Fast Maintenance of Rectilinear Centers

Sergei Bespamyatnikh[1] and Michael Segal[2]

[1] Department of Computer Science, University of British Columbia,
Vancouver V6T 1Z4, Canada
es @cs.u c.ca, htt :// .cs.u c.ca/s ide / es
[2] Department of Communication Systems Engineering,
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
se a @cs. u.ac.i , htt :// .cs. u.ac.i /~se a

Abstract. We address the problem of dynamic maintenance of 2-centers in the plane under rectilinear metric. We present two algorithms for the continuous and discrete versions of the problem. We show that rectilinear 2-centers can be maintained in $O(\log^2 n)$ time. We give an algorithm for semi-dynamic (either insertions only or deletions only) maintenance of the discrete 2-centers in $O(\log n \log m)$ amortized time where n is the number of customer points and m is the number of possible locations of centers.

1 Introduction

Given two sets S, C of points in the plane of size n and m, respectively we wish to maintain dynamically (under insertions and/or deletions of points of S)

1. **Rectilinear 2-center:** two squares that cover S such that the radius of maximal square is minimized.
2. **Discrete Rectilinear 2-center:** two squares that cover S centered at points of C such that the radius of maximal square is minimized.

We also consider the generalization of problem 2 for the case of rectangles, where one wants to minimize the largest perimeter. There are several results for the static version of the problems above. A linear time algorithm for the planar rectilinear 2-center problem is given by Drezner [4]. The $O(n \log n)$ time solution for the discrete rectilinear 2-center was given by Bespamyatnikh and Segal [3] and the optimality of their algorithm has been shown by Segal [6]. To our best knowledge nothing has been done regarding the dynamic version of the rectilinear 2-center problem. Bespamyatnikh and Segal [3] considered also a dynamic version of the discrete rectilinear 2-center. They have been able to achieve an $O(\log n)$ update time, though the actual query time is only $O(m \log n(\log n + \log m))$.

For the dynamic rectilinear 2-center problem we present a scheme which allows us to maintain an optimal solution under insertions and deletions of points of S in $O(\log^2 n)$ time (both update and query), after $O(n \log n)$ preprocessing time. For the semi-dynamic discrete rectilinear 2-center problem we give an algorithm for maintaining the optimal pair of squares under insertions only (resp. deletions only) of points of S in amortized $O(\log n \log m)$ time (both update and query), after $O(n \log n)$ preprocessing time. Our solution for the semi-dynamic

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 633–639, 2001.

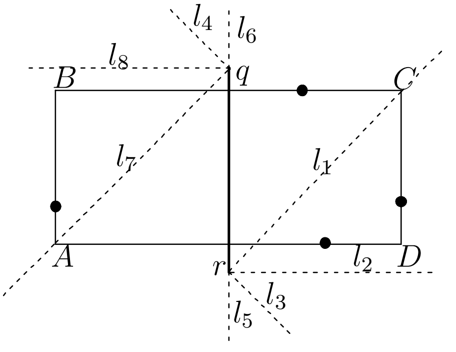

Fig. 1. Subdivision of the bounding box into the ranges.

discrete rectilinear 2-center improves the best previous result by almost linear factor, thus providing first sublinear semi-dynamic algorithm for dynamic maintenance of the discrete rectilinear 2-center.

