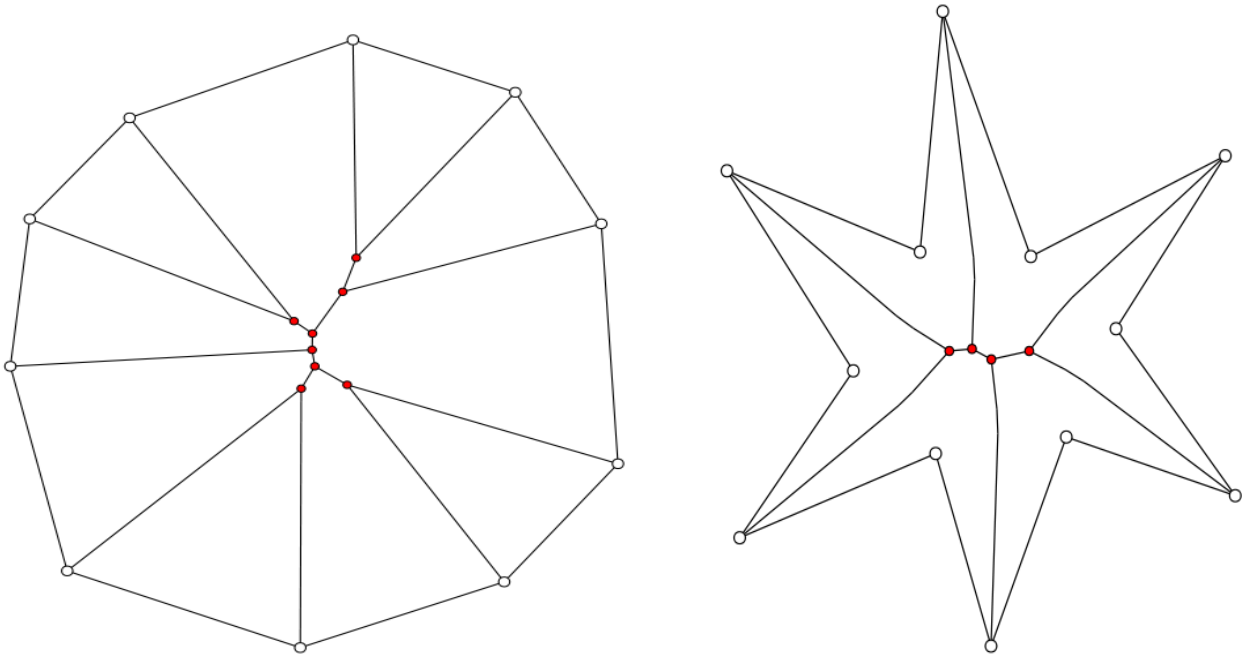
Determining the Skeleton of a Simple Polygon in (Almost) Linear Time

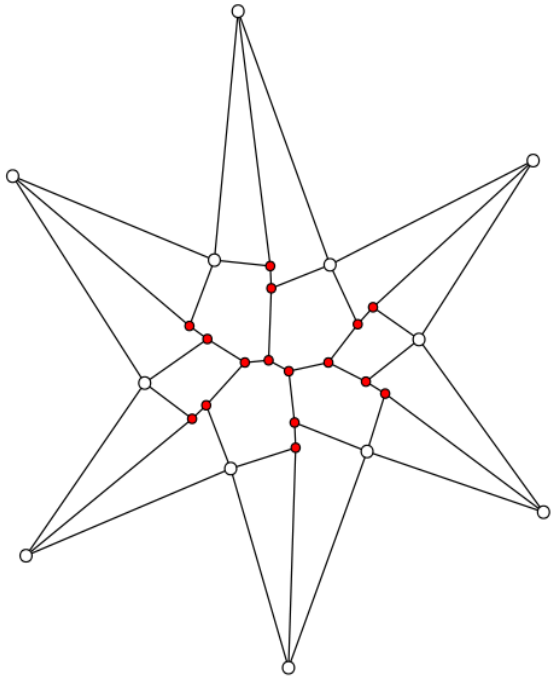
Robert Edwards
Oak Ridge, Tennessee
December 2, 2010

The skeleton of a simple polygon (also known as the medial axis) is a set of rays (lines or parabolas) that have the property that any interior point on a ray is equidistant from the closest a) two sides, b) two reflex corners,[1] or c) an reflex corner and non-adjacent side, whichever combination is closer to the point. The nodes of the skeleton of a simple polygon are the set of intersection points of skeletal rays. The nodes have the property that they are equidistant from three sides, from two sides and an reflex corner, from one side and two reflex corners, or from three reflex corners. Below are examples of skeletons of convex and non-convex polygons:

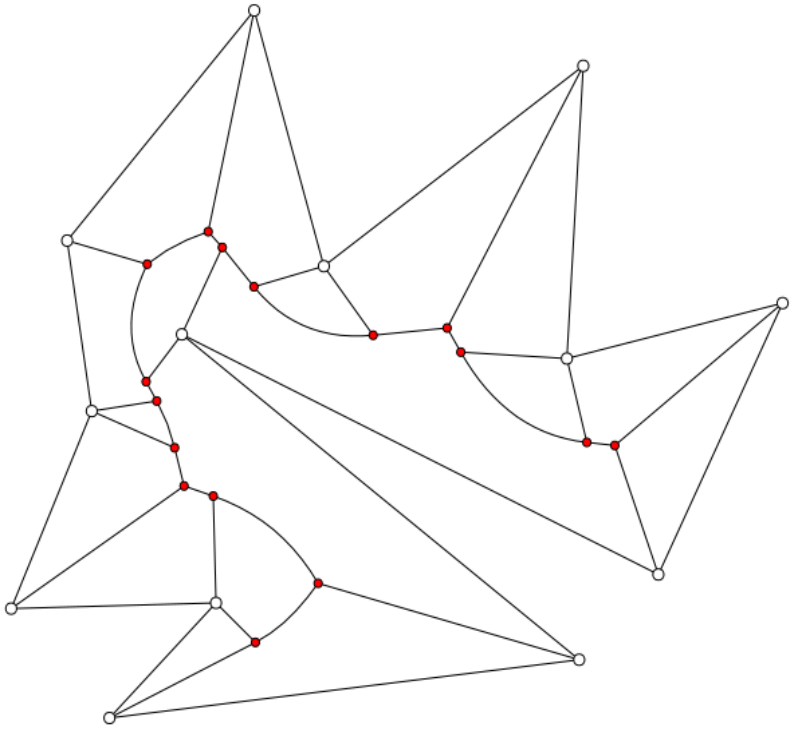


For a convex polygon, all the rays are lines, but in the case of a non-convex polygon, a ray may a line or parabola. The transition point between a line and a parabola occurs at a point on a line emanating from an reflex corner that is perpendicular to one of the sides adjacent to that corner. For the star-shaped non-convex polygon shown above, the additional transition points are shown in the figure on the next page. Note the two perpendiculars emanating from each reflex corner and the additional points on the skeleton indicating line/parabola transition points.

1 The reflex corners of a simple polygon are corners not on the polygon's convex hull.



For this star-shaped example, the line-to-parabola transitions are not readily apparent, but in the following example[2] the transitions can be clearly seen:



2 This example can be found on the internet at http://jeff.cs.mcgill.ca/~mcleish/507/myapplet.html

Determining the skeleton of a simple polygon has been studied extensively. The most straightforward - but most time consuming - method for determining its nodes is to find all circles defined by the four equidistant conditions stated above and lying entirely within the polygon. For the non-convex polygon of just six sides and two reflex angles, this would consists of examining as many as 120 circles for combinations of any three sides, 30 circles for combinations of two sides and a point, 6 circles for combinations of one side and two points - altogether, some 156 circle determinations.

