CS91T: Computational Geometry **Fall 2023**

Partitioning Regular Polygons into Circular Pieces

David Yang

1 Problem Setup

In this project, I explored TOPP Open Problem 59 [1], the problem of partitioning a polygon into pieces, each of which is as "circular" as possible. This problem was first studied by Mirela Damian and Joseph O'Rourke [2].

One notion of the circularity of a given polygon is its *aspect ratio* – the more "circular" a partition, the closer its aspect ratio is to 1.

> **Definition 1** (Aspect Ratio & Circularity)
>
> The **aspect ratio**, or **circularity**, of a polygon is the ratio of the diameters of the smallest circumscribing circle to the largest inscribed disk.
>
> *The distinction betIen circle and disk is made to emphasize that the circumcircles can intersect, whereas the disks cannot.*

In [1] and this project, I focus on minimizing the aspect ratio across the pieces in a convex partition of a regular polygon.

> **Definition 2** (Partition)
>
> A **partition** of a polygon $\mathcal{P}$ is a collection of polygonal pieces $P_1, P_2 \ldots$ such that $\mathcal{P} = \bigcup P_i$ and no pair of pieces share an interior point.

Furthmore, I add the restriction that the partition of a regular polygon P must be convex, meaning that each of its polygonal pieces are themselves convex.

In summary, the problem is as follows:

> **Goal** — Find a convex partition of a regular polygon that minimizes the maximum circularity across all pieces in the partition.

My main goal for this project was to create an interactive visualizer that allows the user to specify a partition of a square, and for the program to then calculate the circularity[1] of the selected partition.

2 Related Problems

Calculating the circularity of an individual polygon is a non-trivial process, and one that [2] glosses over. To find the circularity, we must find the ratio of the radii (or diameters) of the smallest circum-

[1] circularity here refers to the maximum aspect ratio across all pieces of the partition.

scribing circle and largest inscribed disk; these problems themselves are their own Computational Geometry problems.

2.1 Smallest Enclosing Circle

2.1.1 Introduction

Definition 3 (Smallest-Circle Problem)

The **smallest-circle problem** (also known as minimum covering circle problem, bounding circle problem, least bounding circle problem, smallest enclosing circle problem) is the computational geometry problem of computing the smallest circle that contains all of a given set of points in the Euclidean plane.

One can think of this problem as a facility-location problem. Given n customers, by finding the smallest circle enclosing all of customers' locations, we find the optimal location of the facility (the center of the smallest enclosing circle), in the sense that the farthest distance that any customer must travel to reach the facility is minimized.

It turns out this computational geometry problem is precisely what we need to determine the smallest circumscribing circle for each of our polygonal partition pieces, which we will then use in our circularity calculations.

2.1.2 Implementation

Rather than implementing the smallest enclosing algorithm from scratch, I make use of an online implementation [3] that is a variant of Welzl's algorithm; this algorithm runs in expected $\Theta(n)$ time, with randomization. Since the user is not expected to specify a partition into polygons with thousands of vertices, even a brute-force algorithm could do.

2.2 Largest Inscribed Disk (Indisk)

2.2.1 Introduction

The Largest Inscribed Disk problem, which we require here, is similar to the Largest Empty Circle (LEC) problem, discussed in de Berg et. al. and in, surprisingly, a Swarthmore CS97 Senior Conference paper by Megan Schuster in 2008 [4].

However, the Largest Empty Circle[2], which finds the largest circle that contains no points in P and is also centered inside the convex hull of P, is in fact **not** the same as the Largest Inscribed Disk. In fact, notice that the LEC could intersect the convex hull of P, which is not a desired property of the largest indisk of P. That being said, the LEC problem can be solved in $O(n \log n)$ time, and my implementation can be found in *largestemptycircle.py*.

[2]for additional information, the LEC is always centered at either a vertex on the Voronoi diagram for P or at the intersection of the convex hull of P and one of its Voronoi edges. Consequently, the center of the circle that solves the LEC problem is the Voronoi vertex that maximizes the distance to the closest site.

2.2.2 Solution Methods

One solution to the Largest Inscribed Disk problem expands on the solution to the LEC problem. To ensure that the LEC is enclosed in the polygon, one can find the Voronoi diagram of both the vertices of the polygon as well as its edges. VRONI is a tool that seems to do this, but it is not open-source. After finding the Voronoi diagram of the *edges* of the polygon, rather than the vertices, it turns out the center of the Largest Inscribed Disk must be one of the Voronoi vertices. Then, to find the indisk itself, we can find the Voronoi vertex that maximizes the minimum distance to the edges, and take that vertex as the center and the minimum distance as its radius. Unfortunately, I could not find any implementations for Voronoi diagrams of line segments (representing the boundary edges of a polygon) online, so I attempted a different solution, involving a concept known as the Chebyshev center.

Definition 4 (Chebyshev Center)

The **Chebyshev center** of a set C is the center of the largest ball that lies inside C.

Note that when C is a convex polygon as it is in our problem, then finding the Chebyshev Center is a convex optimization problem that can be solved with linear programming.

