Reverse Area Skyline in a Map

Annisa
Graduate School of Engineering,
Hiroshima University, Japan

Asif Zaman
Graduate School of Engineering,
Hiroshima University, Japan

Yasuhiko Morimoto
Graduate School of Engineering,
Hiroshima University, Japan

***Abstract*—Skyline query retrieves a set of data objects, each of which is not dominated by another object. On the other hand, given a query object q, "reverse" skyline query retrieves a set of points that are "dynamic" skyline of q. If q is a given preference of a user, "dynamic" skyline query retrieves a set of points that are not dominated by another point with respect to q. Intuitively,"reverse" skyline query of q retrieves a set of points that are as preferable as q. Area skyline query is a method for selecting good areas, each of which is near to desirable facilities such as stations, warehouses, promising customers' house, etc. and is far from undesirable facilities such as competitors' shops, noise sources, etc. In this paper, we applied reverse skyline concept to area skyline query and proposed Reverse Area Skyline algorithm. Analogically, given an area g, reverse area skyline query selects areas, each of which are as preferable as g. Assume that a real estate company wants to sell an area. Reverse area skyline query must be useful for such company to consider effective real estate developments so that the area attracts many buyers. Reverse area skyline query can also be used for selecting promising buyers of the area.**

Keywords*—skyline query; reverse skyline query; area skyline query

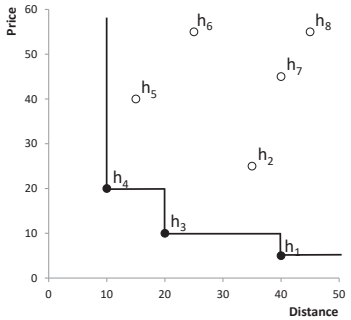
I. Introduction

Skyline query [1] is a widely applicable method for selecting small number of superior data objects. It retrieves a set of data objects, each of which is not dominated by another object. Given D as a d-dimensional database, an object p_i is said to dominate another object p_j if p_i is not worse in any of the d dimensions than p_j, and p_i is better than p_j in at least one of the d dimensions. Fig. 1 shows a typical example of skyline. Consider a typical online booking system. A user can select a hotel from the list in Fig. 1 (a) based on her/his preference on the price and distance of the hotel to the beach. Assume that smaller value is better for each attribute. In this situation, $\{h_1, h_3, h_4\}$ are skyline objects because they are not dominated by another object. Other objects $\{h_2, h_5, h_6, h_7, h_8\}$ are dominated by h_4. Fig. 1 (b) shows skyline hotels from the given hotel list.

Dynamic skyline query [2] and reverse skyline query [3] is a variant of skyline query. Given a query object q, "reverse" skyline query retrieves a set of points that are "dynamic" skyline of q. If q is a given preference of a user, "dynamic" skyline query retrieves a set of points that are not dominated by another point with respect to q. Intuitively,"reverse" skyline query of q retrieves a set of points that are as preferable as q. Assume that a businessman is running a hotel h_2. Reverse skyline query of h_2 retrieves a set of hotels that are as preferable as h_2. Therefore, he/she can expect customers who are interested in reverse skyline hotels of h_2, might also be interested in h_2. Skyline query retrieves candidate hotels from users' perspective, while reverse skyline query retrieves hotels from hotels' perspective.

ID	Distance	Price
h1	40	5
h2	35	25
h3	20	10
h4	10	20
h5	15	40
h6	25	55
h7	40	45
h8	45	55

(a)



(b)

Fig. 1: List of Hotel (a) and Conventional Skyline (b)

Skyline and reverse skyline query are two important methods for selecting smaller number of objects in various applications, one of which is in location selection problem. Choosing good location in a map is very important for many location-based applications. Usually, one would consider a location as a good location if it is near to desirable facilities that would be useful and/or pleasant to her/him such as stations, schools, supermarkets, etc. and is far from undesirable facilities that would unpleasant to her/him, such as competitors, noise sources, pollution sources, high-crime areas, etc.

Area skyline query is a method for selecting good areas, which are near to desirable facilities and far from undesirable facilities. In [4] and [5], the idea of skyline queries [1] is used to select area skyline in a map. We proposed Grid-based Area Skyline (GASky) algorithm in [5], which divides query area into grids as disjoint areas, and calculate minimum (min) and maximum (max) distance of each grid from closest desirable "+" facilities and closest undesirable "-" facilities. An area g dominates another area g' if g has smaller or equal max distance than min distance of g' for all facility types. Shaded grids in map in Fig. 2 shows an example of area skylines.

Area skyline query is important method to select non-dominated area from the users' perspective, who need some good locations based on his/her preference. Assume a real estate company has an area g (grid (2,18) in Fig. 2) to develop apartment, office, or market complex. The company needs to know who will be interested in the area. By using the idea of "reverse skyline", reverse skyline areas of g can be identified. Grey grids in Fig. 3 are dynamic area skyline of grid $(1, 14)$, while grey grids of Fig. 4 are reverse area skyline of g. Notice that g is a dynamic area skyline of grid $(1, 14)$, so that grid $(1, 14)$ is a reverse area skyline of g.

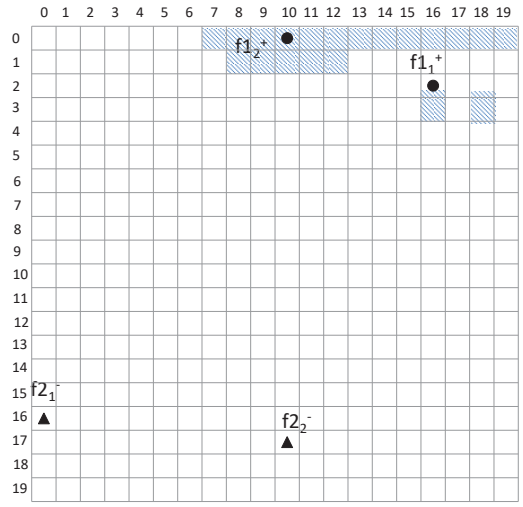


Fig. 2: Area Skyline Queries

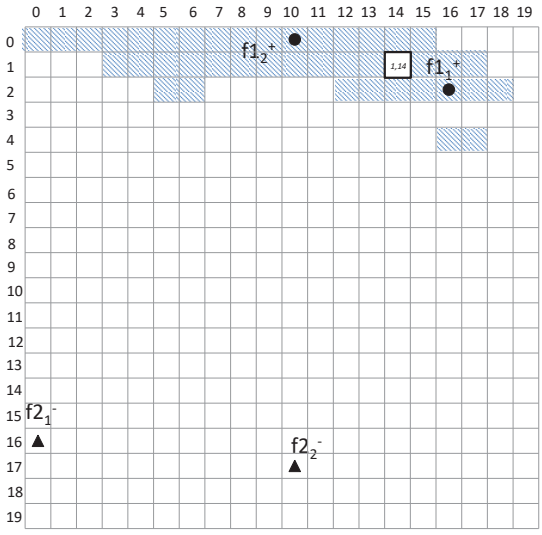


Fig. 3: Dynamic Area Skyline of grid (1,14)

On the analogy of the utilization of "reverse skyline", the "reverse" skyline areas has invaluable information. Let us consider a real estate company that have an area g (grid (2,18) in Fig. 4). Information about reverse area skyline, shaded area in Fig. 4, must be useful for such company to consider effective real estate developments so that the area attracts many buyers. Reverse area skyline query can also be used for selecting promising buyers of the area, since it may give the company clues to find who will be interested in the area. Moreover, it also may help to predict what type of business that would be suitable for the area considering the type of business that had already exist in the reverse area skylines.

In this paper, we present the reverse area skyline query and propose an effective and efficient method to answer reverse area skyline problem.

The contributions of this paper are summarized below:

1) We have introduced a new skyline query, i.e., reverse area skyline query in the literature.
2) We have introduced some important concepts, dynamic area skyline and global area skyline, and propose Reverse Area Skyline (RASky) algorithm to answer the reverse area skyline problem.
3) We have conducted intensive experiments to prove the efficiency of proposed algorithm.

