International Journal of Computer and Information Sciences, Vol. 12, No. 5, 1983

Computing Largest Empty Circles with Location Constraints[1,2]

Godfried T. Toussaint[3]

Received May 1983; revised August 1983

Let $Q = \{q_1, q_2, ..., q_n\}$ be a set of n points on the plane. The largest empty circle (LEC) problem consists in finding the largest circle C with center in the convex hull of Q such that no point $q_i \in Q$ lies in the interior of C. Shamos recently outlined an $O(n \log n)$ algorithm for solving this problem.[9] In this paper it is shown that this algorithm does not always work correctly. A different approach is proposed here and shown to also result in an $O(n \log n)$ algorithm. The new approach has the advantage that it can also solve more general problems. In particular, it is shown that if the center of C is constrained to lie in an arbitrary convex n-gon, an $O(n \log n)$ algorithm can still be obtained. Finally, an $O(n \log n + k \log n)$ algorithm is given for solving this problem when the center of C is constrained to lie in an arbitrary simple n-gon P, where k denotes the number of intersections occurring between edges of P and edges of the Voronoi diagram of Q and $k \leqslant O(n^2)$.

KEY WORDS: Largest empty circle; facility location; polygons; Voronoi diagram; algorithms; complexity; operations research; computational geometry.

1. INTRODUCTION

Let $Q = \{q_1, q_2, ..., q_n\}$ be a set of n points on the plane and let $CH(Q)$ denote the convex hull of Q. The largest empty circle (LEC) problem consists in finding the largest circle the center of which lies in $CH(Q)$ such that no point of Q lies in the interior of the circle. It is clear that the location of the center must be constrained. Otherwise we simply set the center at infinity and obtain an infinitely large circle. It is assumed that the points in Q are specified in terms of their Cartesian coordinates and they lie in

[1] Research suported by NSERC Grant No. A-9293 and FCAC Grant No. EQ-1678.
[2] Technical Repot No. SOCS 83.4.
[3] School of Computer Science, McGill University, Montreal, Quebec, Canada, H3A 2K6.

0091-7036/83/1000-0347$03.00/0

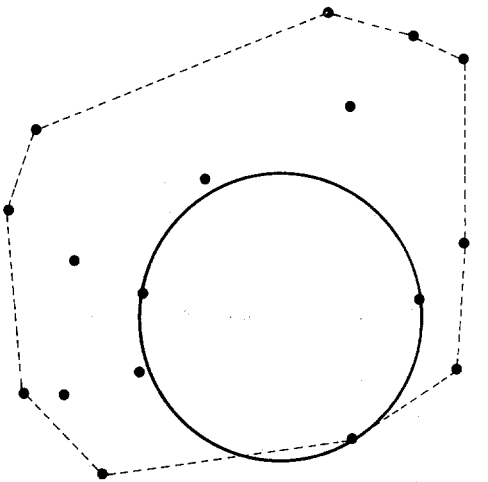

Fig. 1. A set of points, their convex hull, and the largest empty circle.

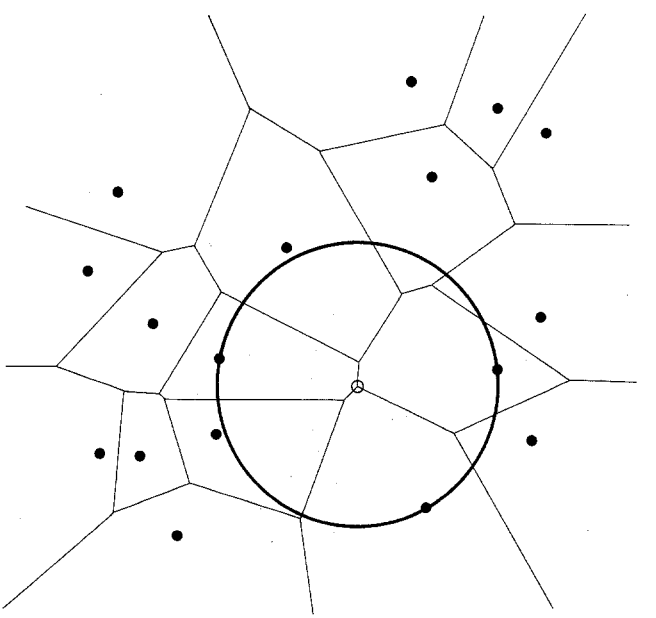

Fig. 2. Illustrating the case where the solution circle is centered on a vertex of the Voronoi diagram.

