Euclidean Steiner Tree Problem

Rowan Shigeno

Introduction

Background: I first came across a specific case of this problem in my Analysis of Algorithms class— for $n = 4$ points (the corners of square) in $\mathbb{R}^2$— introduced as the "4-village road-building problem." The problem: given 4 villages lying at the corners of a unit square (i.e. a mile in width/height), what is the optimal (length-minimizing) way to connect the 4 villages together via a sequence of roads? Here is my own solution:

A first guess would be to just connect the four villages by the edges of the square: here, the total road length would be 4 miles. We can in fact remove one of the four roads (say, the top of the square) and still have all 4 villages be connected in some way, yielding a total of 3 miles. (To distribute travel distance between villages as fairly as possible, we could if desired translate the bottom road upwards, to the center, to make an 'H'.)

An alternative, close-to-optimal solution would be to link the 4 villages together via the diagonals of the square, making an 'X'. The total distance here is $2\sqrt{2} \approx 2.83$. But there is an even better solution.

Consider taking the 'H' diagram, but contracting the intersection points (the midpoints of the left and right roads) to the center of the square, as follows:

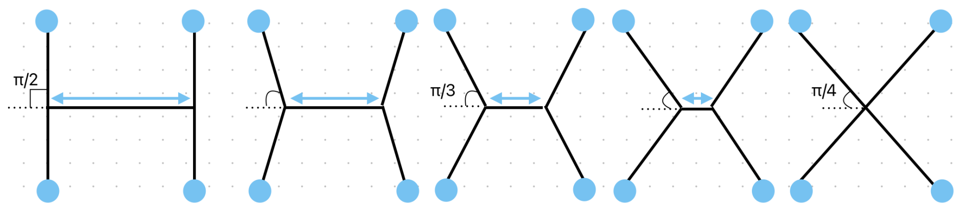

Figure 1: contraction of H to X.

Keeping track of the labeled angle, we can write the total road length L in terms of the angle, using trigonometry:

$$L = \frac{2}{\sin\theta} + 1 - \frac{1}{\tan\theta}.$$

This quantity is minimized when $\theta = \frac{\pi}{3}$, making $L \approx 2.73$. This is the optimal road construction.

- Consider: apply the same process to the corners of a general isosceles triangle.

A more general version of the problem, called the **Steiner tree problem** (for graphs), asks for the tree of minimum weight that contains all terminals (but may include additional vertices), which we shall refer to as the **Steiner minimal tree**, or SMT for short. Notice the distinction from the minimum spanning tree (MST), whose vertices are restricted to the original vertices of the graph. Determination of the SMT is NP-hard [**Gander**, 4]– in contrast, the MST can be determined in $O(n \log n)$. In this project, I intend to investigate the Steiner tree problem for graphs in $\mathbb{R}^2$.

For the classical Steiner tree problem in $\mathbb{R}^2$, it is known that points added to the original graph (called **Steiner points**) must have a degree of at least three (where edges incident to degree-three Steiner point form three 120 degree angles).

Proof that $\deg p > 2$ *for any Steiner point* p. If the degree of p were two, we could remove p and connect the two points directly instead, which would be at least as short. □

The fact that the Steiner points necessarily yield 120 degree angles stems from the fact that the normalized edge vectors will only cancel out when the they are separated by equal angles [**Gander**, 6].

For the specific case of a triangle ($n = 3$) with no angle being greater than 120 degrees, the point which satisfies this property is called the **Fermat point** (AKA the Fermat-Torricelli point [**Gander**, 3]) of the triangle—which is *not* the same as the circumcenter, centroid, orthocenter, or incenter of that triangle). And since the number of triangles in any triangulation of n initial points is $n-1$, it follows that the maximum number of Steiner points that a Steiner tree can have is $n-2$.

The general solution for the SMT of a regular polygon is a honeycomb-esque shape for $n < 7$, but this pattern breaks after a hexagon, as the interior angles subsequently exceed 120 degrees (making the edges of the polygon the most optimal for Steiner tree branches).