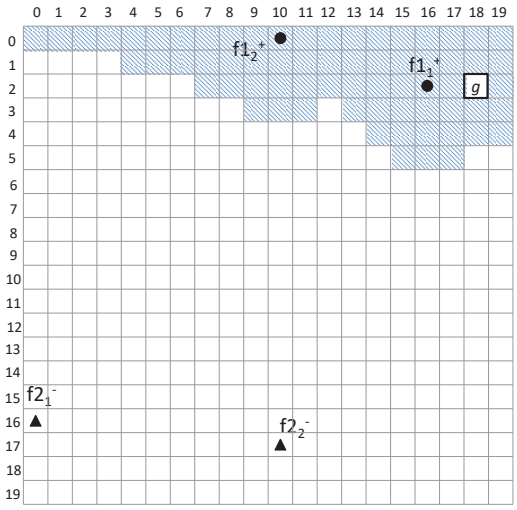


Fig. 4: Reverse Area Skyline Result

The rest of this paper is organized as follows. Section 2 reviews about skyline, dynamic skyline, reverse skyline, global skyline, spatial skyline, and area skyline issues. Section 3 formulates the problem definition and proposes the reverse area skyline algorithm. Section 4 presents the result of experiments, and finally Section 5 gives conclusions and future works.

II. Literature Review

A. Skyline, Dynamic Skyline, Reverse Skyline, and Global Skyline

Skyline query is a popular method for selecting small number of preferred answer from database. Since first introduced in [1], many algorithms have been proposed for answering skyline query problem, [1], [2], [6], [7]. Two important variants of skyline query are dynamic skyline query [2] and reverse skyline query [3]. Currently, the most efficient method in computing skyline and dynamic skyline is Branch and Bound Skyline (BBS), proposed in [2], which is a progressive algorithm using the R-tree, while The Branch and Bound Reverse Skyline (BBRS) algorithm [3] are the state-of-the-art algorithms for answering reverse skyline queries using the global skyline concept. In particular, BBRS is an improved customization of the original BBS algorithm [2]. GSRS is an improvement of BBRS to answer reverse skyline query [8].

In dynamic skyline query, a user specifies her/his preference as a query point in the data space and the query retrieves skyline objects that are not dominated by another with respect to the query object. In reverse skyline query, given a dataset P and a query point q in the space of P, an object p in P called as a "reverse skyline" of q if q is a dynamic skyline of p.

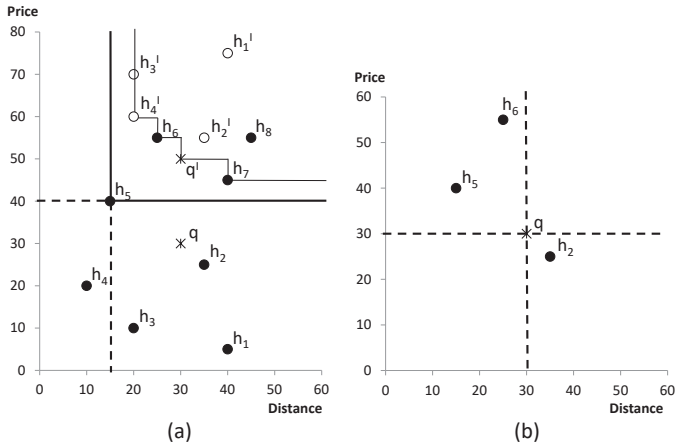


Fig. 5: Dynamic Skyline of h_5 (a) and Reverse Skyline (b)

Fig. 5 shows an example of a dynamic and reverse skyline queries. The example assumes that a user specifies $q = (30, 30)$ as a query point. To find reverse skyline of q, compute dynamic skyline of all objects, and find which objects that have q in its dynamic skyline. Fig. 5 (a) shows dynamic skyline query of h_5, $(x = distance, y = price) = (15, 40)$. To find dynamic skyline based on h_5, we first transform objects as follows. If x value of objects is less than 15, then transform the x value into $15 + (15 - x)$. Similarly, if y value of objects is less than 40, then transform y values into $40 + (40 - y)$. These transformations result in transformed data objects as in Fig. 5 (a). In the example, $h_2 = (35, 25)$ is transformed to $h'_2 = (35, 55)$, $h_4 = (10, 20)$ is transformed to $h'_4 = (20, 60)$, and so forth. We, then, compute skyline query for the transformed data objects as dynamic skyline query for $(15, 40)$, which retrieves $\{h_4, h_6, q, h_7\}$. Since q is in the dynamic skyline h_5, h_5 is a reverse skyline of q. Similarly, we calculate other reverse skyline objects of q as in Fig. 5 (b).

Skyline query in the Fig. 1 retrieves candidate hotels from users' perspective. On the other hand, reverse skyline query in Fig. 5 (b) retrieves hotels from hotels' perspective. Assume that a company is running a hotel whose detail is represented as a query point q. Intuitively, a user that is interested in h_5 hotel may also be interested in q since q is a dynamic skyline of h_5. The similar intuition holds on h_2 and h_6. Therefore, the company can expect users who are interested in h_2, h_5, and h_6 might also be interested in q.

Calculating dynamic skyline for each p in P to find reverse skyline of q needs very large computation. In order to reduce the search space, Dellis and Seeger introduced Branch and Bound Reverse Skyline (BBRS) algorithm using a concept called global skyline in [3].

Given a d-dimensional data set P and a query point q, p_1 said globally dominates p_2 with respect to q if: (1) $(p_1\text{-}q)(p_2\text{-}q)>0$ for all dimension, and (2) distance p_1 to q are smaller or equal than distance p_2 to q for all dimension, and smaller at least in one dimension. Rule (1) is to make sure that p_1 and p_2 are in the same quadrant w.r.t query point, while Rule (2) is dominance rule of global skyline. Evangelos and Seeger [3] have proved that reverse skyline point always a member of global skyline point.

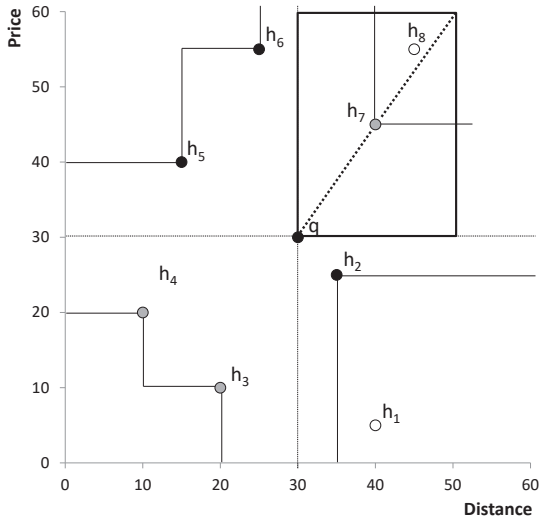


Fig. 6: Global Skyline and Query window

For selecting reverse skyline points from global skyline points, BBRS applies a window query for each global skyline point p. Window Query is an empty range query (boolean range query) which will return either true or false depending on whether there is any object inside the given range or not [9]. In [3], window query is the rectangle area with p as its center, and distance to a given query point q and its extension as its border coordinate. If there is another point inside this rectangle, then p is not a reverse skyline of q, otherwise p is a reverse skyline of q. Moreover, to reduce the number of window query checks, Gao et al. [8] introduced global-1-skyline concept in GSRS algorithm, which is a set of points that globally dominated by at most one global skyline point. They have proved that instead of checking all globally dominated points against all window queries, we simply just need to check whether any global skyline point or global 1-skyline point is inside the window query. Using the concept of global skyline point, BBRS and GSRS only consider points that potentially can be reverse skyline points and prune other points.

