Computing the Dominating-Free Set by Two Point Sets in the Plane

Soo-Hwan Kim, *Member, KIMICS*

***Abstract*— In this paper, we study the dominating-free set which is defined as follows: *k* points called servers and *n* points called clients in the plane are given. For a point *p* in the plane is said to be dominated by a client *c* if for every server *s*, the distance between *s* and *p* is greater than the distance between *s* and *c*. The dominating-free set is the set of points in the plane which aren't dominated by any client. We present an $O(nk\log k+n^2k)$ time algorithm for computing the dominating-free set under the L_1-metric. Specially, we present an $O(n\log n)$ time algorithm for the problem when k=2. The algorithm uses some variables and 1-dimensional arrays as its principle data structures, so it is easy to implement and runs fast.**

***Index Terms*—Dominating-free set, Geometric algorithm, L_1-metric, Orthogonal convex polygon.**

I. INTRODUCTION

There are diverse domination properties considered in research area such as graph theory, game theory, and databases. Specially, the skyline query problem can be found in a wide spectrum of optimization applications [1, 2, 3, 4, 5, 6]. Given a set of points *D*, a skyline query finds the skyline points from *D*, such that every point on the skyline is not 'dominated' by any other point in *D*, i.e., if a point *p* is on the skyline, there exists no data point *q* in *D* such that *q* is pair-wise smaller than *p* for the values in all dimensions. In spatial skyline query problem, the distance between two points is defined using the conventional Euclidean distance(L_2-metric)[3, 4] or Manhattan distance(L_1-metric)[6].

In this paper, we study a variation of the spatial skyline query problem which requires a geometric algorithm is defined as follows: *k* points called *servers* and *n* points called *clients* in the plane are given. For a point *p* in the plane is said to be *dominated by* a client *c* if for every server *s*, the distance between *s* and *p* is greater than the distance between *s* and *c*. The dominating-free set is the set of points in the plane which aren't dominated by any client. Consider the following example to easily understand a dominating-free set in the plane.

Manuscript received January 10, 2011; revised February 3, 2011; accepted February 7, 2011.

Soo-Hwan Kim is with the Department of Embedded IT, Pusan University of Foreign Studies, Busan, 608-738, Korea (Email: shkim@pufs.ac.kr)

There is a big city where the road network consists of horizontal roads and vertical roads. The distance between two locations of the city is computed using L_1-metric. There are *k* apartments and *n* restaurants. If someone wants to open a new restaurant, he would regard a location *p* as a bad location if for an existing restaurant *r*, there is no apartment which is closer to *p* than *r*. Hence, he will consider not bad locations as candidates of a new restaurant.

Fig. 1 shows an example of the region *R* called a *proper dominating-free set* which is composed with not bad locations in the city. A black circle represents an apartment and a white circle represents an existing restaurant. A polygon which is surrounded with solid lines is called a *dominating-free set*, and *R* is the region obtained by deleting the boundary and the dotted line segment from the polygon.

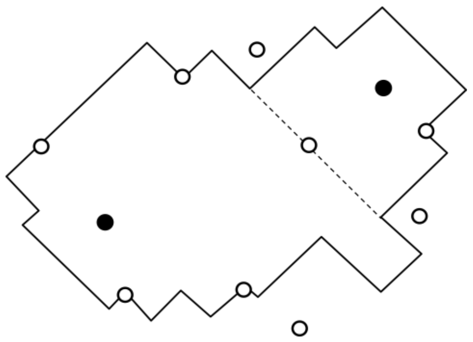

Fig. 1. An example of the proper dominating-free set.

The rest of this paper is organized as follows. In Section 2, some definitions and properties are described, and a simple algorithm of computing a dominating-free set is given. In Section 3, we present an efficient algorithm for two-server problem, i.e., computing the dominating-free set in case that the cardinality of a server set is two. Finally, Section 4 concludes the paper.

