Information Systems 54 (2015) 92–112

Contents lists available at ScienceDirect

Information Systems

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

Reverse nearest neighbor search with a non-spatial aspect

Jeong-hoon Park [a], Chin-Wan Chung [a,b,*], U Kang [a]

[a] The School of Computing, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea
[b] The Chongqing Liangjiang KAIST International Program, Chongqing University of Technology (CQUT), Chongqing, China

ARTICLE INFO

Article history:
Received 16 December 2014
Received in revised form
10 May 2015
Accepted 26 June 2015
Available online 9 July 2015

Keywords:
Reverse nearest neighbor
Spatial
Non-spatial
RNN
LBS

ABSTRACT

With the recent surge in the use of the location-based service (LBS), the importance of spatial database queries has increased. The reverse nearest neighbor (RNN) search is one of the most popular spatial database queries. In most previous studies, the spatial distance is used for measuring the distance between objects. However, as the demands of users of the LBSs are becoming more complex, considering only the spatial factor as a distance measure is not sufficient. For example, through a hotel finding service, users want to choose a hotel considering not only the spatial distance, but also the non-spatial aspect of the hotel such as the quality which can be represented by the number of stars. Therefore, services that consider both spatial and non-spatial factors in measuring the distance are more useful for users. In such a case, techniques proposed in the previous studies cannot be used since the distance measure is different. In this paper, we propose an efficient method for the RNN search in which a distance measure involves both the spatial distance and the non-spatial aspect of an object. We conduct extensive experiments on a large dataset to evaluate the efficiency of the proposed method. The experimental results show that the proposed method is significantly efficient and scalable.

1. Introduction

In recent years, location-based service (LBS) becomes one of the most popular trends in Web-based services. Because LBSs utilize spatial position information, diverse techniques in the spatial database area are employed. Among many spatial database techniques supporting LBSs, the reverse nearest neighbor (RNN) search and its variations are broadly used. Given a query object, the RNN search finds the set of objects that consider the query object as their nearest neighbor. For example, the advertisement target finding is an application using the RNN search. For a given cafe, the service finds the homes for which the cafe is closer than all the other cafes by using the RNN search. Since the homes in the result are considered as the potential customers, the cafe can send the brochures and coupons to the homes.

Lots of researches on the RNN search [1–11] have been conducted over the last decade. In [1–6], methods for finding the reverse nearest neighbor from a snapshot of a dataset are devised. The authors in [7–11] propose methods for the continuous nearest neighbor search which find the reverse nearest neighbor and continuously update the result as the objects change their locations. All the researches use only the spatial distance for measuring the distance between objects. However, in many real-life LBSs, we are given more information than just the physical locations of objects. Recent services

* Corresponding author.
E-mail addresses: jhpark@islab.kaist.ac.kr (J.-h. Park), chungcw@kaist.edu (C.-W. Chung), ukang@cs.kaist.ac.kr (U. Kang).

http://dx.doi.org/10.1016/j.is.2015.06.010
0306-4379/

such as Google Maps,[1] Facebook[2] and Groupon[3] provide users with rating scores of products. Such information is valuable in defining a new distance measure in the RNN search. For example, when choosing a premium-grade steakhouse for dinner, we generally consider not only the spatial proximities of steakhouses, but also the quality of the steakhouse based on items such as the food, atmosphere, price and service. Therefore, in order to choose the better steakhouse, the spatial proximity and the quality of the steakhouse are necessary to be comprehensively considered. In such a case, the traditional distance measure based solely on the spatial distance cannot be used. Consequently, the distance measure based on both the spatial proximity and the quality of the item is more useful and realistic than the traditional measure.

In this paper, given a set of items $\hat{I}$ and a set of users U, we introduce a new distance called the IU (Item-User) distance which is a distance measure between an item and a user considering the spatial distance and the non-spatial aspect of the item. Then, we propose an efficient method for the problem of the reverse nearest neighbor search with the non-spatial score. Specifically, given a query item, the problem is to find the users to which the query item is the nearest item based on the IU distance. Our problem supports updates of users and items, and the values of parameters for the IU distance can be given at the query time instead of being pre-determined. Since the distance measure is different from the previous researches, traditional approaches cannot be used in our case.

In the proposed method, we define the domination relationship among items, by using the properties of the domination relationship, we devise an efficient algorithm, given a query item I_q, to find the items having the domination relationships with I_q. Since an item dominated by another item cannot be a candidate of the nearest item for all the users w.r.t. the IU distance, the dominated items can be pruned without considering users. Then, we propose an efficient algorithm for the RNN search in which a 2-layered structure is devised to avoid redundant visits in synchronously traversing 2 R-tree indexes, one for users and the other for items. In addition, the three pruning techniques are developed and used in the algorithm utilizing the threshold of the IU distance and the spatial distance. These techniques incrementally prune the items and users considering each other.

The applications of our problem include a more practical marketing support system. People want to find a gas station such that the total cost for visiting the gas station and filling the gas is minimum based on their locations. Therefore, the marketing targets of a gas station are promising buyers for which visiting the gas station is more economical than visiting other gas stations. The method for our problem can find such promising buyers by considering the spatial distance and the price as the non-spatial aspect.

To the best of our knowledge, this is the first work that addresses this problem. Note that our problem cannot be a special case of the reverse spatial and textual RNN search [12]. It is because the existing spatial and textual RNN search problem is a monochromatic search which considers single type objects while our problem is a bichromatic search that considers two types of objects.

Our contributions are as follows:

- ***(Introduction of the bichromatic RNN search with a non-spatial aspect)*** We firstly address the problem of the bichromatic reverse nearest neighbor search with a non spatial aspect. In our reverse nearest neighbor search problem, instead of the traditional distance measure, we employ a new distance measure named the IU(Item-User) distance that considers both the spatial proximity and the quality of the item.
- ***(Item pruning method)*** We propose an effective method to filter out items before performing the RNN search. We define the domination relationship among items. Then, only the domination relationships of items with a given query item are used for filtering items. As a result, the proposed method can reduce the search space of items without considering the location of users. In addition, we statistically analyze the performance of the item domination.
- ***(Novel search algorithms)*** We propose a novel search algorithm for the RNN search with a non-spatial aspect, based on a 2-layered structure for maintaining the contexts of search between users and items. By the 2-layered structure, the algorithm avoids redundant computations. In addition, three pruning techniques are devised and used in the algorithm for incrementally reducing the search spaces for users and items.
- ***(Experiments on a large data set)*** We conduct extensive experiments for evaluating the efficiency of the proposed method using synthetic datasets and real datasets. The experimental results show that the proposed method is at least 4 times more efficient than an adapted version of an existing method

This paper is organized as follows. Section 2 reviews related works on the reverse nearest neighbor search. The problem is formally defined in Section 3. In Section 4, we present the index strategy. Section 5 describes the item domination and a method for the item pruning using the item domination relationship. An efficient algorithm for the RNN search is proposed in Section 6. In Section 7, we present experimental results. Finally, conclusions are made in Section 8.

[1] https://maps.google.com
[2] http://facebook.com
[3] http://www.groupon.com

2. Related works

The studies related to the reverse nearest neighbor(RNN) search have been conducted for the last decade. It is divided into two categories – the snapshot reverse nearest neighbor search and the continuous reverse nearest neighbor search.

The snapshot reverse nearest neighbor search is to find the RNN of a given query object from a static set of objects.

In [1], the authors firstly propose a method for processing the RNN query. The method is based on the off-line processing. For each object o_i, the method finds the nearest neighbor o_j, and computes a circle whose center is o_i and radius is the distance between o_i and o_j. The result of the RNN search of the query object q is the set of objects which contain q in their circles.

The authors in [4] propose a query time processing method for the RNN search. In the method, the off-line processing is not necessary. Given a query point q, the method divides the universe area into six fan-shaped regions which have the same area. For each region, only the nearest object from q among the objects in the region is a candidate object. For each of the maximum six candidate objects, if the nearest neighbor of the candidate object is q, the candidate object belongs to the result of the RNN search. By using this property, the method finds the final answer of the RNN search.

The authors in [5] propose a query time processing method for the reverse k nearest neighbor (RkNN) search. The method is based on the perpendicular bisectors among objects. Given an object o and a query object q, by the perpendicular bisector of o and q, the universe region is divided into two regions $H_{o:q}$ and $H_{q:o}$. $H_{o:q}$ contains o and $H_{q:o}$ contains q. For all the objects in $H_{o:q}$, o is closer than q. In contrast, for all the objects in $H_{q:o}$, q is closer than o. If an object is located in more than k number of $H_{o_i:q}$ s, the object cannot be the answer of the RNN search for the query q. By using this property, appropriate algorithms are proposed in [5].

The continuous reverse nearest neighbor (CRNN) search includes not only the snapshot RNN search, but also maintaining the results of the RNN search as objects change their locations.

In [7], the authors firstly devise a method for the CRNN search. However, they assume that there is a moving pattern for each object.

The method proposed in [8] utilizes the six fan-shaped regions based approach proposed in [4]. As the objects change their locations, the method manages the candidate objects and their circles, each of which contains the nearest neighbor of the corresponding candidate object.

The authors in [10] propose a method for the continuous reverse k nearest neighbor (CRkNN) search. It is also based on the six fan-shaped regions based approach proposed in [4]. The method manages the k nearest neighbor in each region, which are the candidates of the result of the RkNN search. In order to continuously maintain the results of the RkNN search, for each candidate object, the method continuously monitors the circle of the candidate object which contains k nearest neighbors.

The authors in [9] propose a method for the CRNN search based on the method proposed in [5] which utilizes the perpendicular bisectors among objects. The method monitors and updates the result of the RNN search by managing grid cells and pruning areas. For the grid cells in a pruned area, the objects moved to the cells are filtered out. For the grid cells in unpruned area, each object moved to one of the grid cells is checked whether the object is an answer or not.

In [11], the authors assume that the query object can change its location. The safe zone for the query object is continuously updated when the query object or the others change their locations. Then, pruning techniques and appropriate algorithms are devised for continuous monitoring of the result of the RkNN search. The purpose of this study is to reduce not only the computational time, but also the cost for communications between a server and clients.

The studies mentioned above are based on the spatial distance. However, in our problem, we use a different distance measure considering not only the spatial distance but also the preference score of an object. Therefore, the techniques devised in the related works are incompatible to our problem.

There are some studies for the RNN search in which the environment is the metric space [13], the high dimensional space [14], or large graphs [15]. Since the environments of these studies are different from those of our problem, details of them are not mentioned in this paper.

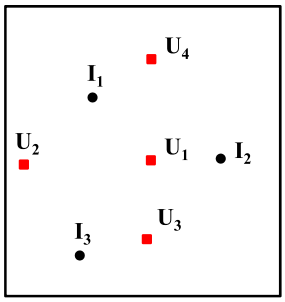
The reverse top k (RTOPk) problem studied in [16–18] is related to our problem. Given a set of user interesting vectors, a set of multi-dimensional items and a query item, the RTOPk problem is to find the set of user interesting vectors such that, for each vector, the ranking score of the query item is better than or equal to the kth ranking score among the items. The item and the user of our problem correspond to the item and the user interesting vector of the RTOPk problem, respectively. The shared features between two problems are as follows:

1. The items are differently ranked by a given ranking function according to users.
2. The result is the set of users who consider the query item as one of the best k items.

