On the minimum color separation circle

Steven Bitner and Yam-Ki Cheung and Ovidiu Daescu*
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75080, USA
stevenbitner@student.utdallas.edu {ykcheung,daescu}@utdallas.edu

September 17, 2009

1 Introduction

Assume we are given two input point sets in $\mathbb{R}^2$, one of size n and the other one of size m. We refer to the two point sets as red set and blue set. We denote the red set by $\mathcal{R}$ and the blue set by $\mathcal{B}$, where $|\mathcal{R}|=n$ and $|\mathcal{B}|=m$. Let $\mathcal{S}$ be the set of circles that enclose all points in $\mathcal{R}$, while minimizing the number of points from $\mathcal{B}$ contained within the same circle. We show how to find the smallest circle $C_{\mathcal{B}}(\mathcal{R})$ in $\mathcal{S}$. We refer to $C_{\mathcal{B}}(\mathcal{R})$ as *the minimum separating circle*. We also give an exact solution for finding the *largest separating circle*, which we call the largest circle in $\mathcal{S}$.

In the case where $\mathcal{R}$ and $\mathcal{B}$ can be completely separated, several results exist in the literature. In [2], Fisk solves the problem of separating $\mathcal{R}$ and $\mathcal{B}$ using the smallest circle in $O(|\mathcal{R}||\mathcal{B}|)$ time and space. This result was improved upon in [3] where the time and space requirements are $O(|\mathcal{R}|+|\mathcal{B}|)$. The problem of separating two sets of polygons in the plane was also solved in linear time [1].

2 Results

We present below an algorithm for finding the smallest separating circle as well as an algorithm for finding the largest separating circle.

Lemma 1. *The center of $C_{\mathcal{B}}(\mathcal{R})$ must lie on an edge of the farthest neighbor Voronoi diagram of $\mathcal{R}$, $FVD(\mathcal{R})$.*

*This research was partially supported by NSF grant CCF-0635013.

Proof. We know that $C_{\mathcal{B}}(\mathcal{R})$ is either the minimum enclosing circle of $\mathcal{R}$, $C(\mathcal{R})$, or passes through two points in $\mathcal{R}$ and one point in $\mathcal{B}$. Hence, $C_{\mathcal{B}}(\mathcal{R})$ must pass through at least two red points. Given the fact that the locus of the centers of enclosing circles of $\mathcal{R}$ through two red points defines an edge of $FVD(\mathcal{R})$ and the center of an enclosing circle of $\mathcal{R}$ passes through three red points and defines a vertex of $FVD(\mathcal{R})$, the proof follows. □

By applying *Lemma* 1, we can compute $C_{\mathcal{B}}(\mathcal{R})$ by executing a simple sweeping procedure on every edge of $FVD(\mathcal{R})$. For every edge e of $FVD(\mathcal{R})$, let q_i and q_j be the points in $\mathcal{R}$ defining e. That is, for any point $c \in e$, the smallest enclosing circle of $\mathcal{R}$ centered at c passes through q_i and q_j. We index these two points such that the directed line segment $\overline{q_iq_j}$ divides the enclosing circle into two regions and the smaller region A is on the left side of $\overline{q_iq_j}$, while the larger region B is on the right side of $\overline{q_iq_j}$.

We initialize our sweeping algorithm on e by constructing an enclosing circle C of $\mathcal{R}$, which passes through q_i and q_j with minimum area. Let $c \in e$ be the center of C. Notice that c is the nearest point on e to q_i and q_j. c is almost always a vertex of $FVD(\mathcal{R})$ unless $C(\mathcal{R})$ is defined by q_i and q_j. To handle the degenerate case, we can either run the sweeping procedure twice in both directions along e or divide e into two separate edges by treating c as a vertex. We expand C by sweeping c along e while still passing through q_i and q_i. See Figure 1 for an illustration. We define a point $p_e \in e$ as an event point if when c sweeps through p_e, circle C sweeps through a blue point or a red point. We compute the num-