Fig. 6 shows an example of global skyline points of q ($\{h_2, h_3, h_4, h_5, h_6, h_7\}$), global 1-skyline points, h_1 and h_8 (white points), and the window query of h_7. Based on rule 1 in global skyline definition, each point is only compared with other points in the same quadrant. There are four quadrants in Fig. 6. h_5 and h_6 are in the same quadrant, h_7 and h_8, h_4 and h_3, and h_2 and h_1 are in the other quadrants. Since h_8 is inside the window query of h_7, then h_7 is not a reverse skyline of q. Applying window query to the rest of global skyline points, we can get the reverse skyline points of q are h_2, h_5, and h_6 (black points). h_7, h_3, and h_4 (grey points) are global skyline points, but not reverse skyline points since there is another point inside their window query.

Due to the importance of its use in various fields of application, research in the reverse skyline has gained many attention in the database research community such as in [3], [10]–[15]. All of the proposed reverse skyline variations above only consider about reverse skyline for zero dimensional data. Specifically, none of them considers about how to select

reverse skyline from two dimensional objects, such as areas in a map. Therefore, all the previous algorithms can not directly be used to answer reverse area skyline problem.

B. Spatial Skyline Query

Spatial skyline query (SSQ) was first introduced in [16]. Given a set of data points P and a set of query points Q, an SSQ retrieves those points of P which are not dominated by any other point in P considering their derived spatial attributes, which is the point's distance to a query point. The difference between spatial skyline query with the regular skyline query is that the domination condition of P depends on the distance to query points Q.

There are several researches works of spatial skyline problem, like in [17]–[23]. 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, which is a zero dimensional data.

C. Area Skyline Query

Area skyline query was introduced in [4] and [5]. Given A as a domain area on map and g as rectangular query area in A. Let $F = \{F1, ...Fm\}$ 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 type has some number of facility objects m_i, for example, a desirable facility $F1^+$ has two objects $F1^+ = \{f1_1^+, f1_2^+\}$. Area skyline query using GASky algorithm consist of two steps. In the step one, GASky would divide A into sxt grids, where s is a number of rows and t is a number of columns. Let us consider a map in Fig. 2. Suppose a company would like to build a new housing complex in a region. To attract customers, the housing complex should be in an area that is near to train stations (point) and far from pollution source (triangle). There are two train stations $(f1_1^+, f1_2^+)$ and two pollution sources $(f2_1^-, f2_2^-)$ in this region. Note that "+" symbol is annotated to train stations, which are desirable facilities, and "-" symbol is annotated to pollution sources, which are undesirable facilities. In this situation, the company has to find two dimensional area on the map. In this example, the region is divided into 20×20 grids, say grid (0,0), ..., grid (19,19), each of which can be identified by row and column number. Then, GASky finds the closest train station and the closest pollution source from each grid and calculates min and max distance to the closest facilities and record the computation result into Minmax table like in Fig. 7. Min distance is the closest distance from grid to the closest facility, while max distance is the farthest distance from grid to the closest facility.

Since each grid is surrounded by four vertexes, to simplify in calculating min and max distance from each grid to closest facility type, GASky calculates distance from vertexes first. Using Voronoi diagram of each facility type, GASky finds the closest facility object for each type to each vertex, and then calculates its distance. Using vertex distance, GASky can calculate min and max distance for each grid easily [5]. For example, the closest $F1$ facility to four vertexes of a grid $g_{0,0}$ is $f1_2^+$ (as shown in Fig. 2). In common case, using these vertexes' distance information, GASky simply add the lowest value as min distance, and the highest as max distance. However, there are two special cases in calculating the min distance: first if the facility is inside the grid, and the other is if the facility is outside of the grid but one of the facility's coordinate is located between the coordinate of two vertexes. In the first case, 0 value simply added to the min distance of the grid, but in the second case we need to recalculate min distance from the facility to the edge connecting those two vertexes.

Fig. 7 is a partial Minmax table that records min and max distance to each facility type for grid $(0, 9)$, $(1, 15)$, and $(0, 19)$ in Fig. 2. In the table, notice that each distance value of undesirable facility was multiplied by -1 and swapped between its min and max value, so that we can say that the smaller value is better.

Grid ID	Closest $F1^+$	Closest $F2^-$	$F1^+$ min	$F1^+$ max	$F2^-$ min	$F2^-$ max
0,9	$f1_2^+$	$f2_1^-$	0.5	1.6	-17.5	-16.5
1,15	$f1_1^+$	$f2_2^-$	0.7	2.1	-17.3	-16.1
0,19	$f1_2^+$	$f2_1^-$	20.8	22.1	-3.5	-2.5

Fig. 7: Minmax Table

After completing Minmax table, in the second step GASky finds area skyline using area skyline dominance rule. A grid g would dominate another grid g' if for all distance to all facility type, max distance of g is smaller or equal than min distance of g'. Area skyline is a set of grids that are not dominated by another grid. Grid $(0, 9)$ and $(1, 15)$ dominates $(0, 19)$. Therefore, $(0, 19)$ is not an area skyline. GASky returns non-dominated areas (records) in the Minmax table (shaded area in Fig. 2) as area skyline.

The computational cost analysis of GASky step 1 shows that GASky takes $O(stm)$ in addition to the Voronoi diagrams' construction time, where s,t, and m are the number of rows, the number of columns, and the number of facility types, respectively. Experiment result in [5] shows that processing time of GASky increases when the number of facility type, the number of objects, and the number of grids increase. The ratio of skyline, which is the ratio between the number of skyline grids compared to all grids, would be high if the number of grids is small and the number of facility type and facility objects are big. If a user prefers small number of area skyline, she/he should increase the number of grids, so that the ratio of skyline areas will decrease. GASky can answer area selection based on user's perspective, but not from the company/business owner perspective. Nevertheless, since GASky operates on two dimensional object, we can use step one method of GASky to calculate Minmax table in reverse area skyline problem.

III. Reverse Area Skyline Query

In this section, we propose a reverse skyline query for grids in a map, which we call "Reverse Area Skyline Query".

A. Problem Definition

Let A be a rectangular target area in which there are spatial objects. Each spatial object can be categorized into one of m facility types. Let Fk be a set of type k ($k = 1, ...m$) objects, which are $Fk = \{fk_1, fk_2, ..., fk_{n_k}\}$ where n_k is the number of objects of the type k facility.

1) Grids and Vertexes: We divide A into $s \times t$ square grids where s is the number of rows, and t is the number of column. We can identify each grid using row number and column number. For example, $g_{i,j}$ is a grid that lies in the i-th row and the j-th column. Each square grid is surrounded by four vertexes, each of which can also be identified by row number and column number. For example, top-left, top-right, bottom-left, and bottom-right vertex of $g_{i,j}$ can be identified as $v_{i,j}$, $v_{i,j+1}$, $v_{i+1,j}$, and $v_{i+1,j+1}$, respectively. In Fig. 8, we defined the query grid g, and divided an area into 12x12 grids. $g_{4,6}$ is surrounded by four vertexes $v_{4,6}$, $v_{4,7}$, $v_{5,6}$ and $v_{5,7}$, respectively. For each vertexes, we find the nearest object of each facility type using Voronoi diagram. After that, we calculate min and max distance from each grid using the same calculation in GASky step 1 as discussed in Section II-C, and record the distances in the Minmax table. From now, we call one record in Minmax table as one object.

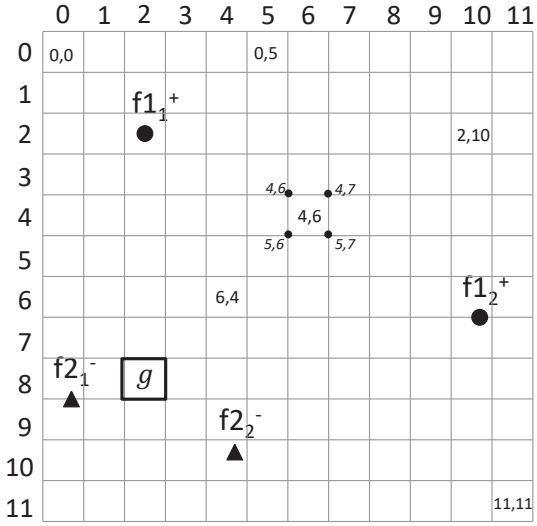


Fig. 8: Target area divided into 12 x 12 grids

2) Dynamic Area Skyline: Let $min(d_k(g))$ and $max(d_k(g))$ be the the min and max distance to facility f_k of grid g, and $min(d_k(q))$ and $max(d_k(q))$ be the the min and max distance to facility f_k of query grid q. In order to calculate dynamic area skyline, we need to transform the distances similar to conventional dynamic skyline. Let $min(d_k(g))^T$ and $max(d_k(g))^T$ are the transformed min and max distance, respectively. There are six cases to transform $min(d_k(g))$ and $max(d_k(g))$ w.r.t query grid q into $min(d_k(g))^T$ and $max(d_k(g))^T$. Fig. 9 illustrates the transformation of six cases in dynamic area skyline.