general position in the sense that no three are collinear and no four are cocircular. Figure 1 illustrates a set of points and the solution circle. Until recently, the best known algorithm for solving this problem required $O(n^3)$ time.[4] More recently, Shamos[9] outlined an $O(n \log n)$ algorithm based upon computing the Voronoi diagram of Q, $V(Q)$. It is assumed here that the reader is familiar with the Voronoi diagram properties discussed in Ref. 9. For example, the Voronoi diagram has at most $m = 2n - 4$ Voronoi vertices (V-vertices) $V = \{v_1, v_2, ..., v_m\}$ and each v_i is the circumcenter of three points of Q determined by a Delaunay triangle in the dual of the Voronoi diagram of Q. Furthermore, these circles are empty. If all v_i in V lie in $CH(Q)$, then one of these vertices is the center of the solution circle. However, not all the V-vertices of $V(Q)$ need lie in $CH(Q)$, in which case the solution circle may be a V-vertex or it may lie at the intersection point of an edge of $V(Q)$ and an edge of $CH(Q)$, as will be shown in Section 2.2. Figure 2 illustrates the Voronoi diagram of a set of points where the solution

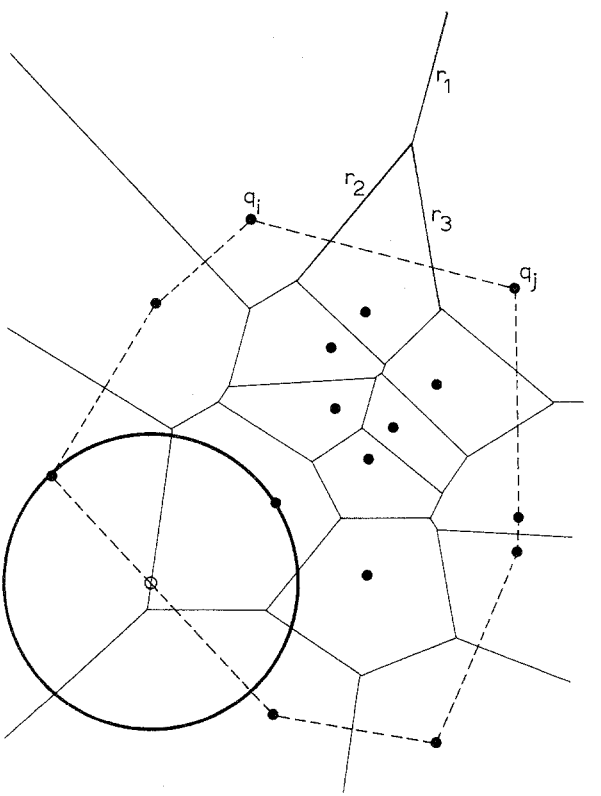

Fig. 3. The case where the solution circle is not centered on a Voronoi vertex.

circle is centered on a V-vertex. Figure 3 illustrates a situation where the solution center is not a V-vertex. Shamos' algorithm consists of two parts: the first part computes the Voronoi diagram of Q and the magnitudes of all the circles centered at V-vertices that lie in $CH(Q)$, and the second part looks for intersection points of V-edges with $CH(Q)$ edges and computes the magnitudes of the empty circles centered at these intersection points (see Fig. 3). Now each CH-edge, such as q_iq_j in Fig. 3, determines an unbounded V-edge r_1 collinear with the perpendicular bisector of the CH-edge. In Ref. 9, Shamos claims that each CH-edge, such as q_iq_j, either intersects r_1 or the two V-edges adjacent to r_1 (r_2 and r_3) in Fig. 3, i.e., that a CH-edge intersects at most two V-edges. Drawing on this, the second part of the algorithm checks each CH-edge with the corresponding one or three (as the case may be) V-edges for intersection points and computes the corresponding empty circles. Since the Voronoi diagram can be computed in $O(n \log n)$ time and for each CH-edge the circles are computed in $O(1)$ time, the algorithm has a total running time of $O(n \log n)$.

Unfortunately, the above claim is false and such an algorithm may miss some circles; indeed, it may miss the largest empty circle itself and thus fail. One counterexample to the above algorithm is illustrated in Fig. 4. Note that

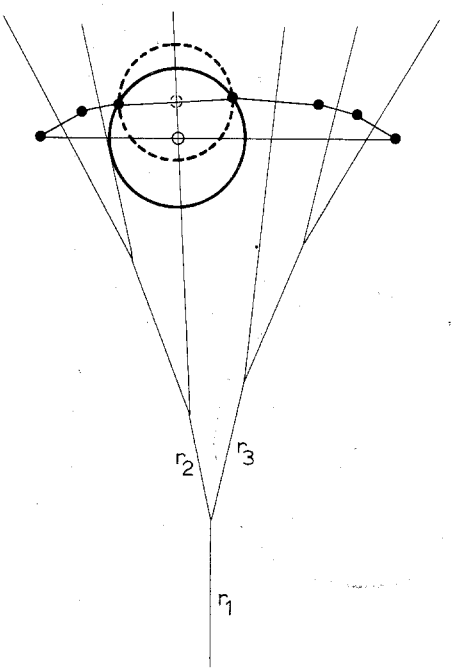

Fig. 4. A counterexample to the algorithm of Shamos.

