: facilities, U: users, q: query, ρ : radius constraint
Output : CRU(q): candidate RNH users
1  begin
2      CRU(q) ← ∅;                                          // initialization
3      for each u ∈ U do
4          D_l ← computePartition(u);
5          if D_l.isPruned() then
6              continue;                                    // Lemma 4.1
7          if d(q, u) < d(NF(u), u) then
8              CRU(q) ← CRU(q) ∪ u;                         // Lemma 2.4
9              continue;                                    // u is a RNN of q
10         c ← computeRNHCenter(u, q, ρ);                   // Lemma 2.2
11         if ∄f ∈ F: d(f, c) < d(q, c) then
12             CRU(q) ← CRU(q) ∪ u;                         // candidate RNH user
```

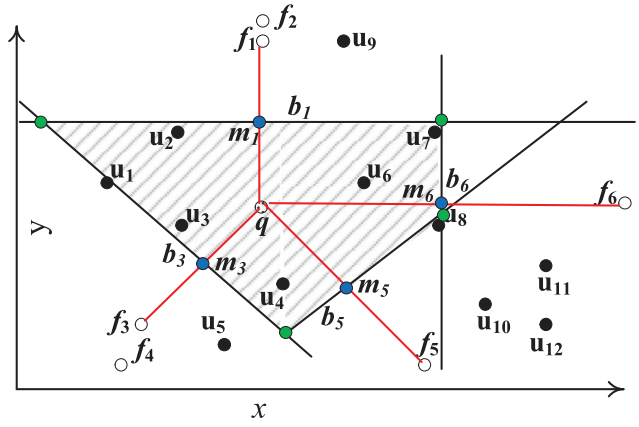


Fig. 5. The patterned region represents the candidate reverse nearest neighborhood centers of the query facility q, where m_i and b_i represent the midpoint between q and f_i and the perpendicular bisector of the line segment between q and f_i, respectively.

D_l. If evaluating Lemma 2.2 takes constant time, then the overall complexity of finding the pruned partitions becomes $\mathcal{O}(n^2)$, where n is the number of partitions in each dimension of the x-y plane.

To compute the candidate reverse nearest neighborhood users of the query facility q based on GIndex, we perform the following steps: (i) read the next user $u \in U$; (ii) compute the partition D_l of u; (iii) if D_l is a *pruned* partition, then go to step (i); (iv) if $d(q,u) < d(NF(u),u)$, then u is a candidate RNH user of q (as per Lemma 2.4) and go to step (i); and (v) compute the center c of the neighborhood $NH(\rho,1)$ consisting of u as per Lemma 2.2, if $\nexists f \in F$ such that $d(f,c) < d(q,c)$, then u is a candidate RNH user of q and go to step (i). The above steps are pseudo-coded in Algorithm 1.

4.1.2. Bound, prune and refine algorithm

Though the GIndex based Algorithm can reduce the search space of the candidate RNH users, it cannot avoid checking the condition given in Lemma 2.2 for a large number of users, specifically for those partitions in GIndex which partly overlaps with the exact search region of the candidate RNH users, i.e., $SRU(q)$ and therefore, might be inefficient. Here, we present a *bound, prune* and *refine* approach (BPRA) of retrieving the candidate RNH users by reducing the $SRU(q)$ significantly. Our approach is based on *influence-zone* based processing of reverse k nearest neighbor (RkNN) queries proposed in [9]. Assume $m_{f:q}$ denotes the midpoint of a facility $f \in F$ and the query facility q, which is computed as $m_{f:q} = (\frac{f^x+q^x}{2}, \frac{f^y+q^y}{2})$. The perpendicular bisector $b_{f:q}$ of the line segment between f and q goes through this midpoint $m_{f:q}$. Now, any neighborhood $NH(\rho,k)$ that has the center on the same side of the bisector $b_{f:q}$ as q does, finds q nearer than f (half-space-based pruning idea exploited for RkNN queries in [3,9]). Let $R_q(f)$ denotes such region in the data space. For example, consider the dataset of facilities $F = \{f_1, f_2, \ldots, f_6\}$ as given in Fig. 1(a), the query location $q = (12, 9)$ and the perpendicular bisectors b_3 as shown in Fig. 5, where b_3 denotes the perpendicular bisector of the line segment between facility f_3 and q. Any neighborhood $NH(\rho,k)$ that has the center on the right-hand side of b_3 finds the query facility q nearer than the facility f_3. Considering all perpendicular bisectors between the query q and the facilities $f \in F$, we get the following lemma.

Lemma 4.2. *The neighborhoods $NH(\rho,k)$ that have centers within $\bigcap_{\forall f \in F} R_q(f)$ find the query facility q as the nearest facility among the facilities $f \in F$.*

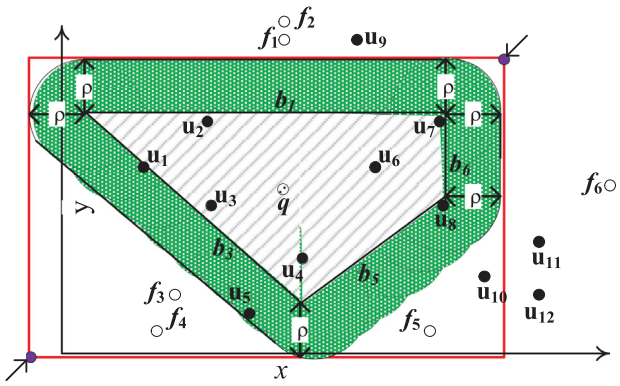


Fig. 6. Any user that falls within the black and green patterned region is a candidate RNH user, i.e., $SRU(q)$.

Lemma 4.3. *The candidate region of RNH centers is equivalent to* $\bigcap_{\forall f \in F} R_q(f)$*, i.e.,* $CRC(q) \equiv \bigcap_{\forall f \in F} R_q(f)$.

A naïve way to discover $CRC(q)$ is to initialize it to the entire data space D and then, refine $CRC(q)$ repeatedly for each facility $f \in F$, i.e., keep intersecting $R_q(f)$ with the $CRC(q)$ found in the previous round. However, this approach requires computing the perpendicular bisector of the line segment between each facility $f \in F$ and q, which is inefficient. Here, we exploit *influence-zone* [9] based technique to discover $CRC(q)$. Assume that the data universe D is bounded by a square. Then, current $CRC(q)$ is always a polygon. Let C_p denotes a circle centered at an arbitrary point p with radius equal to $d(p, q)$. It has been proved in [9] that a facility $f \in F$ can be ignored while updating $CRC(q)$ iff f lies outside C_v for every *convex vertex* v of the current unpruned polygon $CRC(q)$[9]. The *influence-zone* is always a *star-shaped* polygon (for reverse k nearest neighbor (RkNN) queries with $k > 1$) and the query facility q is its kernel point. In our case, we can find $CRC(q)$ of an arbitrary query point q by setting $k = 1$, which is essentially a voronoi diagram of q based on F. Since the kernel point q is known, the containment of a point inside the *influence-zone* can be done in $\mathcal{O}(\log n)$ [12]. It should be noted that every user $u \in CRC(q)$ finds q nearer than any other facility $f \in F$ and therefore, every $u \in CRC(q)$ is a candidate RNH user of q as per Lemma 2.4. However, finding $CRC(q)$ is not enough for discovering candidate reverse nearest neighborhood users $CRU(q)$ as per Lemma 2.5.

Lemma 4.4. *A user* $u \in U \setminus RNN(q)$ *is also a candidate RNH user of* q *if* $d(CRC(q), u) \leq \rho$.

From Lemma 4.4, we can conclude that the search region of the candidate RNH users of q is a superset of $CRC(q)$, i.e., $SRU(q) \supset CRC(q)$ and we can always roll a ρ-radius half-circle along the boundary points of the unpruned polygon $CRC(q)$ to bound $SRU(q)$.

Example 4.3. Consider the datasets given in Fig. 1 and the query $q = (12, 9)$. The search region of the candidate RNH users of q, i.e., $SRU(q)$, is shown in Fig. 6. The $SRU(q)$ here is formed by rolling a 3-radius half circle along the boundary of $CRC(q)$.