In all cases, we assume $min(d_k(q))$ and $max(d_k(q))$ are 5 and 8, respectively. In case 1, assume $min(d_k(g))$ and $max(d_k(g))$ are 9 and 11. Since 9 and 11 are larger than 8, then $min(d_k(q))^T$ and $max(d_k(q))^T$ become 1 (9-8) and 3 (11-8). In case 2, assume $min(d_k(g))$ and $max(d_k(g))$ are 6 and 11. Since 6 is between 5 and 8, and 11 is larger than 8, then $min(d_k(q))^T$ and $max(d_k(q))^T$ become 0 and 3 (11-8). In case 3, assume $min(d_k(g))$ and $max(d_k(g))$ are 6 and 7. Since 6 and 7 are between 5 and 8, then $min(d_k(q))^T$ and $max(d_k(q))^T$ become 0. In case 4, assume $min(d_k(g))$ and $max(d_k(g))$ are 1 and 11. Since 1 is smaller than 5 and 11 is larger and 8, then $min(d_k(q))^T$ and $max(d_k(q))^T$ become 0 and 3, just like in case 2. In case 5, assume $min(d_k(g))$ and $max(d_k(g))$ are 1 and 6. Since 1 is smaller than 5 and 6 is between 5 and 8, then $min(d_k(q))^T$ and $max(d_k(q))^T$ become 0 and 4 (5-1). In case 6, assume $min(d_k(g))$ and $max(d_k(g))$ are 1 and 4. Since 1 and 4 are smaller than 5, then $min(d_k(q))^T$ and $max(d_k(q))^T$ become 1 (5-4) and 4 (5-1).

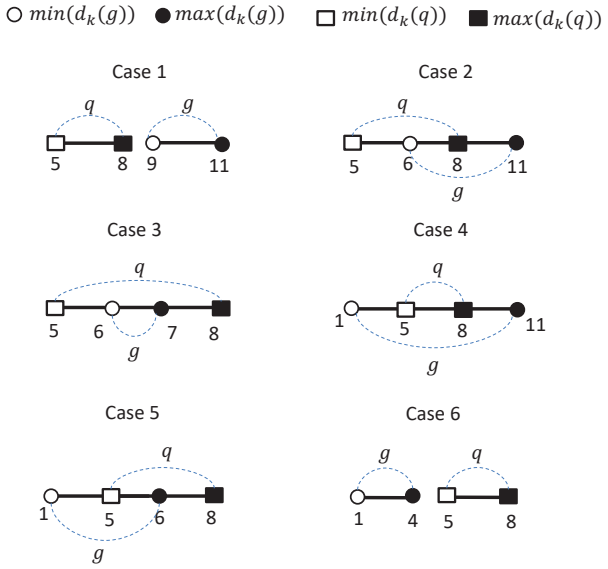


Fig. 9: Transformation cases

We then formally defined Case 1 to 6 as:

if $min(d_k(g)) \geq max(d_k(q))$, then

$$\begin{aligned} min(d_k(g))^T &= min(d_k(g)) - max(d_k(q)) \\ max(d_k(g))^T &= max(d_k(g)) - max(d_k(q)) \end{aligned} \quad (1)$$

if $max(d_k(g)) > max(d_k(q))$ and $max(d_k(q)) > min(d_k(g)) \geq min(d_k(q))$, then

$$\begin{aligned} min(d_k(g))^T &= 0 \\ max(d_k(g))^T &= max(d_k(g)) - max(d_k(q)) \end{aligned} \quad (2)$$

if $min(d_k(g)) \geq min(d_k(q))$ and $max(d_k(g)) \leq max(d_k(q))$, then

$$\begin{aligned} min(d_k(g))^T &= 0 \\ max(d_k(g))^T &= 0 \end{aligned} \quad (3)$$

if $min(d_k(g)) < min(d_k(q))$ and $max(d_k(g)) > max(d_k(q))$, then

$$\begin{aligned} min(d_k(g))^T &= 0 \\ max(d_k(g))^T &= max(d_k(g)) - max(d_k(q)) \end{aligned} \quad (4)$$

if $min(d_k(g)) < min(d_k(q))$ and $min(d_k(q)) < max(d_k(g)) \leq max(d_k(q))$, then

$$\begin{aligned} min(d_k(g))^T &= 0 \\ max(d_k(g))^T &= min(d_k(q)) - min(d_k(g)) \end{aligned} \tag{5}$$

if $max(d_k(g)) \leq min(d_k(q))$, then

$$\begin{aligned} min(d_k(g))^T &= min(d_k(q)) - max(d_k(g)) \\ max(d_k(g))^T &= min(d_k(q)) - min(d_k(g)) \end{aligned} \tag{6}$$

Definition 1. Dynamic Area Skyline Query

For two objects, g and g', we said g dynamically dominates g' w.r.t q, if and only if $max(d_k(g))^T \leq min(d_k(g'))^T$ for all k ($1 \leq k \leq m$). Dynamic area skyline query of q retrieves the set of all area objects that are not dynamically dominated by any other objects w.r.t q.

Based on dynamic area skyline definition, we can formally define the reverse area skyline of query area q.

Definition 2. Reverse Area Skyline Query

Let G be a set of d-dimensional objects. Reverse area skyline query w.r.t query area q retrieves all area objects $g \in G$ where q is in the dynamic area skyline of g. In other words, we said g is reverse area skyline of q if $\nexists g' \in G$ such that $max(d_k(g'))^T \leq min(d_k(q))^T$ for all k ($1 \leq k \leq m$) w.r.t g.

Using Definition 2, we can compute reverse area skyline query by performing dynamic area skyline query for each grid objects, and retrieve set of grid objects which have query area q in their dynamic area skyline result. But as discussed in Section II-A, to compute reverse skyline by computing dynamic skyline for each object is time-consuming. In this paper, we define global area skyline concept to compute reverse area skyline. We extend global skyline concept in [3] so that it can be applied in area skyline.

3) Disjoint, Overlap, Within/Contain: Using information of min and max distances, one object's min and max distance might disjoint, overlap, within/contain with another object's min and max distance. Let us consider the example of Fig. 9 again. We define disjoint objects using case 1 and 6, overlap objects using case 2 and 5, and case 3 and 4 for within/contain objects.

Definition 3. Disjoint Objects

Object g is disjoint with g', if $max(d_k(g)) \leq min(d_k(g'))$ or $min(d_k(g)) \geq max(d_k(g'))$, for all $k \in m$.

Definition 4. Overlap Objects

Object g overlap with g', if $max(d_k(g)) > max(d_k(g'))$ and $max(d_k(g')) > min(d_k(g)) \geq min(d_k(g'))$, or if $min(d_k(g)) < min(d_k(g'))$ and $min(d_k(g')) < max(d_k(g)) \leq max(d_k(g'))$, for at least one $k \in m$.

Definition 5. Within/Contain Objects

Object g is within g', if $min(d_k(g)) \geq min(d_k(g'))$ and $max(d_k(g)) \leq max(d_k(g'))$, for all $k \in m$. Object g contains g' if $min(d_k(g)) < min(d_k(g'))$ and $max(d_k(g)) > max(d_k(g'))$, for all $k \in m$.

