Kinetic Red-Blue Minimum Separating Circle

Conference Paper · August 2011

DOI: 10.1007/978-3-642-22616-8_35 · Source: DBLP

3 authors, including:

Yam-Ki Cheung
Chinese Academy of Sciences

Marko Zivanic
University of Texas at Dallas

Kinetic Red-blue Minimum Separating Circle

Yam Ki Cheung, Ovidiu Daescu, and Marko Zivanic
Department of Computer Science
The University of Texas at Dallas
Richardson, TX USA
Email: {ykcheung,daescu,mxz052000}@utdallas.edu

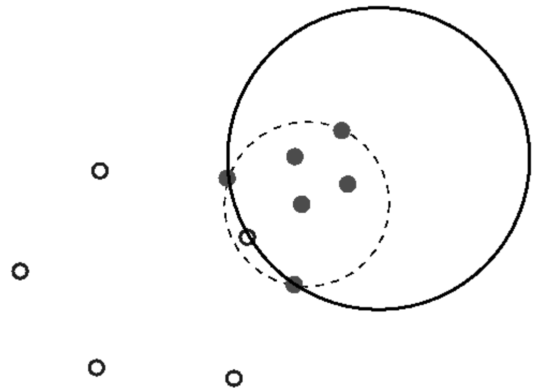

Figure 1: Minimum red enclosing circle (dashed) and red-blue separating circle (solid).

1 Introduction

Let $\mathcal{R}$ and $\mathcal{B}$ be two finite sets of points in $\mathbb{R}^2$, of size $|\mathcal{R}| = n$ and $|\mathcal{B}| = m$, respectively. We refer to $\mathcal{R}$ as the set of red points and to $\mathcal{B}$ as the set of blue points. In [4] the authors define a constrained version of the circular separability problem, called the *minimum separating circle problem*, as follows: Let $\mathcal{S}$ denote the set of circles such that each circle in $\mathcal{S}$ encloses all points in $\mathcal{R}$ while having the smallest number of points of $\mathcal{B}$ in its interior. The goal is to find the smallest circle in $\mathcal{S}$, called the *minimum separating circle* and denoted by $C_{\mathcal{B}}(\mathcal{R})$. See Figure 1 for an illustration.

The problem has applications in military planning. It can be used to determine the best location to deploy an explosive and the amount needed so that all enemy forces, represented by red points, will be impacted while minimizing civilian casulties (blue points). It is also applicable in determining the best set-up of communication devices such that all red devices stay connected and as few blue devices as possible can intercept their communication. Two algorithms for the static version of this problem have been proposed by Bitner et al. [4].

In practice, however, it is possible that not all points (targets) are stationary. In this paper, we study a kinetic version of the red-blue minimum separating circle problem, in which all points are stationary except one red point, which moves along a linear trajectory with constant velocity. We want to find the locus of the minimum separating circle over a period of time.

For the case when the two point sets can be separated, Fisk [6] gave a quadratic time and space algorithm to compute the minimum separating circle. The result was later improved to optimal linear time and space by O'Rourke et. al. [8].

To the best of our knowledge the kinetic version of the minimum separating circle problem has not been studied in the past, but there is a significant number of publications on related topics. Atallah [1] introduced the concept of kinetic computational geometry in a seminal paper on this topic. Basch et al. [3] introduced a set of kinetic data structures that can be used to maintain the convex hull of a moving set of points. Ross [10] gived an algorithm for maintaining the nearest-point Voronoi diagram of a kinetic data set. He presented an update algorithm for the topological structure of the Voronoi diagram of moving points, using $O(\log n)$ time for each change. Demaine et al. [5] presented a kinetic data structure that calculates the minimum spanning circle for a moving set of points. Banik et al. [2] solved the minimum enclosing circle problem of a fixed set of points and one moving point. Their algorithm computes the locus of the center of the minimum enclosing circle in linear time. Rahmati et al. [9] presented a kinetic data structure for the maintenance of the minimum spanning tree on a set of moving points in $\mathbb{R}^2$.

2 Preliminaries

We start by briefly discussing an algorithm proposed by Bitner et al. [4] for the static version of the minimum separating circle problem. The algorithm is based on a sweep procedure on the edges of the farthest neighbor Voronoi diagram $FVD(\mathcal{R})$ of $\mathcal{R}$.