Upper bound of $SRU(q)$. Assume u and v are the lower-left and upper-right corners of the minimum bounding rectangle (MBR) of the vertices of the polygon $CRC(q)$. Then, we construct u' and v' as follows: $(u^x - \rho, u^y - \rho)$ and $(v^x + \rho, v^y + \rho)$.

Lemma 4.5. *The MBR consisting of* u' *and* v' *as the lower-left and upper-right corners is an upper bound of* $SRU(q)$.

Algorithm 2: Bound, Prune and Refine Algorithm for Computing $CRU(q)$

```
Input  : F: facilities, U: users, q: query, ρ : radius constraint
Output: CRU(q): candidate RNH users
1  begin
2      CRC(q) ←influenceZone(F, q);                    // influence-zone [9]
3      SRU(q) ←upperBound(CRC(q));                     // upper bound of SRU(q)
4      root ←constructRTree(U);                        // index U with R-Tree
5      U' ← regionQuery(root, SRU(q))                  // region query on U
6      for each u ∈ U' do
7          if u ∈ CRC(q) then
8              CRU(q) ← CRU(q) ∪ u;                    // Lemma 2.4
9              continue;                               // u is a RNN of q
10         else if d(u, CRC(q)) ≤ ρ then
11             CRU(q) ← CRU(q) ∪ u;                    // Lemma 4.4
```

Example 4.4. Consider the datasets given in Fig. 1 and the query $q = (12, 9)$. The rectangular window (MBR) marked by red color as shown in Fig. 6 is an upper-bound of $SRU(q)$. The lower-left and upper-right corners of the MBR are shown as steep arrows.

Finally, we perform the following steps for computing $CRU(q)$: (i) firstly, we compute $CRC(q)$; (ii) then, we compute the upper bound of $SRU(q)$ based on Lemma 4.5; (iii) then, we perform a region query on U (index it via R-Tree) based on $\overline{SRU(q)}$ to retrieve U'; and (iv) finally, for each user $u \in U'$ we do the following: add u to $CRU(q)$ if $u \in CRC(q)$ or $d(u, CRC(q)) \leq \rho$. The above is pseudo-coded in Algorithm 2.

4.2. RNH query processing

This section presents our approach of processing the RNH queries. Firstly, we insert the candidate RNH users $CRU(q)$ into a min heap called $\mathcal{H}_q$ to retrieve the next nearest candidate RNH user u_{next} and then, construct RNHs for the query facility q by maintaining and growing smallest enclosing circles (SECs) for u_{next}. The overall steps of our RNH query processing framework are as follows.

1. Find the next nearest candidate RNH user u_{next} from $\mathcal{H}_q$.
2. Retrieve all SECs which can be expanded to a larger circle with radius $\leq \rho$ to include u_{next} or split into new maximal SECs including u_{next}. Update such SEC(s) with u_{next}.
3. Repeat steps (1) and (2) until $\mathcal{H}_q$ is empty.
4. Only the SECs with at least k users are reported after computing their corresponding NECs (Definition 2.3).

The above is pseudocoded in Algorithm 3. However, executing the query framework given in Algorithm 3 is not straightforward. There are two main challenges in relation to Step (2) of the framework: (i) SEC retrieval and (ii) SEC maintenance, which are discussed below.

4.2.1. SEC retrieval

Here, we follow similar approach described in [6]. Firstly, we transform the Cartesian coordinate system of the current SECs into Polar coordinate system, in which an SEC C_i is represented by (a) its distance from the query facility, which is denoted by $C_i.d$ and (b) the angle range of its corresponding set of user points, which is denoted by $C_i[\theta_1, \theta_2]$. Now, whenever we retrieve the next candidate RNH user u_{next} from $\mathcal{H}_q$, we find all SECs C_i for step (2) that satisfy the following two conditions: (i) $d(q, u_{next}) - 2\rho \leq C_i.d$ and (ii) $C_i[\theta_1, \theta_2]$ overlaps the angle range of the $2\rho - radius$ circle centered at u_{next}. The above search can be implemented as 3-sided range search and can be solved in $\mathcal{O}(\log N + K)$ using the priority search tree [13], where we need to set the priorities of the SECs to their distances to the query facility.

Algorithm 3: A Framework of Processing RNH Queries

```
Input  : H_q: min-heap on CRU(q), q: query facility, ρ: radius, k: cardinality
Output: N: NECs representing RNHs of q
1  begin
2      R ← null;                                          // initialization
3      SECs ← {};                                         // initialization
4      while H_q ≠ ∅ do
5          u_next ←retriveFront(H_q);          // retrieve the next nearest
           candidate RNH user
6          SECs ←update(SECs, u_next, ρ);                 // update SECs
7      while ∃SEC ∈ SECs do
8          if |SEC|≥ k then
9              R ← add(convertToNEC(SEC, ρ));             // RNH is found
```

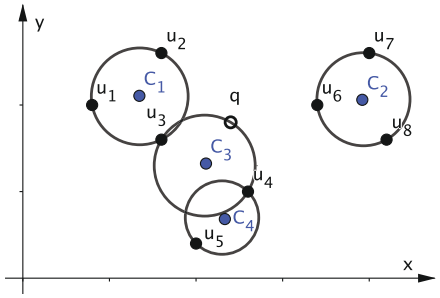


Fig. 7. SECs produced by the RNH querying framework.

4.2.2. SEC maintenance

Once we find all SECs for step (2) in our framework, we then update the SECs with the next user u_{next} accordingly. Assume C be the SEC retrieved for u_{next} and U_C be the set of users currently encircled by C. There are four cases to consider as per [6] and they are discussed below.

Case-1: If $d(C, u_{next}) \leq \rho$, we simply add u_{next} to C, where $d(C, u_{next})$ is the distance between u_{next} and the center of C.

Case-2: Assume $U_C \cup u_{next}$ can be enclosed by a ρ-radius circle i.e., $d(C, u_{next}) + C.r \leq 2\rho$, where $C.r$ is the radius of C. In this case, we need to compute the SEC of U_C and U_{next}. However, recomputing SEC for each SEC relevant to u_{next} is expensive, whenever u_{next} is retrieved. Therefore, similar to the approach in [6], we maintain an approximate SEC $\hat{C}$ for $C \cup u_{next}$ by the SEC of the corners points of the MBR of $U_C \cup u_{next}$ as long as $\hat{C}.r \leq \rho$.[5]

Case-3: Assume that only a subset of users $U'_C \subset U_C$ can be enclosed by a $\rho - radius$ circle including u_{next}, where U'_C is the set of users that are within 2ρ-radius circle centered at u_{next}. In this case, we compute a new exact SEC for $U'_C \cup u_{next}$.

Case-4: Assume that $U'_C \subset U_C$ is the set of users that are within 2ρ-radius circle centered at u_{next}. However, U'_C cannot be enclosed by a ρ-radius circle including u_{next}, but only a subset of $U''_C \subset U'_C$ including u_{next}. In this case, we need to compute all maximal SECs U''_C from C (i.e.e, SEC of U_C) that are not contained in any other SECs. The interested readers are referred to [6] for detailed explanation and technique for this problem.

Unlike the NNH query approach [6], we cannot terminate early while processing RNH queries. We can terminate only when $\mathcal{H}_q$ becomes empty. Our framework ensures that a neighborhood does not completely overlap with another neighborhood, i.e., for a pair of neighborhoods C_1 and C_2, we get $C_1 \setminus C_2 = \emptyset$ (similarly $C_2 \setminus C_1 = \emptyset$). However, two or more neighborhoods can completely cover the users of another neighborhood. Section 4.3 discusses the min query problem for RNH queries in detail.