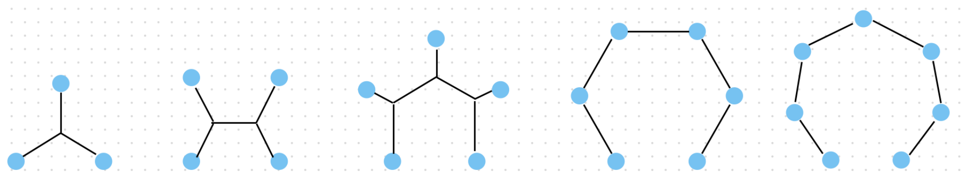

Figure 2: SMTs for regular polygons ($n = 3, 4, 5, 6, 7$).

Further questions to consider:

- General solution for any triangle? Any quad? Any irregular n-gon?
- Any point set with n points?
- Ultimate goal: examine known algorithms, and create & implement an algorithm for solving the Steiner tree problem for an input set corresponding to the vertices of a convex n-gon that is approximately regular, n restricted.

Known model of computation: **GeoSteiner** (http://www.geosteiner.com/), from the University of Copenhagen. Using GeoSteiner, I made the following Steiner trees (see final figure, end of paper). Only the Steiner vertices were outputted, no edges, so I drew them manually in Desmos. (Also, GeoSteiner didn't like certain input points for whatever reason, so I had to make adjustments.)

The GeoSteiner algorithm is "by far the most efficient **exact** algorithm for calculating solutions to the minimum Steiner tree problem in practice, and can calculate these solutions for $n = 10,000$ in less than 500 seconds" [**Brazil**, 1]. The algorithm has two phases: a generation phase, in which potential Steiner tree components are proposed, and a concatenation phase, where the solution is found by combining a subset of these components. The "generation phase" of GeoSteiner for subtrees uses the fact that given any 2 points, all possible Steiner points connecting those points to a third point must lie on a specific arc that passes between the initial points (since the angles emerging from the Steiner point must be 120 degrees).

Other implementations: **FLUTE**, **PHYLIP**, **pcst_fast**, **DSTAlgoEvaluation**, **Salowe's Rectilinear Steiner Trees**, although many of these are of modified versions of the Steiner problem (e.g. finding rectilinear trees) [**Skienna**].

Key ideas

Henceforth, we shall refer to any of the original n points (i.e. non-Steiner nodes in the graph) as **cities**. Letting p be the number of Steiner points and r be the number of roads [**Gander**, 5],

$$r = p + n - 1.$$

Proof. A set of connected edges of minimal length cannot contain a cycle, and is thus a tree with # of edges being $n - 1$. Since each Steiner point (having degree 3) removes the need for two edges while adding 1, adding 1 Steiner point adds 1 road. □

Now for any 3 cities a, b, c in $\mathbb{R}^2$, we have two cases:

(1) One angle in the triangle is ≥ 120 degrees. Then there are no Steiner points, and the SMT consists of two edges meeting the ≥ 120-degree point.

(2) All angles are < 120 degrees. Then the cities are all linked to one Steiner point, which is the Fermat point of the triangle.

Visual proof (adapted from [**AC vol. 29**, 3]): $|cs| + |as| + |bs| = |cs| + |rs| + |rt|$, which is minimized when c, s, r, t are collinear. See figure 3.

For n cities, shortest network has the following properties [**Gander**, 8]:

(1) Two roads connecting a common node have angle greater than or equal to 120.

(2) No node is connected to more than 3 roads.

(3) Any node connected to 3 roads has 120 degree angles.