These disjoint, overlap, and within/contain conditions are two dimensional objects' characteristics that are not exist in zero dimensional objects. Based on these characteristics, we define Lemma 1 and 2 which are very important to efficiently compute reverse area skyline using global area skyline concept.

Lemma 1. *Let q be the query area. If g overlaps with or within/contain q, then g must be a reverse area skyline of q.*

Proof. Assume g is not a reverse area skyline of q. Then, there should be at least one object that dynamically dominates q w.r.t g. If we apply dynamic area skyline of g, since g overlaps or within/contains q, based on case 2, 3, 4, 5 in Section III-A2, $min(d_k(g))^T$ is always be 0, which makes it not possible to be dominated by other objects. It means that q is a dynamic area skyline of g, and consequently, g is reverse area skyline of q. So the assumption is not true and the proof is complete.◇

Fig. 10 shows an illustration of Lemma 1. Fig. 10 (a) shows original min-max distance of q and g, while Fig. 10 (b) shows min-max distance of q^T after we apply dynamic area skyline of g.

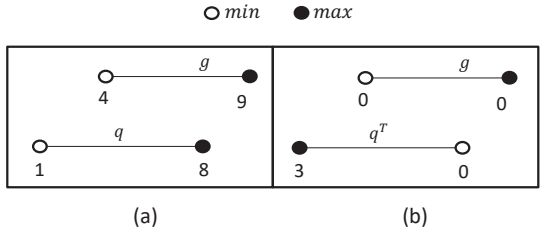


Fig. 10: Lemma 1 situation

Lemma 1 provides an easy selection method for RASky algorithms to directly put overlap/within/contain objects into reverse area skyline result.

Before defining global area skyline, let us consider the definition of global skyline for zero dimensional data in [3]. Given a d-dimensional data set P and a query point q, p_1 is said globally dominates p_2 with respect to q if: (1) (p_1-q)(p_2-q)>0 for all dimension, and (2) distance p_1 to q are smaller or equal then distance p_2 to q for all dimension, and smaller at least in one dimension. Let us consider point h_2 and h_1 in Fig. 6. We said h_2 is globally dominates h_1, because (h_1-q)(h_2-q)>0 and distance h_2 to q are smaller than distance h_1 to q for all dimension. In two dimensional case, the above situations become more complicated for overlap/within/contain conditions. Lemma 2 shows that overlap/within/contain objects can not globally dominate any other objects.

Lemma 2. *Overlap/within/contain objects can not globally dominate any other objects*

Proof. Let assume that g and g' is in the same quadrant w.r.t q. g is an overlap/within/contain object w.r.t q, and g' has $min(d_k(g'))^T \geq max(d_k(g))^T$, for all $k \in m$, so that g globally dominated g', and g' is not a reverse area skyline of q. On the contrary, if we apply dynamic area skyline of g', we can see that g can not dominate q, since g and q are overlap/within/contain objects. This mean that g' is reverse skyline of q and should not be discarded by g.◇

Fig. 11 shows an illustration of Lemma 2. Fig. 11 (a) shows the original min-max distance of q, g, and g'. Fig. 11 (b) shows if g (overlap/within/contain object) globally dominate g' w.r.t q, since $max(d_k(g))^T$ is smaller than $min(d_k(g'))^T$, then g' is not a reverse skyline of q. But this will lead to wrong result. If we apply dynamic area skyline on g' as in Fig. 11 (c), q is in dynamic area skyline of g', which consequently makes g' as reverse area skyline of q. In this situation g should not be allowed to globally dominate g' at the first place, since g' is reverse area skyline of q. Using Lemma 1 and 2, we can reduce the comparison step in calculating global area skyline because all the overlap/within/contain objects do not participate in the comparison process.

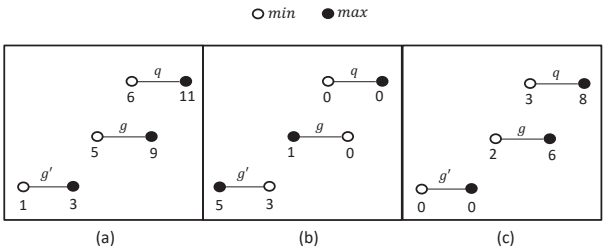


Fig. 11: Lemma 2 situation

4) Global and Global-1 Area Skyline: Based on Lemma 1 and 2, only disjoint objects will participate in global area skyline computation. Let us consider disjoint situations in Fig. 9 case 1 and 6. In Fig. 9 case 1, $min(d_k(g))$ and $max(d_k(g))$ are larger than $max(d_k(q))$, while in Fig. 9 case 6 $min(d_k(g))$ and $max(d_k(g))$ are smaller than $min(d_k(q))$. To differentiate between these two disjoint conditions, we defined $diff\,(g_k)$ as:

$$diff\,(g_k) = \begin{cases} min(d_k(g)) - max(d_k(q)) \\ \qquad\qquad \text{if } max(d_k(q)) \leq min(d_k(g)). \\ max(d_k(g)) - min(d_k(q)) \\ \qquad\qquad \text{if } max(d_k(g)) \leq min(d_k(q)). \end{cases}$$

Notice that the value of $diff\,(g_k)$ could be "positive" (case 1) or "negative" (case 6). Two objects g and g' are in the same quadrant w.r.t q if $(diff\,(g_k))(diff\,(g'_k))$ >0 for all $k \in m$. In Fig. 12, since $max(d_k(g)) \leq min(d_k(q))$,$(15 \leq 20)$, then $diff\,(g_k) < 0$ while $diff\,(g'_k) > 0$, since $min(d_k(g')) \geq max(d_k(q))$, $(30 \geq 25)$. Using Lemma 1, Lemma 2, and $diff$ definition, we define global and global-1 area skyline.

Definition 6. Global and Global-1 Area Skyline For two objects, g and g', we said g globally dominates g' w.r.t q, if and only if: (1)g and g' are disjoint objects w.r.t q, (2)$(diff\,(g_k))(diff\,(g'_k)) > 0$ and (3) $max(d_k(g))^T \leq min(d_k(g'))^T$, for all $k \in m$. Any objects g'' becomes global-1 area skyline if there is only one other object that globally dominates it.

5) Window Query: Window Query of grid $w(g)$ w.r.t q, has minimum and maximum value for each k dimension, $min(w_k(g))$ and $max(w_k(g))$ where $k \in m$. It is defined as follows:

$$min(w_k(g)) = \begin{cases} max(d_k(q)) \qquad \text{if } min(d_k(g)) \geq max(d_k(q)). \\ min(d_k(g)) + diff\,(g_k) \\ \qquad\qquad \text{if } max(d_k(g)) \leq min(d_k(q)). \end{cases}$$

$$max(w_k(g)) = \begin{cases} max(d_k(g)) + diff\,(g_k) \\ \qquad\qquad \text{if } min(d_k(g)) \geq max(d_k(q)). \\ min(d_k(q)) \qquad \text{if } max(d_k(g)) \leq min(d_k(q)). \end{cases}$$

Fig. 12 shows an illustration of window query's minimum and maximum value in one dimension. Assume min and max distance for g, q, and g' are (10,15), (20,25), and (30,35). For $w(g)$, since $max(d_k(g)) \leq min(d_k(q))$, then $diff\,(g_k)$ is -5 (15-20), so that $min(w_k(g))$ and $max(w_k(g))$ are 5 (10 + (-5)) and 20 (same value as $min(d_k(q))$). For $w(g')$, since $min(d_k(g)) \geq max(d_k(q))$, then $diff\,(g'_k)$ is 5 (30-25), so that $min(w_k(g'))$ and $max(w_k(g'))$ are 25 (same value as $max(d_k(q))$) and 40 (35+5).

