# Skyline Query for Selecting Spatial Objects by Utilizing Surrounding Objects

Mohammad Shamsul Arefin, Xu Jinhao, Chen Zhiming, and Yasuhiko Morimoto
Graduate School of Engineering, Hiroshima University, Japan
Email: [d105660, M114625, chan, morimo]@hiroshima-u.ac.jp

*Abstract*— **With the increase of data volume, advanced query operators such as skyline queries are necessary in order to help users to handle the huge amount of available data by selecting a set of promising data objects. In this paper, we present a method for selecting spatial objects, such as houses, based on the objects of the surrounding facilities such as restaurants, supermarkets, and stations. In our approach, at first a user specifies a list of surrounding facilities within a specified distance in his preferred location. Our system then computes a set of spatial objects in the preferred location considering the objects of the surrounding facilities by utilizing the idea of skyline queries. We perform different experiments to show the effectiveness of our approach. Experimental evaluation shows that our approach is well applicable for efficient decision making.**

*Index Terms*— **Skyline queries, surrounding facilities, aggregation R-tree.**

## I. INTRODUCTION

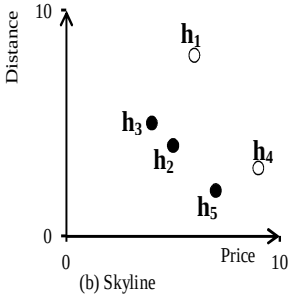
With the increase of data volume, advanced query operators are necessary in order to help users to handle the huge amount of available data by selecting a set of promising data objects. For example, when we want to purchase something, recommendation systems [1]–[4] are helpful for handling huge number of choices. Most of recommendation systems find transactions of similar items and/or similar users and use the similar transactions for selecting items, which we call "collaborative filtering" method.

In our life, selecting a good object such as a house is very important for us and selection of such an object is highly influenced by the co-existence of other facilities in the surrounding area. There is no doubt that we can use collaborative filtering technologies to select such an object. However, the number of transactions is not so large to use the collaborative filtering method. The number of similar companies and similar users those are related to such business are not as large as daily commodities. Therefore, in this paper, we use the idea of skyline queries [5] instead of collaborative filtering for selecting such spatial objects.

Let $p$ and $q$ be objects in $DB$. Let $p.a_l$ and $q.a_l$ be the $l$-th attribute values of $p$ and $q$, respectively, where $(1 \leq l \leq k)$. If we assume smaller value is better, an object $p$ is said to dominate another object $q$, if $p.a_l \leq q.a_l$ for all $k$ attributes $(1 \leq l \leq k)$ and $p.a_j < q.a_j$ $(1 \leq j \leq k)$ on at least one attribute. The skyline is a set of objects which are not dominated by any other object in $DB$. Figure 1 shows a typical example of skyline objects. The table in Figure 1 is a list of hotels, each of which contains two numerical attributes "Distance", which is the distance from the nearest station, and "Price", which is the accommodation charge. In the list, the best choice usually comes from the skyline, i.e., one of $\{h_2, h_3, h_5\}$ (see Figure 1 (b)). A number of efficient algorithms for computing skyline queries from numerical databases have been reported in the literature [5]–[9].

| ID | Price | Distance |
|---|---|---|
| $h_1$ | 6 | 8 |
| $h_2$ | 5 | 4 |
| $h_3$ | 4 | 5 |
| $h_4$ | 9 | 3 |
| $h_5$ | 7 | 2 |

(a) Hotels data




(b) Skyline

Figure 1. Skyline example

If we use the skyline query for selecting spatial objects like houses, we can filter many dominated houses based on features of the houses, such as price, age, and so on. However, the location is also important for selecting a house. For example, a house is convenient if there are many supermarkets within a walking distance. Conventional skyline queries do not take into account such surrounding environment.

In this paper, we propose an efficient method based on skyline queries for recommending such type of spatial objects that takes into account the features as well as the location of the spatial objects.

### A. Motivating Example

Assume the objects of three different facilities in a specific location as shown in Figure 2. The location information of the objects of these facilities is stored in a spatial database as in Table I. Figure 3 (a), (b), and (c) are non-spatial databases for each of these three facilities. For simplicity, throughout the paper, we consider 1 unit of the spatial database is equal to 200 meters and higher values in each dimension of the non-spatial databases are better.


doi:10.4304/jcp.8.7.1742-1749

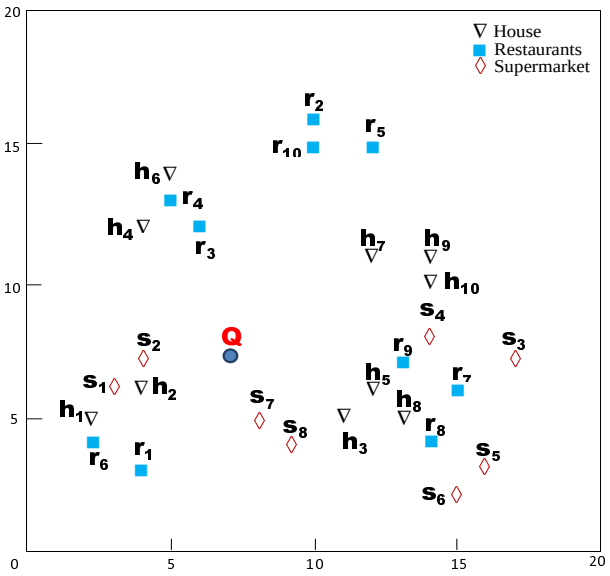



Figure 2. Three different facilities in a location

In our method, a user specifies a list of surrounding facilities within a specified distance with favourable conditions for the objects. Similarly, a user can specify unfavourable conditions for the objects of the facilities if necessary. A user may specify the conditions of favourable and unfavourable.