However, in the detailed level, the ranking function and the dataset setting are different. For instance, at the query time, the spatial distance values of each item are different to different users in our problem while all the values of dimensions of dataset are determined before the query time in the RTOPk search problem. It means that the spatial distance values for items should be computed in the query processing of our problem. In addition, two datasets are weight vectors and items in the RTOPk problem, while two datasets are users and items, and a single weight vector is given at the query time in our method. Because of the differences of the datasets, our problem cannot be directly translated to the RTOPk problem. Especially, the method in [16] that conducts the top k search for each user and reuses the result of the previously conducted top k search can be used to solve our problem by using the ranking function in our problem instead of the ranking function

Table 1
Summary of notations.

Notation	Description
$u^i \in U$	A user
$I^j \in \hat{I}$	An item
$IUD(I_j, u_i)$	The IU distance between I_j and u_i
α	The weighting parameter for the IU distance
$sScore(I_j, u_i)$	The spatial score between I_j and u_i
$MaxD$	The maximum possible spatial distance
$I_j.p$	The score of the non-spatial aspect of I_j
$RNN_{IU}(I_q)$	The set of users to which I_q is their nearest item
R^U	The R-tree for users
R^I	The adapted R-tree for items
N^U, N^I	The nodes of R^U and R^I, respectively
$D(N^U)$, $D(N^I)$	The descendant items of N^U and N^I, respectively
$R(N)$ or $N.MBR$	The minimum bounding rectangle(MBR) of N
$\mathrm{Min}P(N_j^I), \mathrm{Max}P(N_j^I)$	The min. and max. $pScore$among $D(N_j^I)$, respectively



Item	*User*	*sScore*	*IU - Distance*
I_1	U_1	0.29	0.335
I_1	U_2	0.33	0.355
I_1	U_3	0.53	0.455
I_1	U_4	0.25	0.315
I_2	U_1	0.25	0.325
I_2	U_2	0.68	0.54
I_2	U_3	0.39	0.395
I_2	U_4	0.42	0.41
I_3	U_1	0.41	0.42
I_3	U_2	0.37	0.4
I_3	U_3	0.25	0.34
I_3	U_4	0.71	0.57

Fig. 1. An example of the RNN search based on the IU-distance.

in [16]. However, conducting top k search for each user is time consuming task when the dataset becomes large, and the efficiency is sensitive to the order of top k searches. In addition, Vlachou et al. [18] proposes a problem directly translated to a continuous version of our problem by employing a spatial distance as a dimension. However, the method only focuses on the monitoring part. Therefore, our problem cannot be covered by the method in [18].

Several works use the non-spatial aspect for measuring the distance. Top k spatial keyword query [19], LkT query [20], LkPT query [21], and Reverse spatial textual k nearest neighbor query [12] employ the keyword similarity as the non-spatial aspect. The problems in [19–21] are not related to the RNN search. Especially, our problem is not a special case of the problem addressed in [12] because the problem in [12] is a monochromatic search problem which considers single type objects.

3. Problem definition

In this section, we formally define related concepts and the problem to be solved in this paper. In addition, we will discuss about a newly proposed distance measure, and explain the inadequacy of using the existing techniques to solve our problem.

Table 1 summarizes the notations used throughout this paper.

3.1. Definitions

Definition 1 (*User*). A user $u \in U$ is defined as $u = (id, loc)$ where id is the identifier of u, and loc is the location of u in the 2-dimensional geometric space, denoted by $loc = (x, y)$.

We assume that users can change their locations, and the width and height of the universe area are 1s.

Definition 2 (*Item*). An item $I \in \hat{I}$ is defined as $I = (id, loc, p)$ where id is the identifier of I, loc is the location of I in the 2-dimensional geometric space denoted by $loc = (x, y)$, and p is the non-spatial score of I which is called $pScore$. The range of p is from 0 to 1 where the lower value means the better number for users.

We assume that loc and p of an item can be changed.

We define the Item-User (IU) distance. We follow the existing works [21,20,19,12], dealing with subjects different from ours, which consider non-spatial aspects in their distance measures. In these works, the authors employ the weighted sum to combine the spatial distance and the non-spatial score, with a fixed value α as the weight.

Definition 3 (*Item-User distance*). The item-user (IU) distance is the distance between an item and a user considering the spatial distance and the non-spatial score of the item. The IU distance between $u_i \in U$ and $I_j \in \hat{I}$ is formally defined as

$$IUD(I_j, u_i) = \alpha \times sScore(I_j, u_i) + (1-\alpha) \times I_j.p \tag{1}$$

where α is the weighting parameter whose value range is [0,1] and *sScore* is the spatial score between two spatial objects. The spatial score between u_i and I_j is calculated by

$$sScore(I_j, u_i) = \frac{eDist(u_i, I_j)}{\text{Max}D} \tag{2}$$

where *eDist* is the Euclidean distance between the two input objects u_i and I_j, and *MaxD* is the maximum possible Euclidean distance between u_i and I_j. In our assumption of the universe area, *MaxD* is $\sqrt{2}$. Since $sScore(\cdot) \in [0,1]$, $I_j.p \in [0,1]$, and $\alpha \in [0,1]$, we can realize that $IUD(I_j, u_i) \in [0,1]$.

We assume that the value of α is set by domain experts.

For a user $u_i \in U$, the nearest neighbor of u_i among the items in $\hat{I}$ based on the IU distance is $NN_{IU}(u_i, \hat{I}) = \arg\min_{I_j \in \hat{I}} IUD(I_j, u_i)$.

Given a query item $I_q \in \hat{I}$, the problem of this paper is to find the users u_i such that $NN_{IU}(u_i, \hat{I}) = I_q$. It means that $\forall I_j \in \hat{I}, IUD(I_q, u_i) \le IUD(I_j, u_i)$. We denote the reverse nearest neighbor of a query item I_q based on the IU distance as $RNN_{IU}(I_q)$.

Note that our problem is categorized in the bichromatic RNN search that is based on two types of datasets. In the bichromatic RNN searches, the distance between two objects from two different types of datasets, respectively, is used.

Fig. 1 shows an example for the reverse nearest neighbor search based on the IU distance. There are 3 items and 4 users. In the example, the non-spatial scores of the items I_1, I_2, I_3 are 0.38, 0.4, and 0.43, respectively, and the weight factor α is 0.5. In the table in the figure, there are spatial scores and IU distances between items and users. As a result, $RNN_{IU}(I_1) = \{u_2, u_4\}, RNN_{IU}(I_2) = \{u_1\}$, and $RNN_{IU}(I_3) = \{u_3\}$.

3.2. A guideline for deciding α

The newly proposed distance measure, the IU distance, considers not only the spatial factor, but also the non-spatial factor. The parameters for the IU distance are inputted by the query issuer at the query time. Especially, since determining α is a difficult problem, it is necessary to discuss about guidelines for setting α. In order to set α, both *sScore* and *pScore* should be comprehensively considered according to the application domain. We propose a guideline for setting α as follows:

- Assume that we know the costs corresponding to the maximum *sScore* and the maximum *pScore* denoted by *mcost*(*sScore*) and *mcost*(*pScore*), respectively. The texi fare for *MaxD* is an example of *mcost*(*sScore*), and the maximum price of an item is an example of *mcost*(*pScore*). Then, we recommend to set α as $mcost(sScore)/(mcost(sScore)+mcost(pScore))$. For example, if *sScore* is one and the corresponding cost is ten dollars, and if *pScore* is one and the corresponding cost is one hundred dollars. α is set to $\frac{1}{11}$.

3.3. Inadequacy of existing techniques

In this section, we describe the reason why the existing studies cannot be utilized for solving our problem.

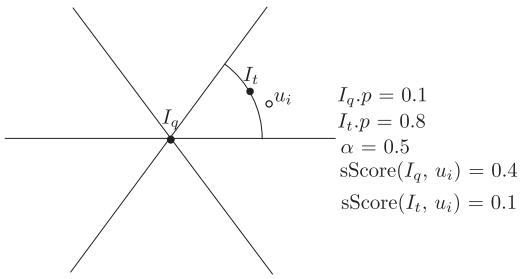


Fig. 2. An example of inadequacy of the six-region based approach.

The two major categories of the solutions for the traditional RNN search are the six-region based approach and the half-space based approach. Since the pruning techniques and algorithms of the methods in the two categories are based on only the spatial distance, they cannot be used for solving our problem because our problem uses the IU distance that considers not only the spatial distance, but also a non-spatial distance.

The spatial and texture reverse nearest neighbor (STRNN) search is similar to our problem in terms of using a non-spatial factor for the distance measure. However, the problem is a monochromatic search problem which uses the single type of objects having the spatial location and the set of keywords. In our problem, we have two types of objects. One type of objects have only the spatial location and the other type of objects have both the spatial location and the non-spatial score. The method for the monochromatic reverse spatial and textual kNN proposed in [12] can be extended to a method for the bichromatic version. However, in order to optimize the solution extended from the monochromatic version, it is necessary to carefully study the features of the bichromatic problem. It is because, in general, there are diverse options to extend a monochromatic method to a bichromatic method. In addition, even though an optimized extended method for the bichromatic problem is devised, there are several issues for utilizing the extended method for solving our problem as follows:

- The pruning methods proposed based on the textual relevancy in [12] are not applicable to our problem.
- The index structure proposed in [12], IUR-tree, is not compatible to the dataset of our problem.
- The clustered IUR-tree and text-entropy based optimization techniques in [12] are only for the spatial and textual queries.

4. Index strategy

4.1. Construction of indexes

In this section, we describe a method to construct the indexes for users and items. In order to support updates of data without a large amount of the computational cost, we use R-tree based indexes for the users and items. The users are indexed by using the original version of the R-tree index [22]. However, for items, the original R-tree index cannot cover *pScore* of the item. In order to support *pScore*, we adapt the R-tree index. In the adapted R-tree, for each R-tree node, we

a

Item	X	Y	p
I_1	5	3	4
I_2	7	6	2
I_3	4	5	1
I_4	5	8	5
I_5	5	9	6
I_6	2	9	2
I_7	2	10	3
I_8	12	6	4
I_9	13	7	5
I_{10}	14	8	7
I_{11}	9	10	2
I_{12}	12	10	3
I_{13}	10	10	4
I_{14}	11	10	8

b

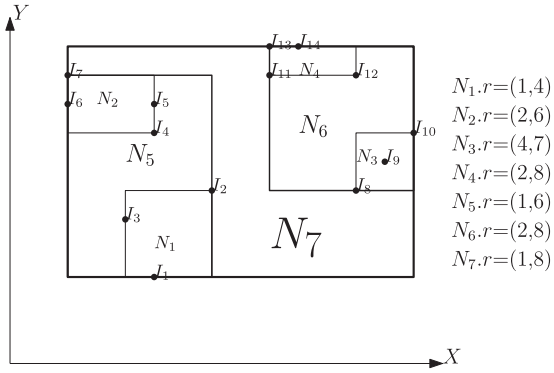


Fig. 3. An example of the adapted R-tree indexing items. (a) A set of items. (b) The adapted R-tree.

record the minimum and the maximum of *pScores* of the descendant items. The descendant items of a node *N* means all the items such that the leaf nodes pointing them are descendant nodes of *N*. We denote *N.r* as the pair of the minimum score and the maximum score. When a node is updated, the maximum *pScore* and minimum *pScore* are updated using those of its child nodes. In the proposed algorithm for the RNN search, the minimum and maximum *pScores* will be used for pruning unnecessary R-tree nodes from the search space.