II. DEFINITIONS AND PRELIMINARIES

The distance between two points *p* and *q* in the plane under the L_1-metric is denoted by *dist*(*p*, *q*). For example, when *p* = (5, 9), *q* = (7, 3), *dist*(*p*, *q*) = |5-7| + |9-3| = 8. Given two distinct points *p* and *q*, a diamond whose center is p and the L_1-distnace between p and a point of it is less than or equal to $dist(p, q)$ is denoted by $D(p, q)$.

Let a client set $C = \{c_0, c_1, \ldots, c_{n-1}\}$ be a point set with n distinct points in the plane and a server set $S = \{s_0, s_1, \ldots, s_{k-1}\}$ be a point set with k distinct points.

The proper dominating-free set by two point sets C and S is formally defined as follows.

Def. 1: A point set Q is called the *proper dominating-free set* by two point sets C and S if for every point $q \in Q$, there exists a pair (s, c) of a server s and a client c such that $dist(s, q) < dist(s, c)$.

For a point q' which doesn't lie on the proper dominating-free set by C and S, there exists a client c such that $dist(s_0, q') \geq dist(s_0, c)$, $dist(s_1, q') \geq dist(s_1, c)$, … , and $dist(s_{k-1}, q') \geq dist(s_{k-1}, c)$. A proper dominating-free set is an open set of points and its interior may have zero or more holes(points or line segments)(see Fig. 1). In order to produce a point set as a simple polygon, a dominating-free set is defined as follows.

Def. 2: A point set Q' is called the *dominating-free set* by two point sets C and S if for every point $q \in Q'$, there exists a pair (s, c) of a server s and a client c such that $dist(s, q) \leq dist(s, c)$.

The dominating-free set by two point sets C and S is denoted by $DFS(C, S)$. The following lemma shows a characterization of a dominating-free set.

Lemma 1: For two point sets C and S, $DFS(C,S) = \bigcap_{0\leq j\leq n-1}\{\bigcup_{0\leq i\leq k-1} D(s_i,c_j)\}$.

Proof. Let $E(c_j) = D(s_0,c_j) \cup D(s_1,c_j) \cup \cdots \cup D(s_{k-1},c_j)$. For every point q of $E(c_j)$, at least one of the following k conditions holds true.

$$dist(s_0, q) \leq dist(s_0, c_j)$$
$$dist(s_1, q) \leq dist(s_1, c_j)$$
$$\ldots$$
$$dist(s_{k-1}, q) \leq dist(s_{k-1}, c_j)$$

But, for a point $q \notin E(c_j)$, above all conditions are false. Hence, all points of $E(c_j)$ are contained in $DFS(C, S)$ and all external points of $E(c_j)$ aren't contained in the set by definition. Therefore, for all j, the intersection of $E(c_j)$'s is the dominating-free set by C and S. □

A *convex polygon* is defined as a polygon P for which the line segment connecting any two points in P lies entirely within P. If we change the "line segment" to "horizontal or vertical line segment", the resultant region is called an *orthogonal convex polygon*. The following lemma shows that a dominating-free set is an orthogonal convex polygon rotated by 90 degrees.

Lemma 2: $DFS(C,S)$ is an orthogonal convex polygon rotated by 90^o.

Proof. Let's consider the coordinate system rotated by 90 degrees. Then a diamond $D(s_i,c_j)$ would be a square under the transformed coordinate system. Since the union of two or more squares which share a point is an orthogonal convex, $D(s_0,c_j) \cup \cdots \cup D(s_{k-1},c_j)$ is orthogonal convex, too. Since the intersection of two orthogonal convex polygons is also orthogonal convex[8], $DFS(C,S)$ is orthogonal convex by Lemma 1. □

From Lemma 2, we can get an algorithm for computing $DFS(C,S)$ as follows: First, compute $E(c_j) = D(s_0,c_j) \cup \cdots \cup D(s_{k-1},c_j)$ for all j. Next, compute incrementally the intersection of $E(c_j)$'s. The union of k squares which share a point can be computed in $O(k\log k)$ time[7]. Since the intersection of two orthogonal convex polygons can be constructed in linear time[8], $DFS(C,S)$ can be computed in $O(nk\log k + \sum_{i=2}^{n} ik) = O(nk\log k + n^2k)$.