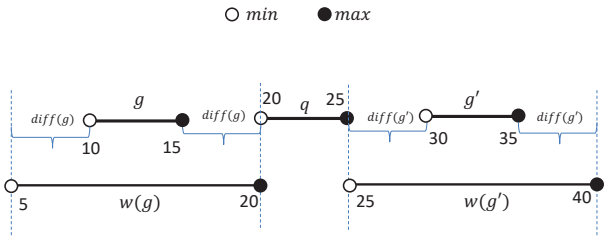


Fig. 12: Diff and Window query

Lemma 3. *Let g be a global area skyline of q, and g' be a global or global 1-area skyline of q with the same quadrant with g. If the window query of g contains g' w.r.t q, then g is not a reverse area skyline of q.*

Proof. If the window query of g contains g', then if we apply dynamic area skyline of g using formula in Section III-A2, we know that $max(d_k(g'))^T$ is always smaller than $min(d_k(q))^T$. It means that g' will dynamically dominate q w.r.t g, therefore g can not be a reverse area skyline of q.$\diamond$

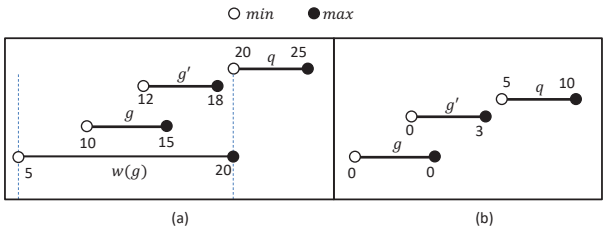
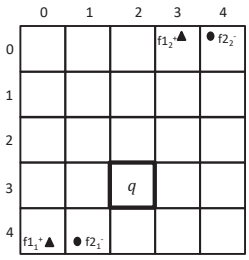


Fig. 13: Lemma 3 situation

Fig. 13 illustrates Lemma 3 situation. Fig. 13 (a) shows that $w(g)$ is contain g', while Fig. 13 (b) shows that g' will dynamically dominate q w.r.t g, so that g is not a reverse area skyline of q. Using Lemma 3, for each global area skyline we simply just check whether at least one of other global or global 1-area skyline is within its window query or not.

B. Reverse Area Skyline (RASky) Algorithm

Reverse area skyline algorithm (RASky) consist of two steps. At step 1, we divide A into grids. For each grid, we find the nearest facility type, calculate its min and max distance, and complete the distance information in Minmax table using the same method in GASky step 1 [5] as explained in Section II-C. In step 2, using information in Minmax table from the first step, we calculate reverse area skyline using global area skyline. In this section, we will focus on the reverse area skyline step 2.



Grid ID	$F1^+_{min}$	$F1^+_{max}$	$F2^-_{min}$	$F2^-_{max}$
0,0	27	38	33	43
0,1	17	28	23	34
0,2	8	18	8	19
0,3	0	10	3	15
0,4	2	14	0	10
1,0	29	39	28	40
1,1	19	32	24	37
1,2	10	25	15	28
1,3	7	19	8	22
1,4	8	21	8	19
2,0	18	29	18	31
2,1	19	30	18	29
2,2	19	32	19	32
2,3	17	28	18	30
2,4	18	30	18	28
3,0	8	19	8	22
3,1	9	23	8	19
3,2	15	29	10	25
3,3	25	37	19	32
3,4	27	39	28	38
4,0	0	10	3	15
4,1	2	14	0	10
4,2	13	24	7	18
4,3	23	34	17	28
4,4	33	44	28	38

(b)

Fig. 14: Sample map (a) and Minmax Table (b)

In this section we use sample map in Fig. 14 (a) and set $g_{3,2}$ as query area q, then divide sample map into 5x5 grids. In this map we have two types of facilities, $F1^+$ and $F2^-$, each of them have two objects $F1^+ = (f1_1^+, f1_2^+)$, $F2^- = (f2_1^-, f2_2^-)$. After completing RASky step 1 in the sample map, we obtain Minmax table like in Fig. 14 (b).

We index the grid by their min and max distance in Minmax table using R-tree structure. Each leaf in the R-tree is in the format $(id, qd, RECT)$, where id is the number of grid in Minmax table, qd is quadrant, and $RECT$ is a bundle of all min and max distance in a grid for all dimension. For example, for 2 facility type, or 2-dimensional, $RECT$ has $(f1min, f2min)$ as bottom-left coordinate and $(f1max, f2max)$ as top-right coordinate. Our query object $RECT_{3,2}$ has bottom-left coordinate (15,10) and top-right coordinate (29,25). Using $RECT$ object, we build R-tree of Minmax table.

1) Building R-tree: RASky read each object in Minmax table. Using Lemma 1 and 2, if the object is an overlap/within/contain object, then it will automatically be a reverse area skyline object, and will be excluded from R-tree and further computation. Only disjoint objects will be inserted into R-tree. Let us consider Minmax table in Fig. 14 (b). Since $g_{0,4}$, $g_{1,0}$, $g_{4,1}$, and $g_{4,4}$ are disjoint objects (rows with bold border in Fig. 14 (b)), they are inserted into R-tree, while others directly become reverse area skyline of q. Fig. 15 shows R-tree after inserting disjoint objects.

2) Finding Global and Global-1 area skyline: RASky insert all root entries into heap H and sort them by their distance from q. Besides H, we also use two additional heap H_g and H_{g1} to maintain global area skyline and global-1 area skyline. Since $N1$ is closest to q, its entry is expanded, and $N1$ is removed from H. Now H contents become $RECT_{0,4}$,

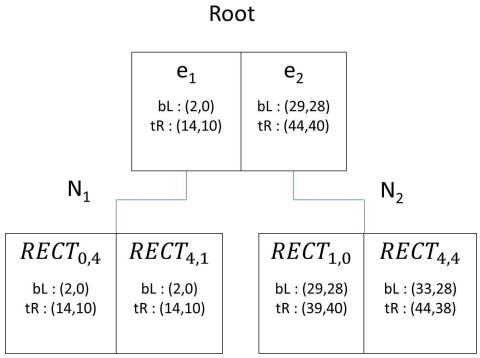


Fig. 15: R-tree of disjoint objects

$RECT_{4,1}$, and $N2$. As top of H, $RECT_{0,4}$ then become the first global area skyline and inserted into H_g. Notice that $RECT_{4,1}$ is in the same quadrant with $RECT_{0,4}$ and it is not globally dominated by $RECT_{0,4}$, so it also inserted into H_g. Next $N2$ is expanded and $RECT_{1,0}$ and $RECT_{4,4}$ is inserted into H. Since $RECT_{1,0}$ is in different quadrant with $RECT_{0,4}$ and $RECT_{4,1}$, $RECT_{1,0}$ also become global area skyline and inserted into H_g. $RECT_{4,4}$ is in the same quadrant with $RECT_{1,0}$, but since it is not globally dominated by $RECT_{1,0}$, then it also inserted into H_g. Since there is no global-1 area skyline in this sample dataset, then H_{g1} is remain empty.

3) Applying Window Query: After getting all global area skyline, we build window query for each entry in H_g. Using window query formula in Section III-A5, Fig. 16 shows bottom-left and top-right coordinate of each window query for query area $RECT_{3,2}$ whose bottom-left and top-right is (15,10) and (29,25), respectively.