A theoretical linear-time algorithm is proposed by Chin, Snoeyink and Wang,[3] but no practical implementation has yet been devised.

A more straightforward method of producing the skeleton is to determine the Voronoi diagram of the corners, then adjusting this diagram to generate a Voronoi diagram of lines segments by sequentially replacing pairs of points corresponding to a sides of the polygon with the lines between them. The time to determine a Voronoi diagram for an arbitrary set of n points is an $n \cdot ln(n)$ process, but it has been shown that it is possible to bring this to a process linearly dependent on the number of polygonal sides. Since the points-to-lines adjustment of the diagram would be proportional to the numbers of sides, this could be a process linearly dependent on the number of sides. Unfortunately, no practical implementation of this method has been presented.

A more general generation algorithm for an arbitrary set of points and line segments is contained in CGAL (Computational Geometry Algorithms Library). Information about this CGAL algorithm is located at http://www.cgal.org. This algorithm is stepwise in that it adjusts an existing graph that results in addition of a point or line. However, the search to determine what objects need by adjusted is again at best an $n \cdot ln(n)$ process.

In 1982, D. T. Lee proposed a method[4] that consisted of defining a diagram consisting of perpendiculars from each reflex angle, then sequentially generating rays from each angle on the convex hull, transitioning from line to parabola and back again as perpendiculars or previously generated rays are encountered. A Java animation by Thomas Wolf of this algorithm is available on the Internet: http://www.lupinho.de/gishur/html/Skeleton.html.

The method we will use is similar to the algorithm proposed by Yao and Rokne in 1991[5], which is a modification of Lee's algorithm. We will show that our algorithm can calculate a polygon skeleton in time proportional (essentially) to the number of sides. Although the proposed method is a major improvement over the Yao/Rokne algorithm is that no backtracking is required, the primary objective is to facilitate implementation of a compact, straightforward computer program.[6]

The proposed algorithm for generating the skeleton of a simple polygon is an extension of a rather obvious method for the skeleton of a convex polygon. The final result skeleton aimed for (from an example in the Yao/Rokne article) is shown at left on the next page. For convex polygons the method begins by creating a double-linked list of rays that are bisectors of each angle. For each ray, distances are determined from a ray's origin (a corner of the polygon) to the intersection with its predecessor ray and with its successor ray.[7] These distances for a ray are called the behind and ahead intersection distances. The reduction then starts with each ray pair such that the ray to the rear has a behind distance **greater** than its ahead distance, and the ray to the fore has a behind distance **less** than its ahead distance.

---

3 Francis Y. Chin , Jack Snoeyink , Cao An Wang, Finding the Medial Axis of a Simple Polygon in Linear Time, Proceedings of the 6th International Symposium on Algorithms and Computation, p.382-391, December 04-06, 1995

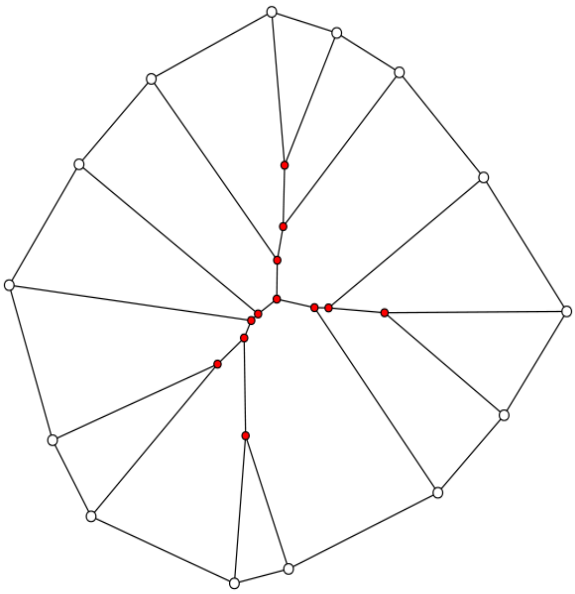
4 D. T. Lee, Medial axis transformation of a planar shape, IEEE Trans. Pattern Anal. Machine Intell. PAMI-4 (1982), pp 363-369.

5 C. Yao and J. Rokne, A straightforward algorithm for computing the medial axis of a simple polygon, International Journal of Computer Mathematics, Vol. 19, pp 51-60 (1991).

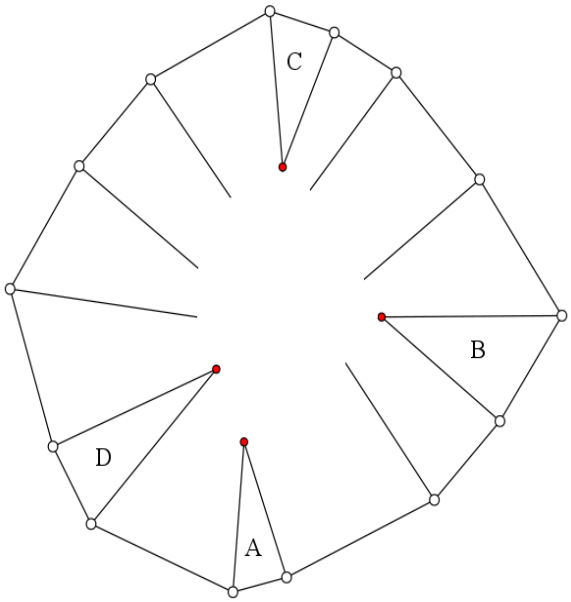
6 Experimental versions of the program in Borland Delphi and C++ Builder are available from the author.

7 Throughout this discussion polygons are assumed to close counterclockwise, so a successor ray is the one first encountered in traversing a polygon in the counterclockwise direction.

In the figure at right, below, the ray pairs adjacent to A, B, C and D are the initial reduction candidates. The rays forming A, B, C and D are eliminated, candidate pairs in the reduced polygon are considered for the next step. Note that the actual distance values associated with the rays are not critical in determining a candidate, only the fact that they increase or decrease from one side to the other.