III. TWO-SERVER PROBLEM

In this section, we consider the two-server problem, i.e., the case that $S = \{s_0, s_1\}$. From Lemma 2, we know that there is an $O(n^2)$ time algorithm for computing $DFS(C,S)$, where $|C| = n$ and $|S| = 2$. We present a more efficient algorithm for computing a dominating-free set.

Without loss of generality, we assume that the x-coordinate of s_0 is less than s_1's and the y-coordinate of s_0 is less than s_1's (see Fig. 2).

Let R_S be the rectangle which is defined by s_0 and s_1 of S. as shown in Fig. 2.

Lemma 3: R_S is contained in $DFS(C,S)$.

Proof. For every point c of C, the polygon $D(s_0,c) \cup D(s_1,c)$ includes R_S. So, the lemma holds true by Lemma 1. □

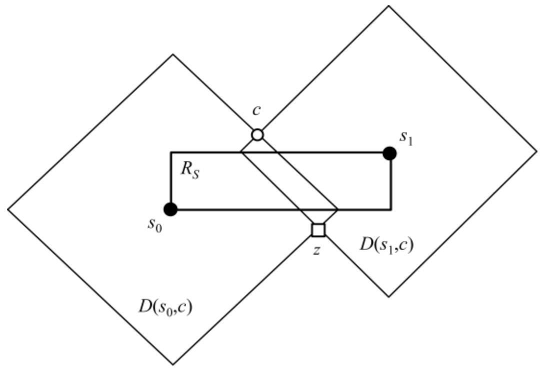

Fig. 2. An example of $D(s,c)$ and R_S.

Now, we define the *event points* as follows: Consider an intersection point z of the boundaries of two diamonds $D(s_0, c)$ and $D(s_1, c)$ for $c \in C$. Let u (resp. v) be a vertex of $D(s_0, c)$ (resp. $D(s_1, c)$) which is adjacent from z. If the chain (u, z, v) is wedge-shaped("V") and the triangle (u, z, v) doesn't intersect with the interior of $D(s_0, c) \cup D(s_1, c)$, then z is said to be an event point. In Fig. 2, c and z are event points.

Extending the edges of R_S, the rectangle defined by s_0 and s_1, the plane is partitioned into the regions R_1(*North-east*), R_2(*North*), R_3(*North-west*), R_4(*West*), R_5(*South-west*), R_6(*South*), R_7(*South-east*), R_8(*East*), and R_S as shown in Fig. 3.

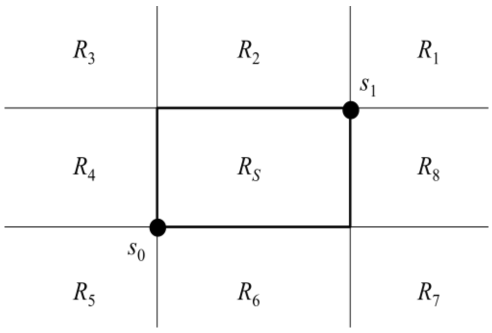

Fig. 3. The regions partitioned by s_0 and s_1.

Lemma 4: All event points by C and S lie on the regions R_2, R_4, R_6, or R_8.

Proof. Since all vertices of $D(s_0, c)$ and $D(s_1, c)$ lie on the lines extending the edges of R_S, we can easily see that there doesn't exist an event point which lies inside the regions R_1, R_3, R_5, or R_7. □

A vertex of a polygon is called *convex* if its internal angle is strictly less than 180^o, otherwise, it is called *reflex*.

Lemma 5: All reflex vertices of $DFS(C, S)$ correspond to event points.

Proof. By Lemma 1, we know that $DFS(C, S)$ is the intersection of $D(s_0, c_i) \cup D(s_1, c_i)$, i = 0, 1, … , n-1. $D(s_0, c_i) \cup D(s_1, c_i)$ is the union of two squares. So, the event points by $D(s_0, c_i)$ and $D(s_1, c_i)$ are reflex vertices of $D(s_0, c_i) \cup D(s_1, c_i)$.