Window Query	$diff(g_1)$	$diff(g_2)$	$\min(w_1(g)), max(w_1(g))$	$\min(w_2(g)), max(w_2(g))$	bL	tR
0,4	-1	0	(1,15)	(0,10)	(1,0)	(15,10)
4,1	-1	0	(1,15)	(0,10)	(1,0)	(15,10)
1,0	0	3	(29,39)	(25,43)	(29,25)	(39,43)
4,4	4	3	(29,48)	(25,41)	(29,25)	(48,41)

Fig. 16: Window query in sample dataset

Let us consider $w(g_{0,4})$, $diff(g_{0,4_1})$ is -1 (14-15) and $diff(g_{0,4_2})$ is 0 (10-10). Using these values, we can compute min and max of $w(g_{0,4})$ in dimension 1 as (2+(-1),15) and in dimension 2 as (0+0,10), so that bL and tR coordinates are (1,0) and (15,10). Since $RECT_{4,1}$ has the same bL and tR coordinate with $RECT_{0,4}$, then min and max of $w(g_{4,1})$ are the same with $w(g_{0,4})$. This mean that $RECT_{4,1}$ always contains $w(g_{0,4})$ and vice versa, so based on Lemma 3, both of them are not reverse area skyline of $g_{3,2}$. Now for $w(g_{1,0})$, $diff(g_{1,0_1})$ is 0 (29-29) and $diff(g_{1,0_2})$ is 3 (28-25). Min and max of $w(g_{1,0})$ in dimension 1 as (29,39+0) and in dimension 2 as (25,40+3), so that bL and tR coordinates are (29,25) and (39,43). Finally, for $w(g_{4,4})$, $diff(g_{4,4_1})$ is 4 (33-29) and $diff(g_{4,4_2})$ is 3 (28-25). Min and max of $w(g_{4,4})$ in dimension 1 as (29,44+4) and in dimension 2 as (25,38+3),

so that bottom-left and top-right coordinates are (29,25) and (48,41). $RECT_{4,4}$ overlaps with $w(g_{1,0})$, while window query of $w(g_{4,4})$ contains $RECT_{1,0}$. Based on Lemma 3, $RECT_{1,0}$ is reverse area skyline of q while $RECT_{4,4}$ is not. From the above computation, we can find that $g_{0,4}$, $g_{4,1}$, and $g_{4,4}$ is not reverse area skyline of $g_{3,2}$ while the others are.

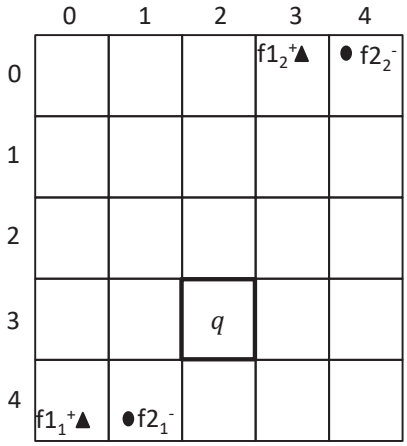


Fig. 17: Reverse area skyline for sample map

Shaded area in Fig. 17 shows reverse area skyline of $g_{3,2}$ in sample map Fig. 14 (a), which is 88% of all grids. Next in the experiment section, we discover that smaller size of query area q will reduce reverse area skyline result.

IV. Experimental Evaluation

We experimentally evaluated RASky algorithm in a PC with Intel Core i5 3.2GHz processor and 4GB of RAM. We conducted three experiments using three synthetic datasets. In each experiment, we repeated five times and reported the average. We examined the effect of parameters such as number of objects, number of types, and number of grids, to the step 1 and step 2 of RASky algorithm. We recorded the processing time for step 1 and step 2 of RASky and the ratio of reverse area skyline resulted from each experiment. Ratio of skyline is the number of reverse area skyline compared with the number of grids in the experiment, Table I lists the synthetic datasets and parameters in these experiments.

TABLE I: Experimental Dataset

Dataset	Objects	Types	Grids
DB1	1k,2k,4k,8k,16k	2	160k
DB2	1k	2,4,8,16	40k
DB3	1k	2	10k,40k,160k,640k

A. Effect of Number of Objects

In these experiments, we examined the performance of RASky on the different number of objects, when the number of facility types and the number of grids are fix, using DB1. Fig. 18 shows the processing time of this experiment. We can see that the increase of the number of objects will increase the total processing time of RASky. In step 1, increasing number of objects will increase the processing time to build Voronoi diagram. However, since the number of Voronoi diagram is fix according to the number of type, increasing number of objects does not have effect in the size of Minmax table. Hence in RASky step 2, increasing the number of objects has less effect, and the processing time tend to decrease when the number of objects increases. The reason is because increasing number of objects, while the number of grids is fix, increases the number of non-disjoint objects. It means less objects will participate in global area skyline computation, since only disjoint objects participates in the computation. Therefore the processing time will be decreased. The ratio of reverse area skyline are increasing as reported in Fig. 19. Increasing the number of objects will cause smaller value on min and max distances, but since the number of grids is fix when the number of objects increase, the ratio of reverse area skyline still will increase.

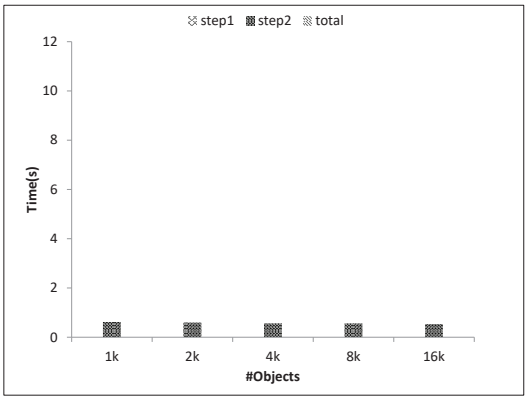


Fig. 18: Processing time of $DB1$

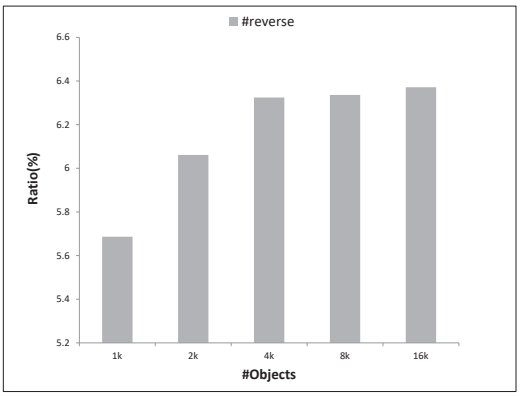


Fig. 19: Reverse area skyline's ratio of $DB1$

B. Effect of Number of Types

In these experiments, we used a synthetic data DB2 that have fix number of objects and number of grids. From the results in Fig. 20, we can observe that the processing time increases with the increase of the number of types. The increasing number of types will require more Voronoi diagrams, which in turn increase the processing time. The result illustrates that increasing the number of types significantly increase the processing time of step 1. Similar with increasing

number of objects, increasing number of types with fix number of grids will decrease the number of disjoint objects. Although the number of disjoint objects decreases, the processing time still increases because increasing facility types also means larger size of Minmax tables. Since the dimension is increasing as the number of facility types increase, the ratio of skyline is also increasing as shown in Fig. 21.

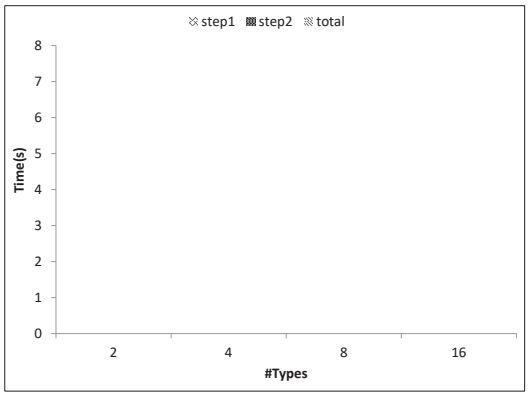


Fig. 20: Processing time of $DB2$

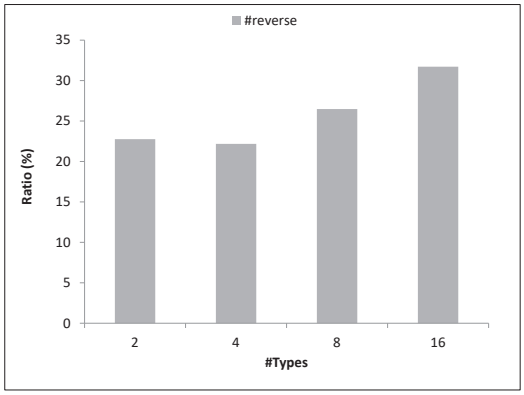


Fig. 21: Reverse area skyline's ratio of $DB2$

C. Effect of Number of Grids

In these experiments, we evaluated the effect of number of grids while the number of objects and number of types are fix, using DB3. Fig. 22 shows that the number of grids affects the processing time of step 1 and step 2. In step 1, increasing the number of grids means more comparison on Voronoi diagrams and more calculation time to obtain min and max distance, and in the same time enlarges the number of record in Minmax table which also cause increasing time needed for step 2 computation. Increasing number of grids while number of objects and number of types are fix also causing the number of disjoint objects to increase. In step 2, increasing number of disjoint objects will increase processing time since more objects will participate in global area skyline computation. In Fig. 23, the effect of number of grids affect ratio of skyline differently compared to the effect of the number of objects and types. Increasing the number of grids will decrease the ratio of skyline. The important reason of that is because more grids has the same meaning of having smaller size of each grid, which significantly decrease the ratio of skyline, since smaller area is likely to be dominated by another area.