Lemma 1 *[4] The smallest separating circle must pass through at least two points from $\mathcal{R}$.*

It follows that the minimum separating circle is either the smallest enclosing circle of $\mathcal{R}$, which can be found in linear time [7], or a circle which passes

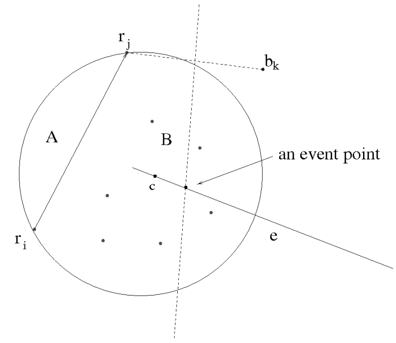

Figure 2: An event point on edge e_{ij}.

through two points from $\mathcal{R}$ and one point from $\mathcal{B}$. Thus, the center of a minimum separating circle lies on an edge of $FVD(\mathcal{R})$.

Following Lemma 1, Bitner et al. [4] proposed a sweep algorithm for computing the minimum separating circle.

Consider a Voronoi edge e_{ij}, defined by two red points r_i and r_j. The sweep procedure on e_{ij} is initialized by constructing an enclosing circle C of $\mathcal{R}$ which passes through r_i and r_j and has the smallest possible radius. This happens at one endpoint of e_{ij}. Let $c \in e_{ij}$ be the center of C. C is grown by sweeping c along e_{ij} and keeping r_i and r_j on the circumference of C.

A point $E \in e_{ij}$ is an event point if, when c sweeps through E, the circle C sweeps through a blue point (see Figure 2 for an illustration). The number of blue points enclosed by C is updated at each event point. The sweep procedure on e_{ij} terminates when the other endpoint of e_{ij} is reached. To total number of event points over $FVD(\mathcal{R})$ is $O(mn)$.

Lemma 2 *[4] A point from $\mathcal{B}$ defines at most one exit event point on $FVD(\mathcal{R})$, and such exit event point can be found in $O(\log n)$ time.*

3 The minimum separating circle with one mobile red point

In this section, we study the minimum separating circle problem for two point sets $\mathcal{R}$ and $\mathcal{B}$, such that all points are stationary except one red point p, which is moving along a linear trajectory with constant velocity. We show how to find the locus of the center of the minimum separating circle $C_{\mathcal{R}}(\mathcal{B})$ over a period of time.

Following Lemma 2, at any time instant t, we can find the minimum separating circle by computing the $FVD(\mathcal{R})$ and checking $O(m)$ exit event points. To find the locus of the minimum separating circle over a period of time, it is sufficient to keep track of the (trajectory of) the exit event associated with each blue point as well as the number of blue points enclosed by the corresponding candidate separating circle. Since one red point is mobile, the topology of $FVD(\mathcal{R})$ changes continuous. In order to keep track of the trajectory of each exit event point, we define the following four classes of events: 1) the appearance/disappearance of a vertex on $FVD(\mathcal{R})$; 2) the appearance/disappearance of a vertex on the boundary of the convex hull $CH(\mathcal{R})$ of $\mathcal{R}$; 3) the exit event point associated with a blue point moves to another Voronoi edge, and 4) the candidate separating circle associated with an exit event point encloses/excludes a blue point.

To avoid ambiguity, we refer to these four classes of events as *instant events*, distinguishing from the event points introduced in Section 2. Notice that case 1 and case 2 instant events indicate that the topology of the $FVD(\mathcal{R})$ need to be updated. Case 3 instant events indicate that the trajectory of some exit event point changes, i.e. it moves along a different line. And case 4 indicates that the number of blue points enclosed by some candidate separating circle needs to be updated.

Lemma 3 *The locus of the center of the minimum separating circle can be discontinuous.*

Lemma 4 *The locus of the center of the minimum separating circle consists of a set of halflines, line segments, or points.*

3.1 Case 1 instant events

These are time instants when a Voronoi edge appears or disappears from $FVD(R)$.

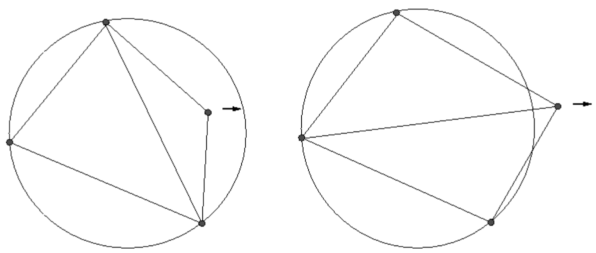

Figure 3: Update of Delaunay triangulation of four points by edge swapping.