For each specified surrounding facility, we count the number of objects those satisfy user defined distance and conditions. We, then, add a new attribute for each chosen surrounding facility that contains the number of objects those satisfies the conditions. Then, we compute the skyline result from the extended database.

Now, assume that a user wants to buy a house within 1200 meters of a well known place $Q$ as shown in Figure 2. Additionally, the user specifies that he wants some favourable restaurants and supermarkets within 800 meters of the house. In the example, houses those are within 1200 meters from $Q$ are shown in Figure 4. If she/he specifies the condition of favourable restaurants and supermarkets, we can filter some objects from Figure 4. Assume that restaurants and supermarkets having not less than 3 in the both attributes are specified to be favourable. Then, we get the information as shown in Figure 5, in which $r_1$ and $r_7$ are eliminated.

Now, considering the non-spatial information of houses $h_1$, $h_2$, $h_3$, $h_4$, and $h_5$ and the information obtained from Figure 5, we get the information as shown in Table II.

After constructing such a table, we can perform a conventional skyline query to return the results for the user. Note that without considering surrounding restaurants and supermarkets the result is only $\{h_1\}$, while considering the surrounding environment, the result becomes $\{h_1, h_3, h_4, h_5\}$ that provides more options to the user.

Conventional skyline algorithms do not provide such type of skyline calculation.

Motivated with the above results, in this paper, we introduce a framework to compute spatial objects like houses considering surrounding facilities. We perform different experiments to show efficiency and robustness of our algorithm.

TABLE I.
SPATIAL DATABASE

| ID | Longitude | Latitude | Type |
|---|---|---|---|
| $h_1$ | 2 | 5 | House |
| $h_2$ | 4 | 6 | House |
| $h_3$ | 11 | 5 | House |
| $h_4$ | 4 | 12 | House |
| $h_5$ | 12 | 6 | House |
| $h_6$ | 5 | 14 | House |
| $h_7$ | 12 | 11 | House |
| $h_8$ | 13 | 5 | House |
| $h_9$ | 14 | 11 | House |
| $h_{10}$ | 14 | 10 | House |
| $s_1$ | 3 | 6 | Supermarket |
| $s_2$ | 4 | 7 | Supermarket |
| $s_3$ | 17 | 7 | Supermarket |
| $s_4$ | 14 | 8 | Supermarket |
| $s_5$ | 16 | 3 | Supermarket |
| $s_6$ | 15 | 2 | Supermarket |
| $s_7$ | 8 | 5 | Supermarket |
| $s_8$ | 9 | 4 | Supermarket |
| $r_1$ | 4 | 3 | Restaurant |
| $r_2$ | 10 | 11 | Restaurant |
| $r_3$ | 6 | 12 | Restaurant |
| $r_4$ | 5 | 14 | Restaurant |
| $r_5$ | 12 | 15 | Restaurant |
| $r_6$ | 2 | 4 | Restaurant |
| $r_7$ | 15 | 6 | Restaurant |
| $r_8$ | 14 | 4 | Restaurant |
| $r_9$ | 13 | 7 | Restaurant |
| $r_{10}$ | 10 | 15 | Restaurant |

| ID | $a_1$ | $a_2$ |
|---|---|---|
| $h_1$ | 5 | 8 |
| $h_2$ | 2 | 3 |
| $h_3$ | 4 | 4 |
| $h_4$ | 5 | 6 |
| $h_5$ | 3 | 5 |
| $h_6$ | 4 | 6 |
| $h_7$ | 2 | 3 |
| $h_8$ | 5 | 4 |
| $h_9$ | 3 | 7 |
| $h_{10}$ | 7 | 6 |

(a) Houses

| ID | $a_1$ | $a_2$ |
|---|---|---|
| $s_1$ | 3 | 6 |
| $s_2$ | 4 | 5 |
| $s_3$ | 6 | 7 |
| $s_4$ | 7 | 6 |
| $s_5$ | 3 | 4 |
| $s_6$ | 5 | 5 |
| $s_7$ | 6 | 4 |
| $s_8$ | 3 | 5 |

(b) Supermarkets

| ID | $a_1$ | $a_2$ |
|---|---|---|
| $r_1$ | 2 | 8 |
| $r_2$ | 6 | 7 |
| $r_3$ | 5 | 6 |
| $r_4$ | 6 | 8 |
| $r_5$ | 4 | 2 |
| $r_6$ | 6 | 4 |
| $r_7$ | 2 | 3 |
| $r_8$ | 4 | 5 |
| $r_9$ | 7 | 5 |
| $r_{10}$ | 1 | 3 |

(c) Restaurants

Figure 3. Databases showing non-spatial features of three facilities

The remainder of this paper is organized as follows. Section II provides a brief review of related works on skyline queries. In section III, we detail the computation framework of our approach. Section IV presents the experimental results. Finally, we conclude and sketch future research directions in Section V.

