Project 3 Report: Divide Polygon into Four Pieces

Deep Ghosh, Donovan Hatch

Objective

Split a polygon into four equal areas by finding and making minimum length cuts. Each area of the polygon must have at least one connection to another area but does not need to be connected to all of the other areas. Determine what kind of line (straight, curve) will make the best minimum length cut in the polygon to make each quadrant.

Possible Applications

There are many different situations where being able to divide a polygon into any number of minimum cuts can be useful. For the following examples, we will assume the area is being split into 4 pieces in every scenario even though they can apply to any N number of splits.

One use case is being able to split up an area into 4 equal pieces for vacuum robots to clean. Each one will have the same amount of space to clean so we know that the work is being done as efficiently as possible. By using the min-cut to calculate the different areas, we also can minimize interference that they can have on each other if they were to go to the same position on the boundary at the same time.

Another use case is load balancing web content delivery networks (CDNs). CDN servers are placed in different states of the U.S.A. for the reason of delivering content to users faster and load balancing. We can use this formula to decide the location of the servers by splitting the country into 4 parts and drawing the min cuts mixed with user density. The min-cut here will help us position the servers in the more dense areas of user usage.

Intellectual challenges

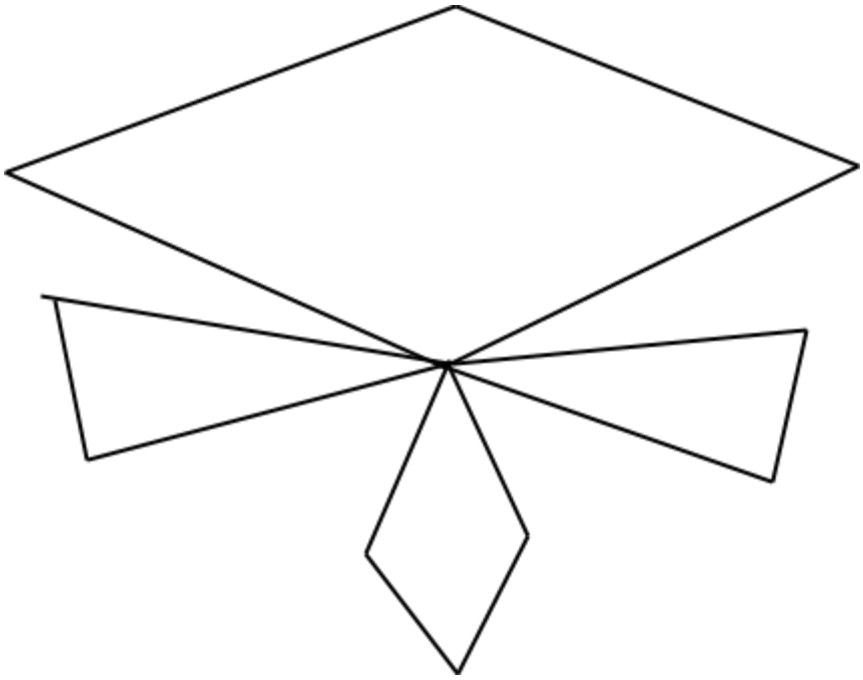
Polygons are made up of and make many different shapes. Some of these shapes are simple and have easier solutions such as a triangle, square, or any other regular polygon. However, if a polygon becomes irregular or has manifold vertices, a solution becomes much harder if not impossible depending on the polygon.

A solution for concave polygons with mincut can be NP hard. Refer to article 7, in the reference section for a proof.

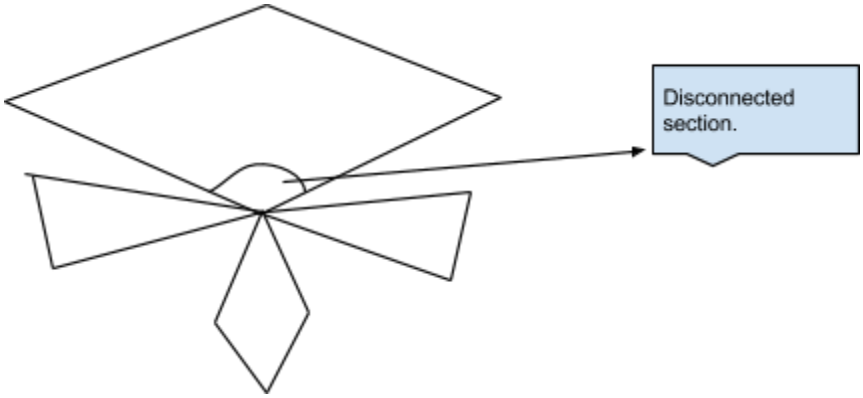
In most cases(regular polygons) a straight line cut will generate a result which is a very close approximation of the minimal cut but not the minimum cut. We can prove that for a given perimeter, a circle covers the largest area. So from this statement we can draw on the fact

that for a fixed area A, a circle will cover the area with the most optimum perimeter. Referring to article 9 in the reference material, we can see that it can be proven for a convex polygon that the minimal cut will form a set of line(s) that has at most a combined length of Square root of (2*PI*A), where A is the area to be cut.