Since every $D(s_0, c_i)$ is a diamond of which the center is s_0 and the radius is c_i, for two integers i and j, there is only the inclusion relation between $D(s_0, c_i)$ and $D(s_0, c_j)$, i.e., $D(s_0, c_i) \subseteq D(s_0, c_j)$ or $D(s_0, c_i) \supseteq D(s_0, c_j)$. So do $D(s_1, c_i)$'s.

Therefore, for two distinct integers i and j, any intersection points of $D(s_0, c_i)$ and $D(s_1, c_j)$ cannot be reflex vertices of $DFS(C, S)$(see Fig. 4). □

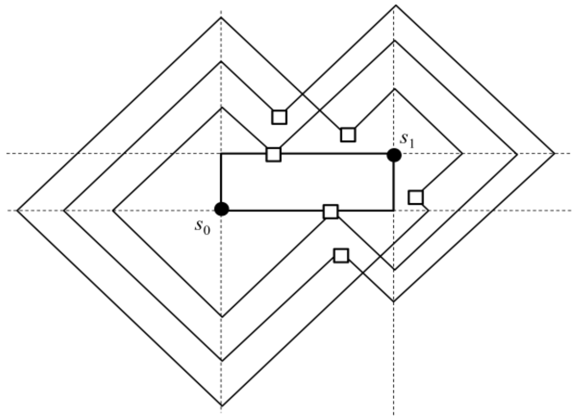

Fig. 4. Illustration for the proof of Lemma 5.

The region $D(s_0, c_i) \cup D(s_1, c_i)$ has at most two event points which lie on R_2, R_4, R_6, or R_8. Now, we introduce *the pseudo-event points* for $E(c_i) = D(s_0, c_i) \cup D(s_1, c_i)$ which are the intersection points between $E(c_i)$ and the lines extending the edges of R_S. Then, each region of R_2, R_4, R_6, and R_8 has at least one event point or pseudo-event point. In Fig 5, a white circle represents a client point, a white square represents an event point, and a gray square represents a pseudo-event points.

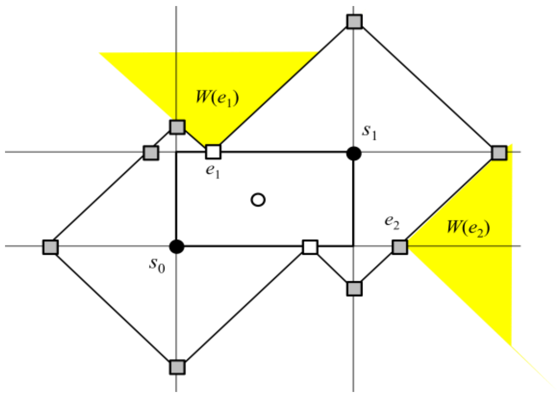

Fig. 5. Event points and pseudo-event points.

For an event point or a pseudo-event point z, a wedge region $W(z)$ is defined as follows: If z is on R_2(resp. R_4, R_6, or R_8), $W(z)$ is the region on the upper (resp. left, lower, or right) side of R_S among the regions which are surrounding with two lines which pass by z and have the scope +1 and -1. By the definition, a wedge region doesn't intersect any inner points of $DFS(C, S)$ (see Fig. 5).

For an event point or a pseudo-event point e which lies on R_i(i = 2, 4, 6, 8) called a *local minimum* if $W(e)$ is not a subset of $W(e')$ for every event point or pseudo-event point e'. For two event points or pseudo-event points $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$, $W(e_1)$ includes $W(e_2)$ if

$$x_2+y_2 \geq x_1+y_1 \text{ and } y_2-x_2 \geq y_1-x_1, \quad e_1, e_2 \in R_2$$
$$x_2+y_2 \leq x_1+y_1 \text{ and } y_2-x_2 \geq y_1-x_1, \quad e_1, e_2 \in R_4$$
$$x_2+y_2 \leq x_1+y_1 \text{ and } y_2-x_2 \leq y_1-x_1, \quad e_1, e_2 \in R_6$$
$$x_2+y_2 \geq x_1+y_1 \text{ and } y_2-x_2 \leq y_1-x_1, \quad e_1, e_2 \in R_8$$

Lemma 6: All reflex vertices of $DFS(C,S)$ are event points which are local minima, and *vice versa*.