2 Dynamic Rectilinear 2-Center

Denote by $|pq|$ the L_∞ distance between two points p, q in the plane. We observe as in [2] that two pairs of the diagonal vertices of the bounding box of S play a crucial role in defining two minimal squares that cover S. More precisely, let us consider a pair of diagonal vertices A and C of the bounding box of S in Figure 1. For the vertex A we find the farthest neighbor point $p' \in S$ (in L_∞ metric) among the points that are closer to A than to C. We repeat the similar procedure for vertex C, obtaining point p''. It can be done efficiently by constructing a rectilinear bisector l_4qrl_3 and dividing the obtained regions into the wedges, see Figure 1. The main property of such subdivision is that the largest distance from a point $p_i \in W$ (W is a wedge) to corresponding vertex (A or C) is either x- or y-distance between p_i and the corresponding vertex. For example, consider the diagonal vertex C in Figure 1 and associated with C wedges: l_4ql_6, l_6qrl_1, l_1rl_2, l_2rl_3 (we should consider all these wedges since it may happen that points q and r will be inside of the bounding box of S). We can use the orthogonal range tree data structure [1] in order to find the required largest distance. For the case of wedge l_1rl_2, only the y-coordinate of any point of S lying in this wedge determines the distance from this point to C. We construct a range tree T in the new system of coordinates corresponding to the directions of l_1 and l_2. The main structure of T is a balanced binary tree according to the "x"-coordinate of points. Each node v of this tree corresponds to the balanced binary tree (secondary tree) according to the "y"-coordinate of points whose "x"-coordinate belongs to the subtree rooted at v. We augment this data structure by keeping an additional value for each node w in the secondary data structures as the minimal value of the actual x-coordinates of the points corresponding to the nodes in the subtree rooted at w. In order to find the farthest l_∞ neighbor of C in the wedge l_1pl_2, we perform a query on t by taking this wedge as a range. At most $O(\log^2 n)$ nodes of the secondary data structure are taken into account and we collect all the minimal x-values that are kept in these nodes. A point that has a minimal x-coordinate is a farthest neighbor of C in the wedge l_1pl_2.

We apply the similar technique for the remaining wedges. The entire update and query procedure takes $O(\log^2 n)$ time after initial $O(n \log n)$ time for the construction of the orthogonal range trees. In this way we can compute points p' and p''. Let δ_1 be the maximal value between $|Ap'|$ and $|Cp''|$. Using the same searching farthest neighbor technique for a different pair of diagonal vertices B and D, we obtain points $q', q'' \in S$ such that $|Bq'| = \max_{q\ \ S,|Bq|\ \ |Dq|} |Bq|$ and $|Dq''| = \max_{q\ \ S,|Dq|<|Bq|} |Dq|$. Let $\delta_2 = \max(|Bq'|, |Dq''|)$. Finally, the smallest value between δ_1 and δ_2 defines the size of the squares and their position in the optimal solution of the rectilinear 2-center problem.

3 Dynamic Discrete Rectilinear 2-Center

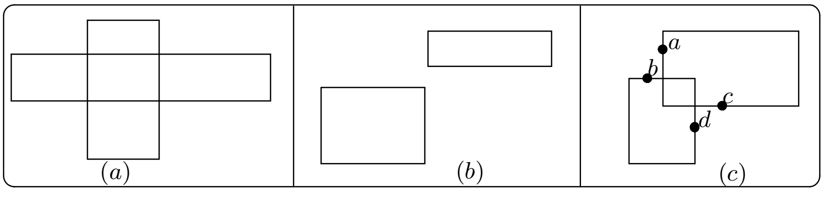

Fig. 2. Different configurations of bounding boxes defined by two optimal discrete squares.

First, we consider an optimal solution for the static discrete rectilinear 2-center problem. Let s_1 and s_2 be two optimal discrete squares centered at points of C that cover S. Consider the bounding boxes B_1 and B_2 of points covered by s_1 and s_2, respectively. Three different configurations of B_1 and B_2 are possible, see Figure 2. (In fact, in our analysis a few more different configurations appear, but they are symmetrically opposite to the configurations described below). Denote by $bb(S)$ the bounding box of S. We call a point of S a *determinator* if it lies on one of the edges of $bb(S)$. Normally, $bb(S)$ has four determinator points $r, l, t, b \in S$ that lie onto the right, left, top and bottom sides of $bb(S)$, respectively. Configuration (a) is characterized by fact that each one of the bounding boxes B_1 and B_2 has two opposite determinators on its sides, e.g., r and l lie on the edges of B_1 while b and t lie on the edges of B_2. In configurations (b) and (c), each one of the bounding boxes B_1 and B_2 has two adjacent determinators on its sides, e.g., l and b lie on the edges of B_1 while r and t lie on the edges of

B_2. The main difference between these two configurations is that in case (b) B_1 and B_2 are totally disjoint, while in case (c) B_1 and B_2 intersect.