## II. LITERATURE REVIEW

### A. Skyline Query

Borzonyi et al. [5] first introduced the skyline operator into database systems and proposed Block Nested Loop (BNL), Divide-and-Conquer (D&C), and B-tree based algorithms forskyline computation from a sole database. As a variant of BNL, Chomicki et al. [6] improved BNL algorithm with the help of a Sort-Filter-Skyline (SFS) algorithm. In SFS, data needs to be pre-sorted using

| ID | Restaurants | Supermarkets |
|---|---|---|
| $\mathbf{h_1}$ | $r_1, r_6$ | $s_1, s_2$ |
| $\mathbf{h_2}$ | $r_1, r_6$ | $s_1, s_2$ |
| $\mathbf{h_3}$ | $r_8, r_9$ | $s_7, s_8$ |
| $\mathbf{h_4}$ | $r_3, r_4$ | -- |
| $\mathbf{h_5}$ | $r_7, r_8, r_9$ | $s_4, s_8$ |

Figure 4. Surrounding facilities satisfying the distance

| ID | Restaurants | | Supermarkets | |
|---|---|---|---|---|
| | Objects | Total count | Objects | Total count |
| $\mathbf{h_1}$ | $r_6$ | 1 | $s_1, s_2$ | 2 |
| $\mathbf{h_2}$ | $r_6$ | 1 | $s_1, s_2$ | 2 |
| $\mathbf{h_3}$ | $r_8, r_9$ | 2 | $s_7, s_8$ | 2 |
| $\mathbf{h_4}$ | $r_3, r_4$ | 2 | -- | 0 |
| $\mathbf{h_5}$ | $r_8, r_9$ | 2 | $s_4, s_8$ | 2 |

Figure 5. Surrounding facilities satisfying both the distance and the condition

a monotone scoring function, which can simplify the selection of skyline objects. Tan et al. [7] proposed two progressive algorithms: Bitmap and Index. The bitmap algorithm represents points in bit vectors and performs bit-wise operations. On the other hand, the index approach uses data transformation and B+-tree indexing. Kossmann et al. [8] proposed a Nearest Neighbor (NN) method. It selects skyline points by recursively invoking R*-tree based depth-first NN search over different data portions. Papadias et al. [9] proposed a Branch-and-Bound Skyline (BBS) method based on the best-first nearest neighbor algorithm.

### B. Spatial Skyline Queries

In [10], Sharifzadeh et al. first addressed the problem of spatial skyline queries. They proposed two algorithms, $B^2S^2$ and $VS^2$, for static query points and one algorithm, $VCS^2$, for the query points whose location change over time. $VCS^2$ exploits the pattern of change in query points to avoid unnecessary re-computation of the skyline. The main limitation of $VS^2$ algorithm of this paper is that it can not deliver correct results in every situation. Son et al. [11] first noticed the problem of $VC^2$ algorithm. They then presented a simple and efficient algorithm that can compute the correct results. Guo et al. [12] introduced the framework for direction-based spatial skyline computation that can retrieve nearest objects around the user from different directions. They also developed an algorithm to support continuous queries. However, their algorithm for direction-based spatial skyline can not handle more than one query point.

There are several works about spatial skyline computation in road networks. Deng et al. [13] first proposed multi-source skyline query processing in road network. They proposed three different skyline query processing algorithms for the computation of skyline points. In [14], Safar et. al considered nearest neighbour based approach for calculating skylines over road networks and claimed that their approach performs better than the approach presented in [13]. Huang et al. [15] proposed two distance-based skyline queries techniques those can efficiently compute skyline queries over road networks.

The main limitations of the above works is that they do not consider the features of the facilities in skyline computation. However, consideration of features of facilities is very much important as without considering features of facilities we may not be able to retrieve good objects of the facilities. Our approach in this paper has considered both location and features of the facilities in skyline computation.

TABLE II.
DATABASE CONTAINING NON-SPATIAL INFORMATION OF TARGET FACILITY AND SURROUNDING INFORMATION

| SID | Price | Age | Restaurants-count | Supermarkets-count |
|---|---|---|---|---|
| $\mathbf{h_1}$ | 5 | 8 | 1 | 2 |
| $\mathbf{h_2}$ | 2 | 3 | 1 | 2 |
| $\mathbf{h_3}$ | 4 | 4 | 2 | 2 |
| $\mathbf{h_4}$ | 5 | 6 | 2 | 0 |
| $\mathbf{h_5}$ | 3 | 5 | 2 | 2 |

### C. Preference Based Skyline Query

Till now there is very little consideration about preference based skyline query processing. There is some consideration about preference based skyline queries in [16], [17]. As for the preference issue, authors in [16] provides a framework for skyline queries considering only one facility type. They did not consider surrounding facilities. On the other hand, our work in this paper considers the surrounding facilities in the choice of a skyline facility. Wong et al. [17] provides skyline queries based on users preference order in nominal attributes. They introduce the concept of converting the partial order of each nominal attribute to a complete order and then evaluate the skyline queries using the concept of implicit preference order tree. Their paper does not consider the issue of user location and surrounding environments. Our paper considers both users location and surrounding environments though we do not consider any nominal attribute.

## III. SKYLINE QUERIES FOR SELECTING SPATIAL OBJECTS

### A. Problem Formulation

Let us again consider the spatial information of three different facilities of Figure 2 as shown in Table I and non-spatial features information of these facilities as shown in Figure 3. We used a variant of $R$-tree index structure called $aR$-tree [18] to keep both spatial and non-spatial information of each facility.