Lemma 5 *There are at most $O(n)$ instant events in case 1. For each instant event, $FVD(\mathcal{R})$ can be updated in constant time.*

Proof. Instead of working on $FVD(\mathcal{R})$ directly, we turn our attention to the dual graph of $FVD(\mathcal{R})$, the Delaunay triangulation $DT(\mathcal{R})$.

The topology of $DT(\mathcal{R})$ could change when four red points, including the mobile red point p, on the boundary of $CH(R)$ are cocircular, or more specifically, when p enters/leaves a circle passing through three fixed red points, and enclosing all fixed red points in $\mathcal{R}$. See Fig. 3. Observe that the center of each such enclosing circle defines a vertex of $FVD(\mathcal{R}/\ p)$, so there are $O(n)$ such enclosing circles. $DT(\mathcal{R})$ can be updated in constant time by edge swapping. □

3.2 Case 2 instant events

These are time instants when a vertex of $CH(R)$ appears or disappears.

Lemma 6 *There are $O(n)$ instant events in case 2, and each event can be identified in constant time.*

Proof. Case 2 instant events occur when p passes through the intersection between a line supporting some edge of $CH(\mathcal{R}/\ p)$ and the trajectory of p. Hence, each instant event can be identified in constant time. See Fig. 4 for an illustration

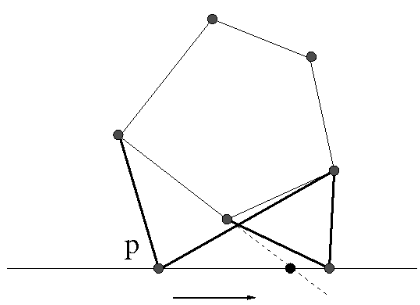

Figure 4: A vertex of $CH(R)$ appears.

□

3.3 Case 3 instant events

These are time instants when an exit event point moves to another edge of $FVD(\mathcal{R})$.

Lemma 7 *Each case 3 instant event can be identified in constant time.*

Lemma 8 *There are $O(nm)$ instant events in case 3.*

3.4 Case 4 instant events

These are time instants when the candidate separating circle centering at an exit event point encloses/excludes a blue point.

Lemma 9 *Each case 4 instant event can be found in linear time.*

Proof. Let e be a Voronoi edge defined by a fixed red point r_i and the mobile red point p and assume the exit event point of a blue point b lies on e. The candidate separating circle centering at the exit event point of b is the circumcircle of r_i, p, and b. The locus of the exit event point is a line segment supported by the bisector between r_i and b. We can keep track of the number of blue points enclosed by the candidate separating circle by running a sweep algorithm along the locus of the exit event point. This sweep algorithm resembles the one used to compute the minimum separating circle of two sets of fixed points. The only difference is that the center of the candidate separating circle moves along the bisector between r_i and b instead of a Voronoi edge.

The case 4 instant event is the moment when the exit event point crosses an event point defined on the bisector between r_i and b. □

Lemma 10 *There are at most $O(m^2 n)$ case 4 instant events.*

Proof. As shown in Lemma 8, one exit event point can traverse $O(n)$ Voronoi edges. For each edge traversed by the exit event point there can be $O(m)$ event points on the bisector between the blue point and the fixed red point which defines the Voronoi edge. Each event point corresponds to one case 4 instant event. Hence, each exit event generates at most $O(mn)$ case 4 instant events. □

3.5 Finding the locus of the center of the minimum separating circle

First, we need to determine the exact trajectory of each exit event point. If in a given time interval $[t_1, t_2]$, the exit event point of a blue point b lies on a Voronoi edge defined by two fixed red points, the exit event point is stationary. However, if the Voronoi edge is defined by one fixed red point a, and the mobile red point p, then the corresponding exit event point is the intersection between the bisector $B(a, b)$ between a and b and the bisector $B(b, p)$ between b and p. The trajectory of p can be expressed parametrically as $x = v_x t$ and $y = m_p v_x t + q_p$, where v_x is a constant and m_p and q_p are the slope and the intercept of the trajectory of p, respectively. It is not difficult to show that the trajectory of the intersection between $B(a, b)$ and $B(b, p)$ can be expressed as:

$$x = \frac{t^2 + c_1 t + c + 2}{c_3 t + c_4},$$

$$y = \frac{t^2 + c_1 t + c + 2}{c_3 t + c_4} m' + q',$$

where c_1, c_2, c_3, c_4 are constants and m' and q' are the slope and intercept of $B(a, b)$, respectively.