For each configuration we find an optimal pair of discrete squares as follows. First, we consider case (a). We show that one of the squares s_1 and s_2 contains three determinators. Without loss of generality we assume that the width of the bounding box of S is greater or equal to its height. Suppose that B_1 contains left and right determinators. Then s_1 contains either upper or lower (or both) determinator. Therefore we may assume that B_1 and B_2 are totally disjoint; moreover, one of them contains three determinators. This case can be solved easily by applying a binary search on the sorted list of x-coordinates (y-coordinates) of the points of S. Each step of the binary search splits the points of S into two subsets $S_1, S_2 \subset S$. For each subset S_i, we compute its bounding box $B_i, i = 1, 2$ (we can assume that one of the bounding boxes contains three determinators).

Now, we need to find two smallest discrete squares s_1 and s_2 that cover B_1 and B_2, respectively. Consider the bounding box B_1 and its center c_1. Without loss of generality the width of B_1 is greater or equal to its height. Our goal is to find the closest L_∞ neighbor point $q \in C$ to the vertical segment A_1A_2 (note that that is defined as follows. A_1A_2 passes through center c_1, the ray emanating from left-bottom corner of B_1 in direction to A_2 makes $45°$, and A_1 lies on $(-45°)$-ray from left-top corner of B_1, (see Figure 3). This point q will define the center of the discrete square s_1. We can find q using orthogonal range trees by the similar technique described in the previous section. We divide the search region into the wedges as shown in the Figure 3, such that the smallest distance from a point $q_i \in W$ (W is a wedge) to A_1A_2 is either x- or y-distance (depending on the wedge) between q_i and A_1A_2.

After we found the locations and sizes of s_1 and s_2 we guide a binary search in order to get an optimal size for the squares for this configuration. Notice that the configuration (b) can be solved by the same method by applying two binary searches on the points (according to x- and y-coordinates) of S. In each step of a binary search we obtain disjoint boxes B_1 and B_2. We find a minimal discrete square that covers B_i, $i = 1, 2$ using orthogonal range trees. The total time required for case(b) (and case(a)) is $O(\log n \log m)$.

The case (c) is most interesting and it can be solved using the following approach. The bounding boxes B_1 and B_2 form two orthogonal corners with four points $a, b, c,$ $\in S$, see Figure 2(c). The additional property is that $B_1 \cap B_2 \neq \emptyset$. We conclude that the points a, b form a single link in the upper-left staircase chain of the points of S and the points $c,$ form a single link in the lower-right staircase chain of the points of S. These two chains correspond the maximal upper-left (north-west) and lower-right (south-east) points of S (similar to set of maxima of S and set of minima of S, Chapter 4 [5]). Each pair of corners: one from the upper-left staircase and one from the lower-right staircase define a configuration with two discrete squares that cover S. For each corner on the upper-left staircase we find the best corresponding corner (in terms of the largest

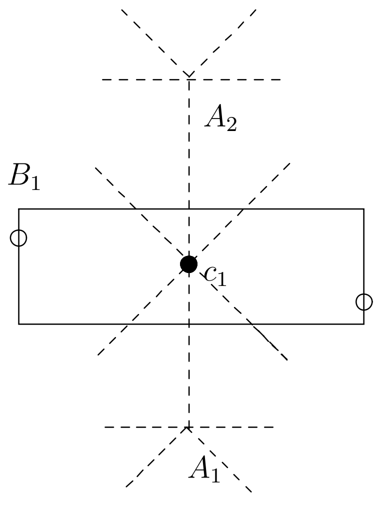

Fig. 3. Regions for point $q \in C$.

size of two obtained squares) on the lower-right staircase and put a *pointer* between them, see Figure 4.

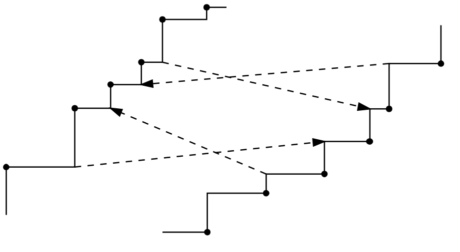

Fig. 4. Pointers between staircases.

