Circumcircular Range Searching in Higher Order Delaunay Triangulations

Ahmad Biniaz*

Range searching is one of the classical problems in computational geometry. Let P be a set of n points given in d-dimensional real space, $\mathbb{R}^d$, and a family of subsets of $\mathbb{R}^d$ called *ranges* is considered. The goal is to preprocess P into a data structure so that for a given query range Q, the points in $P \cap Q$ can be counted or reported efficiently. The simplest among nonlinear ranges are circular discs in the plane. A *circular range searching* query has the form $query(q, r)$, and we have to report the points of P which are lying in the sphere with the center q and radius r.

The most of time optimal solutions of the 2-dimensional circular range searching problem use *higher order Voronoi diagrams* (HOVD) [7]. Bentley [2] presented a technique which extracts the points of the Voronoi cell containing q in the m-order VD of P for $m = 2^0, 2^1, 2^2, ...$ consecutively. It has the query time $O(\log n \log\log n + k)$, and the space complexity of $O(n^3)$, where k is the number of the points P lying in the query disk. Chazelle [3] improved the query time to $O(\log n + k)$ and the space complexity to $O(n(\log n \log\log n)^2)$ by the concept of *filtering search*. Agarwal [1] reduced the space requirement with *compacting* techniques to $O(n \log n)$.

This paper studies the problem of *circumcircular range searching* in 2D triangulations. Let T be a triangulation of the points in P, and a triangle $t \in T$ is given. The problem is to compute all the points of P that fall within the circumcircle of t. This problem frequently arises in *higher order Delaunay triangulations* (HODT) [4, 5, 6].

We are concerned about $Q(n)$, the *query time*; $S(n)$, the size of the data structure; and $P(n)$, the *preprocessing time* of circumcircular range searching algorithms. Our goal is that, the query answering takes time not much larger than the output size, without any preprocessing. We will discuss a technique that is efficient in practice, uses no preprocessing time, no additional storage, and, as a bonus, could not be easier to implement. The proposed technique starts at t, and walks through its neighbors, and then "walk from neighbors to neighbors" in breath-first search (BFS) manner across the triangulation and reports the points belong to the circumcircle of t. This method in its full generality deals with any triangulation. This work, focuses on range searching in Delaunay triangulations.

*Department of Computer Engineering, Azad University of Lamerd, biniaz@cse.shirazu.ac.ir

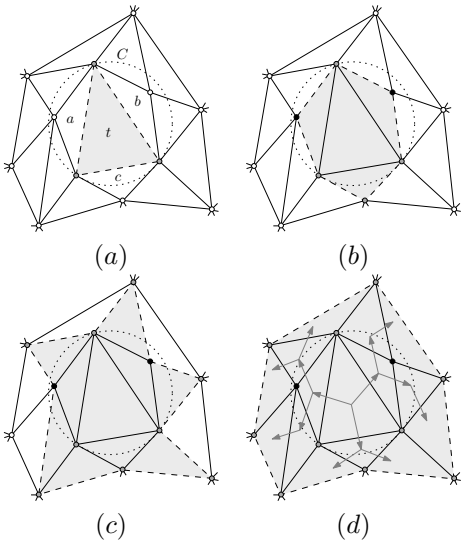

Figure 1: Main idea of the algorithm.

Assume that we are given a query triangle t in a planar triangulation T (Figure 1). Let the circle C with radius r, which centered at q, denotes the circumcircle of t. The search procedure performed beginning at triangle t, visit t and its three vertices (gray points in Figure 1-a). Next, the unvisited triangles adjacent to t are visited in alphabetical order. Therefore, the triangles a, b and c are visited (see Figure 1-b). The three newly appeared vertices, incident to these triangles tested for *in-circle property*. The vertices which satisfy the property are reported (black vertices). The algorithm surrounds the visited triangles by a bordering polygon ($frontier$); plotted by dashed lines in Figure 1. In each iteration, the algorithm visits the unvisited triangles adjacent to the frontier. Thus, in the next iteration the six new triangles adjacent to the frontier are visited and the frontier is updated (see Figure 1-c). The algorithm terminates after all the triangles $\triangle$, with $\triangle \cap C \neq \emptyset$, are visited. So, the frontier is extended in Figure 1-d until surrounds C.

The frontier is a triangulated simple polygon of size f, witch has κ additional steiner points inside. The number of already visited triangles inside the frontier is $\tau = f + 2\kappa - 2$. We denote the maximum size of frontier as $\mathcal{F} = \max\{f\}$, and the number of all visited triangles as $\mathcal{T}$. In other words, $\mathcal{T}$ shows the number of triangles $t' \in T$ such that $t' \cap C$ is not empty; $\mathcal{T} = |T \cap C|$. Thus the time needed to answering a query is $O(\mathcal{T})$.

The proposed method uses the triangulation T as a spanner with $O(n)$ edges and visits all the edges having at least one endpoint not farther from q than r. Thus, it has an expected running time of $O(\delta(n)k)$, where $\delta(n)$ denotes the expected degree of each of k vertices inside C. Dwyer [8] showed that $\delta(n) = O(1)$ for points that chosen uniformly at random in a d-dimensional ball. It is a fair assumption that for Delaunay triangulations we can argue that $\delta(n) = O(1)$, yielding an expected running time of $O(k)$.

