Kinetic Minimum Spanning Circle

Erik D. Demaine* Sarah Eisenstat* Leonidas J. Guibas† André Schulz‡

Abstract

In this paper, we present a kinetic data structure for calculating the minimum spanning circle of a moving point set. Our data structure generates $O(n^{3+\varepsilon})$ events for n points with polynomial motion of fixed degree. In addition, we show a lower bound of $\Omega(n^2)$ on the number of events that can be triggered by a set of points with linear motion functions. As a result of this, we show that our data structure has efficiency $O(n^{1+\varepsilon})$, the best that can be achieved for any data structure based on Delaunay triangulations.

1 Introduction

Kinetic data structures [5] were developed to handle queries on data that change in a regular way over time. For example, consider a set of n points moving at fixed velocities along the real line. With no preprocessing, it takes $\Theta(n)$ time to calculate the minimum value at time t. If we then want to compute the maximum at time $t+\varepsilon$, it would be inefficient to perform $\Theta(n)$ work to get the same answer. Kinetic data structures take advantage of the continuity of the motion to create more efficient data structures.

A kinetic data structure has two components: a core data structure which can be used to answer queries for the current timestamp; and a set of *certificates* which prove that the core data structure was correctly constructed. A certificate can be any boolean function of time which has $O(1)$ times when it switches from true to false. Given a certificate, we need to be able to compute the next time at which it will *fail* — that is, when it will switch from true to false. If we know this time for every certificate, then we know that the core data structure will not change until we reach the minimum failure time. When this time is reached, it triggers an update to the core data structure known as an *event*. There are two types of events: an *external event* is any event that affects the results of a query, and an *internal event* is any event that is not external.

Kinetic data structures have several measures of efficiency. The *responsiveness* of a kinetic data structure is the cost of updating the core data structure and its certificates when a certificate failure occurs. The *efficiency* is the ratio between the total number of events and the number of external events. The *locality* is the maximum number of certificates that a single point may be involved in. The *compactness* is the total number of certificates.

Kinetic data structures have been constructed for a variety of extent problems, such as convex hull [4], minimum box [2], and diameter [2]. In this paper, we present a data structure for another extent measure: the minimum spanning circle. The *minimum spanning circle* of a point set S is the smallest circle containing all points in S.

*MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA, {edemaine,seisenst}@mit.edu

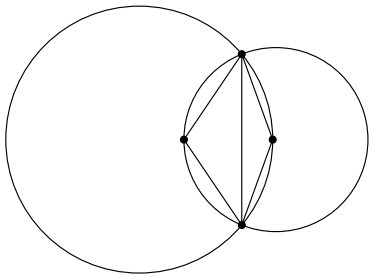
†Department of Computer Science, Stanford University, Stanford, CA 94305, USA, guibas@cs.stanford.edu

‡Institut für Mathematische Logik und Grundlagenforschung, Universität Münster, andre.schulz@uni-muenster.de

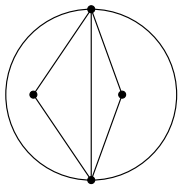
2 Relation to Delaunay

The *farthest-point Delaunay triangulation*, or *FPDT*, of a set of points S is the dual of the *farthest-point Voronoi diagram* of those points — a diagram where each cell consists of all of the points on the plane which are farthest from a particular point in S. Equivalently, the FPDT is a triangulation of the convex hull of S such that the circumcircle of each triangle $\triangle ABC$ contains each point $D \in S$.

The survey "Kinetic Data Structures: A State of the Art Report" [5] presents a sketch of a kinetic data structure that attempts to use this property of the FPDT to calculate the minimum spanning circle. Unfortunately, it is insufficient to consider only the circumcircles of the triangles in the FPDT when calculating the minimum spanning circle. A counterexample can be seen in Figure 1.



(a) FPDT with circumcircles.



(b) The minimum spanning circle.

Figure 1: An example where the minimum spanning circle is not a circumcircle of a triangle in the FPDT.

However, we can still use the FPDT to determine more about the MSC. We define a new concept: the *diameter circle* of two points A and B is the circle centered at $(A+B)/2$ with a radius of $\left|\overline{AB}\right|/2$. In other words, it is the circle such that A and B lie on the circle and on the same diameter.