Our method is based on four computation steps. First, a user specifies a place $Q$ and distance ($\epsilon_1$). Based on this information, at first we select spatial objects of the target facility like houses within the specified distance from $Q$. Second, the user specifies preferable surrounding facilities and conditions those influence the quality of

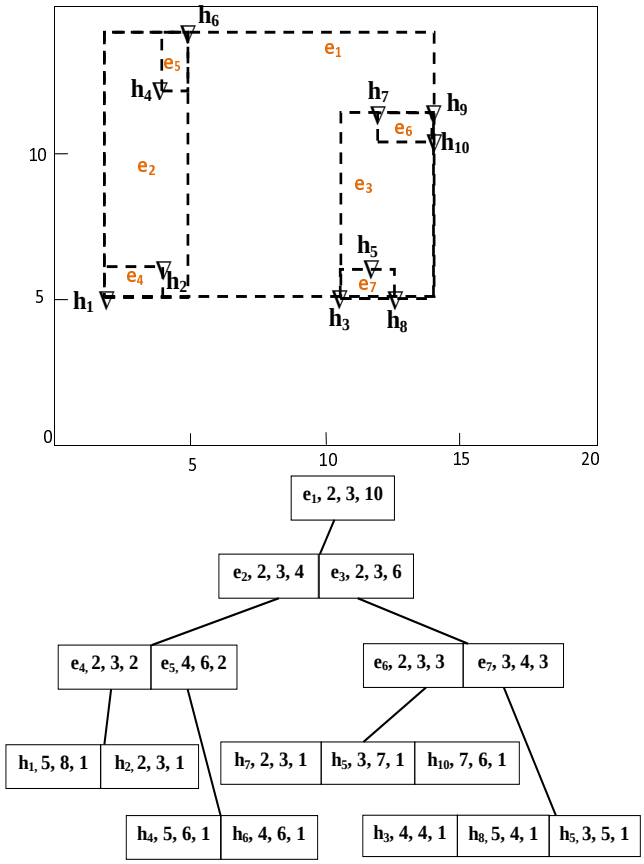



Figure 6. Objects of houses database indexed by an $aR$-tree

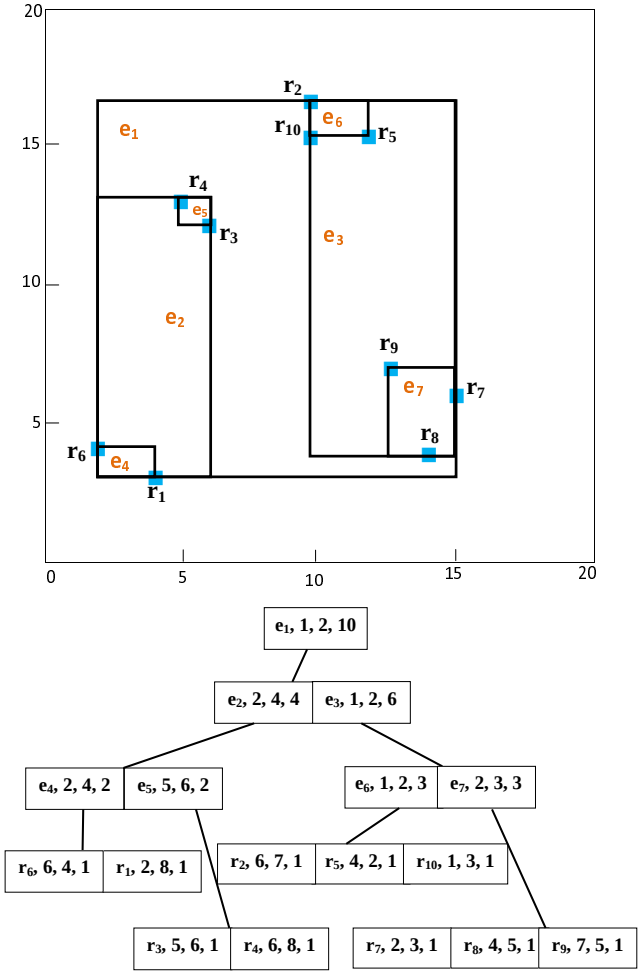



Figure 7. Objects of restaurants database indexed by an $aR$-tree

the selected objects. We, then, count the number of such objects for each selected surrounding facility. Third, we combine the count of surrounding facilities and the non-spatial information and make a table like Table II. Finally, we perform skyline queries to select spatial objects from the combined table.

### B. The Aggregation R-Tree Structure

The aggregation $R$-tree ($aR$-tree) is an $R$-tree each node of which corresponds to minimum bounding rectangle (MBR) that contains objects in a plane. Figure 6 depicts the MBRs and corresponding $aR$-tree for the houses $h_1, \cdots, h_{10}$. A leaf node in the $aR$-tree contains objects and their corresponding information. An internal node contains the minimum value in each attribute of its descendent objects and total number of descendent objects. For example, in Figure 6, the left most leaf node contains the information of $h_1$. The parent node $e_4$, which is MBR $e_4$, contains two objects $h_1$ and $h_2$. The minimum values of $e_4$ in attributes $a_1$ and $a_2$ are 2 and 3, respectively. Therefore, the node has an entry $(e_4, 2, 3, 2)$. Similarly, the root node has an entry $(e_1, 2, 3, 10)$.

We can construct the $aR$-trees for restaurants and supermarkets as shown in Figure 7 and 8, respectively.

### C. Computing Candidate Objects of Target Facility

We first select spatial objects of the target facility that are within the specified distance ($\epsilon_1$) from a given query point. We call such spatial objects as "candidate objects". We are considering that the houses those are within 1200 meters from $Q$ are "candidate objects". We can select candidate objects efficiently by using the $aR$-tree.