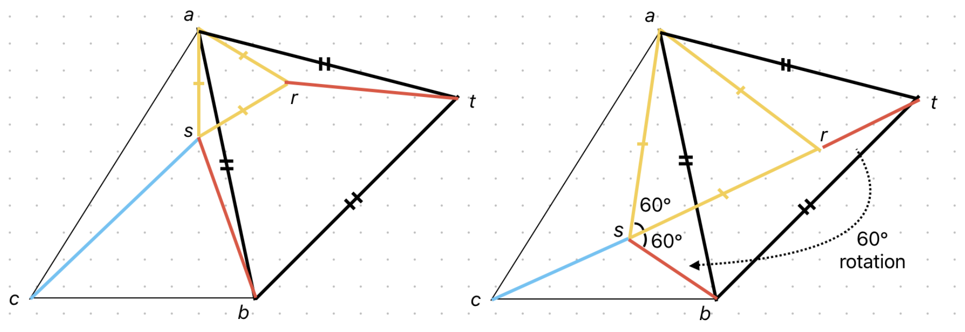

Figure 3: derivation of Fermat point.

(4) All Steiner points are connected to exactly 3 nodes.

One idea for an approximation algorithm for a set of points in general position could be to first take the MST: then, at each angle in the MST less than 120 degrees, add a Steiner point nearby, and iterate through this angle-checking process until a valid Steiner tree is produced. This simple approach had initially been suggested by Thompson [**Zachariasen**, 2]. Note that this is not guaranteed to produce the SMT: the MST itself (with no alterations) could be as far as a factor of $2/\sqrt{3}$ or roughly 15.5% from the optimum [**AC vol. 29**, 23], although on average for $n = 100$ points distributed randomly within a square, the average weight reduction from the MST to the SMT is only about 3.2% [**Zachariasen**, 2]. However, simply generating an SMT from the MST could easily result in a suboptimal topology (see below): so although such an approach could be fairly accurate (i.e., at least having somewhat less weight than the MST in most cases), it would only produce the optimal solution in some cases.

Now in determining the SMT for any set of points, it is crucial to consider the different possible topologies of the resulting Steiner tree. For instance, given 4 points A, B, C, and D, is it more efficient to connect A to B and C to D before linking the Steiner points, or perhaps A to D and B to C? Answering such a question is in fact the most time-consuming step in computing the SMT for a point set. In fact,

Theorem (Melzak-Hwang)**.** *Given a full Steiner topology $\mathcal{T}$ with n terminals, there is an $O(n)$ time algorithm to either compute a full Steiner tree for $\mathcal{T}$ or decide that no such tree exists [**AC vol. 19**, 20].*

The Melzak-Hwang algorithm uses breadth-first search, merging, and reconstruction to construct the SMT in $O(n)$ time from the given topology, and is described further in the next section. Referring to figure 4 for examples of plausible topologies on a set of 5 points: reading from left to right (with vertices lettered clockwise beginning with the top vertex as point A), these topologies can be notated as:

1. $(AB)(CD)E$
2. $A(BC)(DE)$

3. $(AB)C(DE)$
4. $(BC)D(EA)$
5. $(EA)B(CD)$

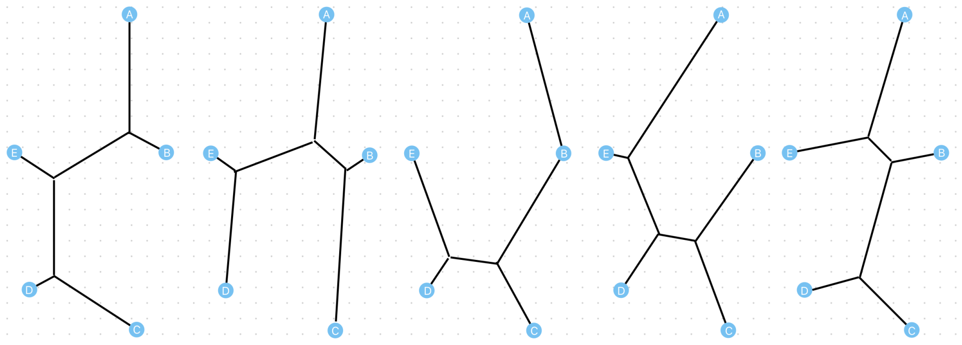

Figure 4: Steiner topologies on a set of 5 points.