4.2.3. A running example

Consider the datasets given in Fig. 1 and the query $q = (12, 9)$. First, we retrieve $CRU(q)$, which is $\{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}$ and insert them into the min heap $\mathcal{H}_q$. The running steps of Algorithm 3 are then as follows. The algorithm retrieves u_3 as the next user and compute SEC for it consisting of u_3 only, i.e., $C_3 = \{u_3\}$. Then, the algorithm retrieves u_4 and Case-2 (Section 4.2.2) is satisfied. Therefore, the algorithm computes an approximate SEC for C_3 consisting of $\{u_3, u_4\}$. Then, the algorithm retrieves the next user u_6 from $\mathcal{H}_q$ and form an SEC for it, i.e., $C_2 = \{u_6\}$ as there exists no overlapped SEC for u_6. Then, the algorithm retrieves the next user u_2 from $\mathcal{H}_q$ and Case-3 is satisfied. In this case, the algorithm retrieves the subset $\{u_3\}$ from C_3 and forms a new SEC C_1 consisting of $\{u_2, u_3\}$. Then, the algorithm retrieves the next user u_5 from $\mathcal{H}_q$ and again Case-3 is satisfied and the algorithm retrieves the subset $\{u_4\}$ from C_3 and forms a new SEC C_4 consisting of $\{u_4, u_5\}$. Then, the algorithm retrieves u_1 and finds $C_1 = \{u_2, u_3\}$ and $C_3 = \{u_3, u_4\}$ as the overlapped circles (as per Section 4.2.1). Now, for C_1 Case-2 is satisfied and therefore, the algorithm forms an approximate SEC C_1 consisting of $\{u_1, u_2, u_3\}$. For C_3, Case-3 is satisfied and therefore, the algorithm forms a new SEC C'_3 consisting of $\{u_1, u_3\}$. However, we ignore C'_3 as $C'_3 \subset C_1$. Then, the algorithm retrieves u_7 and u_8, and merge them into C_2 as Case-2 is satisfied. The above SECs are visalized in Fig. 7. Finally, the algorithm returns only C_1 and C_2 as the RNHs of the query facility q as they satisfy the cardinality constraint, which is 3.

4.3. Variants of RNH queries

This sections presents the variants of RNH queries in spatial databases and provides our solution for these queries.

4.3.1. Top-l RNH queries

Assume that a facility center wants to reach the maximum number of users within a certain budget and capable of reaching out to only l neighborhoods through its special deals. We term this kind of top-l reverse nearest neighborhood queries in spatial databases as top-l RNH queries.

Definition 4.1 (*Top-l RNH Query*)**.** Given a set of users U, a set of facilities F, a distance parameter ρ, a query location q and positive integers k and l, the top-l reverse nearest neighborhood query of q, denoted by $TlRNH(q)$, discovers l neighborhoods $\{NH_i(\rho, k)\}$, $i \in \{1, 2, \ldots, l\}$ such that (i) $d(q, NH_i(\rho, k)) \leq d(f, NH_i(\rho, k))$, $\forall f \in F$ and $\forall i \in \{1, 2, \ldots, l\}$; and (ii) $|\bigcup_{i=1}^{l} \mu(NH_i(\rho, k))|$ is maximized, where $\mu(NH_i(\rho, k))$ returns the users in the neighborhood $NH_i(\rho, k)$.

Unfortunately, processing TlRNH queries in spatial data-bases is a non-trivial problem. The problem can be reduced to maximum l-coverage problem [14], which requires exhaustive search over all possible l-cardinality reverse nearest neighborhoods of q.

A greedy algorithm. We propose a greedy algorithm to answer TlRNH queries in spatial databases based on the generic l-stage covering algorithm provided in [14], which guarantees $1 - 1/e$ approximation of the optimal solution. Before presenting our approach we define the ith stage gain of each neighborhood which is not selected in the answer set yet. Assume $\mathcal{S}^{i-1}$ is a set of the users of the reverse nearest neighborhoods of the query facility q selected till $(i-1)$th stage. Then, the ith stage gain of a neighborhood $NH(\rho, k)$ that has not been selected yet, defined by $\mathcal{G}^i_{\mathcal{S}^{i-1}}(NH(\rho, k))$, is defined as follows:

$$\mathcal{G}^i_{\mathcal{S}^{i-1}}(NH(\rho, k)) = |\mu(NH(\rho, k))| - |\mathcal{S}^{i-1} \cap \mu(NH(\rho, k))| \quad (1)$$

[5] It should be noted that $\hat{C}.r \leq \sqrt{2}C.r$.

Algorithm 4: Greedy Top-l RNH Query Processing

```
Input  : RNH(q): RNH query result of q
Output: 𝒩: Top-l RNH query result
1  begin
2      i = 1; 𝒮 ← ∅; 𝒩 ← ∅;                                  // initialization
3      while i ≤ l do
4          maxGain← 0; NH_1 ← null;                          // initialization
5          foreach NH(ρ, k) ∈ RNH(q) which is not selected do
6              gain←computeGain(NH(ρ, k), 𝒮);                // as per Eq. (1)
7              if gain≥maxGain then
8                  maxGain← gain; NH_1 ← NH(ρ, k);           // update
9          𝒮 ← add(μ(NH_1)); 𝒩 ← add(NH_1);                  // update
10         i ← i + 1;                                        // increment i
```

Algorithm 5: Greedy Min Cover RNH Query Processing

```
Input  : RNH(q): RNH query result of q
Output: 𝒩: Min Cover RNH query result
1  begin
2      𝒮 ← ∅; 𝒩 ← ∅;                                        // initialization
3      while true do
4          maxGain← 0; NH_1 ← null;                          // initialization
5          foreach NH(ρ, k) ∈ RNH(q) which is not selected do
6              gain←computeGain(NH(ρ, k), 𝒮);                // as per Eq. (1)
7              if gain>maxGain then
8                  maxGain← gain; NH_1 ← NH(ρ, k);           // update
9          if maxGain= 0 then
10             break;                                        // terminate the processing
11         else
12             𝒮 ← add(μ(NH_1)); 𝒩 ← add(NH_1);              // update
```

The greedy steps of our TlRNH query processing algorithm in spatial databases are as follows: (1) initialize $\mathcal{S}$ to $\emptyset$; (2) compute the ith stage gain $\mathcal{G}^i_{\mathcal{S}^{i-1}}(NH(\rho,k))$ of each reverse nearest neighborhood of q, $NH(\rho,k)$, that has not been selected yet and select the one that has the highest gain as per Eq. (1) and update $\mathcal{S}$ accordingly, $\forall i \in \{1,2,\ldots,l\}$. The above is pseudocoded in Algorithm 4. Our algorithm guarantees $1-1/e$ approximation of the optimal solution as per [14].

4.3.2. Min cover RNH queries

The *min cover RNH query* of an arbitrary facility q returns the minimum number of neighborhoods that cover all of its reverse nearest neighborhood users (recall that the definition of reverse nearest neighborhood user is given in Definition 2.4). The min cover RNH query is formalized below.

Definition 4.2 (*Min Cover RNH Query*)**.** Given a set of users U, a set of facilities F, a distance parameter ρ, a query location q and a positive integer k, the min cover reverse nearest neighborhood query of q, denoted by $MCRNH(q)$, discovers the minimum number of reverse nearest neighborhoods of q to cover all reverse nearest neighborhood users of q, i.e., minimum number of reverse nearest neighborhoods of q to cover all users $u \in U$ if $\exists NH(\rho,k)$ such that $u \in NH(\rho,k)$ and $d(q,NH(\rho,k)) \leq d(f,NH(\rho,k))$, $\forall f \in F$.

Like TlRNH queries, the processing of MCRNH queries is also non-trivial. The MCRNH query problem can be reduced to minimum set cover problem and is therefore NP-complete.

A greedy algorithm. The greedy steps of our min cover RNH query processing algorithm are as follows: (1) initialize $\mathcal{S}$ and $\mathcal{N}$ to $\emptyset$; (2) compute the ith stage gain of each reverse nearest neighborhood of q that has not been selected yet and select the neighborhood $NH(\rho,k)$ that has the highest gain as per Eq. (1) and add $NH(\rho,k)$ to $\mathcal{N}$: if $\mathcal{G}^i_{\mathcal{S}^{i-1}}(NH(\rho,k)) > 0$ and update $\mathcal{S}$ accordingly, $\forall i \in \{1,2,\ldots,|RNH(q)|\}$, else terminate. $\mathcal{N}$ is the min cover RNH query result and again, our greedy algorithm is guaranteed to produce $1-1/e$ approximation of the optimal result as per [14]. The pseudo-code of the above greedy steps is given in Algorithm 5.

