Electronic Preprint for Journal of Information Processing Vol.24 No.6

Regular Paper

Area Skyline Query for Selecting Good Locations in a Map

Annisa[1,2,a)] Asif Zaman[1,3,b)] Yasuhiko Morimoto[1,c)]

Received: March 20, 2016, Accepted: July 8, 2016

Abstract: We present a method for selecting good locations, each of which is close to desirable facilities such as stations, warehouses, promising customers' house, etc. and is far from undesirable facilities such as competitors' shops, noise sources, etc. Skyline query, which selects non-dominated objects, is a well known method for selecting small number of desirable objects. We use the idea of skyline queries to select good locations. However, locations are two dimensional data, while objects in the problem of conventional skyline queries are zero dimensional data. Comparison of two dimensional data is much more complicated than that of zero dimensional data. In this paper, we solve the problem of skyline query for two dimensional data, i.e., areas in a map. Experimental evaluations of the proposed method shows that our approach is able to find reasonable number of desirable skyline areas and can help users to find good locations.

Keywords: area skyline, spatial skyline, grid data structure, Voronoi diagram

1. Introduction

It is important to select good locations from a map in many location-based applications. In general, a better location is close to desirable facilities such as bus/train stations, warehouses, potential customers' house, etc. and is far from undesirable facilities such as competitors, noise sources, etc.

Skyline query [4] is a well known method for selecting small number of data objects. Let D be a d-dimensional database. A point p is said to dominate another point q if p is not worse than q in any of the d dimensions and p is better than q in at least one of the d dimensions. A skyline query retrieves a set of points, each of which is not dominated by another point.

Figure 1 shows a typical example of skyline. The table in this figure shows a list of hotels with two numerical attributes: price and rating, in a typical on-line booking system. A user can choose a hotel from the list according to her/his preference. In the example, we assume that smaller value is better in each attribute. In this situation, $\{h_1, h_3, h_4\}$ (see Fig. 1 (b)) are skyline objects. Object h_2 and object h_5, are dominated by h_3, while others (the skyline objects) are not dominated by another.

A number of efficient algorithms for computing skyline objects have been reported in Refs. [4], [5], [9], [14], [17]. In this paper, we use the idea of skyline queries to select good locations. However, locations are two dimensional data, while objects in the problem of conventional skyline queries are zero dimensional data. Comparison of two dimensional data is much more complicated than that of zero dimensional data.

[1] Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima, Hiroshima 739–8511, Japan
[2] On leave from Bogor Agricultural University, Indonesia
[3] On leave from Rajshahi University, Bangladesh
[a)] d144809@hiroshima-u.ac.jp
[b)] d140094@hiroshima-u.ac.jp
[c)] morimoto@mis.hiroshima-u.ac.jp

Example 1

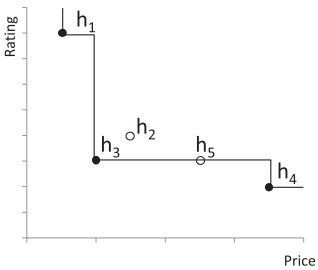
Consider a situation that a businessman wants to open a new supermarket. She/He would like to open the supermarket in a building for rent. She/He prefers a building that is close to desirable facilities such as a bus/train station, a university, and so forth, but it should be far enough from some undesirable facilities, such as a similar supermarket (a potential competitor).

Let P be a set of spatial points, which are buildings for rent, to be chosen. Let F be a set of facilities, which can be categorized into m types, $F1$, $F2$, ..., Fm. Each type is classifies into desirable or undesirable. We annotate "+" mark on the facility symbol of desirable facilities like F^+, while we annotate "−" mark on undesirable ones like F^-. **Figure 2** (a) shows an example of conventional skyline queries for spatial points. Points p_1, p_2, and p_3 ($\in P$) in this figure are buildings for rent. Points illustrated with star symbol, $F1^+ = \{f1_1^+, f1_2^+, \ldots, f1_{m_1}^+\} \in F$, represent locations of universities, which are desirable facilities. Another desirable facilities are stations, which are illustrated with triangle symbol, $F2^+ = \{f2_1^+, f2_2^+, \ldots, f2_{m_2}^+\} \in F$. Points with square symbol, $F3^- = \{f3_1^-, f3_2^-, \ldots, f3_{m_3}^-\} \in F$, represent competitors' supermarkets, which are undesirable facilities.

Based on the map of Fig. 2, we calculate a table as in Fig. 2 (b). In the table, we record distance from a building to the closest fa-

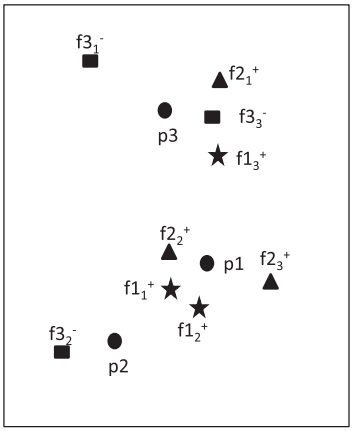
ID	Price	Rating
h_1	3	8
h_2	5	4
h_3	4	3
h_4	9	2
h_5	7	3

(a) Hotels



(b) Skyline

Fig. 1 Conventional skyline.



a) Map on Example 1

Point	F1+	F2+	F3-
p1	4	3	-10
p2	5	6	-2
p3	4	4	-6

b) Data Table on Example 1

Fig. 2 Spatial points in a map and data table.

cility of each of $F1^+$, $F2^+$, and $F3^-$ type. For example, the closest university ($F1^+$ facility (star)) from p_1 is $f1_2^+$ and the distance is 4. Similarly, the closest station ($F2^+$ facility (triangle)) from p_1 is $f2_2^+$ and the distance is 3. The closest competitor ($F3^-$ facility (square)) from p_1 is $f3_3^-$ and the distance is 10. We multiply -1 to each distance value of undesirable facilities F^- so that we can say that smaller value is better in each of the attribute. In the example, p_1 dominates p_2 and p_3 since p_1 is located closer to desirable facility and farther to undesirable facility. Therefore, skyline query for the spatial points returns p_1. Note that the selection problem can be solved by a conventional skyline query after we calculate the data table like in Fig. 2 (b).

However, in some real world examples, we cannot assume there are candidate points like $p_1, \ldots, p_3$, for the selection problem on a map. For example, the businessman wants to build a new supermarket if there is a good vacant area. The businessman may also want to take over a building that locates in a good area at any cost. In this situation, the candidate points are not given and the businessman has to find a good location in an area on the map. In other words, she/he has to find two dimensional area on the map.

Example 2

Assume that the businessman wants to find a place whose location is close to desirable facilities and is far from undesirable facilities. For example, the businessman has to find an area on the map in Fig. 2 without the candidate points like "p". Since comparison of areas is much more complicated than that of points, skyline query for areas is challenging.

We have published a feasibility report of this problem in Ref. [1]. Through the feasibility study, we have improved efficiency of the query processing and have solved technical problems found in Ref. [1] and have made the idea practical in this paper.