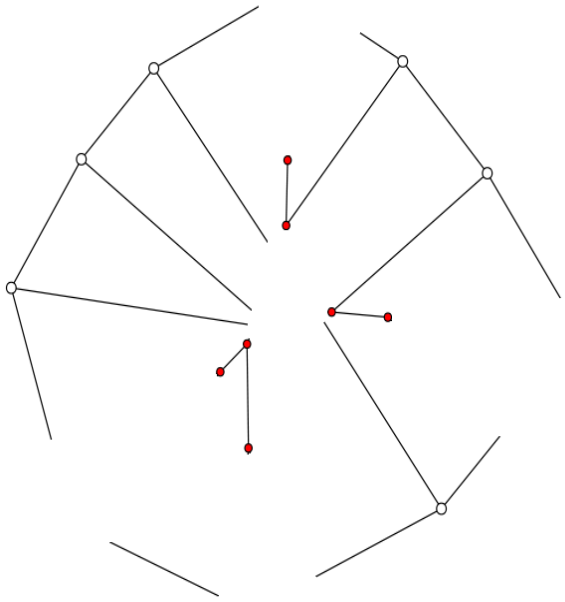


Skeleton Final Result



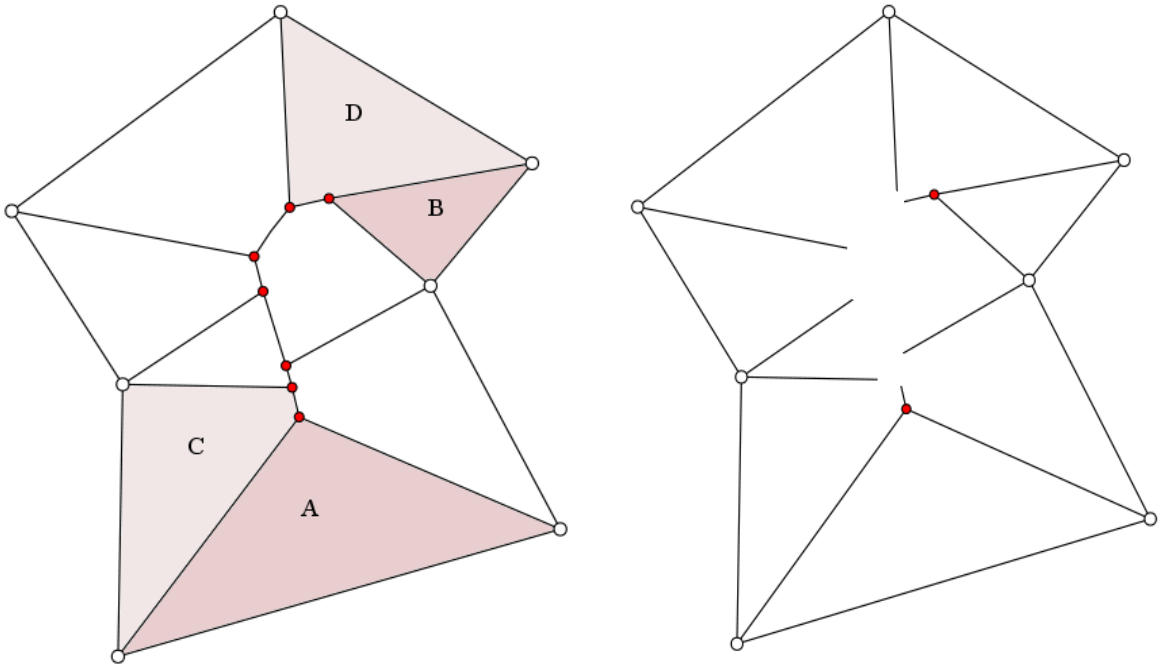
Step 1

The reduction step is to replace the identified ray pairs with a single ray defined on the sides adjacent to the pair being reduced, and starting where the two reduced rays intersect. The behind and ahead distances for the result rays are determined as in the initial step and the process is repeated. The figure below shows the situation after the first reduction step and showing the next three pairs that are candidates for reduction.

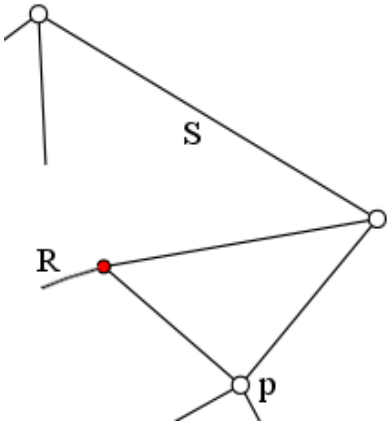


After Step 1 and Identification of Step 2 Candidates

To extend the method just described for a convex polygon to the case where the polygon may not be convex (example polygon and skeleton result at left, below), auxiliary perpendicular rays are constructed at each of the reflex angles (figure at right, below).



The algorithm proceeds as with the convex example: rays adjacent to sides corresponding to A and B are candidates in the first step, C and D in the second step, and so on. However, a complication now arises in that a resulting ray may be a section of a parabola. The following figure shows detail of the new ray at B. The new ray R is a section of a parabola whose focus is at the reflex corner p and has as its directrix the polygonal side S. Note that the behind and ahead distances for R involve line/parabola intersections.



To handle the complication of dealing with both parabola and line rays, rays are typed depending on how a they are defined.

Ray type 1: A line ray defined by two sides, just as in the case of a convex polygon

Ray type 2: A line ray at the back of a succeeding side defining an reflex angle.
The back ray of the pair at B is a type 2 ray.

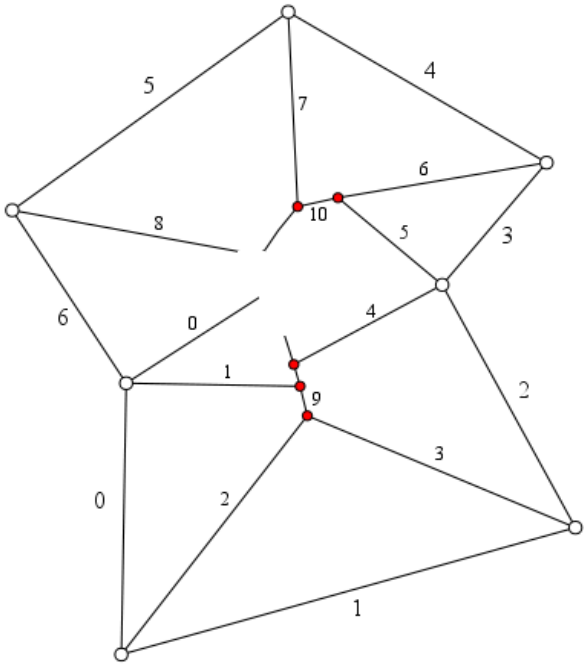
Ray type 2a: A parabolic ray whose focus is at an reflex corner, lies in the area between the two perpendiculars at the corner, and whose direction is away from a type 2 ray.

Ray type 3: The complement of a type 2 ray at the fore of a side preceding an reflex angle.

Ray type 3a: A parabolic ray whose focus is at an reflex corner, lies in the area between the two perpendiculars at the corner, and whose direction is away from a type 3 ray.

Ray type 4: A line ray between two reflex corners.

Given these definitions, we can show the first two steps leading to the determination of the skeleton of the example:



In the figure above we have numbered the sides and rays, and have determined that ray pairs 2,3 and 5,6 are the candidates for the first reduction. The table on the next page is the initial behind/ahead distances for the rays as numbered above.

Ray No	Behind Dist	Ahead Dist
0	137	Inf
1	Inf	127
2	213	190
3	181	223
4	139	Inf
5	Inf	85
6	132	157
7	122	158
8	183	189

Observe that the behind distance for 2 is greater than the ahead distance, and the behind distance of 3 is less than the ahead, so pair 2/3 meets the condition for a reduction. The same is true of pair 5/6.

The first step is to determine ray 9, a type 1 ray (a line) by reducing rays 2 and 3. The origin point for ray 9 is the intersection of rays 2 and 3, and its direction is along the bisector of sides 0 and 2. The behind distance of ray 9 is defined by the intersection of ray 9 with the ray behind ray 9, which is ray 1. The ahead distance of ray 9 is defined by the intersection of ray 9 with the ray ahead of ray 9, which is ray 4.

