CDT

July 2022

1 ALGORITHM

1. Input:
   (a) A cavity given by s and its point sequence $\{p_0, p_1, ..., p_n\}$.
   (b) p_0 and p_n are the endpoints of s.
   (c) Let P be the set of the cavity points. Point repetitions are possible in the point sequence.
   (d) Suppose a counterclockwise order.

2. $o = (a_1, a_2, ..., a_{n-1})$ a random permutation of the cavity points $\{p_1, ..., p_{n-1}\}$.

   Note : Each cavity point can be represented by an arc (Voronoi edge) on the boundary of $VR(s, P \cup s)$. An arc for point p_i is a piece of bisector $J(s, p_i)$. For points $\{p_1...p_{n-1}\}$ these are parabolic arc. For p_0 and p_n the arcs are line segments perpendicular to s. We build nodes that represent the arcs along the boundary of $VR(s, P \cup s)$. Equivalently we build nodes that represent the occurrences of the cavity points along $VR(s, P \cup s)$. During the algorithm we may create additional "auxiliary nodes".

3. **Phase 1** : Delete points in P, 1-by-1 in reverse order while recording their neighbors at the time of deletion. E.g: p_3 neighbors (p_1, p_6). This means that when we will insert p_3 we will insert it between p_1 and p_6 (unless p_6 and p_1 are the same point, in which case we split the node of $p_1 = p_6$). If no auxiliary nodes are present between p_1 and p_6 during insertion then we will create edges (p_1, p_3) and (p_3, p_6).

4. **Phase 2**:
   (a) Insert nodes in the order of o.
   Start with triangle (p_0, a_1, p_n)
   (b) At step i, insert a_i between its two recorded neighbors from phase-1: (r_i, n_i). If r_i and n_i share the same node, i.e., their points are the same point, then we split the node associated with r_i, n_i. Call SPLIT(α, β), where α is the node of r_i, n_i and $\beta = a_i$.

(c) Assume otherwise.
We start at $\alpha = r_i$;
let $\gamma = r_i.next$;
let $\beta = a_i$.
Due to the presence of auxiliary nodes, γ may be $\neq n_i$.
Now check for the following conditions:

i. If the point of β = point of α then simply add $\beta = a_i$ to the input points associated with the node α. Done.
Respectively if β = point of γ.

ii. Now assume points $\beta \neq \alpha \neq \gamma$.
Perform an orientation test to determine if the points (α, β, γ) are oriented ccw or cw.
If ccw, insert the node of β between the nodes of α and γ and create the triangle $T(\alpha, \beta, \gamma)$. This is the normal case like in Chew.
Continue normally: Fix the Delaunay property by potential flips.
Flips = deg(β) (-2).

iii. If cw:

A. determine if the triangle incident to $\alpha\gamma$ is destroyed (or not) by β: Perform inCircle test involving the triangle vertices and β.
Note: Orientation test itself is not enough because cw orientation for normal/original nodes:(α, β, γ) would imply deletion of edge (α, γ) but node γ may not necessarily be an original node—–it could be an auxiliary node. Hence the inCircle test.

B. If not (i.e., the corresponding V. vertex remains/circle does not contain β), there are two possibilities: either β splits α, OR γ is an auxiliary node and β needs to be inserted after γ. In the latter case advance α to γ and γ to next node and repeat the insertion process. In the former case call SPLIT($\alpha\beta$).
Note:Reason for the latter case: If β would have destroyed the triangle then the β arc would have encapsulated the voronoi vertex corresponding to the triangle however that is not the case. The triangle survives hence the voronoi vertex between arc γ and arc α exists. So arc β does not occur between the other two hence $\gamma \neq n_i$. This means that *gamma* is a fabricated node hence an auxiliary node. Thus β must occur after auxiliary node γ.

Examples:

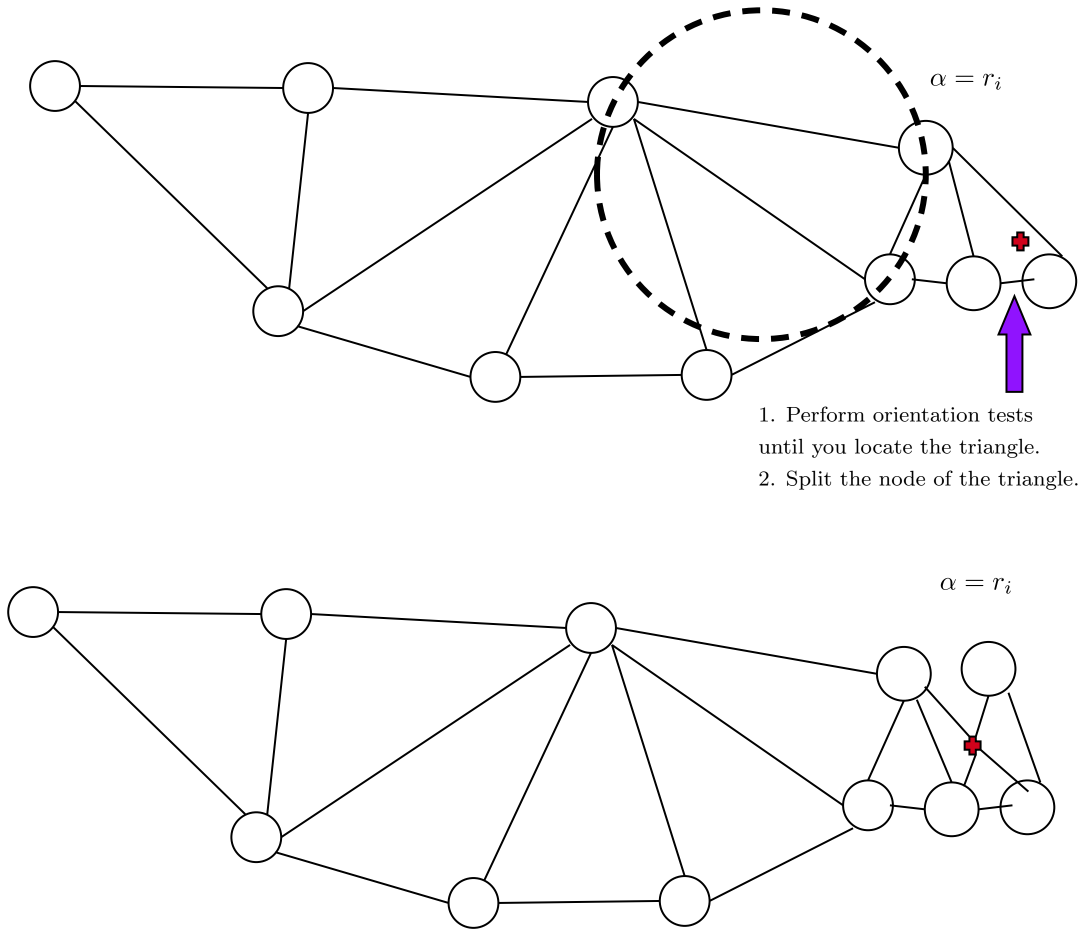

Figure 1: Split case: Triangle survives and it is a split case. As entry and exit points are now both α, all triangles incidental on α should be checked.