The contributions of this paper are summarized below:

(1) We have introduced a new skyline query, i.e., area skyline query in the literature.
(2) We have proposed an efficient and practical solution of the area skyline query.
(3) We have conducted intensive experiments to prove the efficiency of our algorithm.

The rest of this paper is organized as follows. Literature review about some related works on skyline queries and spatial skyline queries are presented in Section 2. Section 3 presents the notions and properties about the area selection problem and skyline area. This section also gives brief explanation of our proposed approach and our previous work in Ref. [1]. In Section 4, we present the result of our experiments. Finally, we conclude the paper and show directions of our future works in Section 5.

2. Related Works

2.1 Skyline Query

Borzsonyi et al. first introduced the skyline operator over large databases and proposed three algorithms: *Block-Nested-Loops* (BNL), *Divide-and-Conquer* (D&C), and B-tree-based schemes [4]. *Sort-Filter-Skyline* (SFS), which improves BNL by presorting, was proposed by Chomicki et al. as a variant of BNL [5]. Two progressive methods Bitmap and Index for computing skyline have been proposed by Tan et al., which improve previous algorithm [16]. Currently, the most efficient method in computing skyline is *Branch-and-Bound Skyline* (BBS), proposed by Papadias et al., which is a progressive algorithm based on the *Best First-Nearest Neighbor* (*BF-NN*) algorithm [13].

2.2 Spatial Skyline Query

Spatial skyline query was first introduced by Sharifzadeh et al. [15]. Let the set P contain points in the d-dimensional space R_d, and $D(.,.)$ be a distance metric defined in R_d where $D(.,.)$ obeys the triangle inequality. Given a set of d-dimensional query points $Q = \{q_1, \cdots, q_n\}$ and the two points p and p' in R_d. In their definition, p spatially dominates p' with respect to Q iff $D(p, q_i) \leq D(p', q_i)$ for all $q_i \in Q$ and $D(p, q_j) < D(p', q_j)$ for some $q_j \in Q$ [15]. Spatial skyline is a set of points that are not dominated by another point.

There exists other research works of spatial skyline problem such as in Refs. [3], [7], [8], [10]. In spatial skyline, distance is the first parameter that needs to be considered. In addition to that, sometimes users also consider their preferences. In Ref. [8], Kodama et al. not only consider the nearest distance but also consider the type of restaurant, which is a non-spatial preferences.

Conventional skyline queries compare non-spatial attributes of the candidate skyline points. However, surrounding facilities, which are not taken into account in the conventional skyline queries, are also important especially in spatial databases. Arefin et al. utilized surrounding facilities for calculating the importance of locations and demonstrated that surrounding environment is as important as other attributes for selecting spatial objects [3].

Guo et al. presented direction based spatial skyline [7]. A candidate object, in their query, not only has its distance but also has its direction from a moving user. The nearest object in the same direction to the user heads toward would be considered as preferable spatial object.

In Ref. [10], Lin et al. introduced the problem of spatial skyline query with different types of facilities, called general spatial skyline query. General spatial skyline query tries to find skyline objects that has smaller distance from every type of facilities.

Different from Sharifzadeh work, the problem of the farthest

spatial skyline queries is proposed in Ref. [18]. Given data points P and query points Q in two dimensional space, the farthest spatial skyline query retrieves the data points which are farther from at least one query point than from all the other data points. This method is helpful to identify spatial locations which are far from undesirable locations. Therefore, this problem is important in perspective of business location. In their paper, You et al. proposed baseline algorithm called *Threshold Farthest Spatial Skyline* (TTFS) and improved the baseline algorithm in *Branch*-and-*Bound Farthest Spatial Skyline* (BBFS). BBFS uses top-down branch-and-bound search on R-tree to access nodes in decreasing order of the sum of distances from query points.

Lin et al. combined the farthest and nearest problem and proposed EFFN algorithm to find targets with the nearest desirable neighbors and farthest undesirable neighbors in Ref. [11]. Consider a set of data points that represent desirable locations f and another set of data points which represents undesirable locations df. Given another set of data points P in the space as candidate locations, the EFFN algorithm is able to find the skyline location according to the nearest distance from desirable locations and the farthest distance from undesirable locations. They used quad-tree based data structure to find the nearest neighbor object from desirable facilities, and built quadrant area to find the farthest neighbor.

Arefin et al. proposed a spatial skyline query for group of users located at different positions [2]. Their method can select a convenient place for all users of a group if the group want to hold a meeting in a restaurant for example. They used Voronoi diagram to retrieve non-dominated objects in spatial sub-space, because Voronoi diagram is able to find the nearest spatial object from a given query point efficiently.

All of the above studies are based on the assumption that there are candidate points to choose skyline location and focused only on spatial data points. In this paper, different from previous researches, we focused on the problem of area selection in which the location of candidate query points q are not given.

3. Area Skyline Query

In this section, we propose a selection method of areas in a map, which we call "Area Skyline Query".

3.1 Problem Definition

Let A be a rectangular target area, where the businessman wants to build his supermarket, on a map. Let $F = \{F1, \ldots, Fm\}$ be a set of facility types, which can be categorized into m types. Each type is classified into desirable (annotated by + mark) or undesirable (annotated by − mark). Each facility has several facility objects, for example, a desirable facility $F1^+$ has three objects $F1^+ = \{f1_1^+, f1_2^+, f1_3^+\}$.

3.1.1 Grids and Vertexes

For simplicity, we assume the rectangular target area A is a square region. We, first, divide A into $s \times s$ grids. **Figure 3** is an example of such grids. To identify a grid, we assign unique ID number to each of the grid from top-left $g_{(0,0)}$ to bottom-right $g_{(s-1,s-1)}$.

Each grid is surrounded by four vertexes, which we denoted as $v_{(i,j)}$, $v_{(i,j+1)}$, $v_{(i+1,j)}$, and $v_{(i+1,j+1)}$, for the top-left (tL), top-right (tR), bottom-left (bL), and bottom-right (bR) of $g_{(i,j)}$ ($0 \le i \le s$, $0 \le j \le s$), respectively.

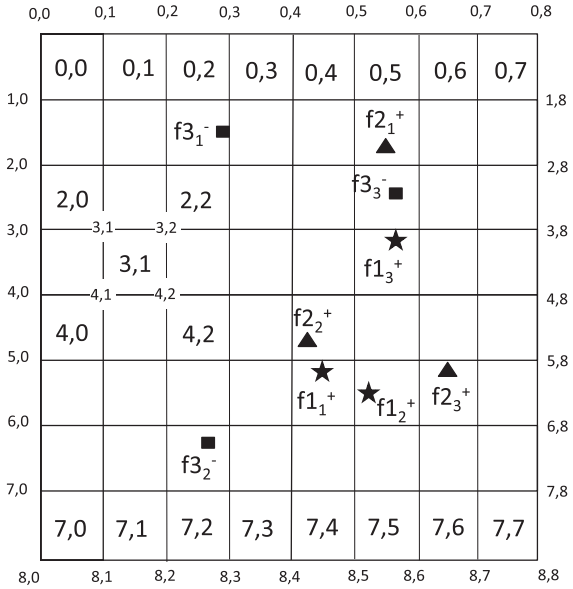


Fig. 3 Targeted area divided into square grids.