Once the trajectory of the exit event point is known, the function of the square radius of the corresponding candidate separating circle is a function $f_b(t)$ of constant degree.

Following Lemma 8, each exit event point cannot cross more than $O(n)$ Voronoi edges. For a time interval $[t_{init}, t_{end}]$, $f_b(t)$ consists of $O(n)$ pieces of curves or horizontal line segments if the exit event point is stationary.

Plotting all functions $f_b(t)$ for $t \in [t_{init}, t_{end}]$ and $b \in \mathcal{B}$ on the same coordinate system gives us an arrangement H of curves of complexity $O(nm^2)$, since all functions are x-monotone and two such functions intersect no more than $O(n)$ times. H gives us the relative size between all candidate circles over time.

However, we also need to consider the blue points enclosed by the candidate circles. We further decompose H by dividing each function $f_b(t)$ at every case 4 instant event t_o generated by b by introducing a vertex at $(t_o, f_b(t_o))$. As a result, each portion of $f_b(t)$ on the new arrangement H' represents the square radius of the candidate circle for a time interval during which the blue points enclosed by the circle remain the same. The new arrangement H' has complexity $O(m^2n)$. We call each portion of $f_b(t)$ on H' a *simple curve*.

Let the blue point count of a simple curve on H' be the number of blue points enclosed by the corresponding candidate circle. The last step to compute the locus of the minimum separating circle is to extract the lower envelope of curves with the lowest blue point count. Each curve on the lower envelope gives us the minimum separating circle for the interval spanned by the curve, hence, the locus of the center of the minimum separating circle.

We use a plane sweep to extract such lower envelope. We sweep H' by a vertical line. At any moment, the sweep line intersects with at most m functions $f_b(t)$, for $b \in \mathcal{B}$. Each function is indexed by its blue point count at the current moment. We build a hash table for functions intersected by the sweep line. For each entry of the hash table, all functions are maintained in a balanced tree by the order intersected by the sweep line. Note that we only have to update the hash table at vertices of H'. If two functions with the same blue point count intersect, we need to exchange their position in the corresponding tree. If the sweep line crosses a vertex introduced by a case 4 instant event, the blue point count of a function changes. The corresponding function will be moved to the appropriate entry of the hash table. It takes $O(\log m)$ time to update the hash table for each instance.

Theorem 11 *The locus of the center of the minimum separating circle has complexity of $O(m^2n)$ and can be found in $O(m^2n\log m)$ time.*

References

[1] Mikhail J. Atallah. Dynamic computational geometry (preliminary version). In *FOCS*, pages 92–99, 1983.

[2] Aritra Banik, Bhaswar B. Bhattacharya, and Sandip Das. Minimum enclosing circle of a set of fixed points and a mobile point. In *Proceedings of the Workshop on Algorithms and Computation*, New Delhi,India, 2011.

[3] Julien Basch, Leonidas J. Guibas, Craig Silverstein, and Li Zhang. A practical evaluation of kinetic data structures. In *Symposium on Computational Geometry*, pages 388–390, 1997.

[4] Steven Bitner, Yam Ki Cheung, and Ovidiu Daescu. Minimum separating circle for bichromatic points in the plane. In *ISVD*, pages 50–55, 2010.

[5] Erik Demain, Sarah Einsenstat, Leonidas Guibas, and Andre Schulz. Kinetic minimum spanning circle. In *Proceedings of the Fall Workshop on Computational Geometry*, New York, NY, USA, 2010.

[6] S. Fisk. Separating point sets by circles, and the recognition of digital disks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 554–556, July 1986.

[7] N. Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

[8] J. O'Rourke, S. Kosaraju, and N. Megiddo. Computing circular separability. *Discrete Computational Geometry*, 1:105–113, 1986.

[9] Zahed Rahmati and Alireza Zarei. Combinatorial changes of euclidean minimum spanning tree of moving points in the plane. In *CCCG*, pages 43–45, 2010.

[10] Thomas Roos. Voronoi diagrams over dynamic scenes. *Discrete Applied Mathematics*, 43(3):243–259, 1993.