So we can derive that in the worst case scenario for a given regular polygon, a curve would be better suited to create a minimal cut.



We have provided the above image as an example of a manifold polygon which will not have a solution. The four “leaves” of the image are of unequal area. Trying to generate the minimal cut to divide the example polygon into 4 parts would result in a cut in one of the quadrants which generates a disconnected piece. Below image is representative of how we picture the problem to be.



Reduced Versions of Problem

More complex polygons can be split up into smaller pieces in order to make the problem simpler. For example, any polygon can be split up into triangles. As per our assertions made in the above section, we will split the problem into regular convex polygons, regular concave polygons, Manifold polygons and Self Intersecting polygons.

Possible Solutions and Prior Art

Liquid Density

If we imagine our polygon as a vertical vase, we can consider pouring in N different liquids will all different densities but have the same area with a total equaling the vase total area. When this is done, a heavier density liquid will sink while a lighter density liquid will float. If we pour all of the liquids in, they will be mixed and fill up all of the pixels. We can start to move the denser pixels in the direction designated down direction (while staying inside of the constraints of the polygon) of the vase and swap with lighter pixels as we pass it.

Sweep Line Algorithm

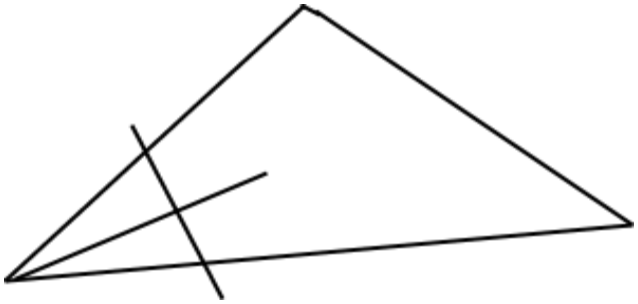
This algorithm is described as part of the article “The Area partitioning problem” by Hannah Bast and Susan Hert. Reference to this article is provided below as [7]. The algorithm consists of 2 parts.

The first part of the algorithm starts by breaking a non convex polygon into a collection of convex polygons, which do not contain Steiner points. The convex decomposition of the non-convex polygon is done by first simply merging adjacent convex polygons. When the condition to merge polygons are not met then only the sweep line is used to divide the convex pieces.

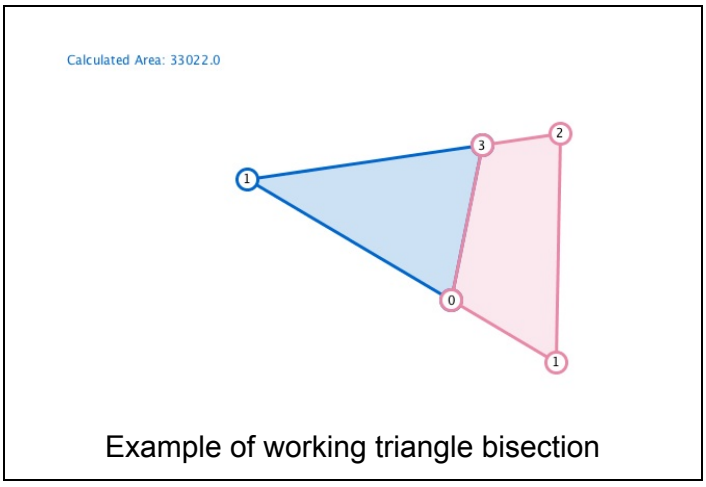
This algorithm achieves O(pn) running time, where p is the number of connected pieces of given area and n is the number of vertices.

Theoretical Solution on Reduced Problem Polygon

We are going to present a solution which is not optimal but produces a reasonable area partition. Also all of the minimal cut length that are generated using this algorithm are well within the range specified above (Square root of (2*PI*A)). We are using a modified version of the Sweep line algorithm. Let us take for example a triangle bisection problem.



We start by creating sweep lines perpendicular to the angle bisectors of each vertice. Let the perpendicular be PL and the angle bisector be AB for the vertex V. As stated earlier we know that for a regular convex polygon the maximum perimeter length is Square root of (2*PI*A). Since A is known to us, we can deduce the radius of the circle which can have semi perimeter of Square root of (2*PI*A). Once we receive the radius we move the line PL by that distance, say d, from the vertex V. That is the maximum distance our minimum cut can be from the vertice. Then we start moving the line backward and forward while checking the area covered by the bisection polynomial to be as close to the percentage of the target area. Then we repeat this process for all three vertices, and then calculate the piece with the least cut length.