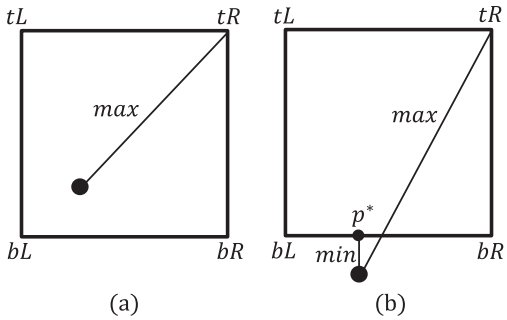


Fig. 4 Min. and Max. distance between grid and point.

3.1.2 Distance between Grid to Point

Given two points p and q in A, let $dist(p, q)$ be the Euclidean distance between p and q. We have to calculate the distance from a given query point q and a grid g in A. Since g is a two dimensional region, we define two distance functions between g and q, $dist_{min}(g, q)$ and $dist_{max}(g, q)$, which we call "minimum distance" and "maximum distance", respectively. The definition of $dist_{min}(g, q)$ and $dist_{max}(g, q)$ are as follows:

$$dist_{min}(g,q) = \begin{cases} 0 & \text{if } q \text{ lies inside } g. \\ min\{dist(p,q) \mid {}^{\forall}p \text{ inside } g\} & \\ & \text{if } q \text{ lies outside } g. \end{cases}$$

$$dist_{max}(g,q) = max\{dist(p,q) \mid {}^{\forall}p \text{ inside } g\}$$

Figure 4 shows examples of "minimum distance" and "maximum distance". Figure 4 (a) shows an example if the query point (black dot) is inside the grid. The "minimum distance" of the grid from the query point is 0, and the "maximum distance" is the distance from the query point to the farthest vertex, which is (tR), of the grid. In Fig. 4 (b), the min distance from the query point (black dot) to the grid is the distance between the point to p^*, which is the closest point from the query point inside the grid. The max distance is the distance between the query point to tR.

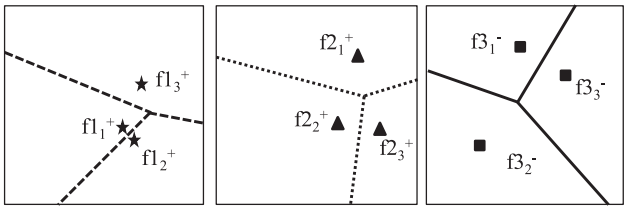
3.1.3 Voronoi Diagram

Given a query point q in A. To find the closest object of a facility type F from the query point, we used Voronoi Diagram. Given a set of n objects of F, the Voronoi diagram of the facility is the subdivision of A into n disjoint Voronoi regions. Each Voronoi region contains an object, say f_i where $1 \le i \le n$, which is called the Voronoi point of the region $V(f_i)$. In the Voronoi diagram of F, a query point q lies in the region $V(f_i)$ if and only if $dist(q, f_i) \le dist(q, f_j)$ for each $f_j \in F$ with $j \ne i$, where $dist(q, f)$ denotes the distance between q and f. For example, if the query point lies in the region that contains $f1_1^+$ in **Fig. 5** (a), the closest university (star) from the query point is $f1_1^+$.

3.1.4 Grid Dominance and Area Skyline

Let fk^{min*} be the object whose $dist_{min}(g, fk^{min*})$ is smaller than or equal to those of any other object in Fk. Similarly, let fk^{max*} be the object whose $dist_{max}(g, fk^{max*})$ is larger than or equal to those of any other object in Fk. Let $dist_{min}(g, Fk)$ and $dist_{max}(g, Fk)$ be the minimum distance and the maximum distance, respectively, from grid g to fk^{min*} and fk^{max*}. For two grids, g_i and g_j, we say g_i dominates g_j iff $dist_{max}(g_i, Fk) \le dist_{min}(g_j, Fk)$ for all k $(1 \le k \le m)$. Area skyline of A is the set of all non-dominated grids in A.

Figure 6 illustrates the grid dominance. Let f, the black dot, be the nearest Fk facility for $g(1,1)$, $g(1,2)$, $g(2,1)$ and $g(2,2)$. For grid $g(1,1)$, $dist(f,u)$ is the minimum distance of Fk. Similarly, $dist(f,t)$, $dist(f,r)$, and $dist(f,s)$ are the minimum distance for $g(1,2)$, $g(2,1)$, and $g(2,2)$, respectively. For grid $g(1,1)$, $dist(f,x)$ is the maximum distance of Fk. Similarly, $dist(f,w)$, $dist(f,t)$, and $dist(f,v)$ are the maximum distance to $g(1,2)$, $g(2,1)$, and $g(2,2)$, respectively. The (blue) circle is the circle whose radius is the maximum distance from f to $g(2,1)$. As we can see in the figure, $g(1,1)$ and $g(2,2)$ intersect with this circle, which means that those two grids have smaller minimum distance from f. In this situation, we say that $g(2,1)$ and $g(1,1)$ are uncomparable with respect to f. Similarly, $g(2,1)$ and $g(2,2)$ are uncomparable. On the other hand, $g(1,2)$ does not intersect with the circle. In this situation, we say that $g(2,1)$ dominates $g(1,2)$ with respect to f.



(a) F1⁺ (b) F2⁺ (c) F3⁻

Fig. 5 Voronoi diagram for $F1^+$ (a), $F2^+$ (b), and $F3^-$ (c).

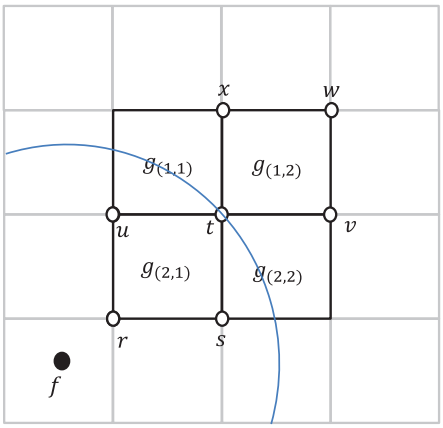


Fig. 6 Grid dominance situation.

3.2 Area Skylines Algorithm

To handle the area skyline query, we first divide the target area into square-grids. Next, for each grid, we calculate the minimum and maximum distances of each type of facilities. Using the min and max distances, we retrieve non-dominated grids, which are skyline areas.