Where the adjacency of any two elements using this notation indicate that they are connected (either directly, or through a Steiner point). In general, the number of possible topologies on a set of n points is the $(n-2)$ Catalan number:

$$C_{n-2} = \frac{(2n-4)!}{(n-1)!(n-2)!}$$

Which grows asymptotically as $O(n^4)$ [**AC vol. 29**, 14]. Thus we see that determination of the correct Steiner topology is what makes this problem NP-hard.

Algorithms

A list of several algorithms for finding the SMT.

1. **Melzak-Hwang algorithm** takes a given topology and constructs the corresponding Steiner tree in $O(n)$. We additionally assume that the resulting tree is non-degenerate, i.e. all edges of the relatively minimal tree have non-zero length. Letting $\mathcal{T}$ be a full Steiner topology will $n \geq 3$ terminals, we define a basic recursive algorithm.

   Let s be a Steiner point with neighboring vertices a, b, and c, with a and b terminals, and let e_{ab} be the equilateral point of a and b (i.e. the point such that Δabe_{ab} is equilateral), such that s is located on the opposite side of $\overline{ab}$ to e_{ab}. Then s must lie on the Steiner arc— that is, the arc formed from sweeping ae_{ab} to be_{ab}, hinged on e_{ab}. e_{ab} and s can be constructed in constant time from a and b. Further, the **Simpson line**— the line that is drawn from e_{ab} to s— intersects c. Now since $|se_{ab}| = |sa| + |sb|$, we can simple replace the points a, b, s with the point e_{ab} as the "pseudo-terminal", and proceed by taking this modified tree to be the new Steiner topology, $\mathcal{T}'$. This is the "merging step".

The "reconstruction step" is as follows. Given $\mathcal{T}'$ and its corresponding tree T', the location of s can be determined by taking the intersection of the line ce_{ab} with the Steiner arc. The complete algorithm consists of $n-2$ merging steps and $n-2$ reconstruction steps. After $n-2$ merging steps, the topology of the tree is simply two terminals connected by a line. Additionally note that the algorithm orders the Steiner points by depth by performing a breadth-first search (BFS) from the root r, which we obtain by selecting an arbitrary terminal.

2. The **GeoSteiner algorithm** is by far the most efficient currently known exact algorithm. It is, in a sense, a framework that can be filled out in various ways. Broadly, it is as follows:

   (a) Generation phase: construct a full set of Steiner tree candidates $\mathcal{F} = \{T_1, T_2, \ldots, T_m\}$ by efficient enumeration.

   (b) Concatenation phase: identify a subset $\mathcal{F}^* \subseteq \mathcal{F}$ such that $\mathcal{F}^*$ interconnects N and has minimum total length.

   To significantly reduce the amount of work required in enumeration of full Steiner tree candidates (FSTs), GeoSteiner uses implicit instead of explicit enumeration.

3. Experimental evidence shows that MSTs should be used as backbones in constructing good approximations to SMTs; however, it is sometimes necessary to deviate from the MST [**Zachariasen**, 10]. Thus, we attempt here to construct our own algorithm for the MST of a point set, starting with the MST and adding Steiner points.

   While the proposal of iteratively adding Steiner points angle in the MST less than 120 degrees may be fruitful in some cases, it does not take into account the fact that previously added Steiner point may need to be readjusted with each new addition of a Steiner points to maintain 120 degree angles; see figure 5. As a result, the following algorithm, constructed by myself, may be more practical. We shall refer to this algorithm as the **MST-polygonal algorithm**.

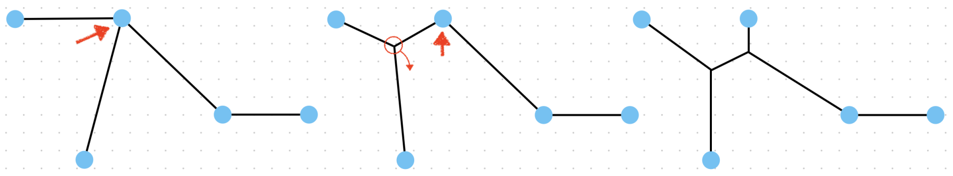

Figure 5: Steiner point adjustment.