Lemma 4.6. *A reverse nearest neighborhood $NH(\rho,k)$ of a facility q is a member of min cover RNH query of q if $NH(\rho,k)$ has at least one user that is not included in any other reverse nearest neighborhoods $NH_1(\rho,k)$ of q which are spatially overlapped with $NH(\rho,k)$.*

Any optimal solution must add a neighborhood, which satisfies Lemma 4.6, in the min cover RNH query answer set. The gain based greedy algorithm for min cover RNH queries described before does not give any special treatment to these neighborhoods to improve the min cover answer set. Consider the three neighborhoods of an arbitrary query q as visualized in Fig. 8. In this example, the greedy algorithm will include all three neighborhoods C_1, C_2 and C_3 in the min cover. One can easily verify that both C_1 and C_3 satisfy Lemma 4.6 and therefore, if we prioritize their inclusion in the answer set before C_2, the min cover could report only C_1 and C_3 as the answers. We exploit this observation to provide a better greedy approximation for the min cover RNH queries.[6] However, the problem of discovering neighborhoods that satisfy Lemma 4.6 is a non-trivial problem. The complexity of discovering these neighborhoods without exploiting their geometrical properties is $\mathcal{O}(n^2)$, where n is the number of reverse nearest neighborhoods of q, which is inefficient. In this paper, we develop an algorithm by exploiting the geometrical properties of the neighborhoods with far lower complexity, which is $\mathcal{O}(n(\log n + K))$, where K is the average number of overlapped neighborhoods for any neighborhood. To implement this improved greedy min cover RNH query algorithm, we first represent each neighborhood (the corresponding SEC) into polar coordinate system and insert them in a priority queue, which is similar to the idea of SEC retrieval described in Section 4.2.1. To identify the neighborhoods that satisfy Lemma 4.6, we search for the overlapped neighborhoods C_i of the neighborboohd C_j ($C_i \neq C_j$): C_i and C_j overlaps with each other if (i) $d(C_i,q) - 2\rho \leq d(C_j,q)$ and (ii) $C_i[\theta_1,\theta_2]$ and $C_j[\theta_1,\theta_2]$ have angle overlap. Then, we check whether C_j has any member user that is not covered by any C_i. If so, C_j can be added to the min cover RNH query result. For the rest of the uncovered true RNH users, we can run gain based greedy min cover RNH query Algorithm 5.

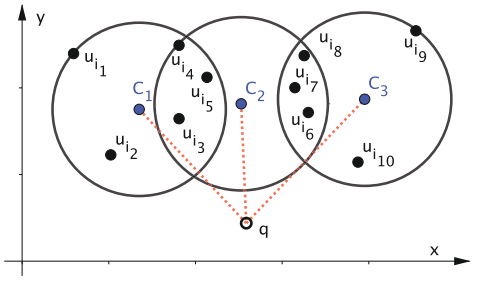


Fig. 8. Circles C_1 and C_3 satisfy Lemma 4.6.

5. Experiments

In this section, we demonstrate the efficiency of our approach of processing reverse nearest neighborhood (RNH) queries in spatial databases. Given that no prior algorithm exists for processing

[6] To the best of our knowledge, we also do not find any work that exploits this property for the classical minimum set cover problem.

Table 2
Test datasets (TD) varieties.

TD#	NE		RR		CAS		SYN	
	Facilities	Users	Facilities	Users	Facilities	Users	Facilities	Users
TD1	192	123 396	239	257 698	216	196 681	400	199 266
TD2	1003	122 585	1313	256 624	1210	195 687	2165	197 501
TD3	2415	1 211 173	3119	254 818	2972	193 925	3470	196 196
TD4	3680	119 908	4718	253 219	4576	192 321	3901	195 765
TD5	5604	117 984	7075	250 862	7000	189 897	4264	195 402

RNH queries, we compare the algorithms presented in this paper only.

5.1. Setup

This section describes our tested datasets, parameters, environment and algorithms experimented.

5.1.1. Datasets

We experiment based on three real world datasets which are namely NE, RR, and CAS including 123,593, 257,942 and 196,902 points of interest. These datasets are originated from TIGER project at the US Census Bureau and they can be downloaded from the Chorochronos website,[7] which was previously known as R-Tree portal. These datasets have been extensively experimented in spatial data management research [6,9,15]. We also create a synthetic dataset, called SYN, consisting of 200,000 uniformly distributed random two dimensional data points in the *x*-*y* plane.

To create the facilities and users from these datasets, we divide each dataset into 125×125, 100×100, 80×80, 50×50 and 20×20 grids to create a variety of test datasets (TDs). For each grid, we randomly select one point of interest as facility point and the rest as the user points. This gives us the varied cardinalities in each dataset and a cluster of users around each facility. For each tested dataset, we also randomly select 10 facilities as test queries by following the distribution of the dataset. The data values are normalized in the range [0, 1] in each tested dataset to conduct all experiments. The statistics of the tested datasets are summarized in Table 2.

5.1.2. Parameters

The radius in the neighborhoods ranges from 0.03 to 0.012 in our experiments. The cardinality constraint (i.e., k) of the neighborhoods varies from 10 to 25. The MAX #entries in a R-tree node is set to 50 by default and varied from 10 to 70 for experimenting its effect on the efficiency of our bound, prune and refine algorithm. The grids in GIndex vary from 50 to 100.

5.1.3. Environment

All of the proposed algorithms are implemented in Java and Eclipse environment. For all real datasets (NE, RR and CAS), we conduct all of our experiments on a Mac laptop with 2 GHz Intel Core i7 CPU and 8 GB main memory. For the synthetic dataset SYN, all of our experiments are conducted on a 64-bit Windows PC with 3.6 GHz Intel Core i7 CPU and 32 GB main memory (see Fig. 11).

5.1.4. Algorithms

The following algorithms are tested in our experiments for demonstrating their efficiencies of retrieving candidate RNH users and processing RNH queries in spatial databases:

- **Online GIndex based Algorithm (OnGIBA)**. In this version of the algorithm, we need to compute the nearest facility of a user u, $NF(u)$, online for line 7 of Algorithm 1.
- **Offline GIndex based Algorithm (OfGIBA)**. In this version of the algorithm, the nearest facility of a user u, $NF(u)$, is precomputed for line 7 of Algorithm 1.
- **Online Bound-Prune-Refine Algorithm (OnBPRA)**. This is simply the proposed algorithm as pseudocoded in Algorithm 2.
- **Offline Bound-Prune-Refine Algorithm (OfBPRA)**. In this version of the algorithm, the nearest facility of an arbitrary user u, $NF(u)$, is known for Algorithm 2. Therefore, we can avoid the containment checking for users, where $d(q, u) < d(NF(u), u)$, in lines 7–9 of Algorithm 2.

We also evaluate top-l RNH query processing (TlRNH), greedy min cover RNH query processing (Greedy MCRNH) and improved greedy min cover RNH query processing (Improved Greedy MCRNH) algorithms.

5.2. Efficiency evaluation

This section evaluates the efficiency of our algorithms.