Fig. 3 shows an example of the adapted R-tree indexing the items. Note that, for easy understanding, we do not normalize the values in this example. Each node has a range of scores from the minimum *pScore* to maximum *pScore*. For the node N_6 in Fig. 3(b), the seven items from I_8 to I_{14} are the descendant items. As we can see in Fig. 3(a), I_{11} has the minimum *pScore* 2 and I_{14} has the maximum *pScore* 8. We denote the R-tree index for users as R^U, and the adpated R-tree for items as R^I.

Since R-tree like structures are broadly used in spatial databases, in many cases, the proposed method based on the R-tree like indexes is compatible to the existing spatial databases without additional efforts for adjusting indexes.

4.2. Update of indexes

In this section, we explain how to efficiently update the indexes as the set of users and the set of items are updated. Since the original R-tree index is constructed for users, we follow the R-tree update method proposed in [22] for the index for users. However, since the adapted R-tree for items manages not only the spatial information of items, but also the non-spatial scores of items, we need to discuss how to support the update of items. We categorize the updates of items into the insertion, deletion, and modification. Since the modification is done by the insertion after deletion, we only focus on the insertion and deletion methods.

(Insertion) When a new item is added to the index, the proposed method basically follows the insertion method of the original R-tree. During the update of the R-tree nodes in the insertion method, our method updates the minimum and maximum scores of each node *N* such that the inserted item is in *D*(*N*) where *D*(*N*) is the set of items indexed in the subtree rooted at *N*. Algorithm 1 is an algorithm for adding an item to the adapted R-tree. The algorithm inserts the new item into a leaf node in Lines 1–4. In Lines 5–18, the algorithm splits nodes in the adapted R-tree and computes the minimum and maximum *pScores* for each updated node. In Lines 19–23, the *pScore* information of the ancestor nodes of the updated nodes are updated.

Algorithm 1. An algorithm for adding an item into the adapted R-tree.

```
   input  : R^I - The adapted R-tree indexing items
            I_new - The item to insert
1  begin
2      N := chooseLeafNode(R^I, I_new);
3      Insert I_new into N;
4      Compute N.r and N.MBR;
5      while N needs to be split do
6          (N_1, N_2) := split(N);
7          Compute N_1.r, N_2.r and MBRs of N_1 and N_2;
8          if N is the root node then
9              N := Create a new node;
10             Insert N_1 and N_2 into N;
11             Compute N.r and N.MBR;
12             Make N the root node of R^I;
13             Break;
14         else
15             Delete N from the parent node of N;
16             N := the parent node of N;
17             Insert N_1 and N_2 into N;
18             Compute N.r and N.MBR;
19     while N is not the root node do
20         N := the parent node of N;
21         Compute N.r;
22         if N.r and N.MBR are not changed then
23             break;
```

(Deletion) When an existing item is deleted from the index, the proposed method basically follows the deletion method of the original R-tree. During the update of the R-tree nodes in the deletion method, our method additionally updates the minimum and maximum scores of each node *N* such that the deleted item was in *D*(*N*) where *D*(*N*) is the set of items indexed in the subtree rooted at *N*. Algorithm 2 is an algorithm for deleting an item from the adapted R-tree. The algorithm deletes an item from a leaf node in Lines 1–4. In Lines 5–14, the algorithm merges nodes in the adapted R-tree and computes the minimum and maximum *pScores* for each updated node. In Lines 15–19, the *pScore* information of the ancestor nodes of the updated nodes is updated.

Algorithm 2. An algorithm for deleting an item from the adapted R-tree.

```
input  : R^I - The adapted R-tree indexing items
         I_del - The item to delete
begin
    N := the leaf node referring to I_del;
    Delete the pointer to I_del from N;
    Compute N.r and N.MBR;
    while N needs to be merged do
        if N is the root node then
            Break;
        N_pnt := the parent node of N;
        N_m := chooseSiblingNode(N, R^I);
        N_new := merge(N_m, N);
        Delete N and N_k from N_pnt;
        Insert N_new into N_pnt;
        Compute N_new.r, N_pnt.r, and MBRs of N_new and N_pnt;
        N := N_pnt;
    while N is not the root node do
        N := the parent node of N;
        Compute N.r and N.MBR;
        if N.r and N.MBR are not changed then
            break;
```

5. Item domination and pruning

As described in Appendix A, a relative safe zone is bound by a curve or a line according to the *pScores* of two items. We find that, given two items I_x and I_y where $I_x.p > I_y.p$, the relative safe zone of I_x is the empty area when the difference of *pScores* is large. In such a case, the safe zone of I_x is also the empty area since the safe zone is the intersection of all the relative safe zones of I_x. Also, I_x cannot affect the safe zone of I_y since the relative safe zone of I_y w.r.t. I_x is the universe area. We say that I_y dominates I_x in this case. We can use the domination relationship for pruning items in order to reduce the search space. Given a query item, if the query item is dominated by at least one item, we can guarantee that the answer of the RNN search is the empty set. If an item is dominated by the query item, we can guarantee that the dominated item does not need to be considered for the RNN search of the query item.

In this section, we will formally present the principle of the item domination, the statistical analysis for the performance of the item domination, and the method for the item pruning.

5.1. Item domination

As described above, given two items, one item can have an empty relative safe zone w.r.t. the other item. In order to find such cases, we provide Lemma 1.

Lemma 1. *Given items I_x and I_y, if $((1-\alpha)/\alpha)(I_x.p - I_y.p) > sScore(I_x, I_y)$, for all users $u_i \in U, IUD(I_y, u_i) < IUD(I_x, u_i)$.*

Proof. We will use contradiction. Assume that there exists a user u^* s.t. $IUD(I_y, u^*) \geq IUD(I_x, u^*)$ where $((1-\alpha)/\alpha)(I_x.p - I_y.p) > sScore(I_x, I_y)$. Then,

$$\begin{aligned} &\alpha \cdot sScore(I_y, u^*) + (1-\alpha) I_y.p \geq \alpha \cdot sScore(I_x, u^*) + (1-\alpha) \cdot I_x.p \\ &\Leftrightarrow sScore(I_y, u^*) - sScore(I_x, u^*) \geq \frac{1-\alpha}{\alpha}(I_x.p - I_y.p) \end{aligned} \tag{3}$$

By the triangle inequality of the Euclidean distance

$$\begin{aligned} &eDist(I_y, u^*) \leq eDist(I_x, I_y) + eDist(I_x, u^*) \\ &\Leftrightarrow sScore(I_y, u^*) \leq sScore(I_x, I_y) + sScore(I_x, u^*) \\ &\Leftrightarrow sScore(I_y, u^*) - sScore(I_x, u^*) \leq sScore(I_x, I_y) \end{aligned} \tag{4}$$

Consequently,

$$\frac{1-\alpha}{\alpha}(I_x.p - I_y.p) \leq sScore(I_y, u^*) - sScore(I_x, u^*) \leq sScore(I_x, I_y) \tag{5}$$

Then, $((1-\alpha)/\alpha)(I_x.p - I_y.p) \leq sScore(I_x, I_y)$ is the contradiction w.r.t. $((1-\alpha)/\alpha)(I_x.p - I_y.p) > sScore(I_x, I_y)$ in the assumption about u^*. □

Definition 4 (*Item domination*). Given items I_x and I_y, we say that I_y dominates I_x if $((1-\alpha)/\alpha)(I_x.p - I_y.p) > sScore(I_x, I_y)$.

Theorem 1. *If an item I_q is dominated by at least one of other items, $RNN_{IU}(I_q)$ is $\varnothing$.*

Proof. By Lemma 1, if a query item is dominated by at least one of other items, the query item cannot be the nearest item of any user. It is because, for all the users, the dominating item is closer than the query item based on the IU distance. Therefore, the result of the reverse nearest neighbor of I_q is empty. □

5.2. Statistical analysis for the performance of the item pruning

For the analysis, we assume that the non-spatial scores of items follow a normal distribution. The rationale is derived from [23] which analyzes the user ratings in IMDB.[4] IMDB is one of the most popular rating services for movies. Also, we assume that the locations of items follow a uniform distribution as many researches in spatial databases assume the distribution.

Theorem 2. *Given items I_x and I_y, the probability that I_x is dominated by I_y is $\int_0^{\sqrt{2}} F_X(t) f_Y(t)\,dt + \int_{\sqrt{2}}^{\infty} f_Y(t)\,dt$ where F_X is the cumulative distribution function of $X = eDist(I_x, I_y)$, and f_Y is the probability density function of $Y = (\sqrt{2}(1-\alpha)/\alpha)(I_x.p - I_y.p)$.*

Proof. The probability that I_x is dominated by I_y is $P(sScore(I_x, I_y) < ((1-\alpha)/\alpha)(I_x.p - I_y.p))$. Then,

$$\begin{aligned} &P\left(sScore(I_x, I_y) < \frac{1-\alpha}{\alpha}(I_x.p - I_y.p)\right) \\ &\quad = P\left(eDist(I_x, I_y) < \frac{\sqrt{2}(1-\alpha)}{\alpha}(I_x.p - I_y.p)\right) \\ &\quad = P(X < Y) \\ &\quad = \int_{-\infty}^{\infty} F_X(t) f_Y(t)\,dt \end{aligned} \tag{6}$$

In order to find the distribution functions of X, we can utilize the 'Square Line Picking' problem which is, given $d \in [0, \sqrt{2}]$, when randomly picking two points in the unit square, finding the probability that the distance between the two points is less than or equal to d. Then, by [24],

$$F_X(t) = \begin{cases} \frac{1}{2}t^4 - \frac{8}{3}t^3 + \pi t^2 & \text{for } 0 \le t \le 1 \\ -\frac{1}{2}t^4 - 4t^2 \arctan\left(\sqrt{t^2-1}\right) + \frac{4}{3}(2t^2+1)\sqrt{t^2-1} + (\pi-2)t^2 + \frac{1}{3} & \text{for } 1 \le t \le \sqrt{2} \\ 1 & \text{for } \sqrt{2} \le t \end{cases} \tag{7}$$

In addition, let the random variable of the non-spatial score of I_x be $Z_x \sim N(\mu, \sigma^2)$ and that of I_y be $Z_y \sim N(\mu, \sigma^2)$. Since $Y = (\sqrt{2}(1-\alpha)/\alpha)(Z_x - Z_y)$, we can realize that $Y \sim N(0, K^2)$ where $K^2 = 4((1-\alpha)/\alpha)^2\sigma^2$. Therefore, $f_Y(t) = 1/\sqrt{2\pi}K^2 \exp[-t^2/2K^2]$. Since $F_X(t)$ is 0 for $t < 0$, $\int_{-\infty}^{\infty} F_X(t) f_Y(t)\,dt = \int_0^{\infty} F_X(t) f_Y(t)\,dt$. In addition, since $F_X(t)$ is 1 for $\sqrt{2} \le t$, $\int_{\sqrt{2}}^{\infty} F_X(t) f_Y(t)\,dt = \int_{\sqrt{2}}^{\infty} f_Y(t)\,dt$. As a result, $\int_{-\infty}^{\infty} F_X(t) f_Y(t)\,dt = \int_0^{\sqrt{2}} F_X(t) f_Y(t)\,dt + \int_{\sqrt{2}}^{\infty} f_Y(t)\,dt$

Fig. 4 shows the probabilities that I_y dominates I_x varying on α where the non-spatial scores of items follow the default normal distribution $N(0.5, 0.01)$, which is based on our dataset for the experiments in Section 7. In the graph, we can realize that the probability is high when α is near 0, and low when α is near 1. Also, the probability heavily depends on α that is various according to the application area.