   (a) Compute the MST T of the given point set P.

   (b) Examine all the angles in T, and create a list of all sequences of neighboring convex angles with at least one of the angles in each sequence being less than 120°.

(c) For each convex angle sequence (this is $n-2$ angles for an n-point sequence), compute the Steiner tree S of the corresponding n-gon, and replace the corresponding edges in T with those in S. (If the angles spiral inward, only consider the largest possible convex n-gon within this spiral.) Repeat this step with the new tree T', until there are no more convex angle sequences with ≥ 1 angle in each sequences being $< 120°$. The order in which we look at angle sequence is determined by some priority measure (e.g. sequences with smallest angles, smallest sum of weights of edges in the sequence, non-edge-sharing sequences first, etc.).

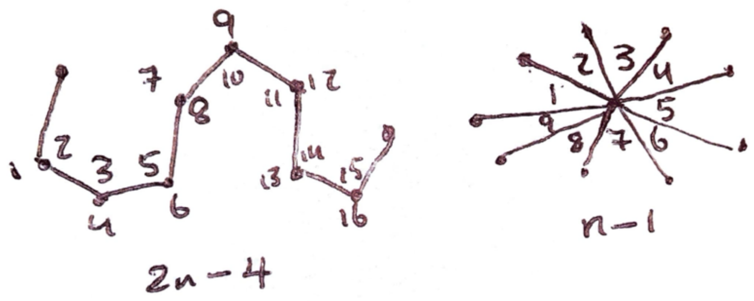

Figure 6: Number of angles in a tree.

Theorem. *In a tree consisting of n points, the total number of angles in the tree A (not including* $180°$ *angles around terminal points) is between* $n-1$ *and* $2n-4$*.*

Proof. See figure 6. $A = n-1$ occurs when there is only point with degree > 1, connected to all other points. Here, there are $n-1$ points around that central point and thus $n-1$ angles. $A = 2n-4$ occurs when points are connected sequentially by edges, giving all but the initial and final points (both terminals) a degree of 2. Each non-terminal point yields 2 angles, so $A = 2(n-2) = 2n-4$. □

As a result of the theorem, $n-1 \leq A \leq 2n-4$. However, a degree-2 point can only have 1 convex angle. Now beginning with a 3-point tree ($A = 1$), the addition of a point either increases A by 2 (if the point is added on the non-convex side of the only degree-2 point) or by 1 (if it is added anywhere else. Thus the upper bound for A will be less than $2n-4$— but regardless, A is linear in n.

(For n points, there will be no more than $n-2$ Steiner points— each of which has 3 angles— the total of $2n-2$ points will contain $3n-6$ angles, and so the number of angles in a Steiner tree will be approximately $3/2$ times the number of points, including Steiner points.)

Since the total number of convex angle sequences in any tree will be anywhere between 1 and $n-1$, we can conclude that the complexity of the MST-polygonal algorithm will be $O(nQ)$ (assuming we do not recount the angle sequences which we have already accounted for), where Q is the complexity of whatever n-gon Steiner algorithm we choose to utilize. Consequently, it would be useful to construct an algorithm that computes the Steiner tree on the vertices of a convex n-gon.

4. We now provide an algorithm which, given a point set P that corresponds to the vertices of a convex n-gon that is approximately regular, outputs the approximate Steiner points of that point set. The algorithm is in fact exact for all quads with rotational symmetry that are not "too thin" (as the edge between the Steiner points will then intersect the centroid of the quad) and any triangle.

   (a) If $n = 3$ and no angle is greater than 120°: return the Fermat point of the triangle.

   (b) If $n = 4$: return the 2 Steiner points. This is done by first taking the centroid of the quad, then for each of the 2 pairs of neighboring points in combination with the centroid, return the Fermat point.