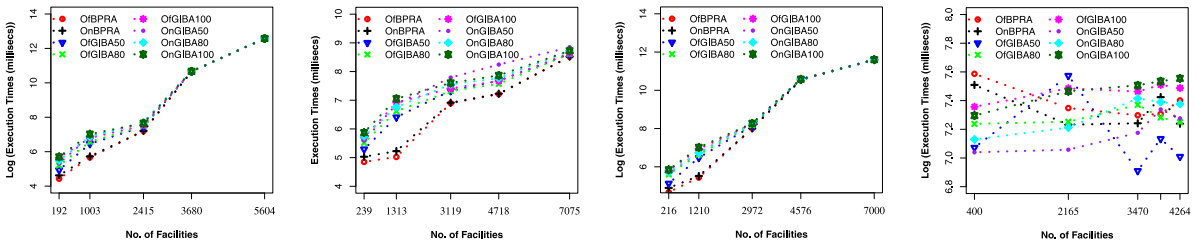
5.2.1. GIBA vs. BPRA based RNH query processing

Here, we compare the performance of our BPRA based RNH query processing with GIBA based RNH query processing algorithms. We set the grid size for GIBA to 50, 80 and 100, and call them GIBA50, GIBA80 and GIBA100, respectively. We create both offline and online versions of BPRA and GIBA algorithms as described in Section 5.1.4. The tested algorithms: OfBPRA and OnBPRA include the run times for both candidate RNH users retrieval based on BPRA as described in Section 5.1.4 and RNH query processing time based on Algorithm 3. Similarly, OfGIBA50, OfGIBA80, OfGIBA100, OnGIBA50, OnGIBA80 and OnGIBA100 include the run times for both candidate RNH users retrieval based on GIBA as described in Section 5.1.4 and RNH query processing time based on Algorithm 3.

Effect of cardinalities in the dataset. The effect of the number of facilities on the efficiencies of BPRA and GIBA based RNH query processing algorithms on all datasets is given in Fig. 9 (the times are plotted in logarithmic scale). It is easy to verify that the proposed BPRA based RNH query processing algorithms outperform the GIBA based RNH query processing algorithms in all tested datasets except for certain test cases in SYN dataset. The performance of GIBA based RNH query processing algorithms could be tuned further by setting the grid size optimally, e.g., for SYN dataset the GIBA based RNH query processing algorithms outperform the BPRA based RNH query processing algorithms for certain grid settings as it can be observed from 9(d). However, we believe that the BPRA based RNH query processing algorithms are more reliable as the optimal setting of grid sizes for GIBA based RNH query processing algorithms is difficult to achieve in reality.

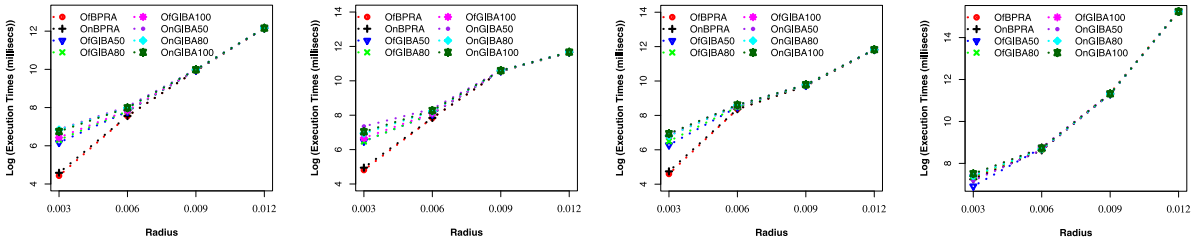
Effect of query parameter. The proposed BPRA based RNH query processing algorithms are more tolerant to the radius constraint of the neighborhoods in comparison to the GIBA based

[7] http://chorochronos.datastories.org/?q=user/15/track.



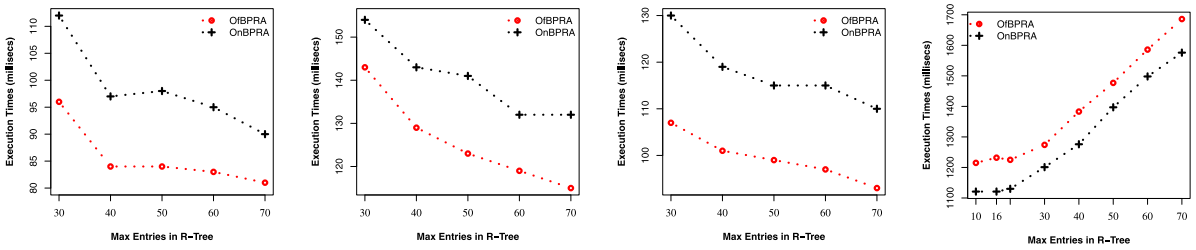
(a) NE (b) RR (c) CAS (d) SYN

Fig. 9. Effect of facilities on the efficiency of BPRA and GIBA based RNH query processing algorithms: (a) NE, (b) RR, (c) CAS and (d) SYN TD1–TD5 datasets ($\rho = 0.003$, $k = 10$).



(a) NE (b) RR (c) CAS (d) SYN

Fig. 10. Effect of radius on the efficiency of BPRA and GIBA based RNH query processing algorithms: (a) NE, (b) RR, (c) CAS and (d) SYN TD3 datasets ($k = 10$).



(a) NE (b) RR (c) CAS (d) SYN

Fig. 11. Effect of MAX entries in the R-tree on the efficiency of BPRA algorithms: (a) NE, (b) RR, (c) CAS and (d) SYN TD3 datasets ($\rho = 0.003$, $k = 10$).

RNH query processing algorithms as it is evident from Fig. 10 (the times are plotted in logarithmic scale). Again, the BPRA based RNH query processing algorithms outperform the GIBA based RNH query processing algorithms in all TD-3 datasets except for certain test cases in SYN dataset (see Fig. 10(d)). However, the performances of the GIBA based RNH query processing algorithms are dependent on grid settings and the optimal settings of grid is data and query dependent.

Effect of R-Tree MAX entries on BPRA based RNH query processing algorithms. We find that the efficiencies of both OfBPRA and OnBPRA based RNH query processing algorithms improve with the increase in the number of MAX entries in the R-Tree in all real datasets and stabilize at 70 as can be seen in Fig. 11(a)-(c). For SYN dataset, we find that the efficiencies of BPRA based RNH query processing algorithms improve when the number of MAX entries in the R-Tree decreases and stabilize in the range 10–20 as can be seen in Fig. 11(d).

5.2.2. Evaluation of RNH query processing framework

Here, we examine the performance of our proposed RNH query processing framework, i.e., Algorithm 3, only.

Effect of radius constraint. The efficiency of the RNH query processing framework (Algorithm 3) is hugely dependent on the radius constraint as it is evident from Fig. 12, i.e., deteriorates with the increase in radius. This is because the number of candidate RNH users also increases if there is an increase in radius value, which ultimately affects the efficiency of the framework.

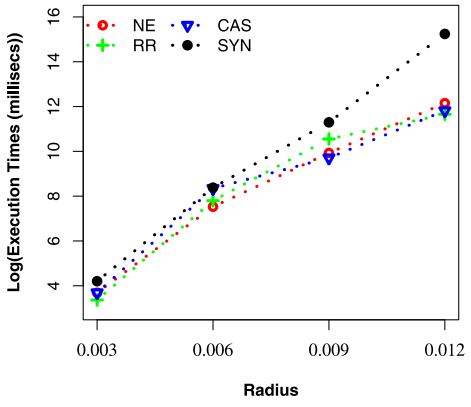


Fig. 12. Effect of radius on the efficiency of RNH query processing framework in TD3 datasets ($k = 10$).

Effect of cardinality constraint. In our experiments, we do not observe any influence of the cardinality constraint k on the efficiency of the proposed RNH query processing framework (Algorithm 3) as it is evident from Fig. 13. This is because we maintain SECs of any size untill we finish accessing all candidate RNH users from heap and finalize the RNHs as the last step in the framework.

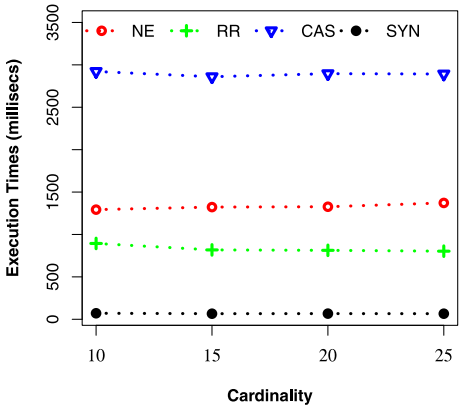


Fig. 13. Effect of cardinality constraint (k) on the efficiency of RNH query processing framework in TD5 datasets ($\rho = 0.003$).