3.2.1 Generate Square-grid Sub Areas

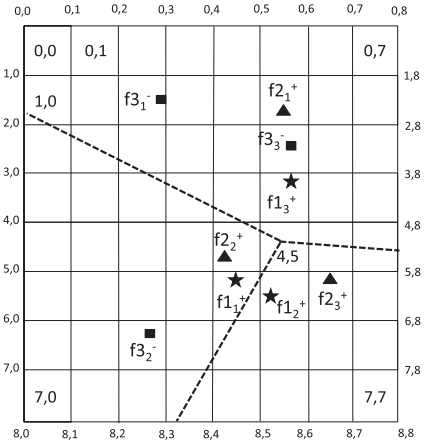
Suppose an area A in Fig. 2 (a) has been divided into 8×8 grids ($s = 8$). Figure 3 shows the grids. Each of the grids and its surrounding vertexes have identification number. For example, grid $g(3,1)$ is surrounded by vertexes $v(3,1)$, $v(3,2)$, $v(4,1)$, and $v(4,2)$.

3.2.2 Min-Max Distance Calculation

In this step, we build Voronoi diagram for each type of facilities. Consider Example 2 and Fig. 3 again. In the example, there are three types of facilities. Therefore, we need to build three Voronoi diagrams. Then, we find the closest facility for each type from each grid as follows.

At first, we find the closest facility from each of the surrounding vertexes. **Figure 7** (a) shows an example of Voronoi diagram for facilities of type $F1^+$ (star symbol). Figure 7 (b) shows the closest $F1^+$ from each of the surrounding vertexes, $v(0,0)$, ..., $v(8,8)$.

We calculate the minimum distance for Fk type for each grid as follows. First of all, for each Fk object, find the grid that contains the Fk object and set the minimum distance for Fk of the grid to zero. Next, for each non-zero grid g, we calculate the minimum distance by using the distances of the vertexes to their closest facility object. These distances are recorded in a table like Fig. 7 (b), that shows the distance from vertexes to their closest facility type $F1^+$. Since a grid has four vertexes and each vertex can be located in different Voronoi cell, the closest object of each



(a)

Vertex ID	F1	Dist
0,0	$f1_3^+$	140
0,1	$f1_3^+$	115
0,2	$f1_3^+$	93
...	...	...
4,4	$f1_1^+$	...
4,5	$f1_3^+$	...
...	...	...
6,1	$f1_1^+$	...
7,4	$f1_2^+$	...
...	...	...
8,7	$f1_2^+$	...
8,8	$f1_2^+$	...

(b)

Fig. 7 (a) Voronoi diagram for facility $F1^+$ (*star*), (b) Closest $F1^+$ for each vertex.

facility type for each vertex on one grid may be different. For example, Grid $(4,5)$ in Fig. 7 (a), the closest $F1^+$ facility of surrounding vertex $v(4,5)$ and $v(4,6)$ is $f1_3^+$. The closest $F1^+$ facility of $v(5,5)$ is $f1_1^+$. The closest $F1^+$ facility of $v(5,6)$ is $f1_2^+$.

Figure 8 shows how to calculate minimum and maximum distance for non-zero grid. Let (x,y), (x',y), (x,y'), and (x',y') be the coordinate of four vertexes of grid g. Since we have calculated minimum distance of all vertexes like in Fig. 7 (b), let's assume that (x,y), depicted by the white dot, has the smallest distance value among the 4 surrounding vertexes of g. Let (a,b), depicted by the black dot, be the coordinate values of the closest Fk object from (x,y). The distance between (a,b) and (x,y) is not always be the minimum distance of (a,b) to g.

There are three cases to calculate minimum distance of a grid, which will be explained as follows. Let (x',y) and (x,y') be coordinate values of the two adjacent vertexes of (x,y). First case, if a is between x and x' and b is not between y and y', then the minimum distance value is the difference between y and b. Second case, if a is not between x and x' and b is between y and y', then the minimum distance value is the difference between x and a. Otherwise, for the third case, the minimum distance value of g to Fk is the Euclidean distance between (x,y) and (a,b). In Fig. 8, minimum distance of g for Fk (the closest Fk is (a,b)) is $|b-y|$, shown as straight line. **Figure 9** (a), (b), (c) shows the three cases to calculate minimum distance of a grid. The maximum distance of g for Fk (the closest Fk is (a,b)) is the distance from (a,b) to the farthest surrounding vertexes of g, which is (x',y'). The dashed line in Fig. 8 shows the maximum distance.

After calculating minimum and maximum distance for each facility type for each grid, we record them into minmax table T.

3.2.3 Calculate Non-dominated Grid

To simplify the skyline query calculation, we multiply the min and max distance values for undesirable facilities by -1. After this simplification, smaller value is better in each of the distance values. We record $dist_{min}$ and $dist_{max}$ for each grid in the minmax table T, then calculate area skyline from T using grid dominance condition in Section 3.1.4. **Figure 10** shows Grid-based Area Skyline (*GASky*) Algorithm. After calculating min and max distance for each grid, we can retrieve the skyline records from the minmax table T as area skylines.

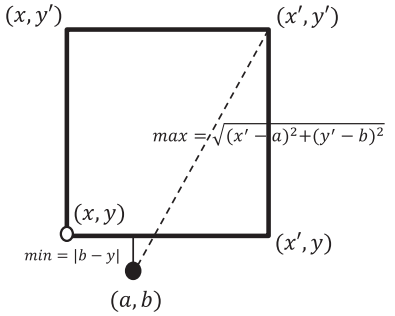


Fig. 8 Min-max calculation for non-zero grid.

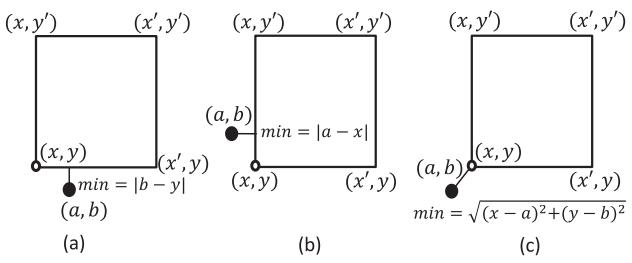


Fig. 9 Minimum distance calculation.

3.3 Unfixed-shape Area Skylines Query

In our previous feasibility study for "area skyline query" [1], we have considered area skyline query for unfixed-shape areas.

Consider Example 2 in Section 1. In the example, there are three types of facilities. For each facility, we first divide the targeted area A into several disjoint sub areas by using Voronoi diagram. Figure 5 shows three Voronoi diagram of the three facilities. Using the three Voronoi diagrams, we divide the targeted area into 13 disjoint areas, say $a_1, a_2, \ldots, a_{13}$, as shown in **Fig. 11**.

For each disjoint area, we calculate the two distances, which are min distance and max distance to the closest facility for each F^+ and F^-. The calculation of the min and max distance is similar to that of Section 3.1.2.