If a point $p$ is given, we find a top most MBR that contains $p$ and an internal node $e$ that corresponds to the MBR. Let $mindist(p, e)$ and $maxdist(p, e)$ denote the minimum and maximum possible distance between $p$ and any point in $e$.

In order to find objects that are within a user specified distance ($\epsilon_1$) from a query point $p$, we first check $mindist(p, e_{root})$. If we find the $mindist(p, e_{root})$ is less than or equal to $\epsilon_1$, we recursively continue the searching the child nodes. An MBR having $mindist(p, e)$ larger than $\epsilon_1$ will be pruned and will not be considered for the further processing. When we reach at a leaf node, we select spatial objects based on the distance from $p$.

Figure 9 shows the exploration of the nodes of the $aR$-tree when we search houses that are within 1200 meters from $Q$. From the $aR$-tree in Figure 9, shaded rectangles satisfy the condition. In this step, we find $h_1, h_2, h_3, h_4, h_5$ as the candidate houses.

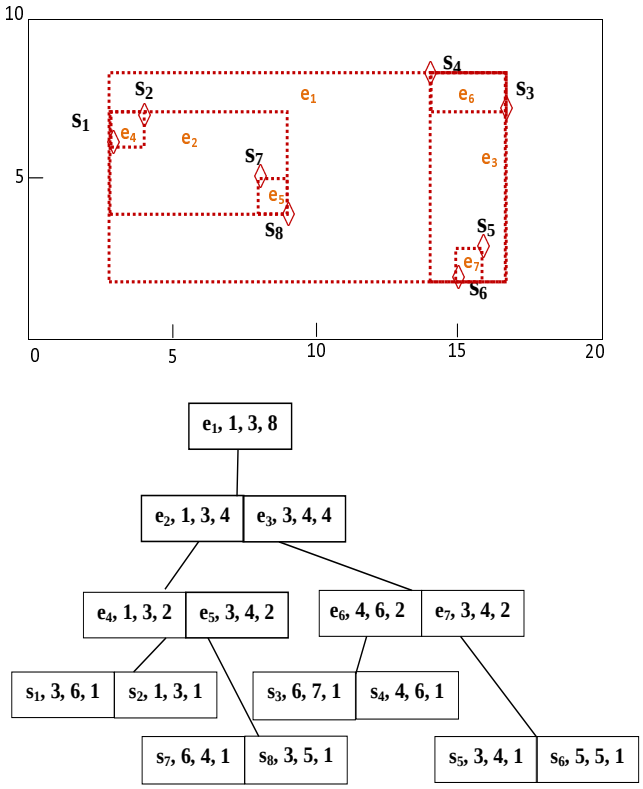



Figure 8. Objects of supermarkets database indexed by an $aR$-tree

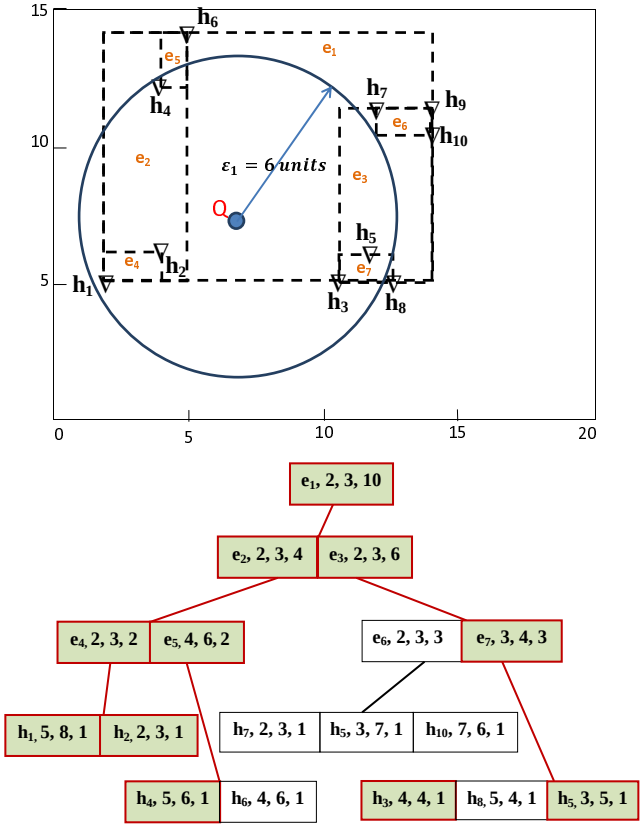



Figure 9. Candidate objects selection of target facility (house)

### D. Calculating Surrounding Facility Count

Let us consider that a user specify $S$ favourable surrounding facilities for selecting spatial objects. For selecting such spatial objects, we count the number of objects of the favourable facilities such as restaurants and supermarkets from each candidate object within the the user specified distance $\epsilon_2$.

We use the $aR$-tree to compute the surrounding facilities count for each candidate object. For an object $p$ in the candidate, we compute the surrounding facility count by traversing nodes from the root of the $aR$-tree. In a node $e$ of the tree, (1) if $mindist(p, e) > \epsilon_2$, we prune the subtree of the node. (2) If $maxdist(p, e) < \epsilon_2$ and value in each attribute satisfies favourable condition, we increment the surrounding facility count by the node's count without traversing its subtree. (3) Otherwise, we recursively traverse each child of $e$.