2.2.3 Implementation

To find the Chebyshev center, I set up a linear program. The goal is to maximize the distance from the Chebyshev center to the nearest edge of the polygon, with linear constraints defined as the edges of the polygon itself (to ensure that the Chebyshev circle is fully contained in the polygon). The code for this linear program is available in *chebyshevcenter.py*, and was adapted from Scipy's Halfspace Intersection page.

3 Circularity of Partitions Visualizer

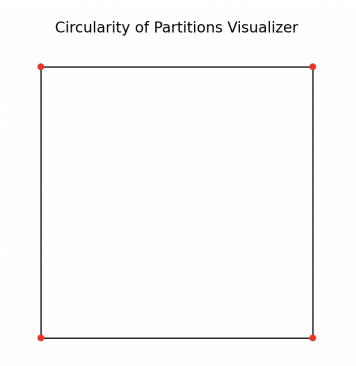
3.1 Using the Visualizer

Note: when using the visualizer, keep the Terminal open to view the output.

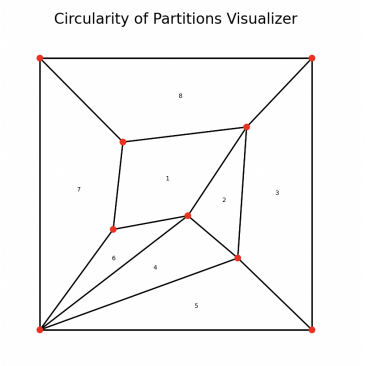
The user input process is complicated, and works as follows:

- Hit the 't' key to set up the visualizer
- Left click to add vertices to the diagram
- Hit the 'v' key to switch to partition mode
- While in partition mode, split the regular polygon into convex polygonal pieces, with vertices selected in previous step
  - Left click the vertices, in clockwise order, of each piece you want to create
  - Right click when done with the polygon
  - Continue this process until the entire square is split into convex, polygonal pieces, each of which has been selected (there should be a number label covering each partition region).

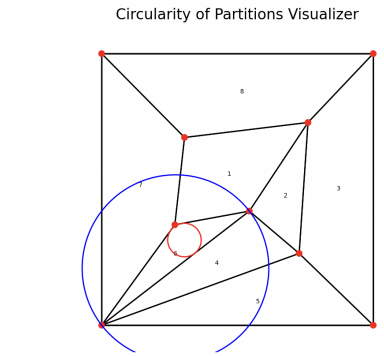
- Hit the ‘d’ key to calculate circularities. The smallest circumcircle and largest indisk for the limiting polygonal piece (the piece with the greatest circularity) will be drawn and the circularity of that piece will be printed.



Setup Scree (‘t’ key)



User-Specified Partition (‘v’ key)



Circumcircle and Indisk (‘d’ key)

```
The maximum circularity for a polygonal piece in the given partition is 5.565725209105727
```

The above process can be repeated for any user-specified partition of the square, given that all the polygonal pieces in the partition are convex. The maximum circularity over all pieces will be calculated and printed.

4 Future Work: Extensions

In my visualizer, I focus on the case where the regular polygon to be partitioned is a square; this is because its optimal circularity remains to be studied [2]. However, I could and should extend my work to visualize any regular polygon.

Furthermore, there is some functionality in the visualizer that I could add and fix. I implemented an undo key to help in my debugging process, but there are aspects of the undo that remain to be completed (for example, undoing a polygon currently does not remove its corresponding label). I would also try to add buttons to my visualizer to allow for better user interaction, rather than just printing the output to the terminal as it currently stands.

Finally, it may help for my visualizer to be dynamic, in the sense that it may allow for users to move the selected vertices/for the polygons to change as the user moves them. This would be an ideal piece of functionality that will be much more difficult to implement, but could lead to stronger visualization.

5 Conclusion

I created an interactive visualizer where a user specifies a convex partition of the square. The program will calculate the maximum circularity over all polygonal pieces and draw the largest inscribed disk and smallest circumscribing circle for that polygonal piece.

This project is part of the broader problem of finding the optimal partition (minimum circularity over all pieces) of a regular polygon, which has many applications, as specified in [2].

Thank you to Professor Neil Lutz for his guidance and support, and for a great semester in CS91T: Computational Geometry.

References

[1] Erik D. Demaine, Joseph S. B. Mitchell, Joseph O'Rourke. *The Open Problems Project.* Originated from Computational Geometry Column 42. Internat. J. Comput. Geom. Appl., 11(5):573-582, 2001. Also in SIGACT News 32(3):63-72 (2001), Issue 120.

[2] Mirela Damian and Joseph O'Rourke. *Partitioning Regular Polygons into Circular Pieces I: Convex Partitions.* In Proc. 15th Canad. Conf. Comput. Geom., pages 43-46, 2003. arXiv:cs.CG/030402.

[3] Nayuki. *Smallest Enclosing Circle.* Nayuki Smallest Enclosing Circle GitHub

[4] Megan Schuster (2008). *The Largest Empty Circle Problem.* Swarthmore College, CS97: Senior Conference, 2008