5.3. Item pruning algorithm

By using the item domination relationship, we can prune unnecessary items. There are two tasks in the item pruning method. One is, given a query item, to check if there exists an item which dominates the query item. If there exists an item dominating the query item, we can return the empty set as the result without the RNN search. The other task is to find and mark all the items which are dominated by the query item. If there does not exist an item dominating the query item, the dominated items that are marked can be used for improving the efficiency of the RNN search.

All the items in $\hat{I}$ are indexed in the adapted R-tree R^I in Section 4. Algorithm 3 is an efficient algorithm, associated with the adapted R-tree, for checking the existence of the dominating items, and marking the dominated items. The algorithm gets the weight parameter, the adapted R-tree R^I, and a query item I_q as the inputs. Then, the algorithm

[4] http://www.imdb.com

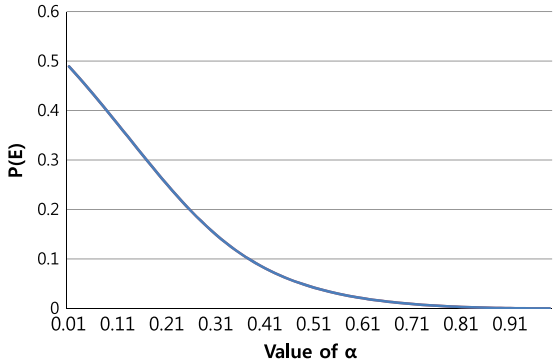


Fig. 4. Given I_x and I_y, the probability that I_y dominates I_x for varying α.

traverses R^I from the top to the bottom in the best-first manner by using a priority queue. In Lines 2–3, the priority queue pq is initialized. In pq , the elements are nodes of R^I and they are always sorted by $E1$ defined in Eq. (8) in the ascending order.

Algorithm 3. An algorithm for checking the existence of a dominating item and marking the dominated items.

input : α - The weight parameter for the IU distance
R^I - The adapted R-tree indexing items
I_q - The query item
output : Yes or No - Yes if there exists a dominating item, and No if does not exists a dominating item.

```
1  begin
2      Initialize a priority queue as pq;
       ;     /* priority is E1 in the ascending order */
3      Insert the root node of R^I into pq;
4      while pq is not empty do
5          N:=Dequeue from pq;
6          if N is a leaf node then
7              for each item I_j to which N points do
8                  if (1-α)/α (I_q.p - I_j.p) > sScore(I_q, I_j) then
9                      return Yes;
10                 if (1-α)/α (I_j.p - I_q.p) > sScore(I_q, I_j) then
11                     Mark I_j as a dominated item;
12         else
13             for each child node N' of N do
14                 if E4(I_q,N') > 0 then
15                     Mark N' as a dominated node;
16                     Continue to the next loop;
17                 if E1(I_q,N') > 0 then
18                     if Max(E2(I_q, N'),E3(I_q,N')) > 0 then
19                         return Yes;
20                     Enqueue N' into pq;
21     Return No;
```

During the traversal, the algorithm stops and returns *YES* when it is guaranteed that there exists at least one item dominating I_q. At the same time, the algorithm marks some nodes and dominated items. If a node is marked, it implies that all the descendant items of the marked node are dominated by I_q. This marking information is utilized in the RNN search algorithm that will be explained in Section 6.

In the algorithm, for checking and marking an item, Lemma 1 is directly used for computing the domination relationship between the item and the query item. This part corresponds to Line 6–11 in the algorithm. For checking and marking a node, the algorithm utilizes Lemma 2–5 as described in Lines 14–20 in the algorithm. For the convenience, before describing the lemmas, we define four equations used in the lemmas as follows:

$$E1\left(I_q, N^I\right) = \frac{(1-\alpha)}{\alpha} \cdot \left(I_q.p - \mathrm{Min}P\left(N^I\right)\right) - \mathrm{Min}S\left(I_q, N^I\right) \tag{8}$$

where Min$P(N^I)$ is the minimum *pScore* of all the descendant items of N^I and Min$S(I_q, N^I)$ is the minimum *sScore* between I_q and the minimum bounding rectangle (MBR) of N^I.

$$E2\left(I_q, N^I\right) = \frac{(1-\alpha)}{\alpha} \cdot \left(I_q.p - \mathrm{Min}P\left(N^I\right)\right) - \mathrm{Max}S\left(I_q, N^I\right) \tag{9}$$

where Max$S(I_q, N^I)$ is the maximum *sScore* between I_q and the MBR of N^I.

$$E3\left(I_q, N^I\right) = \frac{(1-\alpha)}{\alpha} \cdot \left(I_q.p - \mathrm{Max}P\left(N^I\right)\right) - \mathrm{MinMax}S\left(I_q, N^I\right) \tag{10}$$

where Max$P(N^I)$ is the maximum *pScore* of all the descendant items of N^I and MinMax$S(I_q, N^I)$ is $\mathrm{Min}_{f \in FS(N^I)}$ (Max$S(I_q, f)$), $FS(N^I)$ is the set of faces of the MBR of N^I, and Max$S(I_q, f)$ is the maximum among the *sScores* from I_q to all the points on f.

$$E4\left(I_q, N^I\right) = \frac{(1-\alpha)}{\alpha} \cdot \left(\mathrm{Min}P\left(N^I\right) - I_q.p\right) - \mathrm{Max}S\left(I_q, N^I\right) \tag{11}$$

The lemmas used for the algorithm are as follows:

Lemma 2. *Given a query item I_q and a node N^I of R^I, if $E1(I_q, N^I) \le 0$, all the descendent items of N^I cannot dominate I_q.*

Proof. For all descendant item I_j of N^I, it holds that $((1-\alpha)/\alpha) \cdot (I_q.p - I_j.p) - sScore(I_q, I_j) \le ((1-\alpha)/\alpha) \cdot (I_q.p - \mathrm{Min}P(N^I)) - \mathrm{Min}S(I_q, N^I)$. Therefore, if $((1-\alpha)/\alpha) \cdot (I_q.p - \mathrm{Min}P(N^I)) - \mathrm{Min}S(I_q, N^I) \le 0$, we can see that, for all descendant items I_j, $((1-\alpha)/\alpha) \cdot (I_q.p - I_j.p) - sScore(I_q, I_j) \le 0$. According to Lemma 1 and Definition 4, all the descendant items of N^I cannot dominate I_q. □

Lemma 2 is used in Line 17 for checking if node N' can dominate I_q or not.

Lemma 3. *Given a query item I_q and a node N^I of R^I, if $E2(I_q, N^I) > 0$, at least one descendent item of N^I dominates I_q.*

Proof. There must exist an item I^* in the descendant items of N^I whose *pScore* is Min$P(N^I)$· $((1-\alpha)/\alpha) \cdot (I_q.p - I^*.p) - sScore(I_q, I^*) \ge ((1-\alpha)/\alpha) \cdot (I_q.p - \mathrm{Min}P(N^I)) - \mathrm{Max}S(I_q, N^I)$. Therefore, if $E2(I_q, N^I) > 0$, it holds that $((1-\alpha)/\alpha) \cdot (I_q.p - I^*.p) - sScore(I_q, I^*) > 0$. According to Lemma 1, $I^* dominates I_q$. □

Lemma 4. *Given a query item I_q and a node N^I of R^I, if $E3(I_q, N^I) > 0$, at least one descendent item of N^I dominates I_q.*

Proof. There must exist an item I' in the descendant items of N^I, such that $sScore(I_q, I') \le \mathrm{MinMax}S(I_q, N^I)$ according to the property of *MaxNearest* introduced in [25]. We can derive that $((1-\alpha)/\alpha) \cdot (I_q.p - I'.p) - sScore(I_q, I') \ge ((1-\alpha)/\alpha) \cdot (I_q.p - \mathrm{Max}P(N^I)) - \mathrm{MinMax}S(I_q, N^I)$. Therefore, if $E3(I_q, N^I) > 0$, it holds that $((1-\alpha)/\alpha) \cdot (I_q.p - I'.p) - sScore(I_q, I') > 0$. According to Lemma 1, I' dominates I_q. □

Lemmas 3 and 4 are used in Line 18 for checking if there exists a descendant item of N' which dominates I_q.

Lemma 5. *Given a query item I_q and a node N^I of R^I, if $E4(I_q, N^I) > 0$, all the descendent items of N^I are dominated by I_q.*

Proof. For all descendant item I_j of N^I, it holds that $((1-\alpha)/\alpha) \cdot (I_j.p - I_q.p) - sScore(I_q, I_j) \ge ((1-\alpha)/\alpha) \cdot (\mathrm{Min}P(N^I) - I_q.p) - \mathrm{Max}S(I_q, N^I)$. Therefore, if $((1-\alpha)/\alpha) \cdot (I_q.p - \mathrm{Min}P(N^I)) - \mathrm{Min}S(I_q, N^I) \le 0$, we can see that, for all descendant items I_j, $((1-\alpha)/\alpha) \cdot (I_q.p - I_j.p) - sScore(I_q, I_j) \le 0$. □

Lemma 5 is used for checking if I_q dominates all the descendant items of N' or not, in Line 14 of the algorithm.

After terminating the *While* loop by the condition that *pq* becomes empty, the algorithm returns *NO* since the algorithm has not found a dominating item. The marking information is retained to be used in the RNN search.

6. Query processing for the rnn search

In order to find the results of the RNN search based on the IU distance, we propose a branch and bound algorithm. The RNN search is performed when a query is issued to the system and it is guaranteed that there does not exist an item dominating the query item. The purpose of the RNN search is to find $u_i \in U$ such that $\forall I_j \in \hat{I}, IUD(I_q, u_i) \le IUD(I_j, u_i)$. The method can validate the users by synchronously traversing R^I and R^U in a top-down manner. During traversing the trees, the method can prune nodes in R^I and R^U, which are not necessary to be visited considering each other side. In the traversing of R^I, the algorithm ignores the nodes and items marked by Algorithm 3. In order to explain the details of the method for the search, we propose effective pruning techniques in Section 6.1. Then, an efficient algorithm for the RNN search, that uses the pruning techniques, is described in Section 6.2.