Figure 10 and Figure 11 illustrate the computation process of "surrounding supermarkets count" for $h_3$. In this example, we assume that $\epsilon_2 = 4$ and values in each attribute not less than 3 is favourable. The search procedure starts from the root $e_1$ of the $aR$-tree as shown in Figure 10. Since $mindist(h_3, e_1) = 3$ and $maxdist(h_3, e_1) = 8.54$, we examine the children, $e_2$ and $e_3$. In $e_2$, we recursively examine the children and can find that $e_5$ satisfies the condition (2) i.e. $maxdist(h_3, e_5) = 3.16 < \epsilon_2 = 4$ and there no object with value less than 3 in any of their attributes.

Therefore, we increment the count by 2. Note that we can skip the children, which are $s_7$ and $s_8$, of $e_5$. We continue the process from the next node similarly.

Figure 12 shows the tree structure for the search procedure for the restaurants count.

After this process, we get the counts of restaurants and supermarkets for each candidate object of target facility as shown in third and fifth columns of Figure 5.

### E. Combining Information and Generation of Final Result

After computing the count information, we extend the table of candidates by adding the count information. Table II is the example of the extended table of houses. In the table, there are five candidates of the target facility (house). First two numerical attributes represent their non-spatial attributes, while last two are for the count of restaurants and supermarkets. After obtaining such a table, we use Sort Filter Skyline (SFS) algorithm to obtain $h_1$, $h_3$, $h_4$ and $h_5$ as final skyline objects.

## IV. EXPERIMENTS

We simulated the proposed query function in a Mac PC having Intel core i5 processor, 2.3 GHz CPU, and 4 GB main memory. The simulated environment contained seven different facilities. We evaluated our function on synthetic datasets. As benchmark databases, we use the databases containing synthetic data with "anti-correlated" distribution. The parameters and values those were used in our experiments are given Table III.

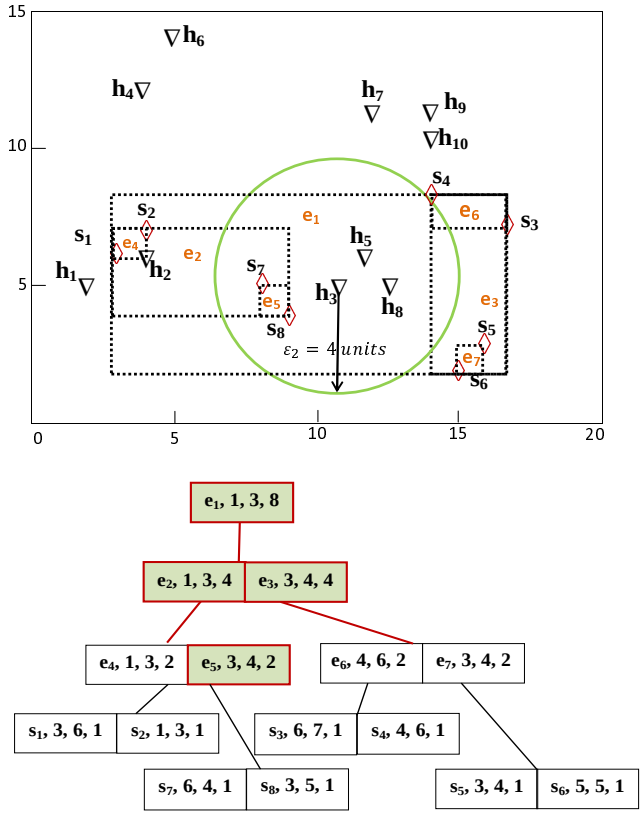



Figure 10. Computation of surrounding supermarket counts

| Step | Heap contents(entry e: mindist(. , .), maxdist(. , .)) | Influence |
|---|---|---|
| 1 | $(e_1: 3, 8.54)$ | 0 |
| 2 | $(e_2: 2, 8.25), (e_3: 3, 6.70)$ | 0 |
| 3 | $(e_5: 2, 3.16), (e_3: 3, 6.70), (e_4: 7.07, 8.25)$ | 0 |
| 4 | $(e_3: 3, 6.70), (e_4: 7.07, 8.25)$ | 2 |
| 5 | $(e_6: 3.16, 6.70), (e_7: 4.47, 5.83), (e_4: 7.07, 8.25)$ | 2 |
| 6 | $(s_4: 4.24, 4.24), (e_7: 4.47, 5.83), (s_3: 6.32, 6.32), (e_4: 7.07, 8.25)$ | 2 |
| 7 | $\phi$ | 2 |