We perform the similar operation for the corners in the lower-right staircase. Thus, we have a collection of at most $2n$ pointers. It may happen that two or more pointers refer to the same corner; in this case we store only one pointer that defines the best two discrete squares. In fact, we keep the sizes of squares in the heap (as an appropriate pointer with associated size of square). Notice that, for a particular corner c as a source, we can find its pointer in $O(\log n \log m)$ using a binary search with orthogonal range trees. For fixed c, the bounding box B_1 (and B_2) has three fixed sides. B_1 changes monotonically when we traverse corners on the opposite staircase. Therefore the size of the discrete square covering B_1 changes monotonically and we can apply binary search. For any corner on the opposite staircase, the discrete square containing B_1 can be obtained in $O(\log m)$ time using range trees. The binary search finds two corners c' and c'' such that corresponding sizes s'_1, s'_2 and s''_1, s''_2 of the squares satisfy the following property: $s'_1 \leq s'_2$ and $s''_1 \quad s''_2$. The total time for finding a pointer is $O(\log n \log m)$.

Consider the insertion of a new customer point p. If p lies between two staircases, then it does not make any change to the staircases and our current solution (case (c)). Suppose that p is above left staircase (the case of right staircase is symmetric), see Figure 5. First, we update the staircase. We find the sequence of corners that are no longer valid. Otherwise, we update a corresponding staircase, remove non-valid pointers and compute two new pointers from the corners defined by the new inserted point and its neighbors in the staircase. If only insertions are allowed (or deletions) the total number of changes in the staircases is $O(n)$ and, therefore, we achieve an amortized $O(\log n \log m)$ time for updates and queries.

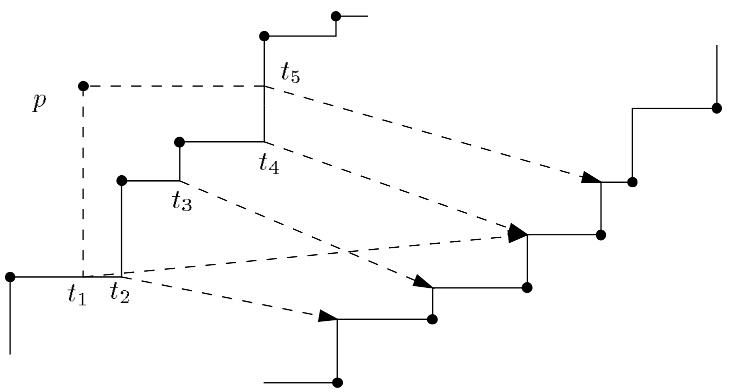

Fig. 5. Insertion of point p. Corners t_2, t_3, t_4 and their pointers are deleted and two corners t_1 and t_5 with pointers are inserted.

Theorem 1. *Rectilinear 2-centers can be maintained in amortized $O(\log n \log m)$ time in semi-dynamic data structure of linear size.*

4 Future Work

In the extended version of this paper we also show how to maintain an $(1+\varepsilon)$-approximated solution for the discrete two-center problem in $O\left(\frac{1}{\varepsilon}\log(n+m)\right)$ time by supporting both deletions and insertions. We also show how to solve efficiently the discrete two-center rectangular problem. A possible future directions for research are containing the extension of the results obtained in this paper to higher dimensions, making algorithms fully dynamic and considering the Euclidean metric.

References

1. M. de Berg, M. van Kreveld, M. Overmars, O. Schwartzkopf *Computational Geometry, Algorithms and Applications*, Springer-Verlag, 1997.
2. S. Bespamyatnikh and D. Kirkpatrick, "Rectilinear 2-center problems", in *Proc. of 11th Can. Conf. Comp. Geom.*, pp. 68–71, 1999.
3. S. Bespamyatnikh and M. Segal, "Rectilinear static and dynamic discrete 2-center problems", in *Int. Jour. of Math. Algorithms*, to appear.
4. Z. Drezner, "On the rectangular p-center problem", *Naval Res. Logist. Q.*, 34, pp. 229–234, 1987.
5. F. P. Preparata and M. I. Shamos, "Computational Geometry: An Introduction", *Springer-Verlag*, 1990.
6. M. Segal, "Lower bounds for covering problems", manuscript, 1999.