6.1. Pruning techniques

To explain the proposed pruning techniques, we employ a query item I_q, a node of R^I which is denoted by N_j^I, and a node of R^U which is denoted by N_i^U. In addition, we define the minimum sScore and the maximum sScore denoted by min$S$ and

max$S$, respectively, as follows:

$$\min S\left(N_j^I, N_i^U\right) = \frac{\min\limits_{(p1,p2)\in R(N_i^U)\times R(N_j^I)} (eDist(p1,p2))}{\text{Max}D} \tag{12}$$

$R(N_i^U)$ and $R(N_j^I)$ are the minimum bounding rectangles(MBR) of N_i^U and N_j^I, respectively.

$$\max S\left(N_j^I, N_i^U\right) = \frac{\max\limits_{(p1,p2)\in R(N_i^U)\times R(N_j^I)} (eDist(p1,p2))}{\text{Max}D} \tag{13}$$

We define $D(N_j^I)$ as the set of items which are indexed by the subtree rooted at N_j^I. In a similar way, $D(N_i^U)$ is the set of users. Also, we define max$P(N_j^I)$ and min$P(N_j^I)$ as $\max_{I_x \in D(N_j^I)} I_x.p$ and $\min_{I_x \in D(N_j^I)} I_x.p$, respectively.

The purpose of the pruning techniques is to find the users and items to be filtered out during the tree traversal. Fig. 5 shows an example. The spatial locations of the MBRs of nodes and query item I_q are presented.

For explaining the pruning techniques, we will use a two dimensional space called *SP* space. One dimension is the spatial score *sScore* and the other is the non-spatial score *pScore*. We define the SP zone (*SPZ*) as the follows:

Definition 5 (*SP zone*). Given tree elements $A \in R^I$ *and* $B \in R^U$, the SP zone (*SPZ*) of a pair of A and B is the region R in the *SP* space such that the minimum *pScore* and maximum *pScore* are min$P(A)$ and max$P(A)$, respectively, and the minimum *sScore* and maximum *sScore* are min$S(A, B)$ and max$S(A, B)$, respectively. It is denoted by $SPZ(A, B)$.

A single point in the *SP* space is like a singleton zone, and it represents the SP zone of the pair of an item and a user (e.g., $SPZ(I_j, u_i)$). The $SPZ(I_j, N_x^U)$ is a line in the *SP* space. Also, $SPZ(N_j^I, u_i)$ and $SPZ(N_j^I, N_i^U)$ are rectangles in the *SP* space. The SP zones for the MBRs and the query item in the example in Fig. 5 are presented in Fig. 6. We note that, for all the points on the dashed line which is perpendicular to the vector $(1-\alpha, \alpha)$, their corresponding pairs of users and items have the same IU distance. It can be derived from the equation of the line which passes through a point (x_0, y_0) in the SP space and is perpendicular to the line $y = (\alpha/(1-\alpha)) \cdot x$.

In addition, SPZs below the dashed line have IU distances lower than those of SPZs above the diagonal line.

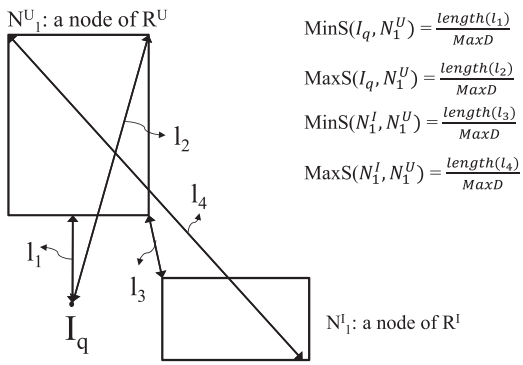


Fig. 5. An example:I_q, N_1^I, and N_1^U.

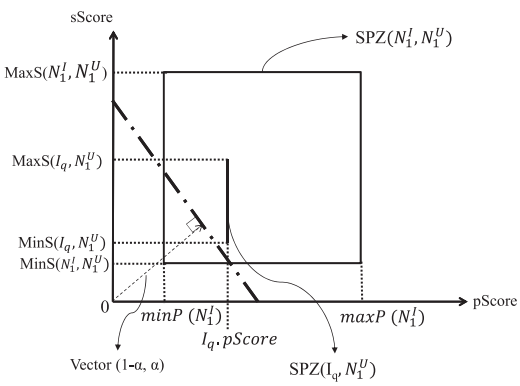


Fig. 6. SP space.

6.1.1. MM-pruning (min–max pruning)

Before describing the min-max pruning, we define *minIUD and tmaxIUD* as follows:

$$\min IUD(N_j^I, N_i^U) = \alpha \cdot \min S(N_j^I, N_i^U) + (1-\alpha) \cdot \min P(N_j^I) \tag{14}$$

$$\text{tmax}IUD(N_j^I, N_i^U) = \alpha \cdot \max S(N_j^I, N_i^U) + (1-\alpha) \cdot \min P(N_j^I) \tag{15}$$

Note that, for using the tighter bound of the maximum IU distance, $\min P(N_j^I)$ is used instead of $\max P(N_j^I)$ in $\text{tmax}IUD(N_j^I, N_i^U)$. We provide two lemmas which presents the basic ideas for the MM pruning.

Lemma 6. *Given I_q, N_i^U and N_j^I, if $\text{tmax}IUD(N_j^I, N_i^U) < \min IUD(I_q, N_i^U)$, then we can prune N_i^U.*

Proof. Since $\min P(N_j^I) = \min_{I_x \in D(N_j^I)} I_x.p$, there must exist an item $I^* \in D(N_j^I)$ s.t. $I^*.p = \min P(N_j^I)$. $\forall u \in D(N_i^U)$, it holds that $IUD(I^*, u) \le \text{tmax}IUD(N_j^I, N_i^U)$. If $\text{tmax}IUD(N_j^I, N_i^U) < \min IUD(I_q, N_i^U)$, we can derive that $\forall u \in D(N_i^U)$, $IUD(I^*, u) < \min IUD(I_q, N_i^U) \le IUD(I_q, u)$. This means that, for all users in $D(N_i^U)$, I^* is closer than I_q w.r.t. the *IU* distance. Consequently, any user in $D(N_i^U)$ cannot be contained in $RNN_{IU}(I_q)$. □

Lemma 7. *Given I_q, N_i^U and N_j^I, if $\text{tmax}IUD(I_q, N_i^U) < \min IUD(N_j^I, N_i^U)$, then we can prune N_j^I in the process evaluating the users in N_i^U.*

Proof. If $\text{tmax}IUD(I_q, N_i^U) < \min IUD(N_j^I, N_i^U)$, it is obviously true that $\forall u_i \in D(N_i^U), \nexists I_j \in D(N_j^I), IUD(I_j, u_i) \le IUD(I_q, u_i)$. Consequently, N_j^I is unnecessary for evaluating the users in N_i^U. □

Fig. 7 shows an example of the min-max pruning. If $SPZ(N_j^I, N_i^U)$ is the rectangle numbered 2, we can prune N_i^U. If $SPZ(N_j^I, N_i^U)$ is the rectangle numbered 1, the diagonal line meeting the left-bottom of the rectangle corresponds to $\min IUD(N_j^I, N_i^U)$. Therefore, for all the users in $D(N_i^U)$, I_q is closer than all the items in $D(N_j^I)$ w.r.t. the IU distance. Consequently, $D(N_j^I)$ can be pruned when evaluating the users in $D(N_i^U)$.

6.1.2. H-pruning(hausdorff IU distance based pruning)

The hausdorff IU distance based pruning method filters out the items which are unnecessary for checking certain users whether the users can be the answer of the RNN search. We define the hausdorff IU distance as follows:

Definition 6 (*Hausdorff IU distance*). Given a set of items $\hat{I}$ and a set of users U^*, the Hausdorff IU distance from U^* to $\hat{I}$ is $\text{HIUD}\ (\hat{I}, U^*) = \max_{u_i \in U^*}(\min_{I_j \in I^*}(IUD(I_j, u_i)))$.

For the convenience, we use $HIUD(N_j^I, N_i^U)$ for representing the hausdorff IU distance from $D(N_i^U)$ *to* $D(N_j^I)$.

Lemma 8. *Given I_q, N_i^U, and N_j^I, for all $I_k \in D(N_j^I)$, if $HIUD(N_j^I, N_i^U) < \min IUD(I_k, N_i^U)$, then I_k is unnecessary for validating the users in $D(N_i^U)$.*

Proof. For each item I_k *in* $D(N_j^I)$, if $HIUD(N_j^I, N_i^U) < \min IUD(I_k, N_i^U)$, we can guarantee that $\forall u_i \in D(N_i^U)$, $\min_{I_x \in D(N_j^I)} IUD(I_x, u_i) < IUD(I_k, u_i)$ by the definition of the hausdorff IU distance. Then, we can remove I_k from the search space. □

Fig. 8 shows an example. The dashed line is for $HIUD(N_i^U, N_j^I)$. $SPZ(I_1, N_i^U), SPZ(I_2, N_i^U), SPZ(I_3, N_i^U)$, and $SPZ(I_4, N_i^U)$ are represented by the lines l_1, l_2, l_3, and l_4, respectively. The shaded area is the pruning area which is the area above the diagonal line. In this example, I_3 and I_4 are pruned. We can compute the hausdorff IU distance by adapting the incremental algorithm proposed in [26]. The adaptation is to replace the Euclidean distance with the IU distance. Also, since the algorithm is based on the R-tree, our index structures can be compatible. However, if we compute *HIUD* multiple times

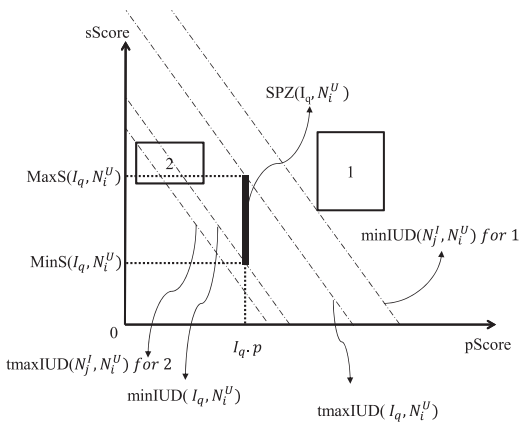


Fig. 7. An example of MM-Pruning.

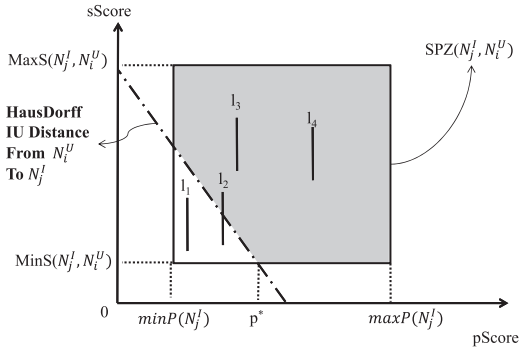


Fig. 8. An example of H-Pruning.

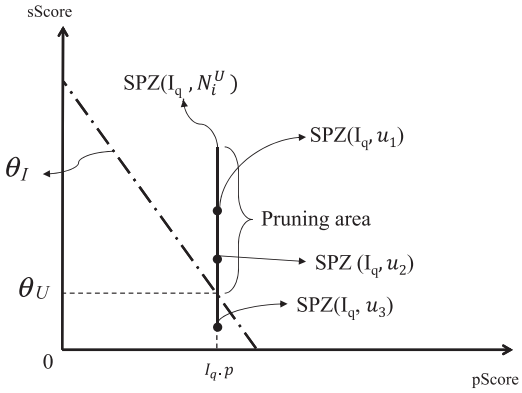


Fig. 9. An example of S-Pruning.

during tree traversing, the overall cost for computing *HIUD*s of multiple pairs of nodes can exceed the benefit from the pruning method. In order to avoid such cases, we compute *HIUD* only for the pair of the root nodes of R^I *and* R^U when the R-trees are updated.

