Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Algorithm for finding the largest inscribed rectangle in polygon

Zahraa Marzeh*[1], Maryam Tahmasbi†[2] and Narges Mirehi‡[3]

[1,2,3]Department of computer science, Shahid Beheshti University, G.C., Tehran, Iran.

ABSTRACT

In many industrial and non-industrial applications, it is necessary to identify the largest inscribed rectangle in a certain shape. The problem is studied for convex and non-convex polygons. Another criterion is the direction of the rectangle: axis aligned or general. In this paper a heuristic algorithm is presented for finding the largest axis aligned inscribed rectangle in a general polygon. Comparing with stare of the art, the rectangles resulted from our algorithm have bigger area. We also proposed an approach to use the algorithm for finding a rectangle with general direction.

ARTICLE INFO

Article history:
Received 11, August 2018
Received in revised form 15, March 2019
Accepted 13 April 2019
Available online 01, June 2019

Keyword: non-convex polygon, IIC, inscribed rectangle, longest path, largest cycle.

AMS subject Classification: 05C78.

1 Introduction

In many industrial and non-industrial applications, it is essential to identify the largest inscribed rectangle in a certain shape. For instance, one of the applications of this problem is for labeling different areas in contours. Most of the research done with respect to

*zahramarzayahoo.com
†Corresponding author: Maryam Tahmasbi. Email: m_tahmasbi@sbu.ac.ir
‡n_mirehi@sbu.ac.ir

search for inscribed rectangle is related to convex polygons. In order to find the inscribed rectangle in a polygon, it is possible to take a specific direction (axis-aligned) into account or not in the sense (arbitrary orientation) that we generally search for the inscribed rectangle (Figure 1).

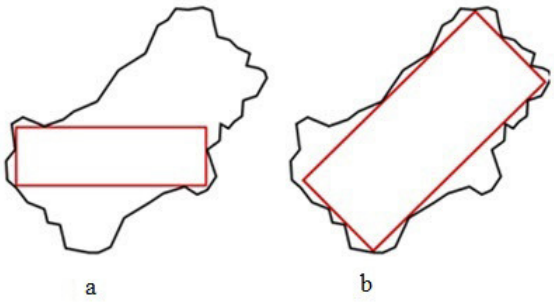

Figure 1: Largest area rectangle with: a) Axis-aligned. b) arbitrary orientation.

In order to measure the largest rectangle in specific direction in a convex closed-contour polygon, it is possible to refer to algorithms proposed by Knauer et al. [3], Cabello et al. [2], and Alt et al. [1]. In this algorithm [3], in order to find the largest rectangle in a polygon, one can choose four groups of edges of given polygons (i.e. Edges of e_1, e_2, e_3, e_4) and separate all of the rectangles that their three vertices are on e_1, e_2, e_3 edges and the last vertex is exactly on e_4 or the edges cut e4 .For example, if the rectangle $R \subset P$ has one corner on polygon area of P or the two forming corners of one edge is on the area of a P polygon, and then by moving one edge it is possible to find an inscribed rectangle with the largest area.
Thus, the largest inscribed rectangle has two statuses: Status 1: having to opposite corner ($ex : a \ \& \ c$) on the area of polygon. Status 2: three corner (a, b, c) on the area of a polygon (see Figure 2). In order to find the largest inscribed rectangle in digital objects,

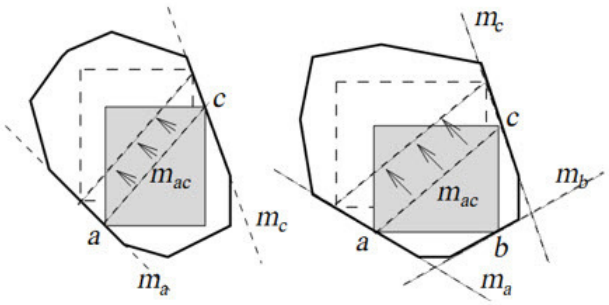

Figure 2: Two possible statuses for the largest inscribed rectangle.