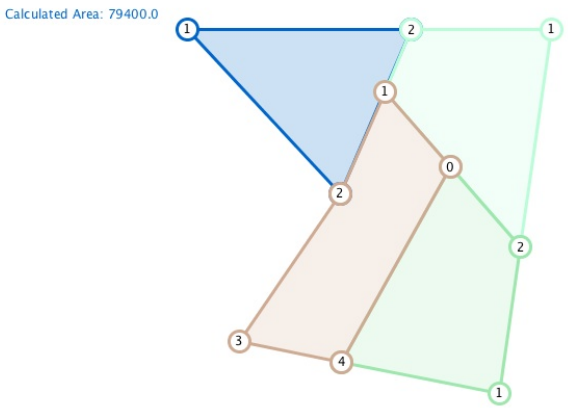


Example of working triangle bisection

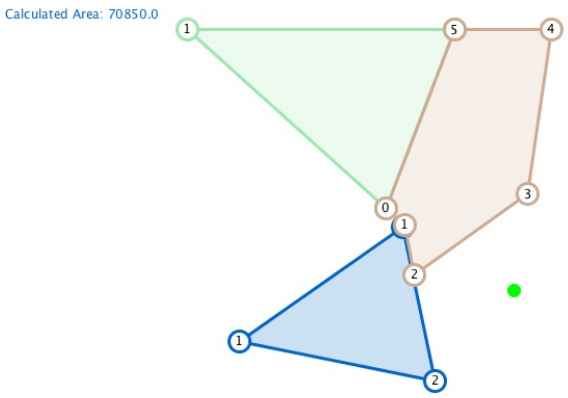
Challenges:

While this implementation works with convex polygons well enough, handling concave polygons is inconsistent. As long as the distance between the concave angle vertice and the non-adjacent edges are large, this implementation will be able to fetch the minimal cuts for the polygon. However, if the angle is too large, the algorithm will fail.

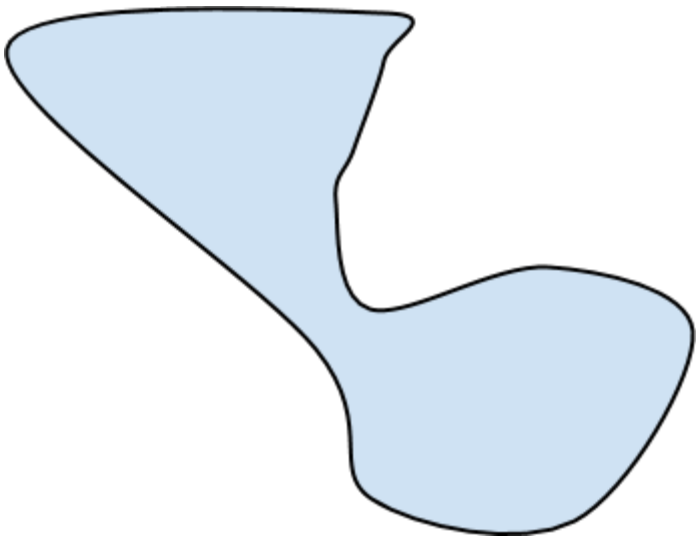
Manifold polygons and self intersecting polygons are not supported.



Example of concave bisection



Example of concave bisection not working



Example of shapes which may not have a straight line solution.

Conclusion

We have tested this algorithm to handle up to 9 splits on a polygon on close to min-cuts. Future work would be to see how we can do this with a curved line to get the minimal cut as well as seeing if we can expand the algorithm to work on all concave polygons.

References

1. http://sunzi.lib.hku.hk/ER/detail/hkul/2687548
2. http://download.springer.com/static/pdf/51/art%253A10.1007%252FBF01388467.pdf?auth66=1411755963_ca7b91d8c92307fce2a363bda87f48e0&ext=.pdf
3. http://www.math.unam.mx/~urrutia/online_papers/dissections.pdf
4. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=CA11B72EEB501BCF4610A0EF18246E98?doi=10.1.1.308.3591&rep=rep1&type=pdf
5. https://www.cs.ucsb.edu/~teo/papers/AAMMel.pdf
6. http://people.seas.harvard.edu/~minilek/papers/sandwich.pdf

7. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=3C982983E1F9E02DFB707BB983940514?doi=10.1.1.29.7473&rep=rep1&type=pdf

8. http://math.science.cmu.ac.th/thaijournal/special%20issue%202007/Spacial%20Issu%202007/10.pdf