Figure 11. Computation process of the supermarket counts for $h_3$

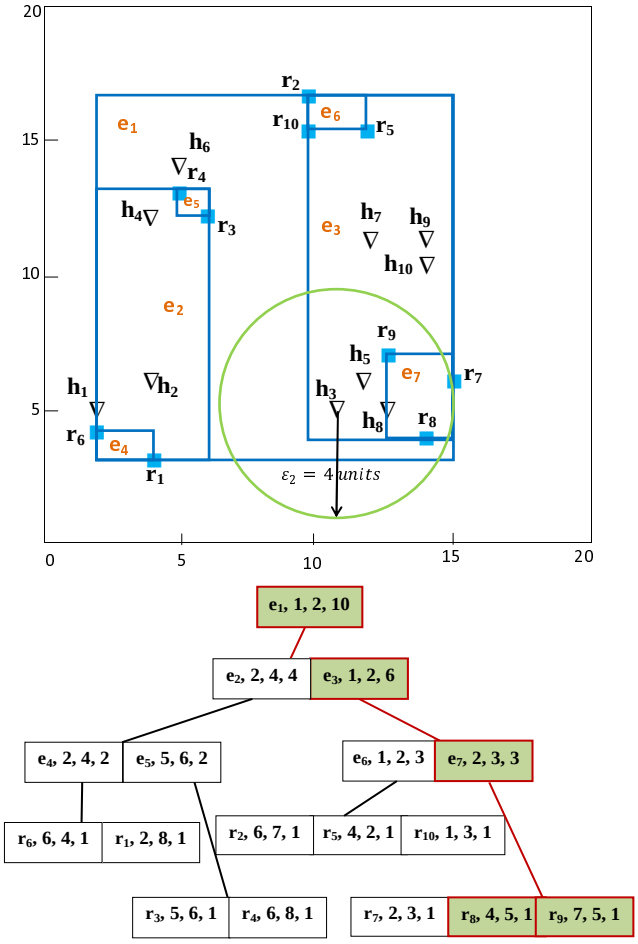



Figure 12. Computation of surrounding restaurants counts

TABLE III.
PARAMETERS AND VALUES

| Parameters | Values | Default Value |
|---|---|---|
| Raw data size of each facility | 20k, 40k, 60k, 80k, 100k | 40k |
| Types of surrounding facilities | 1, 2, 3, 4, 5 | 2 |
| Number of dimension of each object in each facility | 2D, 3D, 4D, 5D | 2D |
| Considerable distance from query point to the target facility in meters | 500, 1000, 1500, 2000, 2500 | 1000 |

We first evaluate the cost of building the $aR$-tree index structure for for each facility. Figure 13 shows the results. Here, we consider 2D, 3D, 4D and 5D cases for each facility type and varied the data size for each facility from 20k to 100k. From the result, we observe that there is an increase in time in building the index structure with the increase of data size. Also the time increases with the increase in data dimensionality. As such index is built in off line, this will not effect the performance of our system.

In the next experiment, we evaluate the retrieval time of the results with varying data size. Figure 14 shows the results. In this experiment, it is observed that response time increases with the increase of data size. It is also observed that response time gradually increases if the dimension increases.

Next, experiment shows the effect of the number of requested surrounding facilities. Figure 15 shows the result. We find that the computation time increases with the increase in the number of surrounding facilities as we need to consider more objects when there are more surrounding facilities.

Our next experiment shows the effect of query time with varying distance between query point and requested target facility. The result is shown in Figure 16. From the figure, we find that the query time increases with the increase of distance between query point and requested target facility. This is because with the increase in distance, we need to consider more objects in computation.

Our final experiment result is shown in Figure 17. It

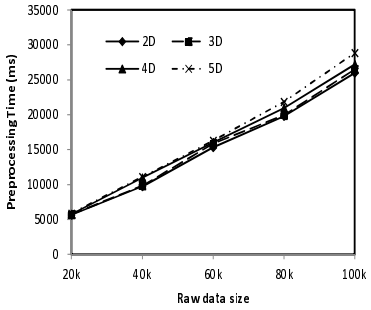



Figure 13. $aR$-tree index building cost

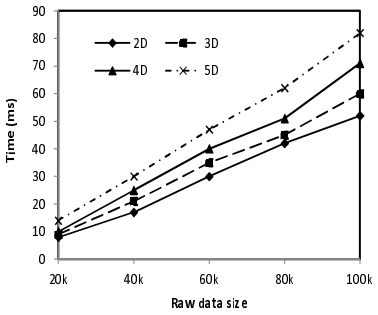



Figure 14. Retrieval time varying data size

shows the comparative analysis of the retrieval of points with and without considering surrounding facilities. Here, we can see that if we consider surrounding facilities more objects are retrieved. Thus a user has more option in his decision making.

## V. Conclusion

In this paper, we have proposed an approach for selecting spatial objects based on the user's choice on the location and surrounding facilities. The main feature of this paper is that we consider the influence of coexistence of some facilities in the surrounding areas. We have proposed an $aR$-tree based computation methodology that can provide real time response to the users efficiently.

In this paper, we simply counted the number of surrounding facilities. However, we have noticed that we should weight each surrounding facility based on distance, quality, and user's preferences in order to improve the result. To use a proper weighting is one of an important open problem of this work.

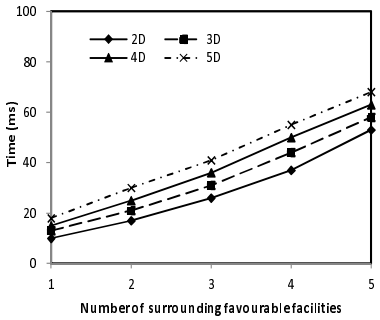



Figure 15. Retrieval time varying number of surrounding facilities

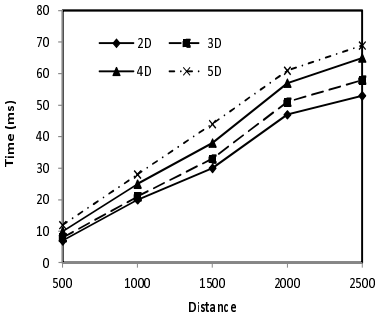



Figure 16. Retrieval time varying the distance of surrounding facilities

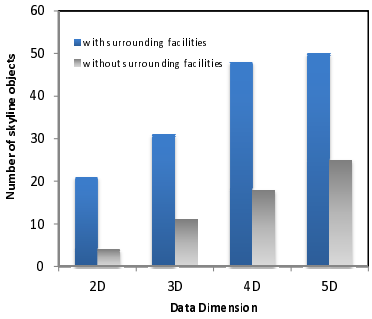



Figure 17. Comparative results

## Acknowledgments

This work was partially supported by KAKENHI (19500123). Mohammad Shamsul Arefin is supported by the scholarship of MEXT Japan.