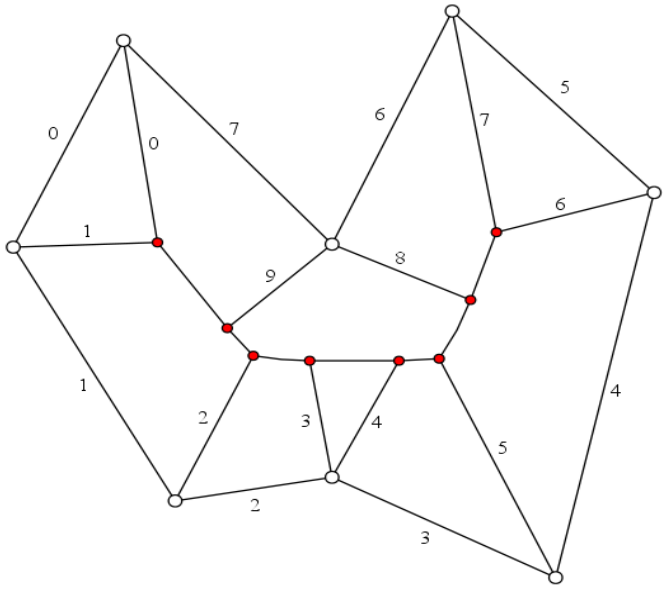
The second step determines ray 10, a type 2a ray. The origin point for ray 10 is the intersection of ray 5 and 6. The direction of ray 10 is a parabola extending away from ray 5, and defined by a focus at the corner between rays 2 and 3 and a directrix of side 4. The behind side distance is the intersection of the parabola and the line ray 4. The ahead distance of ray 10 is the distance between its origin and the intersection of the parabola ray 10 and the line ray 7. Now we can update the table of behind/ahead distances as follows:

Ray No	Behind Dist	Ahead Dist
0	137	Inf
1	Inf	108
2	211	190
3	181	223
4	105	697
5	Inf	84
6	131	158
7	123	158
8	183	189
9	19	34
10	670	26

In this table only the entries 1, 4 and 7 have changed, 9 and 10 have been added and 2,3,5 and 6 have been eliminated. Note that when a pair of rays are reduced, the behind/ahead table changes only by eliminating the two reduced rays, adding a new ray, and recalculating the distances for the two rays preceding and succeeding a new ray. This leads to an optimization for processing the list to fetch ray pairs for subsequent reduction steps: When the list is generated, a link list is created for reduction candidates. As a pair is removed and the list updated, the link list is also updated. Thus processing the sequence of pairs being reduced is dependent only on the total number of pairs reduced.

The next steps are to determine a new ray 11 from rays 1 and 9, and a new ray 12 from 10 and 7. Continuing in this fashion determines the skeleton of the polygon.

Now for a complication. How do we know that the polygon doesn't have "false" reduction candidates[8], as is shown in the following example?



Ray No	Behind Dist	Ahead Dist
0	223.8	138.4
1	86.4	182.2
2	197.9	165.7
3	135.6	Inf
4	Inf	117.0
5	190.5	238.0
6	166.6	98.3
7	152.2	210.7
8	119.4	Inf
9	Inf	124.2

8 False reduction candidates are called "fake rendezvous" in Yao/Rokne.

Clearly, we want pairs 0/1 and 6/7 as reduction candidates. However, rays 2/3 and 4/5 are also candidates by the rules previously established:

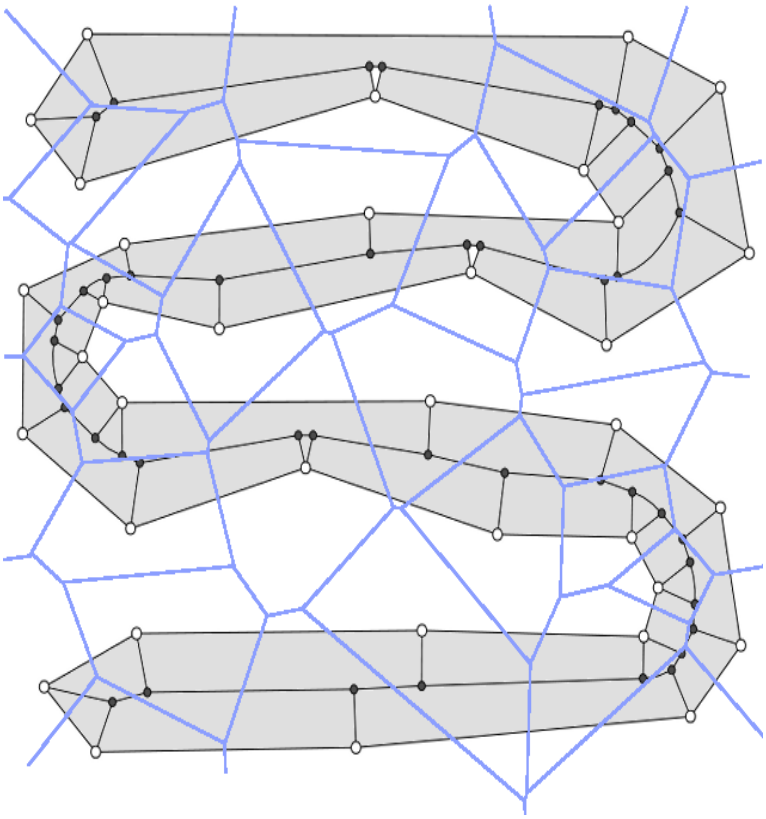
2/3: 197.9 & 165.7 for ray 2 and 135.6 & Inf for ray 3
4/5: Inf & 117.0 for ray 4 and 190.5 & 238.0 for ray 5

We want to bypass candidates 2/3 and 4/5 because their intersections lie closer to point 6 (between sides 6 and 7) than they do to their respective defining sides. So when we determine candidates for reduction, we need to filter out candidates such as 2/3 and 4/5 in the above example.

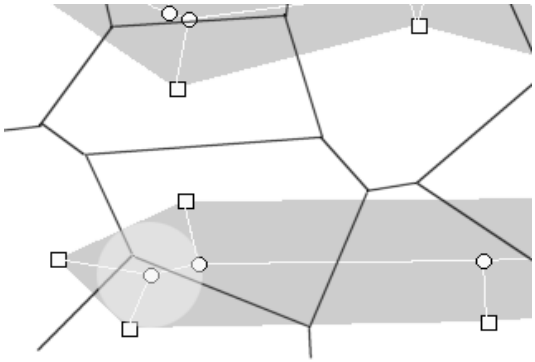
To do this filtering, we must make sure that the intersection of a candidate pair of rays does not lie closer to a non-adjacent vertex or side. So for the general case, this means examining all of the non-adjacent sides and vertices for each candidate – basically turning the algorithm into a n^2 process. So the reasonable next question is: Can the filtering of candidates be done efficiently?

The answer to this question is yes, but with a qualification that we will get into later. The most obvious way to shorten the filtering is to generate the Thiessen polygons of the vertices of the polygon, then determine a list of all of the sides that intersect each containment polygon. Once the Thiessen polygons are generated, the containment of the sides in each polygon can be done by simply traversing the polygon and assigning the sides to polygons as the traversal progresses.

To show a result, the figure below shows a polygon together with its skeleton and Thiessen polygons.



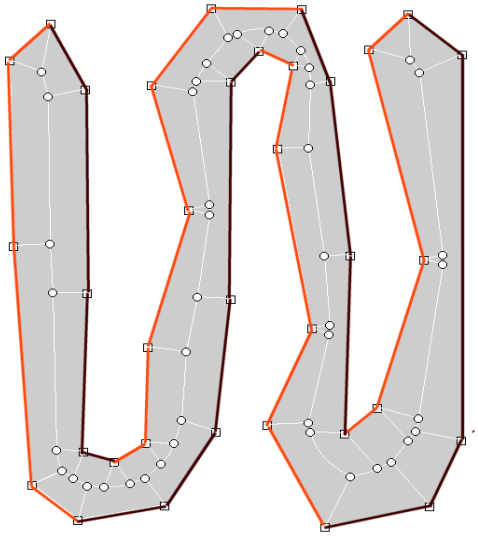
When determining if any point or side of the polygon is closer to the first skeleton point at the lower left of the figure, the only features that need to be examined are those falling in the Thiessen polygons that the candidate falls in and its immediate neighbors. The number of neighbors that have to be examined is dependent on the radius of the circle defined by the candidate point, and whether any neighbor Thiessen polygon intersects that circle:



For the candidate point in the shaded circle, only the polygon features lying in the three Thiessen polygons intersecting the circle need to be examined. All other polygon features are irrelevant.