After the calculation of the min and max distances for all disjoint areas, we can calculate area skyline query. Skyline query

Algorithm 1 : Grid-based Area Skyline Algorithm (GASky Algorithm)

```
Input  : F_{1...k}, s
Output : Sky
1.  for each F_k ∈ F do
2.      build_voronoi V(F_k)
3.      for each vertex v(i,j) from v(0,0) to v(s,s)
4.          find the closest F_k in V(F_k) from v(i,j)
5.      for each grid g(i,j) from g(0,0) to g(s−1, s−1)
6.          calculate dist_min(g(i,j), F_k) and record it into T
7.          calculate dist_max(g(i,j), F_k) and record it into T

    // procedure to remove dominated grids
8.  for each record in T, t_{1...p} do
9.      if t_i.F_{k.max} ≤ t_j.F_{k.min} for all F_k ∈ F do
10.         remove dominated grid t_j from T

11. return Sky
```

Fig. 10 Grid-based area skylines algorithm.

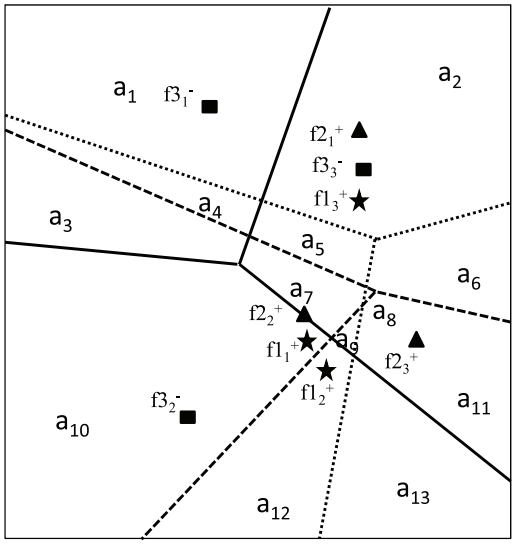


Fig. 11 Disjoint areas divided by 3 Voronoi diagrams.

Algorithm 2 : Unfixed-shape Area Skyline Algorithm (UASky Algorithm)

```
Input  : F_{1...k}
Output : Sky
1.  for each F_k ∈ F do
2.      build_voronoi V(F_k)
3.  divide area by all V(F_k) into m disjoint area a_{1...m}
4.  assign ID for each vertex of divided areas v_{1...n}
5.  for each F_k ∈ F do
6.      for each vertex v_{i ,i=1..n}
7.          find the closest F_k in V(F_k)
8.  for each F_k ∈ F do
9.      for each a_{i ,i=1..m}
10.         calculate dist_min(a_i, F_k) and record it into T
11.         calculate dist_max(a_i, F_k) and record it into T
    // procedure to remove dominated areas
12. for each record in T, t_{1...p} do
13.     if t_i.F_{k.max} ≤ t_j.F_{k.min} for all F_k ∈ F do
14.         remove dominated area t_j from T
15. return Sky
```

Fig. 12 Unfixed-shape area skylines algorithm.

for unfixed shape areas selects all non-dominated areas from the set of the disjoint areas. **Figure 12** shows Unfixed-shape Area Skyline (*UASky*) Algorithm.

Through intensive experiments in unfixed-shape area method in our feasibility study [1], we found that a large area is likely to be selected as skyline. This is because a large area have larger max distance which makes it difficult to be dominated. Moreover, in UASky we cannot change the shape, size and number of disjoint areas because they are produced by intersection of all Voronoi diagrams. Thus, user can not control the number of skyline areas in UASky. In contrast, in GASky user can control the number of grid, which can prevent the problem in UASky.

We will discuss this issue later in Section 4.

3.4 Computational Cost Analysis

Note that step to remove dominated areas in both algorithm is using the same conventional skyline algorithm, therefore the computational cost for this step is the same for UASky and GASky and is not included in this calculation. Both UASky and GASky have same procedures which are generating Voronoi diagram for each type, finding the closest facility for each type to each vertex, and calculating minimum and maximum distance for each area. The difference between UASky and GASky is that in GASky we need to generate $s \times s$ grids, while in UASky we need to divide the whole area by all Voronoi diagrams to get disjoint areas (step 3 in UASky).

In the computational geometry literature, following Voronoi diagram's properties have been studied. The worst time complexity to build a Voronoi diagram of n points is $O(n \log n)$ and the expected time complexity to find the nearest Voronoi point is $O(\log n)$ [6]. If we utilize a quaternary tree together with a Voronoi diagram, the expected time complexity to construct a Voronoi diagram can be reduced to $O(n)$ and the time complexity to find the nearest Voronoi point can be reduced to $O(1)$ [12].

Let m be the total number of facility types, n be the number of objects in each type. The expected time complexity to construct m Voronoi diagrams for GASky and UASky is $O(mn)$. The time to find the nearest facility for each point is $O(m)$.

Since there are $s \times s$ grids in GASky, there are $O(s^2)$ surrounding points. Therefore, GASky takes $O(s^2 m)$ in addition to the Voronoi diagrams' construction time. As for UASky, there are $O(mn)$ Voronoi edges in total. The number of intersections of $O(mn)$ edges is $O((mn)^2)$, which are the number of surrounding vertices. Therefore, UASky takes $O((mn)^2)$ in addition to the Voronoi diagrams' construction time.

4. Experimental Evaluation

In this section, we conduct four experiments to examine selectivity and performance. We performed our experiments in a PC with Intel Core i5 3.2 GHz processor with 4 GB of RAM. We evaluated our algorithm using synthetic datasets. Each experiment is repeated ten times and we reported the average.

Since the step to remove dominated areas is the same for both UASky (Fig. 10 step 8–10) and GASky (Fig. 12 step 12–14), and the performance of this step is not different from other conventional skyline algorithm, we exclude it from the processing time calculation.

4.1 Comparison between GASky and UASky

In these experiments, we compared the performance of the proposed algorithm (GASky) and our previous algorithm (UASky) in the feasibility study [1].

First, we compared processing time of GASky with that of UASky, and second, we compared ratio of skyline areas of both algorithm. We also examined the effect of number of facility type and number of objects in both algorithm.

We used two different synthetic data, say $DB1a$ and $DB1b$ for these experiments.

For $DB1a$, the default number of objects is 128. We varied the number of types to 2, 4, 8, 16, and 32 respectively. In these experiments, the number of desirable types is set to be the same as the number of undesirable types. For $DB1b$, we fixed the number of facility type is 2. Then, we varied the number of objects to 8, 12, 24, 48, 96, and 128.

In $DB1a$, the number of disjoint areas resulted by UASky was around 1,800 and in $DB1b$ it was around 260 disjoint areas. In the first experiment, we set the number of grids in GASky so that the number of grids in GASky becomes almost same to the number of disjoint areas produced by UASky.

The results of first experiments are shown in **Figs. 13** and **14**. From these figures, we can see that GASky is faster than UASky especially when the number of types become large. We can also observe that the number of facility type affect the processing time more than the number of object. One of the main reason is that the number of facility type affects the time to build Voronoi diagrams. The ratios of skyline areas of this experiment are reported in **Figs. 15** and **16**. From these figures, we can see that using the similar number of grids, GASky has better skyline ratio than UASky. In these figures, skyline ratio decreases up to 40%.