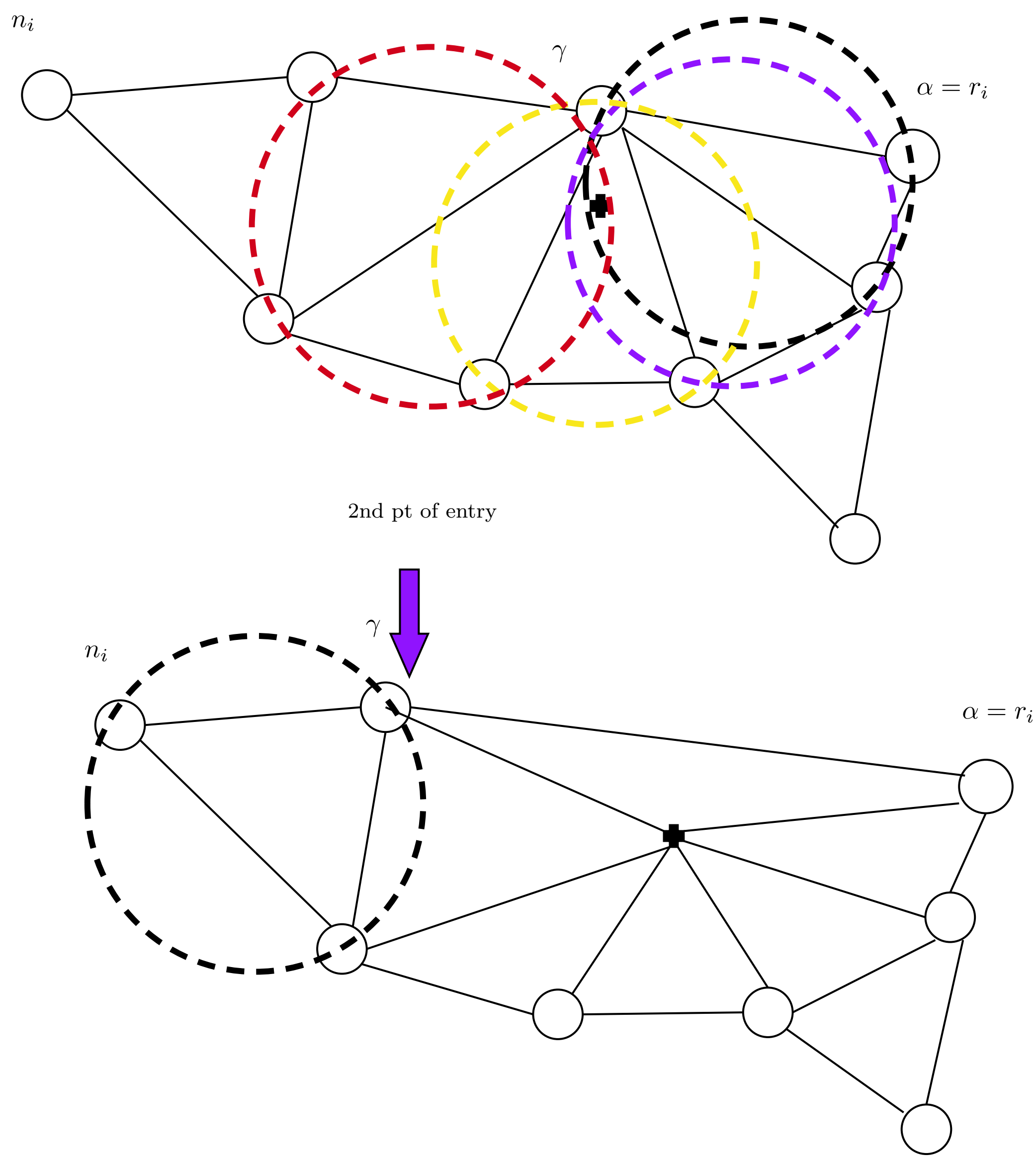

Figure 2: If like in the above figure the triangle containing the point is found, do not stop traversing the faces incident to γ yet; just remember the triangle index. Check the next face. If it survives then start joining edges to β. Else keep traversing the faces until all faces incident to γ gets destroyed. Then go to the triangle index and start joining all nodes except old γ to β. Then delete node old γ. Then triangulate.

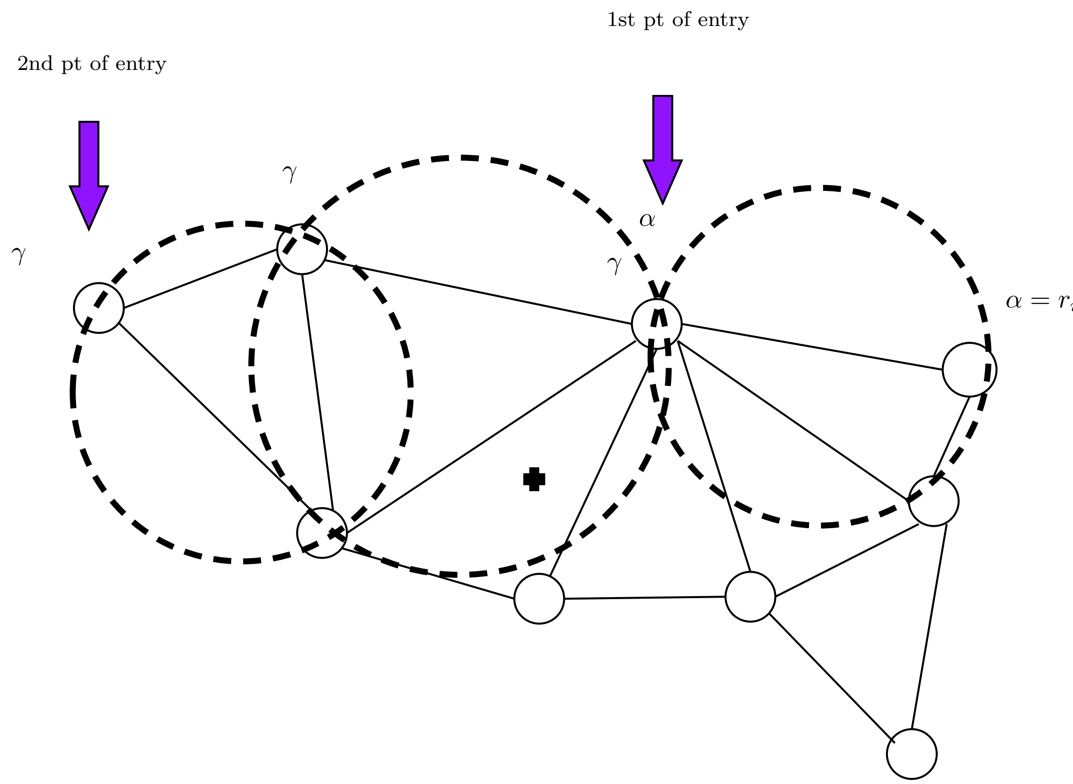

Figure 3: Placement after auxiliary node.