The time for constructing a Thiessen polygons for a arbitrary set of points is proportional to *n log n* (*n* is the number of vertices) if the coordinates of the vertices are not ordered, but only proportional to *n* if they are ordered. So is it possible to order the vertices of a polygon in time proportional to the number of vertices? The answer is "in a way."

To order the coordinates of a polygon in one direction, consider decomposing it into several monotonic chains. To do this, start at the bottom point of the polygon and traverse the two sides emanating from it until a side is encountered that turns back toward the bottom. Traverse this chain until it turns back away from the bottom and reverse the direction of this chain. Continue in this fashion until the polygon has been traversed. The diagram below shows the result of chains resulting from this process. The black chains are traversed from start to finish; the red chains are traversed from finish to start. Thus, all the chains are oriented from bottom to top.



Now we can merge the chains using an "optimal" merge[9] whose speed in the worst case is proportional to *n log k* where *k* is the maximum number of chains encountered by any horizontal line. In the case of the example, *k* is eight for a sweep direction from bottom to top.

So the net of all of this is that a skeleton can be determined in time no longer than proportional to *n log k* where n is the number of sides (or vertices) and k is the maximum number of sides encountered by any line perpendicular to the direction of the sweep line of the Thiessen polygon generation.

So we can make the "weak" claim that the proposed skeleton algorithm is linear with respect to the number of polygonal sides. If we say that the number of polygonal sides increases because of an approximation refinement (that is, the number of chains being merged to generate an ordered set of points for the Thiessen polygon generation step does not increase in refinement), the process is only dependent on the number of polygonal sides.

Comment on optimizing the identification of false reduction candidates

Although the additional complication of determining the Thiessen polygons for the polygon vertices to optimize the filtering of false reduction candidates is attractive from a theoretical view, it is seldom warranted. The test for proximity of a candidate center to a polygonal feature is a simple point-to-line or point-to-point distance test, hence the polygon must have hundreds of sides for this test to be the controlling factor in the speed of the algorithm.

Comment on determining the "ahead/behind" intersection distances for parabolic rays

For a parabola defined in terms of a focus x_f, y_f and directrix $a\,x+b\,y+c=0$, the arc length, *d*, between points on the parabola x_1, y_1 and x_2, y_2 is defined by the formula[10]

$$d=\frac{w}{2}\left[u_1\sqrt{1+u_1^2}+\ln\left(u_1+\sqrt{1+u_1^2}\right)-u_2\sqrt{1+u_2^2}-\ln\left(u_2+\sqrt{1+u_2^2}\right)\right]$$

where

$$w=a\,x_f+b\,y_f+c$$
$$u_0=b\,x_f-a\,y_f$$
$$u_1=(b\,x_1-a\,y_1-u_0)/w$$
$$u_2=(b\,x_2-a\,y_2-u_0)/w$$

One would naively believe that computing this function is necessary in determining the ahead and behind intersection distances for a parabolic ray. However, the simple squared distance between points x_1, y_1 and x_2, y_2 can be used a distance surrogate for this purpose.

Acknowledgement

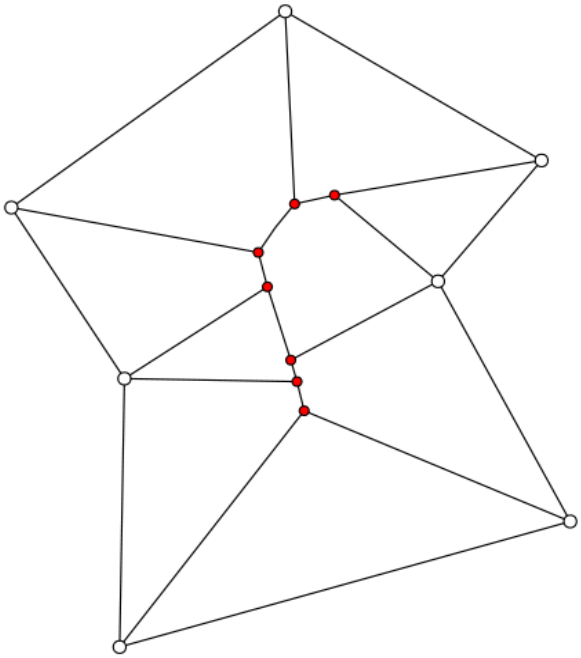
Special thanks to Grimes Slaughter, former physics researcher at Oak Ridge National Laboratory, for review and comments on this paper.

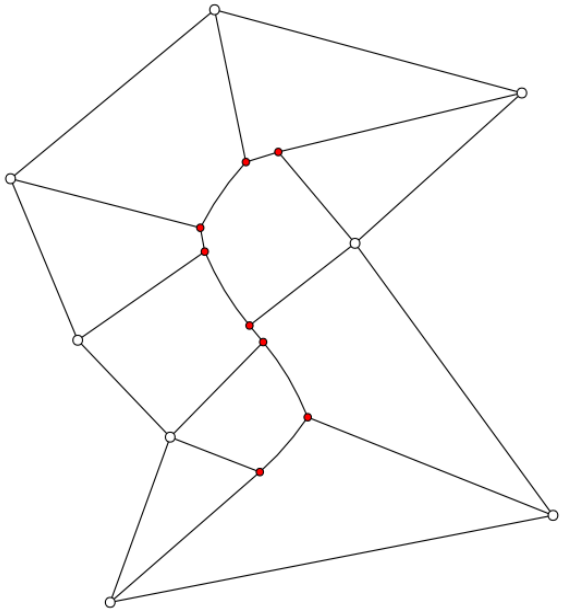
9 The optimal merge is presented in a National Instute of Standards and Technology web page: http://www.nist.gov/dads/HTML/optimalMerge.html

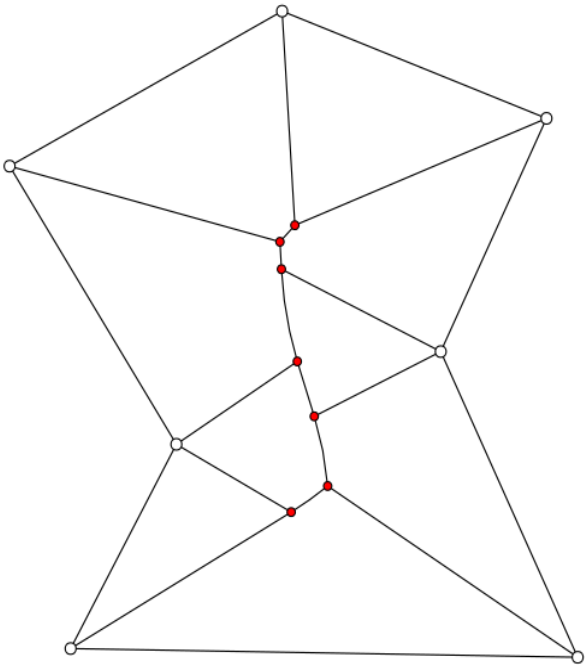
10 Derivation of this formula can be found at http://www.ugrad.math.ubc.ca/coursedoc/math101/notes/moreApps/arclength.html

