Computational Geometry : Project

Charlotte Gullentops - 000463980
Eline Soetens - 000459750

17 November 2021

1 Presentation of the topic

The topic we will present here is how to find a maximum area rectangle inscribed in a convex polygon, taking into account that this rectangle must not necessarily be aligned with the x and y axis. There is a large sample of literature dedicated to finding, in a convex polygon, the largest area rectangle with edges parallel to the axis. However the case of finding rectangles not aligned with the axis is less studied. The main article [1] we used for this topic presents an heuristic algorithm that can find rectangles with an area close to the maximum area possible ($area(R_found) \geq (1-\varepsilon) \times area(R_{optimum})$) with a probability t.

2 Description of the algorithm

The main principle of the algorithm is to find potential directions for rectangles then compute the biggest area rectangle possible for those directions. The algorithm we follow is an heuristic one, meaning that we will not necessarily find the largest rectangle. However, this randomized algorithm has a probability t to find a rectangle inside the polygon which has at least (1-ϵ) times the optimum area.

2.1 Finding ϵ-directions

Mainly Charlotte

The first step is to find directions close enough to the orientation of one of the sides of the optimal rectangle so the area of our rectangle would almost be the largest possible. We will call these direction $\epsilon-$close directions. This kind of direction is defined as follow :

We consider that we know the optimal rectangle (R_{opt}) and we call the intersection of its diagonals s. The segment $\overline{ab}$ is one of its short side and the point d is the midpoint of $\overline{ab}$. The two triangles T_1 and T_2 are defined using vertices s, d and a third vertex $f_1 := d + \epsilon(b - d)$ or $f_2 := d - \epsilon(b - d)$. T_3 and T_4 are obtained using the same principle with the other short side. An $\epsilon-$close direction is a direction that intersects $\overline{f_1 f_2}$ while going through s. Intuitively, we can see that a rectangle closely aligned with the optimal rectangle will have a similar area. The complete proof is detailed in [1].

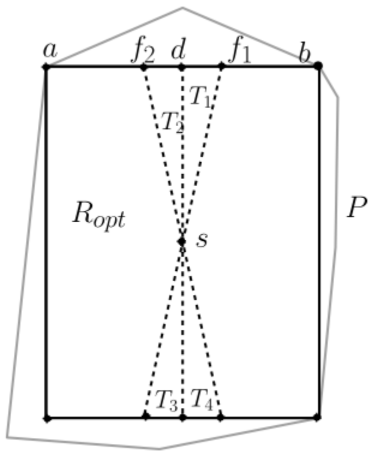

Figure 1: A largest rectangle R_{opt} in a convex polygon P. (from [1])

In practice, the idea to find a set of such $\epsilon-$close directions is to take a set U of $\Theta(1)$ points with a random uniform distribution, then take another set V of $\Theta(1/\epsilon)$ points. The quantity of points taken in the second set is determined by the ratio the user chose at teh beginning. To find those points, Knauer and al [1] suggested several methods which can all be resumed by : take random points inside the polygon. Even though we did not exactly use their method (which consisted in choosing randomly a height between the two extremes y, then taking the largest width of the polygon at that height and finally taking one random point on this line), we did something similar: take a random y between the two y-extremes then take a random x between the x-extremes. If that point is outside the polygon, compute another x-coordinate and check again. We have a good probability of having $\epsilon-$close directions composed by a point in U and a point in V.

Once we have those sets of points, we need to check if they compose an ϵ-direction. To do so, we associate each point u of U with each point v of V and for each pair, compute its x and y deviation. Then, we artificially create a point q whose coordinates are those of s, incremented by the newly computed deviations. Therefore, we can check if the segment $\overline{sq}$ or its projection will ever intersect $\overline{f_1 f_2}$. If yes, we add the pair (u,v) to the set of ϵ-directions.

2.2 Additional part : rotate our polygon accordingly to our direction

2.3 Finding the largest rectangle in a particular direction

Mainly Eline

Once we found all these possible directions, the main part remains; we must compute the biggest area rectangle in that direction which we will do using [2]. In the original article of C. Knauer et al [1], they use H. Alt's work [3] which compute the largest rectangle in logarithmic time. As Alt's article is based on our [2], we took the liberty to apply the earliest one. Nevertheless, it only finds the largest inscribed rectangle in $O(log^2 n)$ so the total complexity of our algorithm is not the same as the one announced in our first article [1]. As those algorithms compute the largest axis-aligned rectangle, we first need to rotate our polygon accordingly to the $\epsilon-$close direction being tested so the algorithm aligns the rectangle with that particular direction and not with the axis.

3 Pedagogical presentation

Here is the link of our Github repository : https://github.com/ElineSoetens/testProjetGeom.

You can also visit our GitHub page and play with the code : https://elinesoetens.github.io/testProjetGeom/

References

[1] J M Schmidt C Knauer, L Schlipf and H R Tiwary. Largest inscribed rectangles in convex polygons. *J. Discrete Algorithms*, 13:78–85, 2012.

[2] Paul Fischer and Klaus-Uwe Höffgen. Computing a maximum axis-aligned rectangle in a convex polygon. *Inf. Process. Lett.*, 51:189–193, 1994.

[3] J. Snoeyink H. Alt, D. Hsu. Computing the largest inscribed isothetic rectangle. *Proceedings of 1995 Canadian Conference on Computational Geometry*, pages 67–72, 1995.