Proof. By definition, an event point which is not a local minimum is contained in the wedge region by an event point or a pseudo-event point which is a local minimum. Since a wedge region lies on the exterior of $DFS(C,S)$, a reflex vertex of $DFS(C,S)$ must be an event point which is a local minimum.

There is an inner point of $DFS(C,S)$ in a very small circle of which the center is an event point which is a local minimum. So, an event point which is a local minimum must be a part of the boundary of $DFS(C,S)$. Hence, all local minima are reflex vertices of $DFS(C,S)$. □

From now on, a pseudo-event point is just called an event point without distinction. Consider the sorted list L_j of the local minima on a region R_j (j = 2, 4, 6, 8). The event points on R_2 and R_6 are sorted along the x-axis, and those on R_4 and R_8 are sorted along the y-axis.

Lemma 7: For each L_j (j = 2, 4, 6, 8), the intersection point of the boundaries of the wedge regions by two adjacent event points which don't lie on the lines extending edges of R_S is a convex vertex of $DFS(C,S)$.

Proof. As shown in Fig. 6, the intersection of the boundaries of the wedge regions by two adjacent event points aren't contained in the interior of any wedge region of an event points. Hence, it is a member of the boundary of $DFS(C,S)$. Since it is not a reflex vertex by Lemma 6, it is a convex vertex of $DFS(C,S)$. □

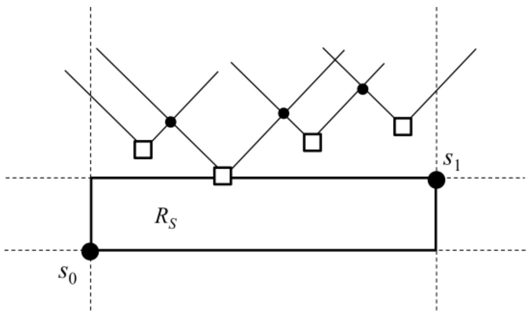

Fig. 6. Illustration for the proof of Lemma 7.

Let z_0 be the intersection point of the edge shared by R_1 and R_2 and the boundary of the wedge region by the last event point of L_2. Similarly, z_1, z_2 , … , and z_7 are defined as shown in Fig. 7.

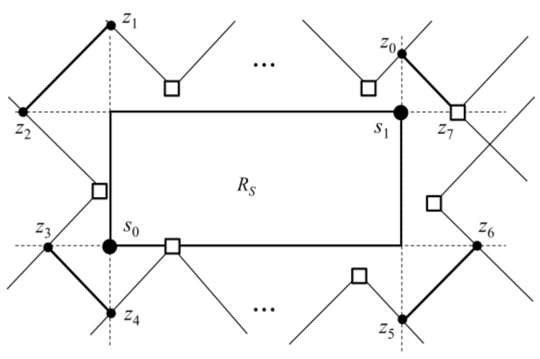

Fig. 7. The intersection points z_0, z_1 , … , and z_7.

Lemma 8: The line segment $l(z_i, z_{i(i+1) \bmod 8})$ (i = 0, 1, … , 7) is an edge of $DFS(C,S)$.

Proof. For all k and j, the edges of $D(s_k, c_j)$ which pass by a region among R_1, R_3, R_5, and R_7 are parallel to each other(see Fig. 4). So, the innermost edge must be an edge of $DFS(C,S)$. By definition of local minima, two end vertices of the edge are corresponding to z_0, z_1, … , or z_7. □

Now, we describe an algorithm for computing $DFS(C,S)$, where $|C|$ = n and $|S|$= 2. An outline of an algorithm is as follows:

1. Find the event points and pseudo-event points.
2. Find the local minima among event points and pseudo-event points, then construct the sorted lists L_2, L_4, L_6, and L_8.
3. Construct the boundary of $DFS(C,S)$.