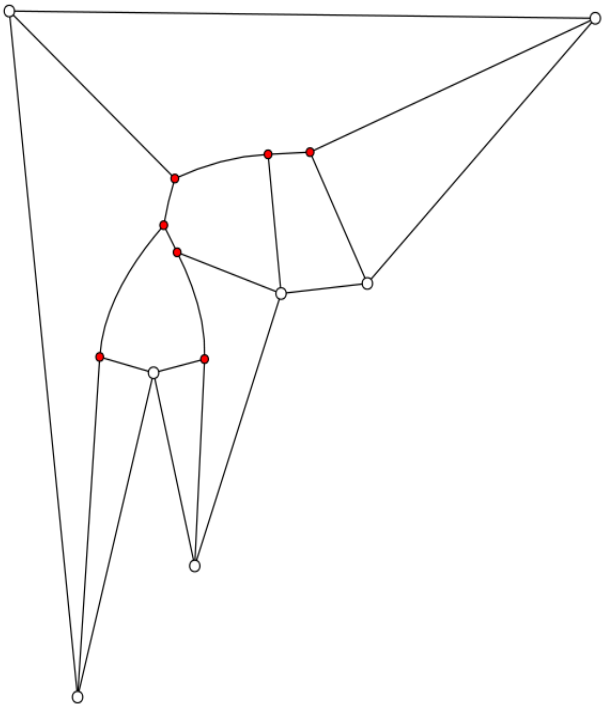
Gallery

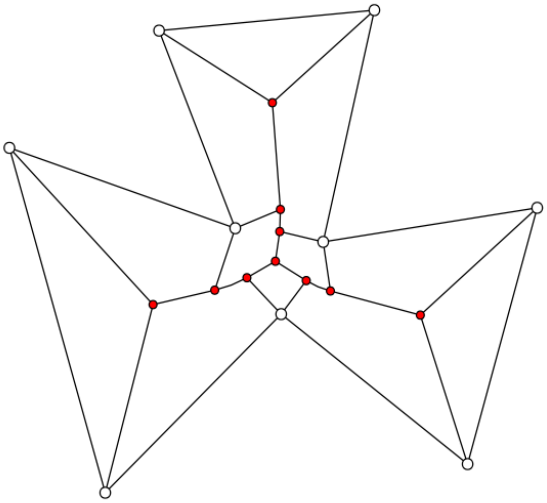
Images of test cases

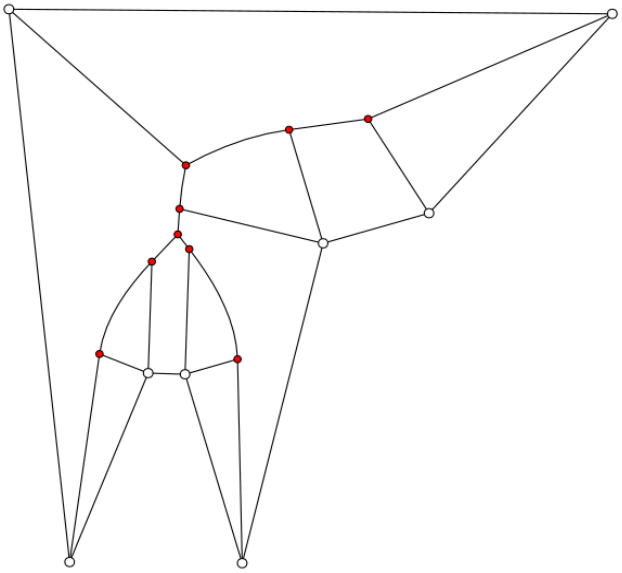


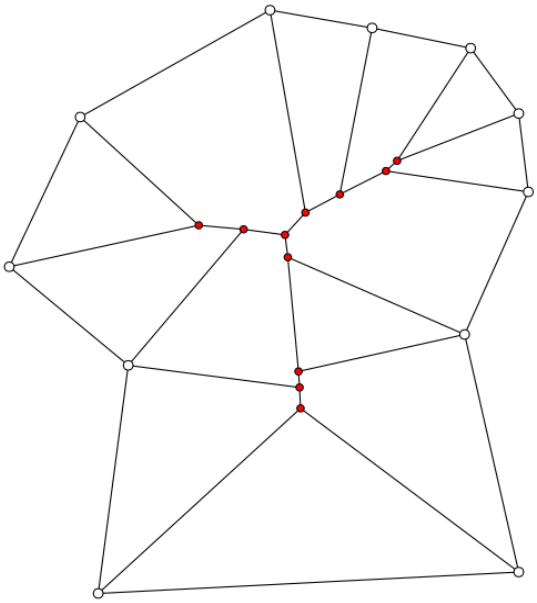


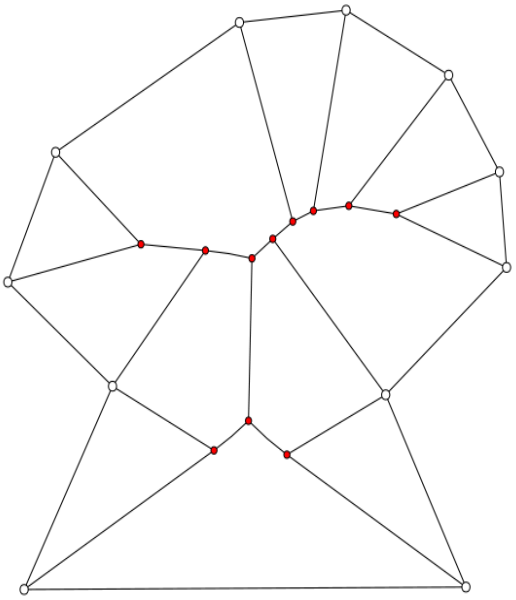


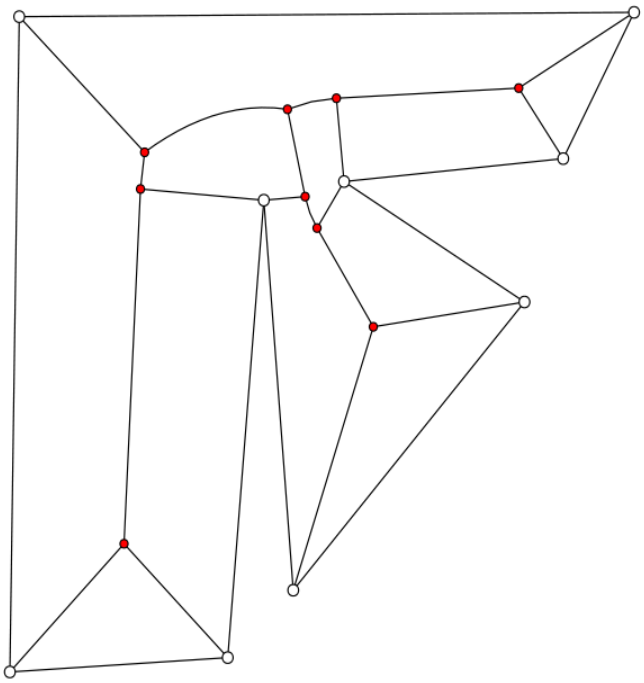