In order to have better skyline ratio, in the second experiments, using $DB1a$ and $DB1b$, we set the number of grids in GASky to be 10,000 and 2,500. The ratios of skyline areas are reported in **Figs. 17** and **18**. Figures 17 and 18 shows that increasing the number of grids can reduce the skyline ratio until 5%.

Figures 17 and 18 also shows that GASky is sensitive to the

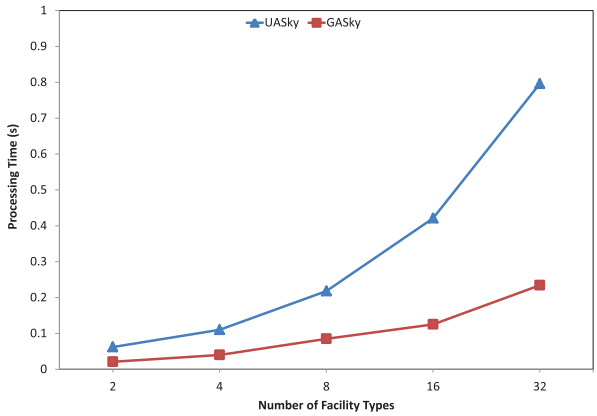


Fig. 13 Processing time of *DB*1*a*.

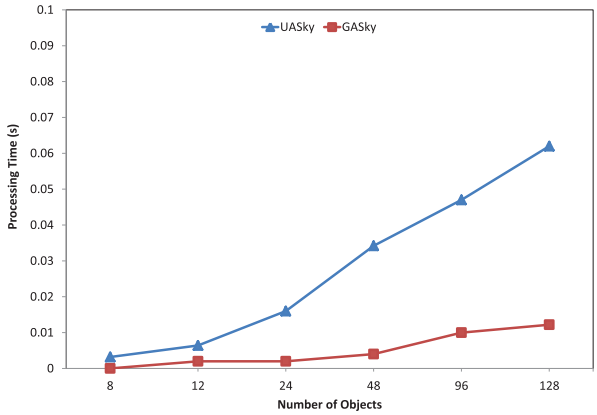


Fig. 14 Processing time of *DB*1*b*.

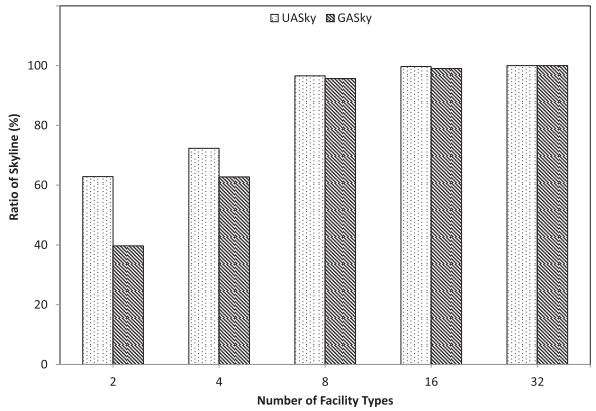


Fig. 15 Skyline ratio of *DB*1*a*.

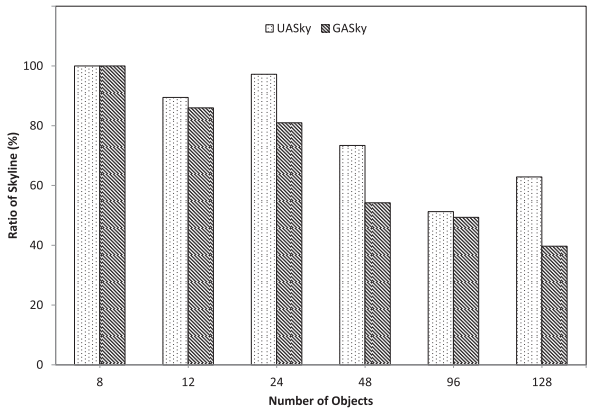


Fig. 16 Skyline ratio of *DB*1*b*.

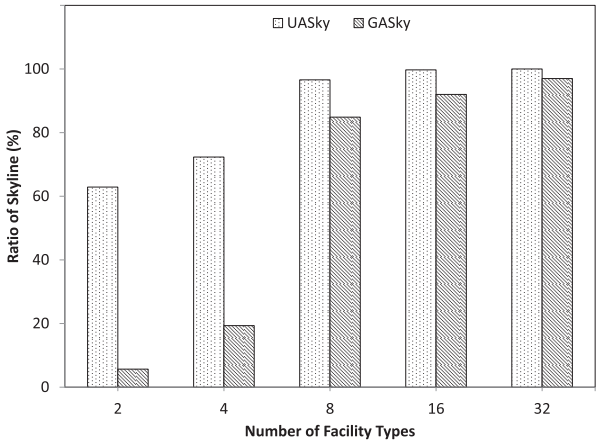


Fig. 17 Skyline ratio of *DB*1*a* with 10,000 grids for GASky.

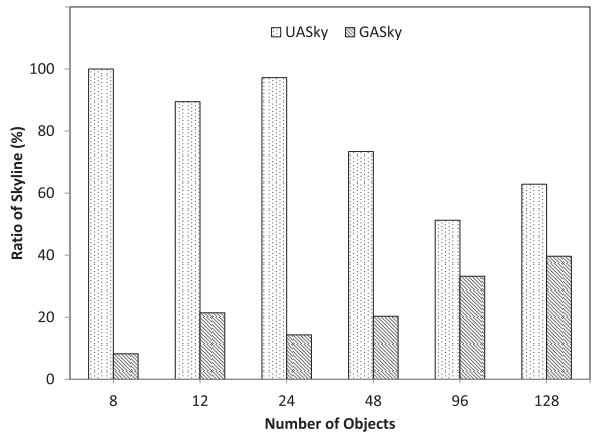


Fig. 18 Skyline ratio of *DB*1*b* with 2,500 grids for GASky.

increase of facility types rather than the increase of objects. One of the main reasons is as follows. The increase of the facility types with fixed number of objects causes decrease of density of each facility. It is equivalent to enlarge each grid, which tends to increase the ratio of skyline areas.

4.2 Effect on Grid Number

In these experiments, we used four different synthetic data, say *DB*2*a*, *DB*2*b*, *DB*2*c*, *DB*2*d*. *DB*2*a* is random data consists of 128 objects with two types of facilities, one is desirable and the other is undesirable facility. *DB*2*b* has the same types of facilities but consists of 256 objects. *DB*2*c* and *DB*2*d* consist of 128 and 256 objects respectively, with four types of facilities, two types are desirable and the others are undesirable facilities.