   (c) Otherwise, return no additional points.

I implemented this algorithm in Python via Google Colab. Steiner point outputs (red) are shown below for four example sets. Link to code here.

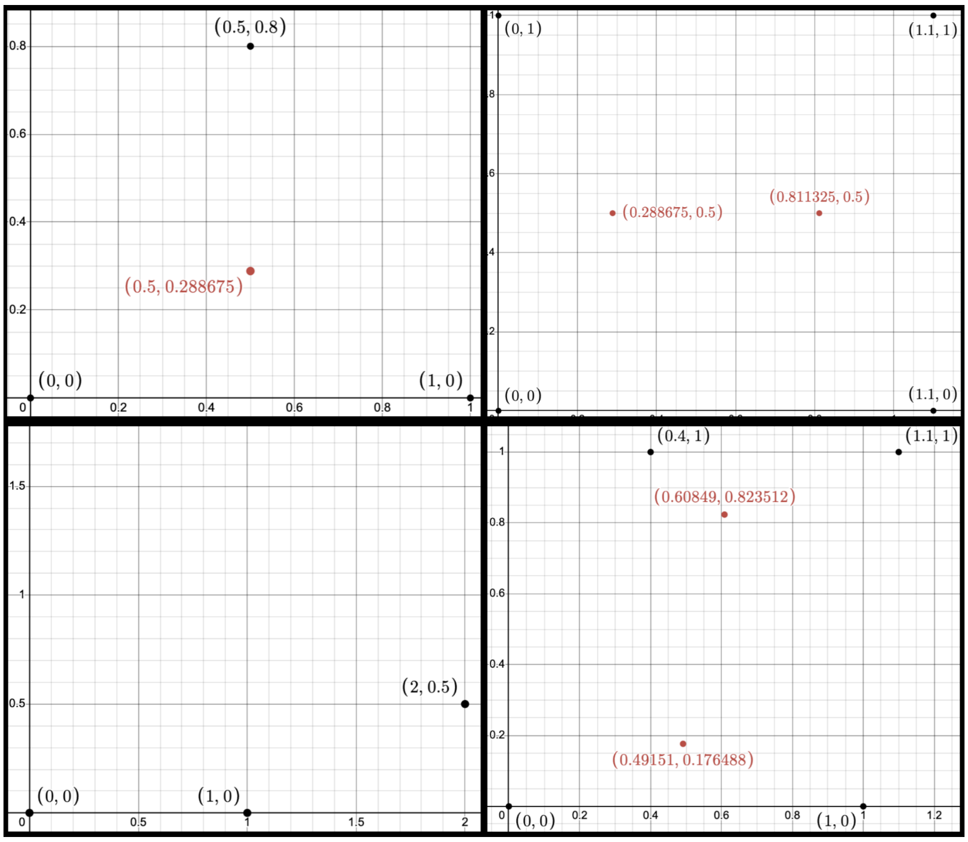

Figure 7: Outputs for implementation of polygonal Steiner point program.

References

[Gander] “Shortest Road Connecting Cities”, 2008. Nice brief article for an introduction and overview of the problem itself.

[Zachariasen] “Concatenation-Based Greedy Heuristics for the Euclidean Steiner Tree Problem”, 1997. Presentation of a class of $O(n \log n)$ heuristics for the Steiner tree problem in the Euclidean plane.

[Skiena] Stony Brook Algorithm Repository. List of known implementations of the Steiner tree problem.

[Brazil] “An exact algorithm for the Euclidean k-Steiner tree problem”, 2024. Describes an algorithm for the k-Steiner tree problem (constraint on number of Steiner points). Also summarizes the GeoSteiner algorithm.

[Swanepoel] “Approximate Euclidean Steiner Trees”, 2016. Provides references to a number of algorithms for the Steiner tree problem.

[AC vol. 29] *Optimal Interconnection Trees in the Plane*. Book from 2015 on the Steiner tree problem.