6.1.3. S-pruning (sScore-based pruning)

The *sScore*-based pruning method filters out the users which cannot belong to the result of the RNN search. The *sScore*-based pruning method is based on the threshold θ_I which is Min(tmax$IUD(I_q, N_i^U), HIUD$). We can realize that

$$\begin{aligned} & IUD(I_q, u_i) > \theta_I \\ & \quad \Leftrightarrow \alpha \times sScore(I_q, u_i) + (1-\alpha) \times I_q.p > \theta_I \\ & \quad \Leftrightarrow sScore(I_q, u_i) > \frac{\theta_I - (1-\alpha) \times I_q.p}{\alpha} \end{aligned} \tag{16}$$

Let $(\theta_I - (1-\alpha) \times I_q.p)/\alpha$ be θ_U. Then, the users in $D(N_i^U)$ whose sScore with I_q is greater than θ_U can be pruned. Fig. 9 shows an example. In the figure, θ_I is $HIUD(N_j^I, N_i^U)$. In this case, among the users in $D(N_i^U)$, u_1 and u_2 are pruned.

6.2. Algorithm for the RNN search

In this section, the algorithm for the RNN search is explained. Algorithm 4 is a branch and bound algorithm for validating the users. Algorithm 4 gets α, I_q, R^U, and R^I as the inputs. The algorithm uses a 2-layered structure which consists of a main queue and sub queues. The element of the main queue is $(U_{elem}, SubQ)$. U_{elem} is an element of R^U which can be a user or a node of the tree. *SubQ* is a sub queue. The element of the sub queue is $(I_{elem}, \min IUD)$. I_{elem} is an element of R^I which can be an item or a node of the tree, and min*IUD* is the minimum IU distance between U_{elem} in the associated main queue element and I_{elem} in the sub queue element. Especially, the sub queue is a priority queue which is sorted in the ascending order of min*IUD*. The semantics of the 2-layered structure is that, for each main queue element $(U_{elem_x}, SubQ_k)$, Algorithm 2 finds users $u_i \in D(U_{elem_x})$ such that $\forall I_j \in \bigcup_{(I_{elem_y}, \min IU_y) \in SubQ} D(I_{elem_j}), IUD(I_q, u_i) \le IUD(I_j, u_i)$. If a user satisfies the above condition, Algorithm 2 adds the user to the result set. The algorithm traverses R^U *and* R^I by replacing a node with its children nodes or

objects. During the traversing, 2-layered structure is used to maintain the search contexts of pairs of U_{elem} s and I_{elem} s of R^U *and* R^I, respectively.

Algorithm 4. A branch and bound algorithm for validating users.

```
   input  : α, I_q, R^U, and R^I
   output : RET - The set of users
1  begin
2      Initialize RET as an empty set, SubQ as a priority queue, and MainQ as a queue ;
3      Enqueue (the root node of R^I, ∞ ) into SubQ ;
4      Enqueue (the root node of R^U, SubQ ) into MainQ ;
5      while MainQ is not empty do
6          (U_elem, SubQ) := Dequeue from MainQ;
7          if SubQ is empty then
8              add U_elem into RET ;
9              Continue;
10         (I_elem, minIUD) := Dequeue from SubQ;
11         if U_elem and I_elem are the root nodes then
12             hiud := HIUD(I_elem, U_elem) ;
13         Initialize U_SET and I_SET as empty sets;
14         if U_elem is a user and I_elem is an item then
15             if IUD(I_q, U_elem) ≤ IUD(I_elem, U_elem) then
16                 add U_elem into RET ;
17         else if U_elem or I_elem is a node then
18             mm:=MM(I_elem, U_elem, I_q);
19             if mm is "Drop" then
20                 Continue ;
21             else if mm is "TakeAll" then
22                 add U_elem into RET ;
23             else
24                 θ_I:=Min(hiud, tmaxIUD(I_q, U_elem));
25                 θ_U:=S(θ_I);
26                 if U_elem is a user then
27                     U_SET:={U_elem};
28                 else
29                     U_SET:=findChildUsers(U_elem, θ_U);
30                 if I_elem is an item then
31                     I_SET:={I_elem};
32                 else
33                     I_SET:=findChildItems(I_elem,θ_I);
34         for each U_elem_i ∈ U_SET do
35             Initialize newSubQ as a priority queue;
36             for each I_elem_j ∈ (SubQ.list() ∪ I_SET) do
37                 Enqueue (I_elem_j, minIUD(U_elem_i, I_elem_j) ) into newSubQ ;
38             Enqueue (U_elem_i,newSubQ) into MainQ ;
39     Return RET ;
```

Algorithm 2 initiates *MainQ and SubQ* in Lines 2–4. The main loop of Algorithm 4 is presented in Lines 5–38. The algorithm loops until *MainQ* becomes empty. For each loop, the algorithm dequeues an element (U_{elem}, *SubQ*) from *MainQ*. If *SubQ* is empty, the algorithm adds U_{elem} into RET and skips to the next loop. It means that, for users in $D(U_{elem})$, there is no more item I_j to be checked if $IUD(I_q, u_i) \leq IUD(I_j, u_i)$. If *subQ* is not empty, the algorithm dequeues an element (I_{elem}, min*Dist*) from *SubQ* in Line 10. Since U_{elem} can be a user or a node, and I_{elem} can be an item or a node, we have 4 combinations of cases. If U_{elem} is a user and I_{elem} is an item, the algorithm compares the IU distances between I_{elem} and U_{elem}, and between I_q *and* U_{elem}, in order to determine whether U_{elem} is to be reported as an answer or not in Lines 14–16. If U_{elem} or I_{elem} is a node, the algorithm performs the *MM* pruning. $MM(\cdot)$ returns "Drop" if $\min IUD(I_q, U_{elem}) > \text{tmax}IUD(I_{elem}, U_{elem})$,"TakeAll" if $\text{tmax}IUD(I_q, U_{elem}) < \min IUD(I_{elem}, U_{elem})$, and *null* otherwise. If the result is "Drop", the algorithm continues to the next step in Lines 19–20. It is because, for all users in D (U_{elem}), I_q cannot be the nearest item by Lemma 6. When the result is "TakeAll", the algorithm adds U_{elem} into *RET* and skip to the next loop in Lines 21–22. Since all the remaining elements of *SubQ* have minIUDs that are greater or equal to $\min IUD(I_{elem}, U_{elem})$ where I_{elem} is in the head of *SubQ*, the results of $MM(\cdot)$ for remaining elements are "TakeAll". Therefore, we can prune all the I_{elem} s in the remaining elements of *SubQ* for validating U_{elem} by Lemma 7. Consequently, all the users in $D(U_{elem})$ are contained in the answer of the RNN search. If the $MM(\cdot)$ returns *null*, the algorithm finds the threshold θ_I *and* θ_U in Lines 24–25. $S(\theta_I)$ returns the maximum threshold of *sScore* computed by the S-pruning method. The algorithm finds the set of children of U_{elem} which do not violate the threshold by using *findChildUsers*(U_{elem}, θ_U). The 'violate' means that the $\text{Min}S(I_q, child_x(U_{elem}))$ is greater than θ_U where $child_x(U_{elem})$ is a child of U_{elem}. Also, *findChildItems*(I_{elem}, θ_I) returns the set of children of I_{elem} such that $\min IUD(child_y(I_{elem}), U_{elem}) \leq \theta_I$ and the child is not marked by Algorithm 3. In Line 34-38, the algorithm updates *MainQ* by

inserting new elements that are generated by the children of U_{elem} in the dequeued *MainQ* element and I_{elem} in the dequeued *SubQ* element. *SubQ.list*() returns the set of I_{elem} s of all the elements of *SubQ*. Finally, the algorithm returns *RET* after terminating the loops.

7. Experiments

In this section, we present the experiments for evaluating the proposed method. PR_FULL is the proposed method which contains all the pruning techniques. For the comparison, we implement the variations: PR_SUB1, PR_SUB2, PR_SUB_NO and PR_ITEM_NO. PR_SUB1 uses the item pruning, the MM pruning and the S-pruning. PR_SUB2 uses the item pruning, the MM pruning and the H pruning. PR_SUB_NO uses item pruning and MM pruning. PR_ITEM_NO uses all the pruning techniques except the item pruning. In our method, the MM pruning is essential to avoid checking all the pairs of users and items. RTA is a method for our problem adapted from the reverse top *k* search problem proposed in [16].

7.1. Dataset and environment

For the evaluation, we conduct experiments on a large synthetic dataset and real datasets.

We generated large synthetic datasets consisting of 1,000,000 users and 1,000,000 items with randomly selected positions in the two dimensional geometric space, and the non-spatial scores are from 0 to 1 which follow normal distributions. The height and width of the universe are 1s. Then, subsets are extracted from the large synthetic datasets for experiments. Table 2 shows the parameters and the variations of their values for the experiments. For each parameter, the value with the bold type is the default.

α is fixed only for the experiment with synthetic datasets, which is about the effects of the dataset cardinality on the performance. As we described in Section 3.2, α is determined by the maximum *sScore and pScore*, and the relative importance of the spatial distance and a non-spatial aspect heavily depends on the extent of the universe area and the application domain. We set the default value of α to 0.5 for the experiments with synthetic datasets because this setting can be a standard situation such that the importance of the spatial distance and that of a non-spatial aspect are the same.

For the distribution of *pScore* of items, a default value is not set since we conduct experiments for all the datasets with the 3 distributions. In addition, 1000 query items are randomly generated for the experiments. All the experimental results are the average values from the RNN searches for all the query items.

For real datasets, we have crawled the data by using Google Places API. The restaurants which have ratings are collected for items, and the houses and hotels are collected for users. We generate three kinds of datasets NY , LD, and SL for New York, London, and Seoul, respectively. Table 3 shows the information of the real datasets.

The experiments are conducted by a single PC with Intel i7 990x CPU, and 24 GB of main memory. Windows 7 64-bit is installed as the operating system on the PC. In addition, an MS-SQL is used for the repository.

For the R-tree index used in the proposed method, we set the maximum node capacity as 30.

7.2. Experimental results

We conduct experiments by varying the number of items, the number of users, and α. For each result, we show the average execution time of the RNN search of all the query items.

Table 2
Setting of synthetic datasets.

Parameter	Range
# of Items	250 K, 500 K, 750 K, **1 M**
# of Users	250 K, 500 K, 750 K, **1 M**
α	0.1, 0.3, **0.5**, 0.7, 0.9
Distribution of *pScore*	N(0.5,0.0025), N(0.5,0.01), N(0.5, 0.04)

Table 3
Real datasets.