For each data, we varied the number of grids to 16, 64, 144, 256, and 400. **Figures 19** and **20** shows the results. From the results in Fig. 19, we can observe that the processing time increases with the increase of the number of grid, and data that has more facility types and more number of objects also has larger processing time. The results in Fig. 20 illustrate that the ratio of skyline decreases with the increase of the number of grids, and data that has smaller number of facility types and number of objects decreased the ratio of skylines. In other words, we can decrease the ratio of skyline area by increasing the number of grids. Thus,

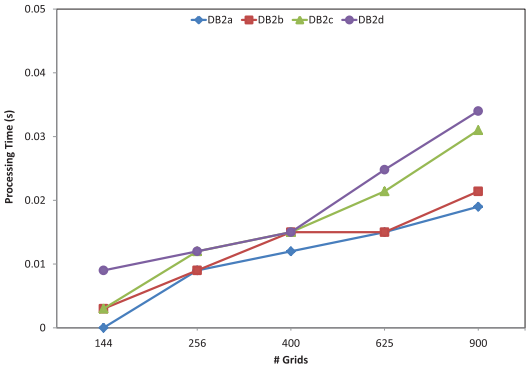


Fig. 19 Processing time varied with number of grids.

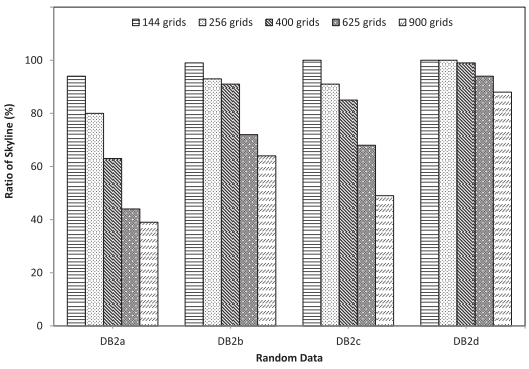


Fig. 20 Ratio of skyline varied with number of grids.

higher number of grid means smaller size of each disjoint area, which in turn will decrease the ratio of skyline. By applying grid data structure, the GASky can control the number of area skyline by changing the number of grids.

In actual usage scenario, if a user prefers selective areas, she/he had better increase the s, which tends to reduce the ratio of skyline areas. Since GASky is sensitive to the increase of facility types, user should use larger s to reduce skyline ratio if she/he increases the number of facility type.

In our motivating example, UASky generated 13 disjoint areas with the ratio of skyline was 100%. Using the same method above, we applied GASky with 225 grids and the skyline ratio was decreased to 30%. **Figures 21** and **22** illustrates skyline area after applying UASky and GASky.

4.3 Effect of Ratio of Desirable and Undesirable Types

In this experiment, we investigated the effect of ratio of desirable (or undesirable) facility among all facilities in UASky and GASky. In this experiment, we set the total objects to 100, the number of facility type to 10, and varied the number of desirable and undesirable facility type to $(10+, 0-)$, $(8+, 2-)$, $(6+, 4-)$, $(4+, 6-)$, $(2+, 8-)$, and $(0+, 10-)$. **Figure 23** shows the results. We can see that the difference of the ratio has no significant effect to the processing time.

Moreover, this result also shows that GASky has better performance than UASky for different ratio of desirable and undesirable facilities.

4.4 Scalability

In this experiment, we examined the scalability of the proposed

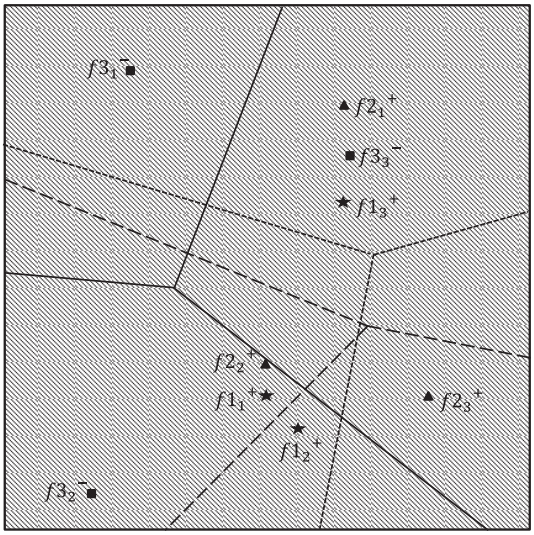


Fig. 21 All targeted area retrieved as skyline area using UASky.

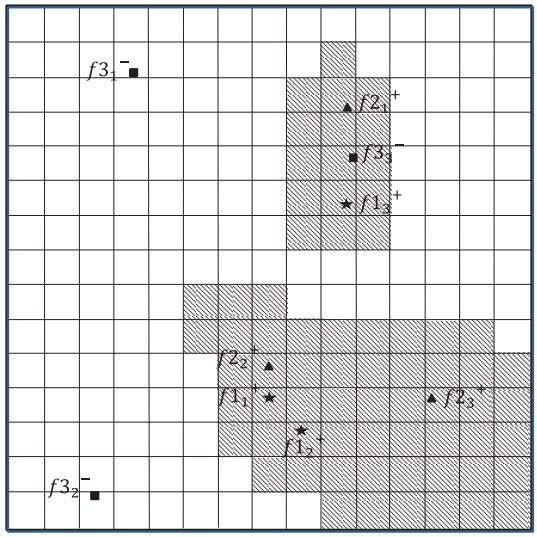


Fig. 22 Decreased skyline area by using GASky.

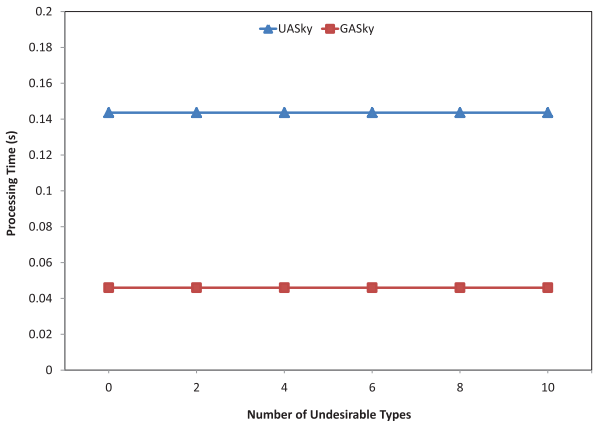


Fig. 23 Effect of desirable and undesirable types' ratio.

algorithm. For this purpose, we set the default number of facility's types to 4, among which two types are desirable facilities and other two types are undesirable facilities. We set the number of grids to 100. We varied the number of total objects to 100K, 200K, 300K, 400K, and 500K, respectively. **Figure 24** shows the results.

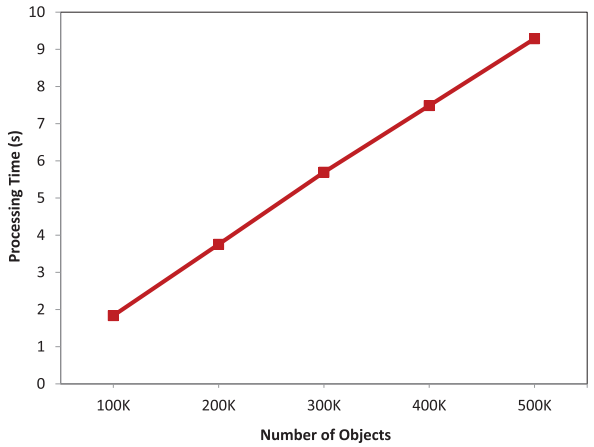


Fig. 24 Scalability of GASky algorithm.