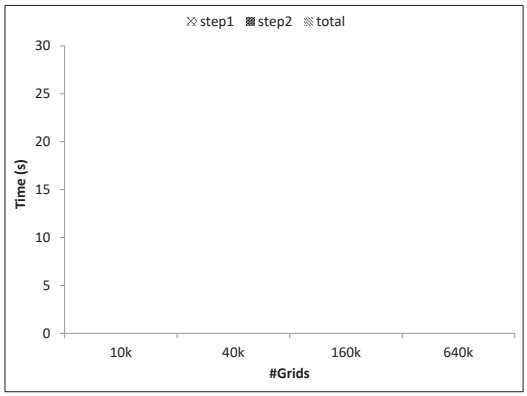


Fig. 22: Processing time of $DB3$

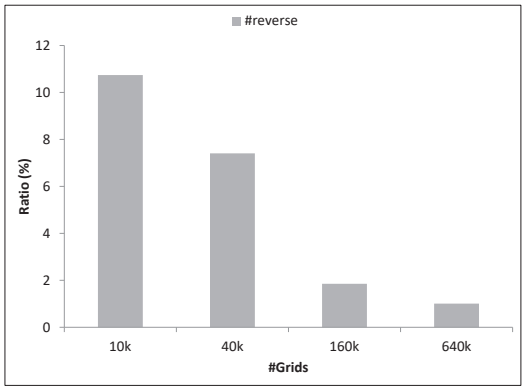


Fig. 23: Reverse area skyline's ratio of $DB3$

From all of experimental results, we can indicate that the total processing time of RASky increases when the number of objects, number of facility types, and the number of grids increases. In addition, the ratio of skyline increases when the number of objects and types increases, and decreases when the number of grids increases.

V. Conclusions and Future Works

In this paper, we define dynamic area skyline, global area skyline, and propose reverse area skyline algorithm (RASky) to solve the reverse area skyline query. This query is very important for location selection in business' or landowners' perspective. RASky has two steps, step 1 to compute Minmax table and step 2 to calculate reverse area skyline. Smaller query area will obtain smaller number of reverse area skyline and vice versa. Reverse area skyline gives invaluable information for landowner to pursue targeted customer or to decide what type of business that would attract more customer. Comprehensive experiments are conducted to show the effectiveness and efficiency of the proposed algorithms. In the future, we will consider another skyline problem in two dimensional objects, such as selecting k-dominant areas. We are also interested in

the application of this method to road network, which also taken into account nonspatial properties such as population density, price, traffic condition, and so on.

ACKNOWLEDGMENT

The authors would like to thank KAKENHI (16K00155) Japan for supporting this research, Indonesian Government DG-RSTHE scholarship for supporting Annisa, and Japanese Government MEXT Scholarship for supporting Asif Zaman.

REFERENCES

[1] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings of the 17th International Conference on Data Engineering (ICDE), April 2–6, Heidelberg, Germany*, 2001, pp. 421–430.

[2] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proceedings of the ACM SIGMOD June 9–12, California, USA*, 2003, pp. 467–478.

[3] D. Evangelos and B. Seeger, "Efficient computation of reverse skyline queries," in *Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment*, 2007, pp. 291–302.

[4] Annisa, M. A. Siddique, A. Zaman, and Y. Morimoto, "A method for selecting desirable unfixed shape areas from integrated geographic information system," in *Proceedings of IIAI*, 2015, pp. 195–200.

[5] Annisa, A. Zaman, and Y. Morimoto, "Area skyline query for selecting good locations in a map," *Information Processing Society of Japan : Database Transaction*, vol. 9, no. 3, pp. 0–0, 2016.

[6] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting," in *Proceedings of the 19th International Conference on Data Engineering (ICDE), March 5–8, Bangalore, India*, 2003, pp. 717–719.

[7] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB), September 11–14, Rome, Italy*, 2001, pp. 301–310.

[8] Y. Gao, Q. Liu, B. Zheng, and G. Chen, "On efficient reverse skyline query processing," *Expert Systems with Applications*, vol. 40, no. 7, pp. 3237–3249, 2014.

[9] H. F. Singh, Amit and A. aman Tosun, "High dimensional reverse nearest neighbor queries," in *Proceedings of the twelfth International Conference on Information and Knowledge Management*, 2003, pp. 91–98.

[10] W. Xiaobing, Y. Tao, R. C.-W. Wong, L. Ding, and J. X. Yu., "Finding the influence set through skylines," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, ACM*, 2009, pp. 1030–1041.

[11] L. Zhu, C. Li, and H. Chen, "Efficient computation of reverse skyline on data stream," in *Computational Sciences and Optimization, International Joint Conference on, vol. 1, IEEE*, 2009, pp. 735–739.

[12] G. Wang, J. Xin, L. Chen, and Y. Liu, "Energy-efficient reverse skyline query processing over wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 7, pp. 1259–1275, 2012.

[13] Q. Liu, Y. Gao, G. Chen, Q. Li, and T. Jiang, "On efficient reverse k-skyband query processing," in *International Conference on Database Systems for Advanced Applications*, 2012, pp. 544–559.

[14] Y. Park, J.-K. Min, and K. Shim, "Parallel computation of skyline and reverse skyline queries using mapreduce," in *Proceedings of the VLDB Endowment 6, no. 14*, 2013, pp. 2002–2013.

[15] M. S. Islam, R. Zhou, and C. Liu, "On answering why-not questions in reverse skyline queries," in *IEEE 29th International Conference on Data Engineering*, 2013, pp. 973–984.

[16] M. Sharifzadeh and C. Shahabi, "The spatial skyline queries," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), September 12–15, Seoul, Korea*, 2006, pp. 751–762.

[17] K. Kodama, Y. Iijima, X. Guo, and Y. Ishikawa, "Skyline queries based on user locations and preferences for making location-based recommendations," in *Proceedings of the International Workshop on Location Based Social Networks (LBSN) November 03, Washington, USA*, 2009, pp. 9–16.

[18] M. Arefin, J. Xu, Z. Chen, and Y. Morimoto, "Skyline query for selecting spatial objects by utilizing surrounding objects," *Journal of Software*, vol. 8, no. 7, pp. 1742–1749, 2013.

[19] X. Guo, Y. Ishikawa, and Y. Gao, "Direction-based spatial skylines," in *Proceedings of the 9th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE), June 6, Indiana, USA*, 2010, pp. 73–80.

[20] Q. Lin, Y. Zhang, W. Zhang, and X. Lin, "Efficient general spatial skyline computation," *World Wide Web*, vol. 16, no. 3, pp. 247–270, 2013.

[21] G.-W. You, M.-W. Lee, H. Im, and S.-W. Hwang, "The farthest spatial skyline queries," *Information Systems*, vol. 38, no. 3, pp. 286–301, 2013.

[22] Y.-W. Lin, E.-T. Wang, C.-F. Chiang, and A. L. P. Chen, "Finding targets with the nearest favor neighbor and farthest disfavor neighbor by a skyline query," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC), March 24–28, Gyeongju, Korea*, 2014, pp. 821–826.

[23] M. Arefin, G. Ma, and Y. Morimoto, "A spatial skyline query for a group of users," *Journal of Software*, vol. 9, no. 11, pp. 2938–2947, 2014.