Lemma 1. *If the FPDT of a set of points includes at least one acute triangle $\triangle ABC$, then the circumcircle of $\triangle ABC$ is the minimum spanning circle.*

Lemma 2. *The FPDT of a set of points contains at most one acute triangle.*

Lemma 3. *If all triangles in the FPDT of a set of points are right or obtuse, then the diameter circle of the longest edge in the FPDT is the minimum spanning circle.*

Lemma 4. *If all triangles in the FPDT of a set of points are right or obtuse, then the diameter circle of diameter in the point set is the minimum spanning circle.*

3 Data Structure

From Lemmas 1 and 4, we know that we must consider two cases: either the FPDT contains an acute triangle, in which case the MSC is the circumcircle of that triangle; or all triangles in the FPDT are right or obtuse, in which case the MSC is the diameter circle of the diameter of the point set. These two cases naturally lead us to break down the data structure into these four pieces:

1. **Kinetic convex hull.** The FPDT is a triangulation of the convex hull, so we maintain the convex hull with the data structure from [4]. This data structure updates $O(\log n)$ certificates when an event occurs, stores a total of $O(n)$ certificates, and generates $O(n^{2+\varepsilon})$ events.

2. **Kinetic farthest-point Delaunay triangulation.** As sketched in [5], we can maintain the FPDT using local constraints. If

$\triangle ABC$ and $\triangle BAD$ are a pair of adjacent triangles in the FPDT, then D must be contained in the circumcircle of $\triangle ABC$. If D leaves the circumcircle of $\triangle ABC$, then the FPDT can be repaired by removing the edge $\overline{AB}$ from the triangulation and swapping in the edge $\overline{CD}$ [5]. Therefore, $O(1)$ certificates are updated when an event occurs. We store $O(1)$ certificates for each of the $O(n)$ triangles, for a total of $O(n)$ certificates. In total, the FPDT generates $O(n^{3+\varepsilon})$ events [1, 3].

3. **Kinetic diameter circle.** We can use the efficient data structure presented in [2] to maintain the diameter of the point set That data structure has "the same overall kinetic performance as the convex hull," according to [5], which means that it updates $O(\log n)$ certificates per event, uses $O(n)$ certificates, and generates $O(n^{2+\varepsilon})$ events in total.

4. **Acuteness certificates.** If A, B, and C are points, then $(A - B) \cdot (C - B)$ is proportional to $\cos(\angle ABC)$. Therefore it will be positive if $\angle ABC$ is acute, negative if $\angle ABC$ is obtuse, and 0 if $\angle ABC$ is right. If we multiply this quantity together for each angle in the triangle, we get a polynomial of fixed degree that allows us to evaluate the acuteness of the triangle.

   We keep one certificate for each triangle in the FPDT. If there is an acute triangle, we store it and keep a certificate of its acuteness. For all other triangles, we keep a certificate of non-acuteness. When a certificate of acuteness fails, we record the fact that there is no acute triangle. When a certificate of non-acuteness fails, we record the triangle which has become acute.

   This portion of the data structure changes $O(1)$ certificates when an event occurs. We store one certificate for each triangle, so the total number of certificates is $O(n)$. To bound the number of events, we note that there are $O(n^3)$ possible triangles, each of which can generate $O(1)$ acuteness events.

The maximum number of certificates that must be updated when processing an event is $O(\log n)$. The insertion and deletion of these events in our priority queue gives us a bound of $O(\log^2 n)$ for responsiveness. The compactness for each component data structure is $O(n)$, so the total compactness is also $O(n)$. The total number of events generated by all data structures is $O(n^{3+\varepsilon})$. Using the lower bound from Section 4, we can calculate the overall efficiency of the data structure:

$$\frac{O(n^{3+\varepsilon})}{\#\text{ of external events}} = O(n^{1+\varepsilon})$$

Unfortunately, this data structure is not very local. First, note that the FPDT of a set of points may be a fan, with one point belonging to a large number of triangles. Hence, it is possible for one point to end up in $\Theta(n)$ certificates. Second, even if there were a more local way to store the FPDT, the diameter data structure which we rely on is also not local, allowing up to $\Theta(n)$ certificates to depend on a single point.