Most of HODT applications use HOVD [7] to determine the order of a given triangle. Using our circumcircular range search (CCRS) scheme instead, the improvements of Table 1 can be achieved without any preprocessing time.

Table 1: Comparison of HOVD and CCRS.

Alg.	$Q(n)$	$S(n)$	$P(n)$
HOVD	$O(k + \log n)$	$O(n \log n)$	$O(nk \log n)$
Here	$O(k)$	$O(k)$	$O(1)$

There are some heuristics that use HODT to generate realistic terrains: *flip*, *hull*, *valley*, *valley reduce*, *elevation*, and *local improvement* [5, 6]. All of them use order-$(k+1)$ VD [7] to determine the order of a given triangle. Using the proposed range-search scheme, the improvements of Table 2 can be achieved.

Gudmundsson [4] determine the order k of a given triangulation in $O(\min(n^{3/2} \log^{O(1)} n, nk \log n \log k))$ using HOVD [7]. Using the proposed algorithm instead, this problem can be easily solved in $O(nk)$.

Table 2: Time complexity of algorithms.

Algorithm	Previous Time	New Time
Flip[5]	$O(nk^2 + nk \log n)$	$O(nk^2 + n \log n)$
Hull[5]	$O(nk^2 + nk \log n)$	$O(nk^2 + n \log n)$
Valley[5]	$O(nk \log n)$	$O(nk + n \log n)$
Val. Red.[6]	$O(nk \log n)$	$O(nk + n \log n)$
Elevation[6]	$O(nk^2 + nk \log n)$	$O(nk^2 + n \log n)$
Loc. Imp.[6]	$O(nk + n \log n)$	$O(nk + n \log n)$

Figure 2-up compares the average query time of HOVD and CCRS; the preprocessing time is excluded. The introduced algorithm is very faster and its runtime is independent of n; specially for small values of k and large values of n. Figure 2-down shows the mean number of visited triangles and max size of frontier per query for different values of k. It shows that the number of visited triangles is almost twice the value of k and increase in k, increases the difference between $\mathcal{F}$ and $\mathcal{T}$.

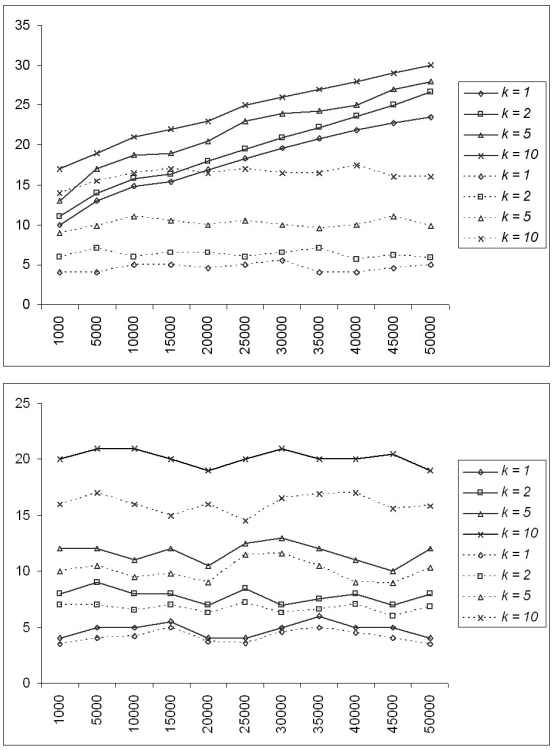

Figure 2: Up: average query times for HOVD (solid) and CCRS (dotted). Down: mean number of visited triangles per query disc (solid) and max size of frontier $\mathcal{F}$ (dotted).

References

[1] P. K. Agarwal, M. Hassen, and T. Leighton. *Solving query retrival problems by compact voronoi diagrams.* In Proc. of 22nd Symp. on Theo. of Comp. (STOC), pp. 331–340. ACM, 1990.

[2] J. L. Bentley and H. A. Maurer. *A note on the euclidean near neighbor searching in the plane.* Inf. Proc. Lett., 8:133–136, 1979.

[3] B. Chazelle, R. Cole, F. P. Preparata, and C. Yap. *New upper bounds for neighbor searching.* Inf. & Cont., 68:105–124, 1986.

[4] J. Gudmundsson, H. Haverkort, and M. van Kreveld. *Constrained higher order delaunay triangulations.* Comp. Geom.: Theo. & App., 30:271-277, 2005.

[5] T. de Kok, M. van Kreveld, and M. Löffler. *Generating realistic terrains with higher-order delaunay triangulations.* Comp. Geom.: Theo. &d App., 36:52-67, 2007.

[6] A. Biniaz and G. Dastghaybifard. *Drainage reality in terrains with higher order delaunay triangulations.* In Adv. in 3D Geoinf. Sys., pp. 199–213, Springer-Verlag, 2008.

[7] E. A. Ramos. *On range reporting, ray shooting and k-level construction.* In Proc. 15th Annu. Symp. on Comp. Geom., pp. 390-399, ACM, 1999.

[8] R. A. Dwyer. *Higher-dimensional Voronoi diagrams in linear expected time.* Disc. & Comp. Geom., 6:343–367, 1991.