In summary, all of the above experiments give the indication that the processing time depends on the number of objects, the number of facility types, and the number of grids. The processing time increases with the increase of the number of facility types, the number of objects, and the number of grids.

5. Conclusions and Future Works

Areas which are close to desirable facilities and far from undesirable facilities are important for various applications. The proposed area skyline queries help to find such areas, which are not dominated by another area. In addition to our motivating example, some concrete examples which could utilize this method are:

- In the business field: suppose a property company would like to build a new housing complex in a new region. To attract customers, the housing complex should be in an area that is close to train stations, shopping centers, and schools, and far from open landfill. The proposed method can help the company finding potential area for it's new project and gain knowledge about the region and it's facilities, thus reducing cost of surveying the whole region.
- In the travel planning: when planning trip to a new area or country, sometimes a traveler would like to stay in an area that will be convenient in location and cost. The proposed method can help a tourist to know which area is close to attraction sites, train stations, and convenience stores and is far from crime areas and polluted areas. After that the tourist can search for place of stay in the skyline areas.

This paper addresses a method to compute area skyline queries using grid data structure. Comprehensive experiments are conducted to demonstrate the effectiveness and efficiency of the algorithms.

In future, we will consider the challenging related open problems such as considering more than one object for each facility type, selecting k-dominant areas and how to utilize non-spatial property such as population density, price, etc., in the selection of areas.

Acknowledgments This work is supported by KAKENHI (23500180, 25.03040) Japan. Annisa is supported by Indonesian Government DG-RSTHE scholarship. A. Zaman is supported by Japanese Government MEXT Scholarship.

References

[1] Annisa, Siddique, M.A., Zaman, A. and Morimoto, Y.: A Method for Selecting Desirable Unfixed Shape Areas from Integrated Geographic Information System, *Proc. IIAI 2015*, pp.195–200 (2015).
[2] Arefin, M., Ma, G. and Morimoto, Y.: A Spatial Skyline Query for a Group of Users, *Journal of Software*, Vol.9, No.11, pp.2938–2947 (2014).
[3] Arefin, M., Xu, J., Chen, Z. and Morimoto, Y.: Skyline Query for Selecting Spatial Objects by Utilizing Surrounding Objects, *Journal of Software*, Vol.8, No.7, pp.1742–1749 (2013).
[4] Borzsonyi, S., Kossmann, D. and Stocker, K.: The skyline operator, *Proc. 17th International Conference on Data Engineering* (*ICDE*), April 2–6, Heidelberg, Germany, pp.421–430 (2001).
[5] Chomicki, J., Godfrey, P., Gryz, J. and Liang, D.: Skyline with Presorting, *Proc. 19th International Conference on Data Engineering* (*ICDE*), March 5–8, Bangalore, India, pp.717–719 (2003).
[6] Edelsbrunner, H.: *Algorithms in Combinatorial Geometry*, Vol.10, Springer Science & Business Media (2012).
[7] Guo, X., Ishikawa, Y. and Gao, Y.: Direction-based Spatial Skylines, *Proc. 9th ACM International Workshop on Data Engineering for Wireless and Mobile Access* (*MobiDE*), June 6, Indiana, USA, pp.73–80 (2010).
[8] Kodama, K., Iijima, Y., Guo, X. and Ishikawa, Y.: Skyline Queries Based on User Locations and Preferences for Making Location-based Recommendations, *Proc. International Workshop on Location Based Social Networks* (*LBSN*) November 3, Washington, USA, pp.9–16 (2009).
[9] Kossmann, D., Ramsak, F. and Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries, *Proc. 28th International Conference on Very Large Data Bases* (*VLDB*), August 20–23, Hong Kong, China, pp.275–286 (2002).
[10] Lin, Q., Zhang, Y., Zhang, W. and Lin, X.: Efficient general spatial skyline computation, *World Wide Web*, Vol.16, No.3, pp.247–270 (2013).
[11] Lin, Y.-W., Wang, E.-T., Chiang, C.-F. and Chen, A.L.P.: Finding Targets with the Nearest Favor Neighbor and Farthest Disfavor Neighbor by a Skyline Query, *Proc. 29th Annual ACM Symposium on Applied Computing* (*SAC*), March 24–28, Gyeongju, Korea, pp.821–826 (2014).
[12] Ohya, T., Iri, M. and Murota, K.: A Fast Voronoi Diagram Algorithm with Quaternary Tree Bucketing, *Information Processing Letters*, Vol.18, No.4, pp.227–231 (1984).
[13] Papadias, D., Tao, Y., Fu, G. and Seeger, B.: An optimal and progressive algorithm for skyline queries, *Proc. ACM SIGMOD*, June 9–12, California, USA, pp.467–478 (2003).
[14] Papadias, D., Tao, Y., Fu, G. and Seeger, B.: Progressive skyline computation in database systems, *ACM Trans. Database Systems*, Vol.30, No.1, pp.41–82 (2005).
[15] Sharifzadeh, M. and Shahabi, C.: The Spatial Skyline Queries, *Proc. 32nd International Conference on Very Large Data Bases* (*VLDB*), September 12–15, Seoul, Korea, pp.751–762 (2006).
[16] Tan, K.-L., Eng, P.-K. and Ooi, B.C.: Efficient Progressive Skyline Computation, *Proc. 27th International Conference on Very Large Data Bases* (*VLDB*), September 11–14, Rome, Italy, pp.301–310 (2001).
[17] Xia, T., Zhang, D. and Tao, Y.: On Skylining with Flexible Dominance Relation, *Proc. 24th International Conference on Data Engineering* (*ICDE*), April 7–12, Cancun, Mexico, pp.1397–1399 (2008).
[18] You, G.-W., Lee, M.-W., Im, H. and Hwang, S.-W.: The Farthest Spatial Skyline Queries, *Information Systems*, Vol.38, No.3, pp.286–301 (2013).

Annisa received her B.Sc. and M.Sc. in Computer Science of Bogor Agricultural University (IPB) and University of Indonesia in 2002 and 2007, respectively. Since 2002–present she is a member of Computer Science Department in Bogor Agricultural University. She is on study leave and currently a Ph.D. candidate at Hiroshima University under supervision of Yasuhiko Morimoto. Her research interests include data mining, skyline query and data management system.

Asif Zaman is a Ph.D. candidate under the supervision of Yasuhiko Morimoto at Hiroshima University, Japan. He completed his M.Sc. and B.Sc. (Hons) in Computer Science & Engineering from the Rajshahi University, Bangladesh. He is a faculty member and on study leave from Department of Computer Science & Engineering, Rajshahi University. His research interests include Data mining, Secure skyline computation, Computer virus & its security.

Yasuhiko Morimoto is an Associate Professor at Hiroshima University. He received his B.E., M.E. and Ph.D. degrees 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 interest include data mining, machine learning, geographic information system and privacy preserving information retrieval.

(Editor in Charge: *Toshiyuki Shimizu*)