4 Lower Bound

To evaluate the efficiency of a kinetic data structure, we wish to find a lower bound on the number of external events. Our lower bound closely follows the lower bound for kinetic diameter from [2].

Say that we have a point whose position is defined by the following linear equations:

$$x_{\phi,\alpha}(t) = (1-t) \cdot \cos(\phi) + t \cdot \cos(\phi + \alpha)$$
$$y_{\phi,\alpha}(t) = (1-t) \cdot \sin(\phi) + t \cdot \sin(\phi + \alpha)$$

If we confine the time to $[0, 1]$, then the point is moving along a chord of the unit circle, where the chord spans an angle of size α. In polar coordinates, this becomes:

$$r_{\phi,\alpha}(t) = \sqrt{(1-t)^2 + t^2 + 2t(1-t)\cos(\alpha)}$$
$$\theta_{\phi,\alpha}(t) = \phi + \frac{\alpha}{2} + \tan^{-1}\left((2t-1)\tan\left(\frac{\alpha}{2}\right)\right)$$

Because $r_{\phi,\alpha}(t)$ has no dependence on ϕ, a set of points with the same value of α can be used to simulate circular motion.

We divide up our points into two sets $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$. Our goal will be to pick P and Q so that each pair p_i and q_j forms a line with the origin at a unique time $t \in [0, 1]$. At that time, we want the diameter circle of p_i and q_j to be the MSC. To that end, we set the Cartesian coordinates of each p_i to be $\left(x_{(\pi+i\delta),\alpha}(t), y_{(\pi+i\delta),\alpha}(t)\right)$. We then set the polar coordinates of each q_j to be:

$$r_{q_j}(t) = \frac{2j}{n} \cdot t + 1 - \left(\frac{j}{n}\right)^2$$
$$\theta_{q_j}(t) = \frac{\alpha}{2} + \tan^{-1}\left(\left(\frac{2j}{n} - 1\right)\tan\left(\frac{\alpha}{2}\right)\right)$$

For sufficiently small δ, this ensures that each p_i will be directly opposite q_j in a small window around the time $t = j/n$. During that window of time, q_j will be at a distance of $\approx 1 + (j/n)^2$ from the origin. For sufficiently small α, this choice of points will ensure that all other points will lie within the diameter circle of p_i and q_j at the time when p_i and q_j are directly across the origin from each other.

5 Open Problems

The efficiency of this data structure is $O(n^{1+\varepsilon})$. This bound is dominated by the FPDT, so any data structure which uses the FPDT to compute the MSC will not improve the efficiency. A major open problem is whether the FPDT (or the nearest-point Delaunay triangulation) incurs $o(n^{3+\varepsilon})$ events. To improve the efficiency of MSC, we must either make progress on Delaunay triangulations, or find a way to maintain the MSC without the FPDT.

In addition, it may be possible to use similar methods to solve related problems, such as computing the smallest enclosing sphere in three dimensions (or the smallest enclosing hypersphere in higher dimensions).

6 Acknowledgements

This work was initiated at the open-problem session organized around the MIT class 6.851: Advanced Data Structures, held in Spring 2010. We thank the other participants of that session for providing ideas and a stimulating environment.

References

[1] P. K. Agarwal, M. Sharir, and P. Shor. Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences. *Journal of Combinatorial Theory, Series A*, 52(2):228 – 274, 1989.

[2] Pankaj K. Agarwal, Leonidas J. Guibas, John Hershberger, and Eric Veach. Maintaining the extent of a moving point set. In *WADS '97: Proceedings of the 5th International Workshop on Algorithms and Data Structures*, London, UK, 1997. Springer-Verlag.

[3] Gerhard Albers, Joseph S.B. Mitchell, Leonidas J. Guibas, and Thomas Roos. Voronoi diagrams of moving points. *Internat. J. Comput. Geom. Appl.*, 8:365–380, 1998.

[4] Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. In *SODA '97: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.

[5] Leonidas J. Guibas. Kinetic data structures: a state of the art report. In *Robotics: the Algorithmic Perspective*, Natick, MA, USA, 1998. A. K. Peters, Ltd.