Dataset	Item size	User size
NY	21,282	55,129
LD	46,016	65,275
SL	56,689	112,123

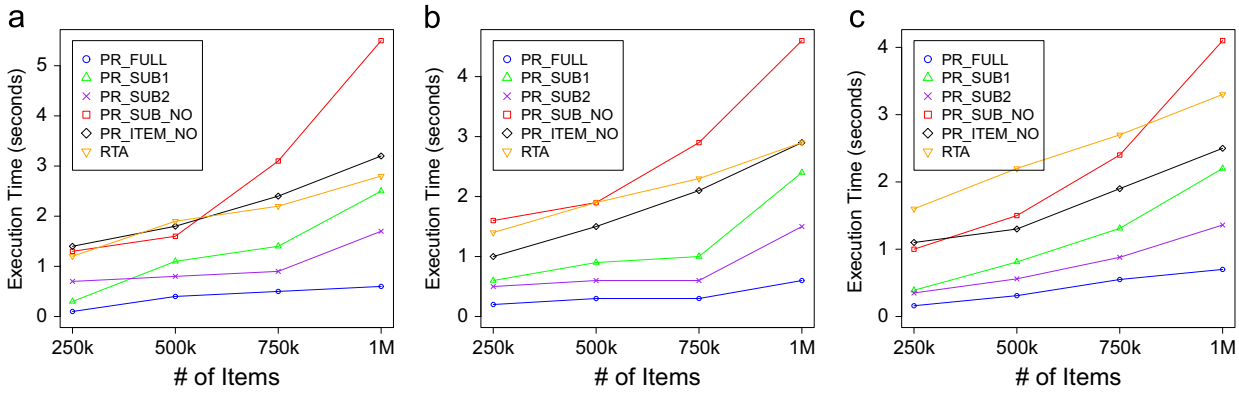


Fig. 10. The execution time by varying the number of items. (a) $N(0.5, 0.0025)$. (b) $N(0.5, 0.01)$. (c) $N(0.5, 0.04)$.

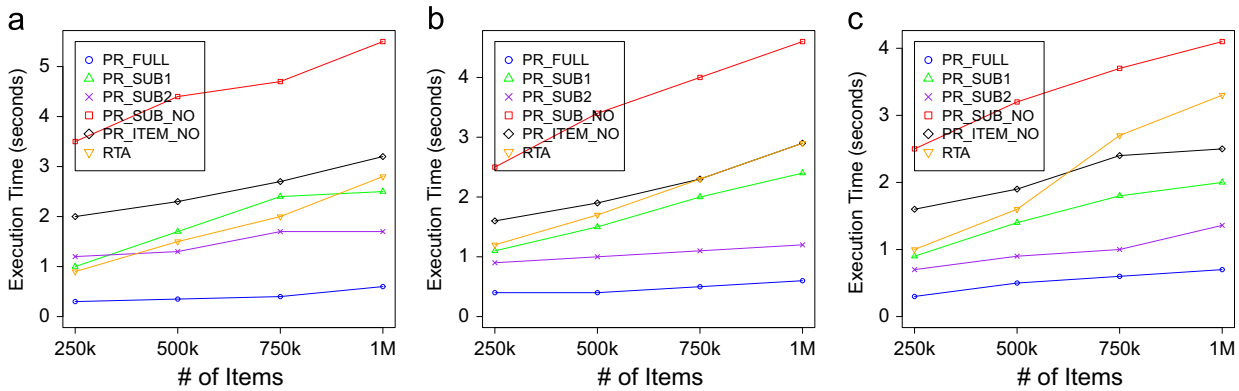


Fig. 11. The execution time by varying the number of users. (a) $N(0.5, 0.0025)$. (b) $N(0.5, 0.01)$. (c) $N(0.5, 0.04)$.

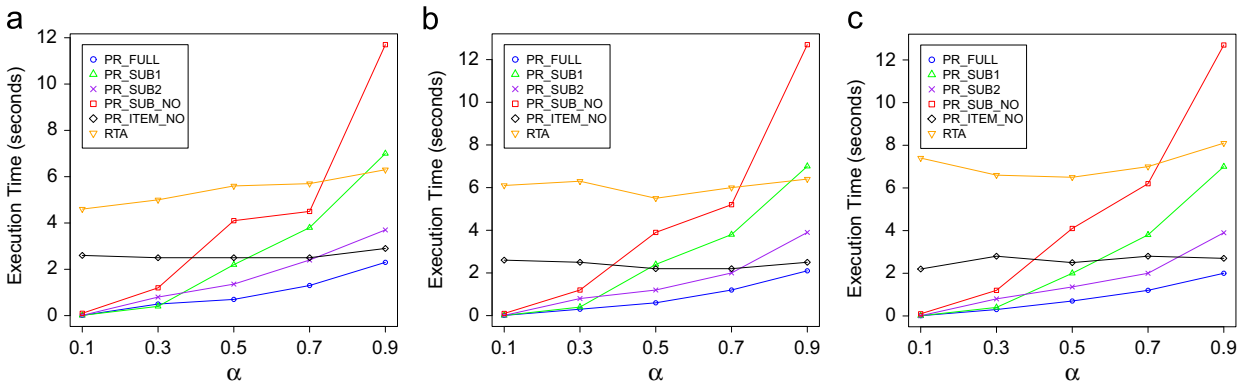


Fig. 12. The execution time by varying α. (a) $N(0.5, 0.0025)$. (b) $N(0.5, 0.01)$. (c) $N(0.5, 0.04)$.

(Effects of the cardinality) Fig. 10 shows the computational time of the RNN search by varying number of total items. We can see that the slope of PR_SUB_NO is the largest among those of the methods. The pruning techniques reduce the sensitivity of the performance on the number of items since the unnecessary items are appropriately pruned by the techniques. The results show that RTA is sensitive to the cardinality of the set of items and the proposed method outperforms RTA for all the cardinalities.

Fig. 11 shows the execution time by varying the number of users. As we can see, the performance is less affected by the number of users in the RNN search. It is because, the second layer of our 2-layered structure in the proposed method for the RNN search becomes rapidly large, as the number of items increases. The results show that RTA is sensitive to the cardinality of the set of users and the proposed method is more efficient than RTA for all the cardinalities.

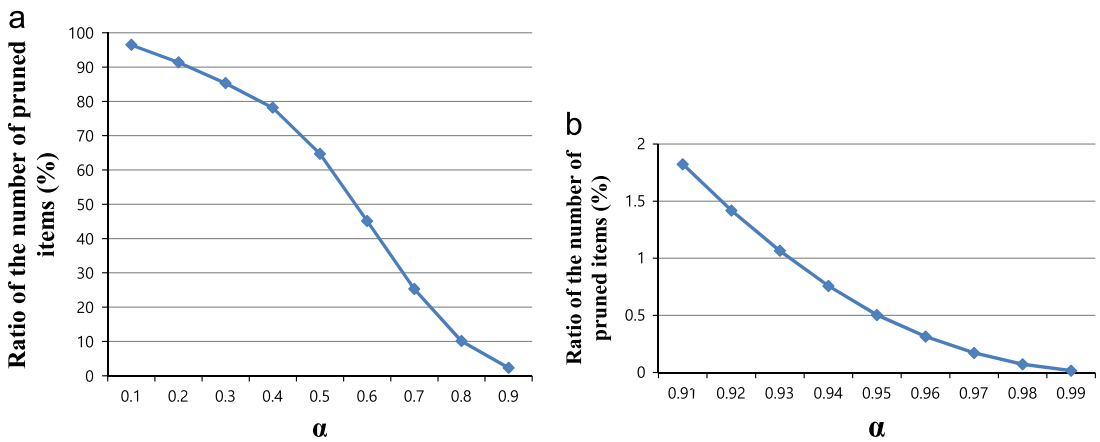


Fig. 13. The ratio of the number of pruned items by varying α. (a) $0.1 \le \alpha \le 0.9$. (b) $0.91 \le \alpha \le 0.99$.

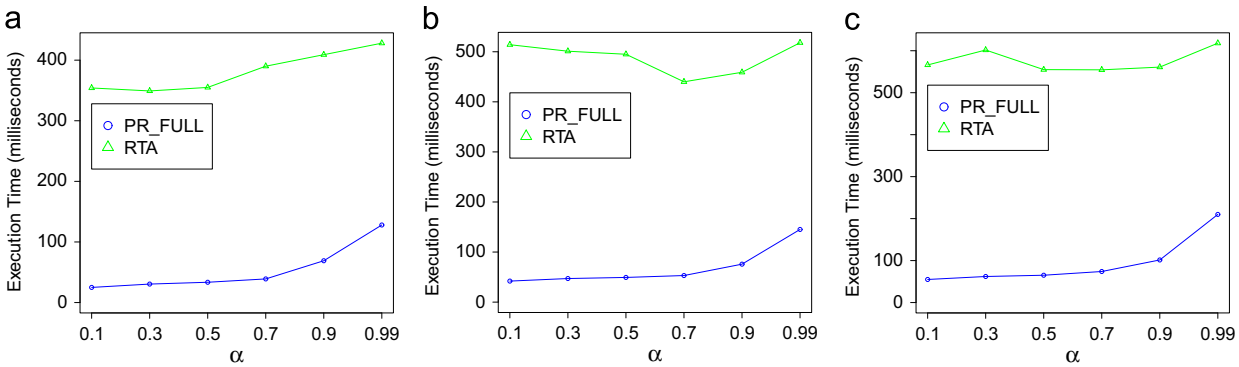


Fig. 14. The execution time for the real dataset by varying α. (a) NY. (b) LD. (c) SL.

Table 4
The average total costs for the nearest neighbor items.

Dataset	With the spatial distance	With the IU distance
NY	\$67	\$43
LD	\$74	\$32
SL	\$39	\$24

(Effects of α on the total execution time) Fig. 12 shows the execution time for varying α. The change of α significantly affects performance of the RNN search because the performance of the item pruning depends on α. PR_ITEM_NO which does not use the item pruning method is not effected by the value of α. In the results, RTA is not very sensitive to α and the proposed method is always better than RTA.

(Effects of α on the pruning power) Fig. 13 shows the average number of items which are pruned by the query item, for varying α. Averagely 65% of 1 M items are pruned by the query item when α is 0.5. Also, when α is 0.9, 2% of items are pruned. As we can see in the result, the item pruning method using the domination relationships effectively prunes items when the weighting parameter α is a low value. Therefore, the higher the weight of the non-spatial aspect is, the more useful the proposed method for the RNN search with a non-spatial aspect is.

(Real datasets) Fig. 14 shows that the execution time of the experiments for PR_FULL and RTA with real datasets. We can see that the result of the experiment based on the real dataset by varying the value of alpha is similar to that of the synthetic dataset. In addition, PR_FULL outperforms RTA for all the real datasets.

(Effectiveness of the IU distance) We demonstrate the effectiveness of the IU distance over the spatial distance by using real datasets. The demonstration is based on the guideline proposed in Section 3.2. We set the cost for 100 km(=MaxD) to 100 dollars, and the maximum price of an item to 100 dollars. Note that we can set any positive values for the costs since the ranking function of our problem is monotonic. If a user selects an item that is 10 km away from the user and the price of the

item is 30 dollars, then the total cost for taking the item is 40 dollars. Given the real datasets, for each user, we find two kinds of the nearest neighbor items. One is based on the spatial distance, and the other is based on the IU distance. For each kind of the nearest neighbor items, we compute the average total cost. Table 4 shows the result of the average cost. The result show that using the IU distance is better than using the spatial distance in terms of the total cost.

8. Conclusion and future works

In this paper, we propose an efficient method to find the reverse nearest neighbor of the query item based on the IU distance. In order to improve the efficiency of the query-time processing, we propose an efficient method for the item pruning. Then, we devise an efficient algorithm for the RNN search. The experimental results show that the proposed method is at least 4 times more efficient than an adapted version of an existing method. As a future work, we will consider the extension of our problem in which the non-spatial score may be different among users. This problem is more practical yet difficult than that of this paper. Since the non-spatial score is personalized for each user, the pruning method with the item domination and the algorithm devised in this paper cannot be directly used. Therefore, new pruning methods and new algorithms should be proposed for solving the problem.