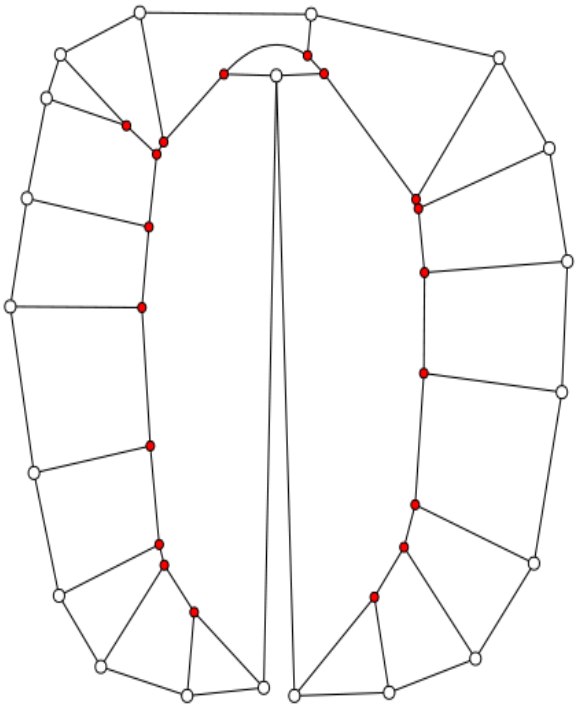


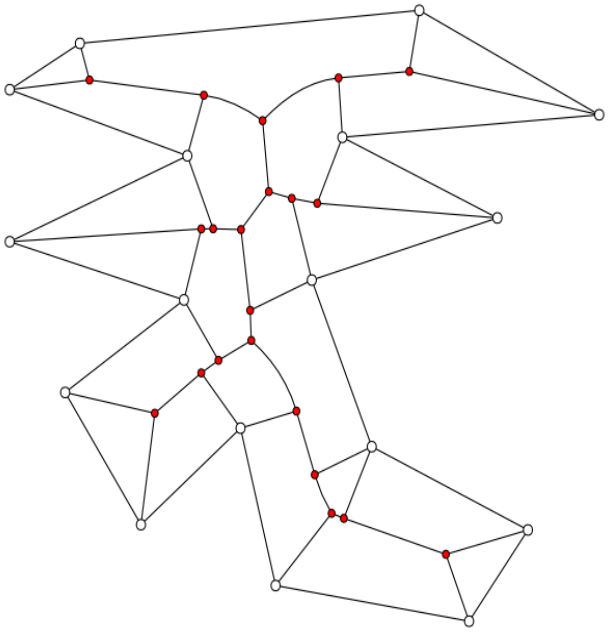


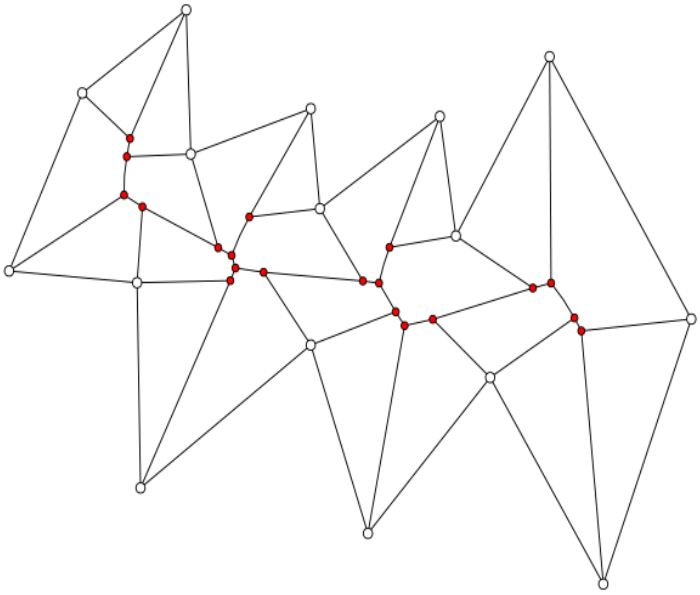


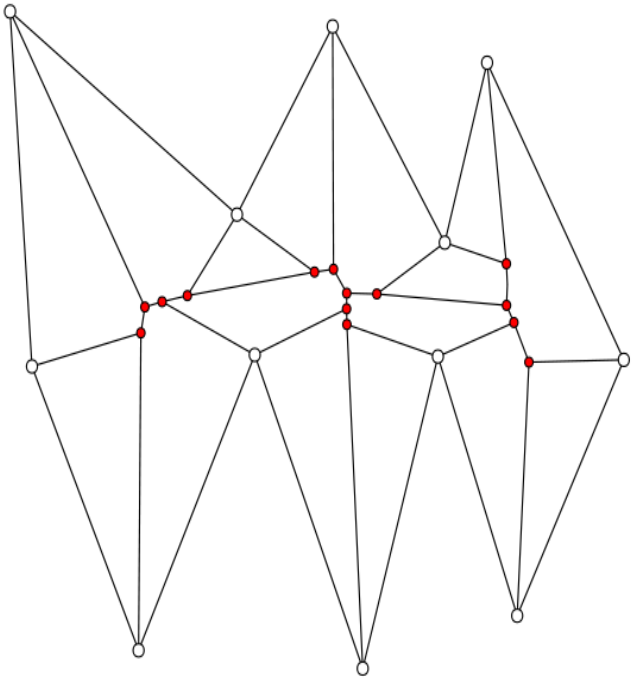


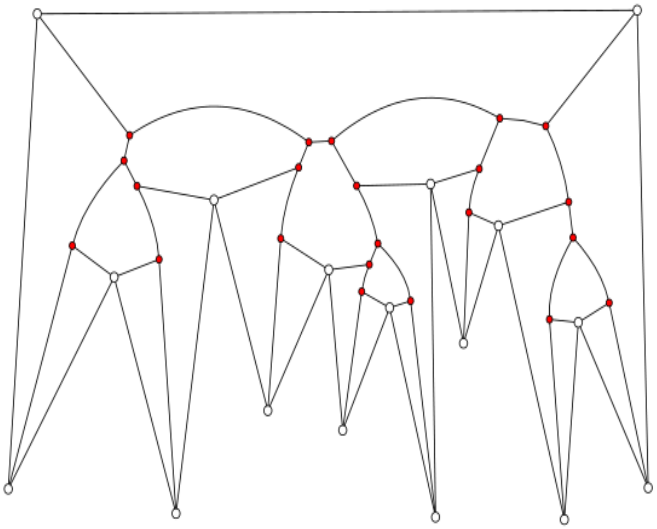


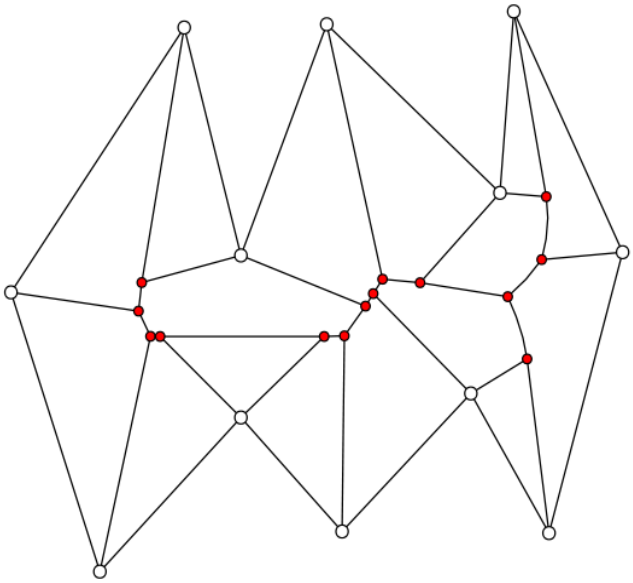


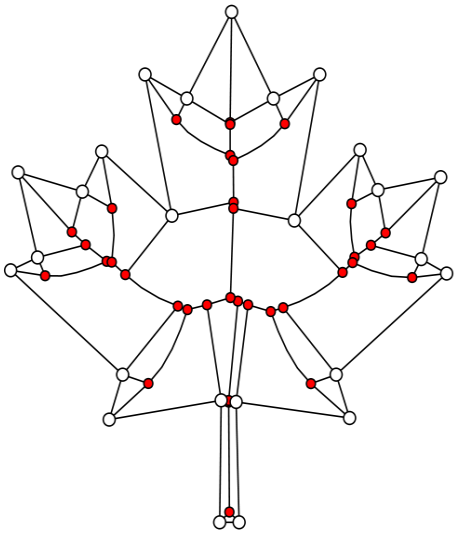


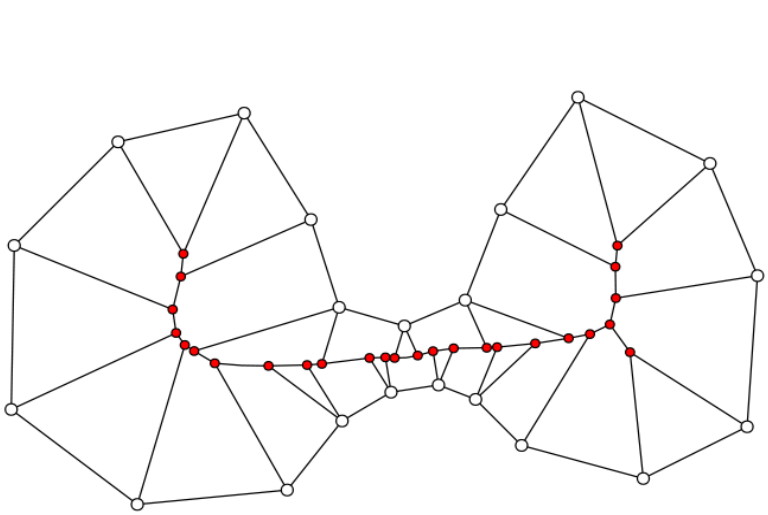