## References

[1] G. Linden, B. Smith, J. York, Amazon.com recommendations:Item-to-item collaborative filtering, *In IEEE Internet Computing*, vol. 7, no. 1, 2003, pp. 76-80

[2] K. K. Ali, and W. V. STAM, Tivo: making show recommendations using a distributed collaborative filtering architecture. *In Proc. of the 10th ACM SIGKDD* , 2004, pp. 394-401.

[3] H. Luo, C. Niu, R. Shen, and C. Ullrich, A collaborative filtering framework based on both local user similarity and global user similarity, *In Machine Learning*. vol. 72, no. 3, 2008, pp. 231-245.

[4] A. S. Das, M. Datar, A. Garg, And S. Rajaram, Google news personalization: Scalable online collaborative filtering. *In Proc. of WWW*, 2007, pp. 271-280.

[5] S. Borzonyi, D. Kossmann, and K. Stocker, The skyline operator, *In Proc. of ICDE*, 2001, pp. 421-430.

[6] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, Skyline with presorting, *In Proc. of ICDE*, 2003, pp. 717-816.

[7] K.L. Tan, P.K. Eng, and B. C. Ooi, Efficient progressive skyline computation *In Proc. of VLDB Conference*, 2001, pp. 301-310.

[8] D. Kossmann, F. Ramsak, and S. Rost, Shooting stars in the sky: An online algorithm for skyline queries, *In Proc. of VLDB Conference*, 2002, pp. 275-286.

[9] D. Papadias, Y. Tao, G. Fu, and B. Seeger, An optimal and progressive algorithm for skyline queries, *In Proc. of ACM SIGMOD Conference*, 2003, pp. 467-478.

[10] M. Sharifzadeh, and C. Shahabi, The spatial skyline queries, *In Proc. of VLDB*, 2006, pp. 751-762.

[11] W. Son, M. Lee, H. Ahn, and S. Hwang, Spatial skyline queries: an efficient geometric algorithm, *In Proc. of SSTD*,2009, pp. 247-264.
[12] X. Guo, Y. Ishikawa, and Y. Gao, Direction-based spatial skylines, *In Proc. of ACM SIGMOD Conference*, 2010, pp. 73-80.
[13] K. Deng, X. Zhou, and H. T. Shen, Multi-source skyline query processing in road networks *In Proc. of ICDE*, 2007, pp. 796-805.
[14] M. Safar, D. E. Amin, and D. Taniar, Optimized skyline queries on road networks using nearest neighbors, *In Journal of Personal and Ubiquitous Computing*, vol. 15, issue 8, 2011 pp. 845-856.
[15] Y. K. Huang, C.H. Chang, and C. Lee, Continuous distance-based skyline queries in road networks, *In Journal of Information Systems*, vol. 37, 2006. pp. 611-633.
[16] K. Kodama, Y. Iijima, X. Guo, and Y. Ishikawa, Skyline queries based on user locations and preferences for making location-based recommendations, *In Proc. of ACM LBSN*, 2009, pp. 9-16.
[17] R. C. Wong, A. W. Fu, J. Pei, Y. S. Ho, T. Wong, and Y. Liu, Efficient skyline querying with variable user preferences on nominal attributes, *In Proc. of VLDB*, 2008, pp. 1032-1043.
[18] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao, Efficient OLAP operations in spatial data warehouses, *In Lecture Notes in Computer Science*, 2001, vol. 2121, pp. 443-459.

**Mohammad Shamsul Arefin** received his B.Sc. Engineering in Computer Science and Engineering from Khulna University, Khulna, Bangladesh in 2002, and completed his M.Sc. Engineering in Computer Science and Engineering in 2008 from Bangladesh University of Engineering and Technology (BUET), Bangladesh. Now he is a Ph.D. candidate at Hiroshima University with support of the scholarship of MEXT, Japan. He is a member of Institution of Engineers Bangladesh (IEB) and currently working as an Assistant Professor in the Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh. His research interest includes privacy preserving data mining, multilingual data management, semantic web, and object oriented system development.

**Xu Jinhao** received his B.Sc. Information management and information system in Computer Science and Engineering from Zhejiang Gongshang University, China, Zhejiang in 2010. Now he is a M.Sc candidate at Hiroshima University, Japan. His research interest includes spatial databases, preference-based queries.

**Chen Zhiming** received his B.Sc. Engineering in Computer Science from South China University of Technology in Guangzhou, China in 2009. Now he is master student in Data Management Laboratory of Hiroshima University, Japan. His research interest includes skyline query and recommender system.

**Yasuhiko Morimoto** is an Associate Professor at Hiroshima University. He received B.E., M.E., and Ph.D. from Hiroshima University in 1989, 1991, and 2002, respectively. From 1991 to 2002, he had been with IBM Tokyo Research Laboratory where he worked for data mining project and multimedia database project. Since 2002, he has been with Hiroshima University. His current research interests include data mining, machine learning, geographic information system, and privacy preserving information retrieval.