5.2.3. Evaluation of RNH query variants

This section evaluates the variants of RNH queries in spatial databases. As explained before that min cover RNH queries are useful in minimizing the number of reverse nearest neighborhood users of an arbitrary query in spatial databases to cover all true reverse nearest neighborhood users as per Definition 2.4. Table 3 demonstrates the effectiveness of min cover RNH query results in comparison to the number of reverse nearest neighborhoods (Org. RNH) returned by the RNH query processing framework. Both greedy min cover RNH query algorithm and improved min cover RNH query algorithms reduces the number of neighborhoods significantly for all tested datasets as it is evident from Table 3.

To validate the run time requirements of the proposed min cover RNH query algorithms, we compare their efficiencies with top-l RNH query algorithms (TlRNH) by setting $l = 3$. It is evident from Fig. 14 that the run time requirements of greedy min cover RNH (G. MCRNH) is comparable to that TlRNH. However, the performance of the improved greedy MCRNH algorithm (Im. G. MCRNH) deteriorates with an increase in the number of neighborhoods returned by the RNH querying framework. This is because we spend much time on discovering the neighborhoods that must be added in the min cover RNH query result whose member users cannot be covered by any other neighborhood. The run time of the Im. G. MCRNH is within secs for all tested datasets which we believe durable as we can minimize the number of neighborhoods further.

6. Related work

This section reviews spatial queries that are similar to the *reverse nearest neighborhood* (RNH) queries.

Reverse nearest neighbor retrieval. The RNH query is a group version of RNN query [2–5]. It has been shown in [3] that the half-space based approach prunes more area than the six-regions based pruning [2] while discovering the RNN points. One may argue that the RNN users of a query facility are the candidate RNH users and thereafter, half-space based approach [3] can be used to prune the search space and discover the candidate RNH users. However, this argument is not true. This is because the nearby facilities surrounding the query facility (half-space based approach [3]) are not sufficient to bound the search space of the candidate RNH users as per Lemma 2.2, i.e., it may miss some of the true candidate RNH users. This paper proves that RNN users of an arbitrary query facility is only a subset of its true candidate RNH users. Thereafter, this paper presents a novel technique to discover a minimal set of existing facilities to bound the candidate RNH centers based on *influence-zone* [9] and then extend this region to bound its candidate RNH users of an arbitrary query facility. It should be noted that *influence-zone* [9] solves k-RNN queries efficiently and it is essentially the Voronoi cell for a query q when $k = 1$. In our case, we need to set $k = 1$ for *influence-zone* based retrieval of existing facilities to bound the candidate RNH centers of the reverse nearest neighborhoods of q, i.e., $CRC(q)$. One may argue that we can retrieve k-RNN users as the candidate reverse nearest neighborhood users for q. However, finding the right k for q would be intractable. Even if we relax it for a large value of k, we may still need to test the condition given in Lemma 2.2 to refine the result, which would be inefficient.

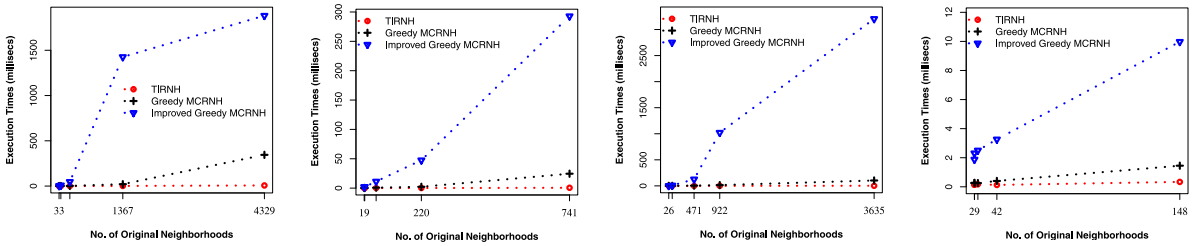
Group of objects retrieval. Given a set of points in the plane, [16] studies the problem of finding k points that form a "small" set under some given measure such as the diameter and presents efficient algorithms to discover such point set. [8] presents an approach for finding the closest facility to a set of user locations. Given a query set, [17] retrieves a point from a given set of points such that the maximum distance of the retrieved point to all query points is minimized. Given a data point set and a set of query points, [18] proposes an approach for processing *group nearest group* query where the total distance of all query points to the nearest database point in a group is minimized. *Circle of friend query* [1] finds a group of friends in a Geo-Social network whose members are close to each other both socially and geographically. Given a query location and a set of keywords, [19] finds top-k nearby groups of relevant objects. Given a set of spatial objects and a query location, [20] presents an approach for retrieving a group of spatial objects such that the group's keywords cover the query's keywords, and such that the group members are nearest to the query location and have the smallest inter-object distances. Choi et al. [6] presents the NNH query where a group of points closest to a given query point is returned. Recently, Ghosh et al. [21] propose an approach to answer top m flexible socio spatial group query to retrieve the top m groups w.r.t. multiple point-of-interests (POIs) where each group satisfies the minimum social connectivity constraints and groups are scored based on social closeness, spatial distance and group size. However, none of the above considers competitor facilities. The proposed RNH query discovers all groups of spatial objects, under a given diameter and cardinality constraint, that find the given query facility nearer than any other facility.

Given a dataset of objects $P = \{p_1, p_2, \ldots, p_n\}$, a query object q and two user given parameters m and k, Jiang et al. [22] solves a spatial group query called reverse top-k group nearest neighbor (RkGNN) query to return k subsets with least aggregate distances such that each subset has m data objects from P and has the query q in its group nearest neighbor. Zhang et al. [23] extended the monochromatic RkGNN query [22] and proposed a bichromatic RkGNN query for two datasets. Given two datasets A and B, a query q and two user given parameters m and k, the bichromatic RkGNN query retrieves k subsets with least aggregate distances such that each subset has m data objects from A and has the query q in its group nearest neighbor in dataset B. Like monochromatic RkGNN query, bichromatic RkGNN query also does not have the radius constraint for the group of data objects as compared to the RNH query proposed in this paper. The cardinality constraint in RNH query is also not an "equality constraint", rather it is a "least constraint". Therefore, the RkGNN query algorithms are not applicable to RNH queries.

Ali et al. [24] study the problem of processing consensus queries in a spatial database where a group of users is interested in discovering a meeting place that optimizes the travel distance for each user in the group. The authors have also addressed the issue of discovering optimal subgroups of all allowable cardinalities. Here, the users are modeled as a set of query points and

Table 3
Avg. number of neighborhoods in RNH, greedy MCRNH and improved greedy MCRNH queries ($\rho = 0.003$ and $k = 10$).

TD#	NE			RR			CAS			SYN		
	Org. RNH	G. MCRNH	Im. G. MCRNH	Org. RNH	G. MCRNH	Im. G. MCRNH	Org. RNH	G. MCRNH	Im. G. MCRNH	Org. RNH	G. MCRNH	Im. G. MCRNH
TD1	4329	118	114.5	741	91	83	3635	166	163	148	49	38
TD2	1367	25.6	24.4	220	17.8	16.8	922	20	19.2	42	16	12
TD3	259	12	11.4	61	8	7.6	471	16.6	15.4	31	13	9
TD4	71	6.2	5.8	21	5	4.6	87	8.4	7.6	29	12	9
TD5	33	5.4	4.8	19	3.6	3.2	26	6	5.4	29	12	8



(a) NE (b) RR (c) CAS (d) SYN

Fig. 14. Effect of the number of original neighborhoods on the efficiency of TIRNH, Greedy MCRNH and improved Greedy MCRNH algorithms: (a) NE, (b) RR, (c) CAS and (d) SYN TD1-TD5 datasets ($\rho = 0.003$, $k = 10$).