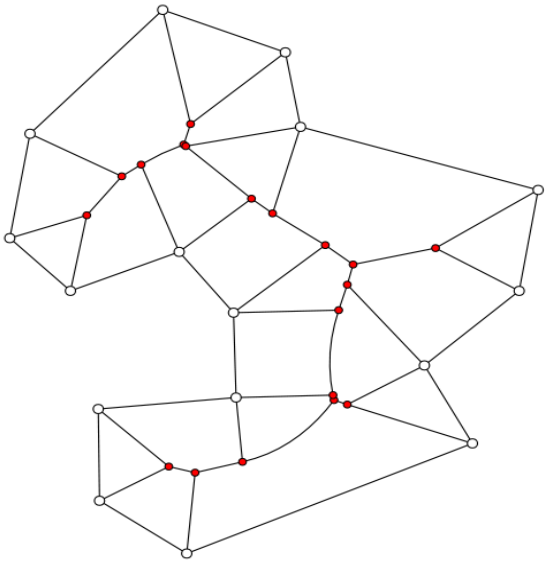


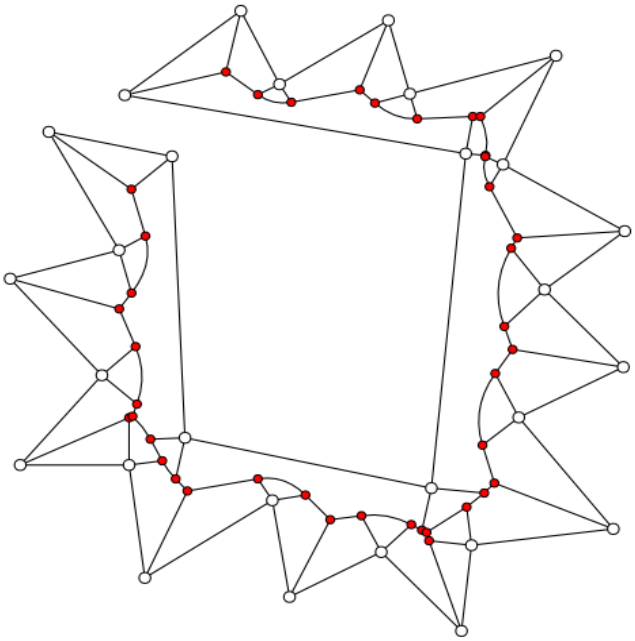


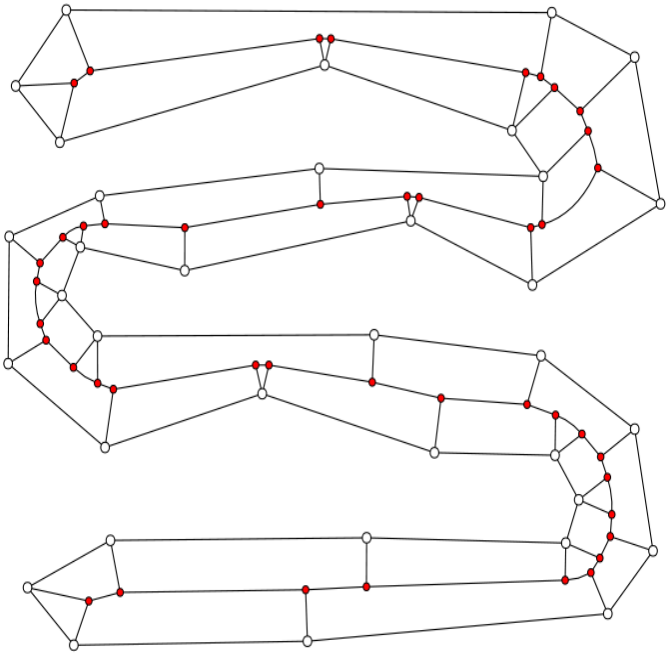


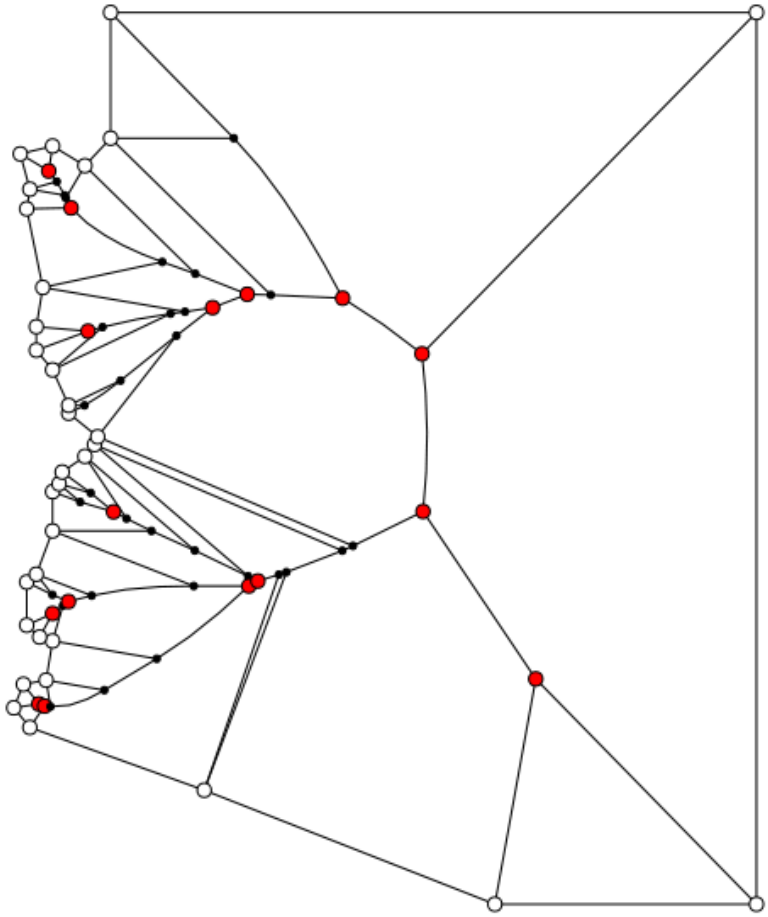












In this example, end points of the perpendiculars starting from reflex angles are shown as small black dots.