all V-vertices lie outside $CH(Q)$, r_2 and r_3 do not intersect the corresponding CH-edge, and indeed a CH-edge may intersect more than two V-edges; in fact, it may intersect $O(n)$ V-edges. In Fig. 4, the LEC (solid curve) is not detected and instead the algorithm exits with a suboptimal solution (dashed curve). These examples reopen the question of whether an $O(n \log n)$ upper bound exists for this problem. Figure 4 suggests that we may be able to efficiently search the Voronoi diagram starting at every unbounded V-edge until all intersections of V-edges with the corresponding CH-edge are found. In fact, earlier work by Shamos[10,11] suggests a depth-first and breadth-first search, respectively, for solving this problem. However, it is not clear what the complexity of such a step would be. Although in Refs. 10 and 11, Shamos claims this step can be done in $O(n)$ time, no proofs are given and the simplistic arguments are not convincing. For example, in Ref. 10 it is claimed that each edge of $V(Q)$ can intersect at most one edge of $CH(Q)$, but, as Fig. 4 illustrates, this is not necessarily true. Also in Ref. 11, it is claimed that in a breadth-first search no V-edge will be examined more than once. However, Fig. 5 illustrates a situation where a

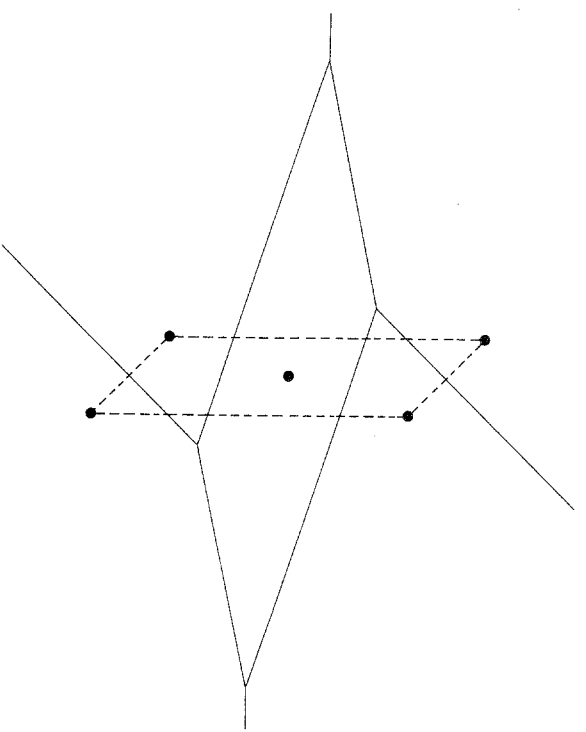

Fig. 5. An example where breadth-first search examines some Voronoi edges twice.

breadth-first search examines some V-edges twice. Rather than try to prove the correctness of the approaches taken in Refs. 10 and 11, in this paper we propose a new approach to this problem that not only yields an algorithm with $O(n \log n)$ running time, but an algorithm that is simpler, admits an easy proof of correctness, and extends to more interesting and applicable generalizations of the LEC problem.

One interpretation of the LEC problem is to let Q denote a set of cities in a country. If it is desired to place an installation such as a nuclear power plant in a location that maximizes the distance from the power plant to the nearest city, then the location is the center of the largest empty circle. However, the convex hull of the cities is not a very realistic constraint on the location of a nuclear power plant. The $CH(Q)$ may intersect other countries and the resulting solution may call for placing the plant in a country that may not grant permission. Alternately, the country in which the cities lie may have a large peninsula protruding from $CH(Q)$. The optimal solution may indeed call for placing the plant on the peninsula, and thus constraining the location to $CH(Q)$ would yield a suboptimal solution. Finally, one may simple wish to constrain the location to some region within the country, such as a desert. Furthermore, this region may not contain a V-vertex and may not intersect V-edges, in which case the CH constraint fails completely. In this paper, it is shown that if the center is constrained to lie in an arbitrary convex polygon with n sides, an $O(n \log n)$ algorithm can still be obtained. Finally, an $O(n \log n + k \log n)$ algorithm is given for solving this problem when the center of the circle is constrained to lie in an arbitrary simple n-sided polygon $P = (p_1, p_2, ..., p_n)$, where k denotes the number of intersections occurring between edges of P and edges of $V(Q)$ and $k \leqslant O(n^2)$. It is assumed that P is given as an array of vertices, in clockwise order, along with their Cartesian coordinates, and that it is in *standard form*, i.e., the vertices are distinct and no three consecutive vertices are collinear. A pair of vertices $p_i p_{i+1}$ defines an edge of the polygon, where $i = 1, 2, ..., n$ and $p_{n+1} = p_1$. Clearly the number of vertices of P need not equal to number of points in Q, but such an assumption will simplify the complexity of notation. If the reader wishes to observe the role played by each, it is straightforward to repeat the complexity analysis when P and Q have different cardinalities.

2. NEW ALGORITHMS

2.1. The Convex Hull Location Constraint

We first consider the LEC problem treated by Shamos[9] and Dasarathy and White,[4] where the center is constrained to lie in $CH(Q)$.