C. To determine which case we have, we perform an inCircle test for $\alpha\beta\gamma$ and s. First assert that the circle $C(\alpha, \beta\gamma)$ is oriented clockwise. If s intersects $C(\alpha, \beta, \gamma)$ we have the split case. Otherwise we have the "advance" case.

   Note: because we have already determined that a V. vertex on $J(\alpha\gamma)$ remains, i.e., s intersects $C(\alpha, \gamma, x)$ while β is out of the circle, there should be no degeneracy issue with the in-circle test involving s, because β should either be to the right or to the left of the projection segments of both alpha and gamma to s. In one case β is "right" of the perpendicular line from α to s, and in the other β is left of the perpendicular line from γ to s. So the two cases should be very well separated.

D. If yes, i.e., the triangle incident to $\alpha\gamma$ is destroyed by β, we have determined that we are inserting β after α. We are trying to compute the entry point of β and it lies between α, β, of which we know α. Now we find γ. Continue visiting

triangles incident to γ, while β destroys them (by in-circle test). Continue until we find a triangle that contains β or a triangle incident to γ that is not destroyed by β or when all triangles incident to γ get destroyed. Once we reach such a triangle, we know β lies to the left of the γ.
In the latter case, γ must be an auxiliary node; delete the node of γ, advance γ to γ.next, and repeat (checking triangles incident to the new-γ until a stopping condition is reached. Once old γ is deleted, α must be connected to the new γ. At the end of this process we either have found a triangle incident to (the potentially new) γ that survives β, or we have found a triangle that contains β.

Note: Before the in-circle test we may perform triple orientation tests to check if β lies in the triangle. Given triangle α, x, γ, compute orientation of α, x, β, x, γ, β and γ, α, β. If all orientations are the same then β lies inside the triangle else it does not. If not, perform the in-circle test.

E. We create edges $\alpha\beta$ and $\alpha\gamma$ (note γ need not be the original one). We have also found several diagonals and triangles that get deleted. Certainly the original diagonal $\alpha\gamma$ is found to be deleted and potentially more diagonals. It remains to find the triangle that contains the point of β (if we have not already found it). Any triangle intersected by $\alpha\beta$ and $\gamma\beta$ must get deleted. We determine any such triangle by continuing our sequential search. We can use segments $\alpha\beta$ and $\gamma\beta$ to guide the search and/or orientation tests. After identifying a number of triangles to be deleted we identify a triangle which contains the point of β. (Such triangle exists and is connected to the original $\alpha\gamma$ by deleted triangles).
F. Once we have located the triangle Δ where the point of β belongs, we connect β to the vertices of the triangle and we propagate Delaunay in-circle tests from the two diagonals of Δ that have not already been marked for deletion, as normally. Note one diagonal of Δ has been marked for deletion. We connect β to α, γ, and all nodes incident to a diagonal marked for deletion. We delete all marked diagonals.
G. **Note:** SPLIT(alpha,beta) Split the node of α in two: α_1 and α_2; create the node of β; Connect edge $\alpha_1\beta$ and $\beta\alpha_2$ Assign the point(s) associated with α to α_1 and/or α_2 according to their indices. Scan triangle incident to α sequentially to determine the first triangle incident to α that gets destroyed by β by running incircle tests. We can start at $\alpha.prev\alpha$ or $\alpha\alpha.next$. The edges alpha.prevα and $\alpha\alpha.next$ will remain. (We can start at both ends visiting diagonals alternating

so that we vist the minimum number of diagonals that will remin in the triangulation). We connect β to all nodes incident to a destroyed triangle.
Note that diagonal p_0p_n will never be deleted
To visualize the triangulation we can place the nodes on the arcs along their implied bisector arcs. But it is not necessary : we can place them on the points on near the points, realizing that sometimes edges may be crossing as shown in Schechuk. For nodes that get split we create 2 different nodes placed appart and near the point they represent.

Note: We might be in a situation where 2 triangles are deleted and we encounter a 3rd triangle such that it survives. We now know the entry point for the point however the earlier orientation tests show that the 2 triangles do not contain the point, in which case the triangles neighbouring the triangles incidental to γ should be checked. Record the edges of the adjacent faces if they will be deleted. Once point is located delete all the marked diagonals.

General rule of thumb: Record all diagonals to be deleted until finding the triangle containing the point. Exception: Situation where all triangles incidental to γ is deleted and β's triangle was found before traversing through all triangles incidental to γ. In which case keep recording faces to delete even after point locating. After finding a surviving triangle, go to point located triangle from before and delete all marked edges.

Updated general rule of thumb: Record all diagonals to be deleted until finding the triangle containing the point and finding the entry points.