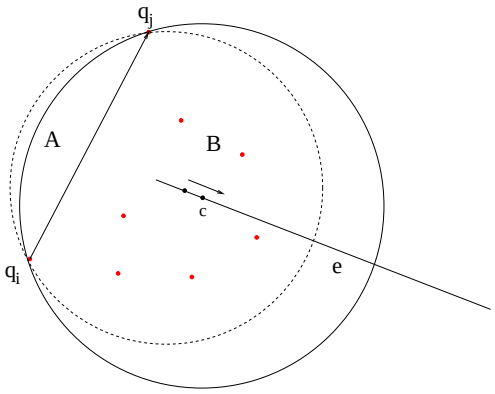

Figure 1: Expand the enclosing circle by sweeping its center c along e.

ber of blue points enclosed by C at each event point. The sweeping procedure terminates when C sweeps through a red point or all event points on e have been processed and returns the local minimum separating circle centered on e. We note that the overall algorithm can output all global minimum separating circles by maintaining a list of circles.

In order to find the largest separating circle, SEP, we first find the list of all minimum separating circles. With k circles in the list, we start with the first circle and find the points from $\mathcal{B}$ that are contained in that circle. We remove those points from $\mathcal{B}$ to obtain $\mathcal{B}_{-1}$. This leaves us with two sets of points that can be purely separated without any overlap. The maximum separating circle between $\mathcal{R}$ and $\mathcal{B}_{-1}$ can be found in $O((n+m)\log(n+m))$ time and this is optimal [3]. We then repeat this process for each of the k smallest circles running the cited algorithm on each of the sets $\mathcal{B}_{-2}$, $\mathcal{B}_{-3}$ ... $\mathcal{B}_{-k}$ and obtain our answer.

Correctness: C remains as an enclosing circle of $\mathcal{R}$ during the process, due to fact that q_i and q_j are the farthest red points to c. By executing the sweeping procedure on each edge, we compute the number of blue points enclosed by every enclosing circle of $\mathcal{R}$ defined by two red points and one blue point as well as $C(\mathcal{R})$.

The proof of the largest circle problem follows from the stated reference for the algorithm and the fact that we analyze all possible sets of blue points that are interior to a minimum circle.

Time: For the minimum separating circle problem, the construction of $FVD(\mathcal{R})$ takes $O(n\log n)$ time. Observe that during the sweeping procedure on edge $e \in FVD(\mathcal{R})$, region A contracts while region B expands. If c sweeps through an event point defined by a blue point q_k on the left (resp. right) side of $\overline{q_iq_j}$, q_k is removed from (resp. added to) C. Hence, the number of blue points enclosed by C at the current event point can be computed in constant time given the number of blue points enclosed by C at the previous event point. The number of event points on each edge is $O(m)$. The sweeping procedure takes $O(m\log m)$ time for each edge. The overall algorithm takes $O(hm\log m + n\log n)$ time, where $h = O(n)$.

For the largest separating circle problem, we must analyze the sets after removing the points contained in the minimum separating circle. Removing these points can be done in $O(m)$ time for each of the k minimum separating circles. Then, for each of the k circles, we use $O((n+m)\log(n+m))$ time for finding the largest separating circle, resulting in an overall running time of $O(k(n+m)\log(n+m))$. This running time is in addition to the running time needed to find the minimum separating circles. So, the overall running time of the largest separating circle algorithm is $O(nm\log m + k(n+m)\log(n+m) + km)$.

Lemma 2. *There are n minimum separating circles in the worst case.*

Proof. Omitted. □

Theorem 1. *Given a red set $\mathcal{R}$ and a blue set $\mathcal{B}$ of points in $\mathbb{R}^2$, one can compute the smallest separating circle, $C_{\mathcal{B}}(\mathcal{R})$, in $O(hm\log m + n\log n)$ time and $O(n+m)$ space where h is the complexity of $FVD(\mathcal{R})$ and compute the largest enclosing circle in $O(nm\log m + k(n+m)\log(n+m))$ time and $O(n+m)$ space, where k is the number of smallest separating circles.*

References

[1] J.-D. Boissonnat, J. Czyzowicz, O. Devillers, and M. Yvinec. Circular separability of polygons. *Algorithmica*, 30(1):67–82, 2001.

[2] 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.

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