the given query points are limited in numbers. Both consensus queries [24] and RNH queries deal with two datasets. But, the idea of groups (neighborhoods) in RNH queries are different from the groups of users studied in [24]. The group of users (neighborhood) in RNH queries are collocated and parameterized by not only a cardinality constraint, but also a distance constraint. Finally, the algorithms developed for consensus queries in [24] are not applicable to solve RNH queries in spatial databases.

Density-based clustering. The most popular density-based clustering algorithm in spatial databases is DBSCAN [25]. DBSCAN repeatedly connects objects within a given distance threshold of each other to construct a cluster. [26] presents top-k *spatial textual clusters* (k-STC) query to return k most textually relevant clusters that are spatially close to a given query location. Though [26] considers to return clusters with respect to a query location, [26] does not consider competitor facilities as we do in RNH queries.

7. Conclusion

This paper proposes a new query called reverse nearest neighborhood (RNH) query for spatial databases. We present an efficient approach for processing RNH queries based on R-Tree data indexing scheme. We also present variants of RNH queries and their solution approaches in this paper. The presented work can be researched further in optimizing the proposed RNH querying framework and all algorithms can be extended for spatio-textual databases.

Declaration of competing interest

Jianxin Li
E-mail: jianxin.li@deakin.edu.au, Deakin University, Australia
Chengfei Liu
E-mail: cliu@swin.edu.au, Swinburne University of Technology, Australia.
Muhammad Aamir Cheema
E-mail: Aamir.Cheema@monash.edu, Monash University, Australia.
Rui Zhou, E-mail: rzhou@swin.edu.au, Swinburne University of Technology, Australia.
Farhana Murtaza Choudhury
E-mail: farhana.choudhury@unimelb.edu.au, University of Melbourne, Australia.
Mohammed Eunus Ali
E-mail: eunus@cse.buet.ac.bd, Bangladesh University of Engineering and Technology, Bangladesh.

Acknowledgments

This work is partially supported by a Griffith University's 2018 New Researcher Grant Australia with Dr. Md. Saiful Islam being the Chief Investigator. We appreciate the anonymous reviewers for their insightful feedback to improve the presentation quality of the paper.

Appendix. Proofs of lemmas

Proof of Lemma 2.1. Any neighborhood with radius ρ consisting of u must have its center within the circle centered at u whose radius is also ρ as shown by the transparent circle in Fig. A.15(a). However, there would be an infinite number of such neighborhoods. A few of these neighborhoods are shown in Fig. A.15(b). Assume that $NH(\rho, 1)$ is a neighborhood consisting of u and the center c of $NH(\rho, 1)$ lies on the line segment between q and u, and c is ρ distance away from u, which is depicted as green patterned circle in Fig. A.15. Again, assume that $NH'(\rho, 1)$ is an another neighborhood consisting of u whose center is at a and a is also ρ distance away from u, but a does not lie on the line segment between q and u, which is depicted as red patterned circle in Fig. A.15(a). Now, imagine a line which is perpendicular to the line segment between q and u passing through c. This line intersects with the line segment between q and a at point a' as shown in Fig. A.15(a). If $c \neq a$, then we get $\angle cqa' > 0$, i.e., $d(q, c) < d(q, a') < d(q, a)$. Therefore, the neighborhood $NH(\rho, 1)$ is nearer to q than $NH'(\rho, 1)$. Hence, the lemma.

Proof of Lemma 2.2. We get the center c of the nearest neighborhood $NH(\rho, 1)$ consisting of u is $c = ((d(q, u) - \rho) \times cos\theta, (d(q, u) - \rho) \times sin\theta)$ in the transformed data space D' as per Lemma 2.1. This neighborhood $NH(\rho, 1)$ is a reverse nearest neighborhood of q iff $\nexists f \in F$ such that $d(f, c) < d(q, c)$ as per Definition 2.3. Hence, the lemma.

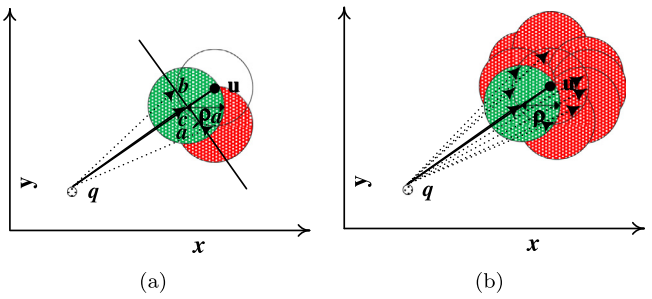


Fig. A.15. Nearest neighborhood of a query facility q consisting of an arbitrary user u.

Proof of Lemma 2.3. If $d(q,c) < \rho$, then the center c of the neighborhood $NH(\rho,k)$ can always be the query facility q itself. That is, $d(q,c) = d(q,q) = 0$. Hence, the lemma.

Proof of Lemma 2.4. As u is a reverse nearest neighbor of q, the center c of the neighborhood $NH(\rho,1)$ consisting of u will be nearer to q than any other facility $f \in F$ as per the definition of reverse nearest neighbor (RNN) and Lemma 2.2. Hence, the lemma.

Proof of Lemma 2.5. All users $u \in RNN(q)$ is a member of $CRU(q)$ as per Lemma 2.4. Now, assume an arbitrary user $u' \in U \setminus RNN(q)$ and $\exists f \in F$ such that $NF(u') = f$ and $d(q,u') = d(f,u') + \epsilon$. Also, assume that $d(q,f) \geq d(q,u)$, and f, q, u' and the center c of the neighborhood $NH(\rho,1)$ consisting of u' are collinear. We get:

$$
\begin{aligned}
d(q,c) &= d(q,u') - \rho && \text{as per Lemma 2.1}\\
d(q,c) - \epsilon &= d(q,u') - \rho - \epsilon &&\\
d(q,c) - \epsilon &= d(f,u') - \rho && \text{as per assumption}\\
d(q,c) &= (d(f,c) - \rho) + \epsilon - \rho && \text{as per Lemma 2.1}\\
d(q,c) &= d(f,c) + \epsilon - 2\rho &&
\end{aligned}
$$

If $\rho \geq \frac{1}{2}\epsilon$, then we get $d(q,c) \leq d(f,c)$. Again, assume that $\nexists f' \in F \setminus f$ such that $d(f',c) < d(q,c)$. This means that u' is an RNH user of q. Therefore, $RNN(q) \subseteq CRU(q)$. Hence, the lemma.

Proof of Lemma 2.6. The proof of this Lemma immediately follows Definitions 2.5 and 2.4. Hence, the Lemma.

Proof of Lemma 4.1. As c_{D_l} is the center of the neighborhood $NH(\rho,1)$ consisting of the closest user u of q in D_l and is computed as per Lemma 2.2, c_{D_l} is the center of the NEC to the query facility q. Since there exists a pivot facility f for which $d(q,c_{D_l}) > d(f,c_{D_l})$, any user from partition D_l cannot be an RNH user of q as per Lemma 2.2 and partition D_l can be pruned. Hence, the lemma.

Proof of Lemma 4.2. Any point c that lies outside of $R_q(f)$ finds f nearer than q, i.e., $d(f,c) < dist(q,c)$ (half-space-pruning [3]). As $\bigcap_{\forall f \in F} R_q(f)$ is the intersection of $R_q(f)$ of all facilities $f \in F$, $\exists f \in F$ for which we get $d(f,c) < d(q,c)$ iff c lies outside of $\bigcap_{\forall f \in F} R_q(f)$. Therefore, any neighborhood $NH(\rho,k)$ finds q nearer than any facility $f \in F$ iff its center c lies within $\bigcap_{\forall f \in F} R_q(f)$. Hence, the lemma.

Proof of Lemma 4.3. The proof of this lemma immediately follows Lemma 4.2.

Proof of Lemma 4.4. As any arbitrary point $c \in CRC(q)$ finds q nearer than any other facility $f \in F$, we can always find a neighborhood $NH(\rho,1)$ with center c consisting of u such that $d(NH(\rho,1),q) \leq d(NH(\rho,1),f), \forall f \in F$ if $d(CRC(q),u) \leq \rho$. Hence, the lemma.