Acknowledgements

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD140022PD, Korea.

Appendix A. The hyperbola-based safe zone

The safe zone of an item I_j is a spatial area such that a user has I_j as the nearest item if and only if the user is located in the area. If the user stays in the safe zone of the item I_q, we can guarantee that I_q is the nearest neighbor of the user. Since our problem employs the IU distance instead of the spatial distance, the traditional safe zones in the previous studies cannot be used in our case. We provide two definitions, the relative safe zone and the safe zone.

Definition 7 (*Relative safe zone*). The relative safe zone of I_x w.r.t I_y is the area R such that, for u_i with $u_i.loc \in R, IUD(I_x, u_i) \leq IUD(I_y, u_i)$. The relative safe zone of I_x w.r.t. I_y is denoted by $H_{x:y}$.

Since the relative safe zone is not symmetric, we can realize that $H_{x:y} \neq H_{y:x}$, and $H_{x:y} \cup H_{y:x}$ is the universe area.

Definition 8 (*Safe zone*). The safe zone of I_x is formally defined by $H_x = \bigcap_{I_y \in \hat{I}, x \neq y} H_{x:y}$.

In order to calculate the safe zone of an item I_q, we use multiple hyperbolas, each of which is formed by I_q and $I_j \in \hat{I}(q \neq j)$. Given two points p_1 and p_2, the hyperbola of the points is defined as the locus[5] of a set of points P such that, for all $p \in P$, the

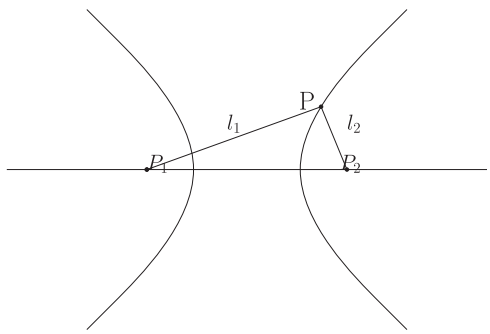


Fig. 15. The hyperbola of the points p_1 and p_2.

difference between the distances $eDist(p_1, p)$ and $eDist(p_2, p)$ is the same.

Fig. 15 shows an example of the hyperbola for the points p_1 and p_2. l_1 is the line between p_1 and p. l_2 is the line between p_2 and p. As the definition of the hyperbola, for all p on the hyperbola, $|Length(l_1) - Length(l_2)|$ is the same. We can see that a hyperbola consists of two disconnected curves called branches.

For finding the safe zone of I_q, we will compute the relative safe zone of I_q w.r.t. $I_j \in \hat{I}$ where $q \neq j$. By using the definition of the IU distance, the boundary of $H_{q:j}$ and $H_{j:q}$ can be determined. The boundary is the set of locations of u_i satisfying the

[5] In mathematics, the locus is the set of points satisfying a particular condition, which often forms a curve.

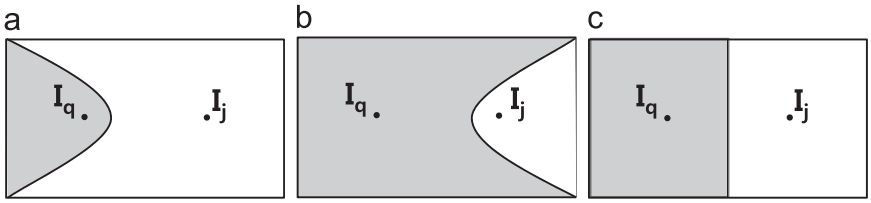


Fig. 16. The area of $H_{q:j}$ depending on the non-spatial scores of I_q and I_j. (a) $I_q.p > I_j.p$. (b) $I_q.p < I_j.p$. (c) $I_q.p = I_j.p$.

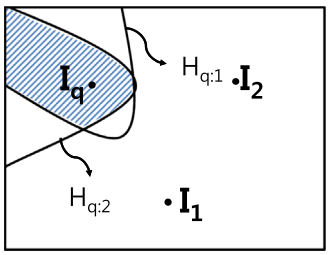


Fig. 17. An example of H_q, the safe zone of an item I_q.

following equation.

$$\begin{aligned} & IUD(I_q, u_i) = IUD(I_j, u_i) \\ & \Leftrightarrow (sScore(I_q, u_i) - sScore(I_j, u_i)) = \frac{1-\alpha}{\alpha} \times (I_j.p - I_q.p) \\ & \Leftrightarrow eDist(I_q, u_i) - eDist(I_j, u_i) = \frac{(1-\alpha) \times \mathrm{Max}D}{\alpha} \times (I_j.p - I_q.p) \end{aligned} \tag{A.1}$$

Since $\alpha, \mathrm{Max}D, I_j.p$, and $I_q.p$ have static values, the locus of the locations of users satisfying the equation is a branch of a hyperbola. The boundary does not contain both the two hyperbola branches because, in the last row of the above equations, the left side does not have the absolute operator like the equation for the locus in Fig. 15 and the right side can have a positive value, a negative value, or the zero. Fig. 16 shows examples of relative safe zones depending on $I_q.p$ and $I_j.p$. The shaded areas represent $H_{q:j}$.

By computing $H_{q:j}$ for all $I_j \in \hat{I} - \{I_q\}$, we can find H_q. Fig. 17 shows an example of the safe zone for I_q. We assume that $\hat{I} = \{I_q, I_1, I_2\}$ such that $I_q.p > I_1.p$ and $I_q.p > I_2.p$. By Definition 8, $H_q = H_{q:1} \cap H_{q:2}$. In this figure, the area with diagonal lines is the safe zone of I_q.

References

[1] F. Korn, S. Muthukrishnan, Influence sets based on reverse nearest neighbor queries, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD'00, ACM, New York, NY, USA, 2000, pp. 201–212.
[2] C. Yang, K.-I. Lin, An index structure for efficient reverse nearest neighbor queries, in: Proceedings of the 17th International Conference on Data Engineering, IEEE Computer Society, Washington, DC, USA, 2001, pp. 485–492. URL ⟨http://dl.acm.org/citation.cfm?id=645484.656392⟩.
[3] K.-I. Lin, M. Nolen, C. Yang, Applying bulk insertion techniques for dynamic reverse nearest neighbor problems, in: 2003 Proceedings of Seventh International Database Engineering and Applications Symposium, 2003, pp. 290–297.
[4] I. Stanoi, D. Agrawal, A.E. Abbadi, Reverse nearest neighbor queries for dynamic databases, in: In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2000, pp. 44–53.
[5] Y. Tao, D. Papadias, X. Lian, Reverse knn search in arbitrary dimensionality, in: Proceedings of the Thirtieth International Conference on Very Large Data Bases, vol. 30, VLDB'04, VLDB Endowment, 2004, pp. 744–755.
[6] W. Wu, F. Yang, C.-Y. Chan, K.-L. Tan, Finch: evaluating reverse k-nearest-neighbor queries on location data, Proc. VLDB Endow. 1 (1) (2008) 1056–1067.
[7] R. Benetis, C.S. Jensen, G. Karciauskas, S. Saltenis, Nearest neighbor and reverse nearest neighbor queries for moving objects, in: Proceedings of the 2002 International Symposium on Database Engineering & Applications, IDEAS'02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 44–53.
[8] T. Xia, D. Zhang, Continuous reverse nearest neighbor monitoring, in: Proceedings of the 22nd International Conference on Data Engineering, ICDE'06, IEEE Computer Society, Washington, DC, USA, 2006, p. 77.
[9] J. Kang, M. Mokbel, S. Shekhar, T. Xia, D. Zhang, Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors, in: IEEE 23rd International Conference on Data Engineering, 2007, ICDE 2007, 2007, pp. 806–815.
[10] W. Wu, F. Yang, C.Y. Chan, K.-L. Tan, Continuous reverse k-nearest-neighbor monitoring, in: Proceedings of the Ninth International Conference on Mobile Data Management, MDM'08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 132–139.
[11] M.A. Cheema, X. Lin, Y. Zhang, W. Wang, W. Zhang, Lazy updates: an efficient technique to continuously monitoring reverse knn, Proc. VLDB Endow. 2 (1) (2009) 1138–1149.
[12] J. Lu, Y. Lu, G. Cong, Reverse spatial and textual k nearest neighbor search, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD'11, ACM, New York, NY, USA, 2011, pp. 349–360.
[13] Y. Tao, M.L. Yiu, N. Mamoulis, Reverse nearest neighbor search in metric spaces, IEEE Trans. Knowl. Data Eng. 18 (9) (2006) 1239–1252.
[14] A. Singh, H. Ferhatosmanoglu, A.c. Tosun, High dimensional reverse nearest neighbor queries, in: Proceedings of the Twelfth International Conference on Information and Knowledge Management, CIKM'03, ACM, New York, NY, USA, 2003, pp. 91–98.
[15] M.L. Yiu, D. Papadias, N. Mamoulis, Y. Tao, Reverse nearest neighbors in large graphs, IEEE Trans. Knowl. Data Eng. 18 (4) (2006) 540–553.
[16] A. Vlachou, C. Doulkeridis, Y. Kotidis, K. Norvag, Reverse top-k queries, in: 2010 IEEE 26th International Conference on Data Engineering (ICDE), 2010, pp. 365–376.

[17] A. Vlachou, C. Doulkeridis, K. Nørvåg, Y. Kotidis, Branch-and-bound algorithm for reverse top-k queries, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD'13, ACM, New York, NY, USA, 2013, pp. 481–492.
[18] A. Vlachou, C. Doulkeridis, K. Nørvåg, Monitoring reverse top-k queries over mobile devices, in: Proceedings of the 10th ACM International Workshop on Data Engineering for Wireless and Mobile Access, MobiDE'11, ACM, New York, NY, USA, 2011, pp. 17–24.
[19] I. De Felipe, V. Hristidis, N. Rishe, Keyword search on spatial databases, in: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, ICDE'08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 656–665.
[20] G. Cong, C.S. Jensen, D. Wu, Efficient retrieval of the top-k most relevant spatial web objects, Proc. VLDB Endow. 2 (1) (2009) 337–348.
[21] X. Cao, G. Cong, C.S. Jensen, Retrieving top-k prestige-based relevant spatial web objects, Proc. VLDB Endow. 3(1–2) (2010) 373–384.
[22] A. Gyttman, R-trees. a dynamic index structure for spatial searching, in: Proceedings of the SIGMOD, 1984, pp. 47–57.
[23] V. Kostakos, Is the crowd's wisdom biased? A quantitative analysis of three online communities, in: Proceedings of the 2009 International Conference on Computational Science and Engineering, vol. 04, CSE'09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 251–255.
[24] E.W. Weisstein, Square line picking. from mathworld—a wolfram web resource, ⟨http://mathworld.wolfram.com/SquareLinePicking.html⟩.
[25] H. Samet, Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
[26] S. Nutanong, E.H. Jacox, H. Samet, An incremental Hausdorff distance calculation algorithm, Proc. VLDB Endow. 4 (8) (2011) 506–517.