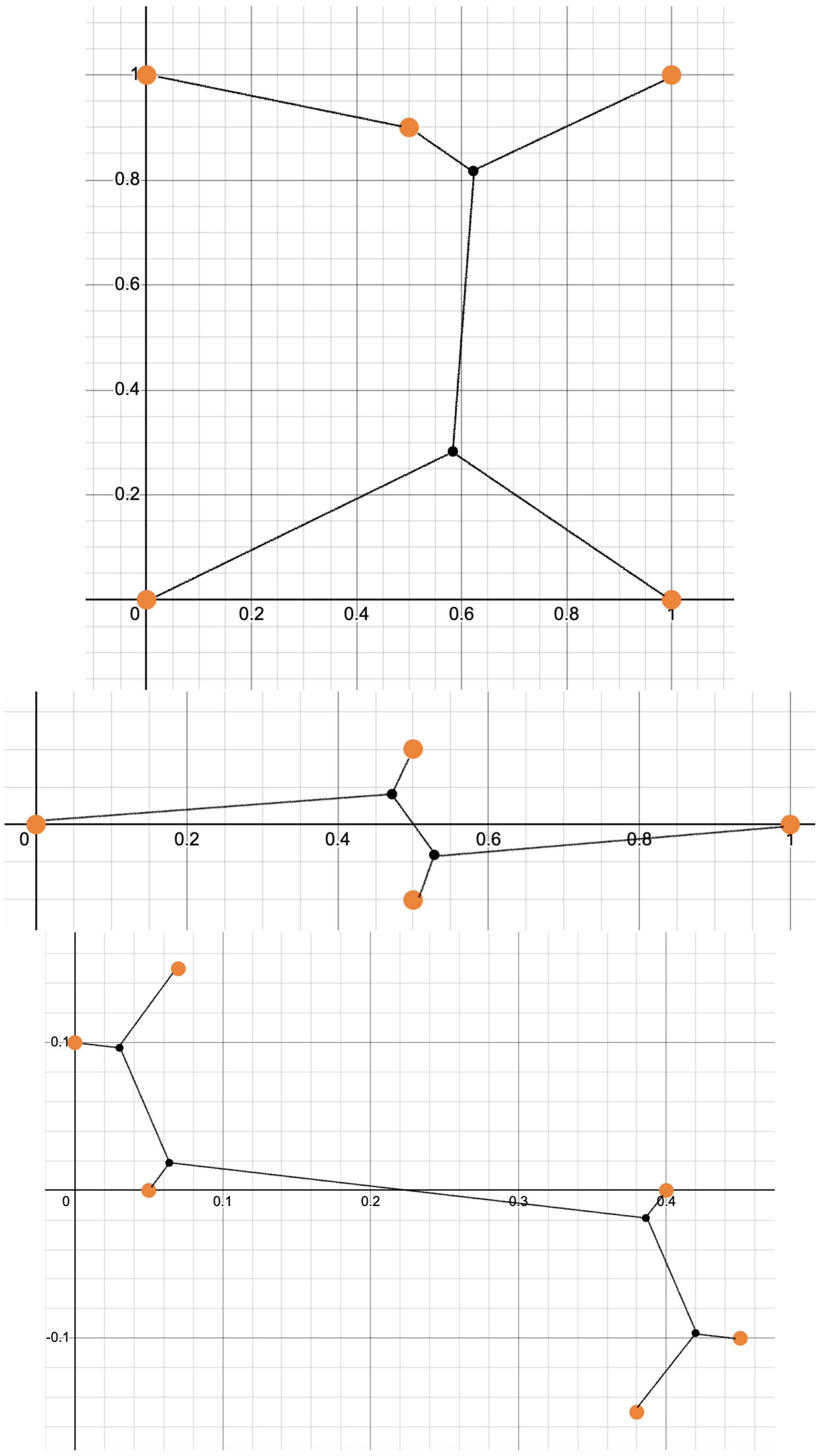

Figure 8: examples using GeoSteiner & Desmos