Proof of Lemma 4.5. Every user $u \in CRC(q)$ finds q nearer than other facility $f \in F$ as per Lemma 4.2, i.e., each user $u \in CRC(q)$ is a candidate reverse nearest neighbor user of q. Now, any user $u_1 \in U$ is a candidate reverse nearest neighbor user of q if $d(u_1, CRC(q)) \leq \rho$ as per Lemma 4.4. Since, vertices u' and v' are constructed by extending the lower-left and upper-right corners of the MBR of the vertices of the polygon $CRC(q)$, the MBR consisting of the vertices u' and v' as the lower-left and upper-right corners is an upper bound of $SRU(q)$, i.e., $\exists u \in SRU(q) \setminus CRC(q) : d(u, CRC(q)) \geq \rho$. Hence, the lemma.

Proof of Lemma 4.6. A reverse nearest neighborhood user of an arbitrary query facility q can be included in more than one neighborhoods of q iff these neighborhoods are spatially overlapped. Since $NH(\rho,k)$ has at least one user that is not included in any other reverse nearest neighborhood $NH_1(\rho,k)$ of q which are spatially overlapped with $NH(\rho,k)$, $NH(\rho,k)$ must be a member of the min cover RNH query of q to cover this user as it is a reverse nearest neighborhood user of q as per Definition 2.4. Hence, the lemma.

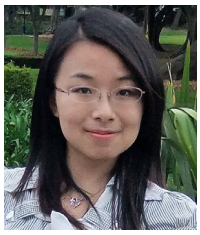
References

[1] W. Liu, W. Sun, C. Chen, Y. Huang, Y. Jing, K. Chen, Circle of friend query in geo-social networks, in: DASFAA, 2012, pp. 126–137.
[2] I. Stanoi, D. Agrawal, A. El Abbadi, Reverse nearest neighbor queries for dynamic databases, in: SIGMOD Workshop, 2000, pp. 44–53.
[3] Y. Tao, D. Papadias, X. Lian, Reverse knn search in arbitrary dimensionality, in: VLDB, 2004, pp. 744–755.
[4] W. Wu, F. Yang, C.Y. Chan, K. Tan, FINCH: evaluating reverse k-nearest-neighbor queries on location data, PVLDB 1 (1) (2008) 1056–1067.
[5] M.A. Cheema, W. Zhang, X. Lin, Y. Zhang, Efficiently processing snapshot and continuous reverse k nearest neighbors queries, VLDB J. 21 (5) (2012) 703–728.
[6] D. Choi, C. Chung, Nearest neighborhood search in spatial databases, in: ICDE, 2015, pp. 699–710.
[7] J. Qi, R. Zhang, Y. Wang, A.Y. Xue, G. Yu, L. Kulik, The min-dist location selection and facility replacement queries, World Wide Web 17 (6) (2014) 1261–1293.
[8] D. Papadias, Y. Tao, K. Mouratidis, C.K. Hui, Aggregate nearest neighbor queries in spatial databases, ACM Trans. Database Syst. 30 (2) (2005) 529–576.
[9] M.A. Cheema, X. Lin, W. Zhang, Y. Zhang, Influence zone: Efficiently processing reverse k nearest neighbors queries, in: ICDE, 2011, pp. 577–588.
[10] M.d. Berg, O. Cheong, M.v. Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, third ed., Springer-Verlag TELOS, Santa Clara, CA, USA, 2008.
[11] M.S. Islam, C. Liu, Know your customer: computing k-most promising products for targeted marketing, VLDB J. 25 (4) (2016) 545–570.
[12] F.P. Preparata, M.I. Shamos, Computational Geometry: An Introduction, Springer-Verlag, Berlin, Heidelberg, 1985.
[13] E.M. McCreight, Priority search trees, SIAM J. Comput. 14 (2) (1985) 257–276.
[14] D.S. Hochbaum, A. Pathria, Analysis of the greedy approach in problems of maximum k-coverage, Nav. Res. Logist. 45 (6) (1998) 615–627.
[15] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, S. Teng, On trip planning queries in spatial databases, in: SSTD, 2005, pp. 273–290.
[16] A. Aggarwal, H. Imai, N. Katoh, S. Suri, Finding k points with minimum diameter and related problems, J. Algorithms 12 (1) (1991) 38–56.
[17] F. Li, B. Yao, P. Kumar, Group enclosing queries, IEEE Trans. Knowl. Data Eng. 23 (10) (2011) 1526–1540.
[18] K. Deng, S.W. Sadiq, X. Zhou, H. Xu, G.P.C. Fung, Y. Lu, On group nearest group query processing, IEEE Trans. Knowl. Data Eng. 24 (2) (2012) 295–308.
[19] A. Skovsgaard, C.S. Jensen, Finding top-k relevant groups of spatial web objects, VLDB J. 24 (4) (2015) 537–555.
[20] X. Cao, G. Cong, T. Guo, C.S. Jensen, B.C. Ooi, Efficient processing of spatial group keyword queries, ACM Trans. Database Syst. 40 (2) (2015) 13.
[21] B. Ghosh, M.E. Ali, F.M. Choudhury, S.H. Apon, T. Sellis, J. Li, The flexible socio spatial group queries, PVLDB 12 (2) (2018) 99–111.

[22] T. Jiang, Y. Gao, B. Zhang, Q. Liu, L. Chen, Reverse top-k group nearest neighbor search, in: Proceedings of the International Conference on Web-Age Information Management (WAIM), 2013, pp. 429–439.
[23] B. Zhang, T. Jiang, Z. Bao, R.C. Wong, L. Chen, Monochromatic and bichromatic reverse top-k group nearest neighbor queries, Expert Syst. Appl. 53 (2016) 57–74.
[24] M.E. Ali, E. Tanin, P. Scheuermann, S. Nutanong, L. Kulik, Spatial consensus queries in a collaborative environment, ACM Trans. Spatial Algorithms Syst. 2 (1) (2016) 3:1–3:37.
[25] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: KDD, 1996, pp. 226–231.
[26] D. Wu, C.S. Jensen, A density-based approach to the retrieval of top-k spatial textual clusters, in: CIKM, 2016, pp. 2095–2100.

Md. Saiful Islam is a Lecturer in the School of Information and Communication Technology, Griffith University, Australia. He has finished his Ph.D. in Computer Science and Software Engineering from Swinburne University of Technology, Australia in February, 2014. He has received his BSc (Hons) and MS degree in Computer Science and Engineering from University of Dhaka, Bangladesh, in 2005 and 2007, respectively. His current research interests are in the areas of database usability, spatial data management and big data analytics.

Bojie Shen is a Ph.D. student in Monash University, Australia. He completed his Bachelor (Honors) in Monash University, Australia in 2019. His current research interests are in the areas of spatial data management, big data analytics and artificial intelligence.



Can Wang received the B.Sc. and M.Sc. degrees from Wuhan University, China, in 2007 and 2009, respectively, and the Ph.D. degree in computing sciences from the Advanced Analytics Institute, University of Technology Sydney, NSW, Australia, in 2013. She worked as a postdoctoral fellow with the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia from 2014 to 2016. She is currently a lecturer in Griffith University, Australia. Her current research interests include data analytics, artificial intelligence, machine learning, and big data.

David Taniar holds Bachelor, Master, and Ph.D. degrees — all in Computer Science, with a particular specialty in Databases. His current research interests cover spatial query processing, and parallel databases. He has published over 130 research papers that can viewed at the DBLP server. He is a founding editor-in-chief of International Journal of Data Warehousing and Mining, International Journal of Web and Grid Services, and International Journal of Web Information Systems. He is currently an Associate Professor at the Faculty of Information Technology, Monash University, Australia.

Junhu Wang received his Ph.D. in Computer Science from Griffith University, Australia in 2003. He is currently an associate professor at the School of Information and Communication Technology, Griffith University. His research interests include query processing, integrity constraint reasoning, and data analytics.