Largest Empty Circle Algorithm Using Voronoi Diagrams

Theoretical Description

The **Largest Empty Circle (LEC)** problem is defined on a set of points $P = \{p_1, p_2, \dots, p_n\}$ in the plane. The goal is to find the largest circle centered inside the **convex hull** of P that contains no points of P in its interior. Formally, the LEC is the circle with center q inside the convex hull of P such that the minimum distance from q to any point in P is maximized.

The LEC problem is closely tied to the **Voronoi diagram** of the point set P. The Voronoi diagram partitions the plane into convex cells, each containing exactly one point of P, such that every location inside a cell is closer to its associated point than to any other.

Key properties relevant to the LEC problem:

- The center of the LEC lies either at a **Voronoi vertex** inside the convex hull of P or on the intersection of a **Voronoi edge** with the convex hull boundary.
- Voronoi vertices are equidistant to three or more points of P and represent local maxima of the minimum distance to points in P.
- Points not on Voronoi vertices or edges cannot be centers of the LEC because they are closer to one particular point in P and thus cannot maximize the minimum distance.

Therefore, the LEC can be found by:

1. Computing the Voronoi diagram $V(P)$ of the point set P.
2. Computing the convex hull $CH(P)$ of P.
3. Identifying all Voronoi vertices inside $CH(P)$.
4. Identifying all intersection points between Voronoi edges and $CH(P)$.
5. For each candidate point (Voronoi vertex or intersection), computing the radius as the minimum distance to the points in P.
6. Selecting the candidate with the largest radius.

Algorithm

Input

- Set of points $P = \{p_1, p_2, \dots, p_n\}$.

Output

- Center q^* and radius r^* of the largest empty circle inside $CH(P)$.

Steps

1. Compute the **Delaunay triangulation** $DT(P)$ of the points in P. The Voronoi diagram $V(P)$ is the dual of $DT(P)$.
2. Compute the **Voronoi diagram** $V(P)$ by dualizing $DT(P)$.
3. Compute the **convex hull** $CH(P)$ of the points in P.
4. Initialize an empty list of candidate centers.
5. For each **Voronoi vertex** v:
   - If v lies inside $CH(P)$, add v to the candidate list.
6. For each **Voronoi edge** e:
   - Compute intersections of e with edges of $CH(P)$.
   - Add all intersection points inside $CH(P)$ to the candidate list.
7. For each candidate center c:
   - Compute radius $r_c = \min_{p \in P} \|c - p\|$.
8. Select the candidate c^* with the maximum radius $r^* = \max_c r_c$.
9. Return (c^*, r^*) as the center and radius of the largest empty circle.

Pseudocode

```
function LargestEmptyCircle(P):
    DT = DelaunayTriangulation(P)
    V = VoronoiDiagram(DT)
    CH = ConvexHull(P)

    candidates = []

    for vertex v in V.vertices:
        if v inside CH:
            candidates.append(v)

    for edge e in V.edges:
        intersections = intersect(e, CH.edges)
        for point in intersections:
            if point inside CH:
                candidates.append(point)

    max_radius = 0
    best_center = None

    for c in candidates:
        r = min_distance(c, P)
        if r > max_radius:
            max_radius = r
            best_center = c

    return best_center, max_radius
```

Complexity Analysis

- **Delaunay triangulation and Voronoi diagram construction:** Both can be computed in $O(n \log n)$ time using well-known algorithms.
- **Convex hull computation:** Can be computed in $O(n \log n)$ time (e.g., Graham scan or Jarvis march).
- **Candidate identification:** There are $O(n)$ Voronoi vertices and edges. Checking whether a vertex lies inside the convex hull and computing edge intersections can be done in $O(nh)$, where h is the number of hull edges.
- **Radius computation:** For each candidate (at most $O(n+nh)$), computing the minimum distance to all points takes $O(n)$, resulting in $O(n^2)$ worst-case time. However, spatial data structures or pruning can improve practical performance.
- **Overall complexity:** Dominated by $O(n \log n + nh + n^2)$ in the worst case. For typical datasets where h is small and spatial indexing is used, practical runtime approaches $O(n \log n)$.

Summary

The Largest Empty Circle problem leverages the geometric duality between Voronoi diagrams and Delaunay triangulations. By restricting the search for the LEC center to Voronoi vertices inside the convex hull and intersections of Voronoi edges with the hull, the problem reduces to evaluating a finite set of candidate points. The algorithm is efficient, well-founded in computational geometry, and widely used in applications such as facility location, robotics, and spatial analysis.

References:

- Megan Schuster, "The Largest Empty Circle Problem," Swarthmore College, 2008. `https://www.cs.swarthmore.edu/ adanner/cs97/s08/papers/schuster.pdf`
- de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O., *Computational Geometry: Algorithms and Applications*, 3rd Edition, 2008.
- Preparata, F.P., and Shamos, M.I., *Computational Geometry: An Introduction*, 1985.