Theorem 1: $DFS(C,S)$, $|C|$ = n and $|S|$= 2, can be constructed in $O(n\log n)$ time and $O(n)$ space.

Proof. By above lemmas, we can see that above algorithm correctly constructs $DFS(C,S)$ for given two point sets C and S.

Now, we consider the time complexity. The union U of two diamonds can be computed in constant time. At most two event points are extracted from U in constant time, and the eight intersection points z_0, z_1, … , and z_7 can be computed in constant time, too. Hence, Step 1 can be done in $O(n)$ time.

For event points and pseudo-event points which lie on R_i (i = 2, 4, 6, 8), the local minima can be found by sorting the event points and pseudo-event points along (x+y, y-x) values and then traversing the sorted lists. L_2, L_4, L_6, and L_8 are consequently obtained through the procedure. Hence, Step 2 can be done in $O(n\log n)$ time.

By Lemma 6, 7, 8, we can see that all vertices of $DFS(C,S)$ are obtained from the lists L_2, L_4, L_6, and L_8. Therefore, sequentially traversing the lists is enough to get $DFS(C,S)$. Hence, Step 3 can be done in $O(n)$ time.

Since only $O(n)$ space is required through the entire steps of the algorithm, it runs in $O(n\log n)$ time and $O(n)$ space. □

In the special case that both of s_0 and s_1 have the same x-coordinates or y-coordinates, an algorithm for computing $DFS(C, S)$ is far simpler than the general case. So, we omit it.

IV. CONCLUDING REMARKS

In this paper, we presented an $O(nk\log k + n^2k)$ time algorithm for computing the dominating-free set by two point set C and S under L_1-metric. Specially, we presented an efficient $O(n\log n)$ time algorithm for computing a dominating-free set when $|S|=2$. The algorithm uses some variables and 1-dimensional arrays as its principle data structures, so it is easy to implement and runs fast.

As the further work, the study for finding a more efficient algorithm of computing a dominating-free set in case that $|S| > 2$ is needed.

REFERENCES

[1] S. Börzsönyi, D. Kossmann, and K. Stocker, “The skyline operator,” *Proc. 17th International Conference on Data Engineering*, pp. 421-430, 2001.

[2] D. Papadias, Y. Tao, G. Fu, B. Seeger, “An optimal and progressive algorithm for skyline queries,” *Proc. the 2003 ACM SIGMOD international conference on Management of data*, pp. 467-478, 2003.

[3] M. Sharifzadeh and C. Shahabi, “The Spatial Skyline Queries,” *Proc. the 32nd international conference on Very large data bases*, pp. 751-762, 2006.

[4] W. Son, M. W. Lee, H. K. Ahn, and S. Hwang, “Spatial Skyline Queries: An Efficient Geometric Algorithm,” *Proc. The 11th International Symposium on Advances in Spatial and Temporal Databases*, pp. 247-264, 2009.

[5] K. Deng, X. Zhou, and H. T. Shen, “Multi-source skyline query processing in road networks,” *Proc. 23th International Conference on Data Engineering*, pp. 796-805, 2007.

[6] Wanbin Son and Hee-Kap Ahn, “Computing the Skyline of a Point Set with a Highway,” *Proc. Korea Computer Congress* 2009, Vol. 36, No. 1, 2009.

[7] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, pp. 345-350, 1985.

[8] J.-R. Sack, “Rectilinear Computational Geometry,” Ph. D. Dissertation, School of Comput. Sci., Carleton University, Ottawa, Canada, 1984.

Soo-Hwan Kim received his B.S. degree in Computer Science & Statistics from Seoul National University, Seoul, Korea, in 1987, M.S. and Ph.D. degrees in Computer Science from KAIST, Korea, in 1989 and 1995, respectively. In 1992, he joined the faculty at Pusan University of Foreign Studies, Busan, Korea, where he is currently a professor in the department of Embedded IT. His research interests include the design of sequential and parallel algorithms, computational geometry, and image processing.