Sarkar et al. [4], first divided the area into a network with stable size and then into a series of convex subareas and also searched the largest inscribed rectangle by defining a

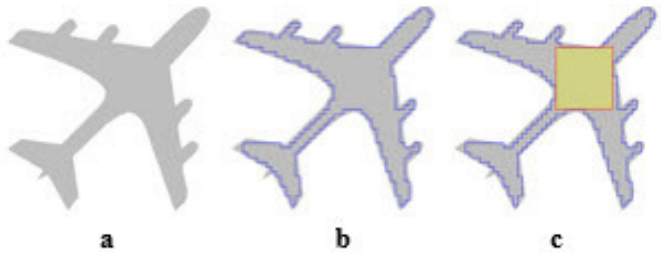

Figure 3: (a) The digital object, (b) Inner isothetic cover, (c) Largest Rectangle.

series of rules for convex subareas (Figure 3). This paper is inspired by the two methods of Alt et al. [1] and Sarkar et al. [4] for finding largest inscribed rectangle in general polygon. In this algorithm, by drawing horizontal and vertical lines from the vertex and some extra points on the edges of polygon, it is divided into rectangular shapes. Then the algorithm searches for the largest inscribed rectangle that is formed by the combination of some rectangular regions.

This algorithm is easier to follow and the performance on some images show better results of this algorithm. Sarkar et al. [4] solution depends on the precision of IIC while in the proposed algorithm, there is no such dependency and the answer is more precise. The steps of our algorithm are as follows:
1. Dividing polygon into subareas
2. Identifying rectangular subareas
3. Forming RA graph
4. Identifying cycles and paths and identifying the largest rectangle
The paper is organized as follow: The main algorithm is presented in section 2. In Section 3 experimental study and comparing the results with other studies is shown. Section 4 presents the conclusion and future study.

2 Main result

2.1 Dividing polygon into rectangular areas.

In this approach, first we divide the polygon to smaller polygons, most of these polygons are rectangle. Then we find the largest inscribed rectangle by joining these rectangles together. The steps are as follows:
1. Finding the smallest rectangle containing (SAR) the polygon (figure 4).
2. Drawing horizontal and vertical lines from the vertices of polygon ignoring the repetition lines (Figure 5). 3. As shown by Alt et al. [1], and it was already stated, the vertices of the largest rectangle are not necessarily on the vertices of the polygon, but may also be on the polygon edges. Therefore, the drawing of horizontal and vertical lines from the polygon vertices is not sufficient. We draw vertical and horizontal lines from the crossing

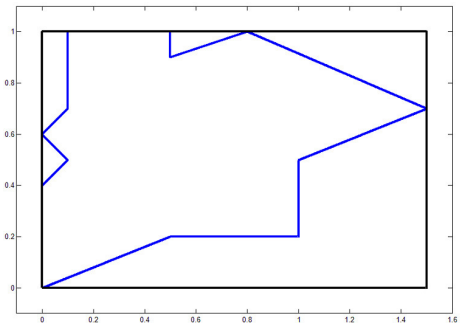

Figure 4: A polygon and its smallest area rectangle.

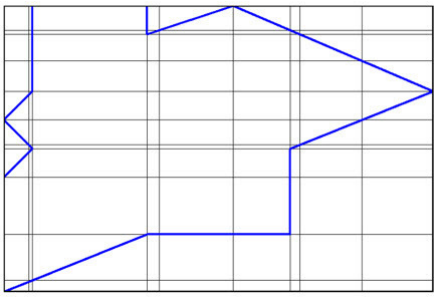

Figure 5: Drawing vertical and horizontal lines from the vertices of polygon.

points of the lines in the previous steps with the edges of polygon and repeat this process for two times (Figure 6).

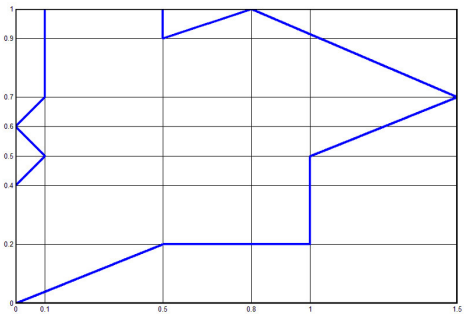

Figure 6: Adding vertical and horizontal lines on the crossing points with the polygon edges in SAR.

As a result, SAR is divided into rectangular subareas. In order to give address for the rectangles in the next steps, they are numbered from top left to bottom right. Since the goal is to look for the largest inscribed rectangle inside a polygon, the rectangles located completely inside the polygon are identified and other rectangles that are outside the polygon are removed. The method was explained in the following section.

2.2 Identifying rectangular subareas outside the polygon.

In order to speed up identifying the rectangular subareas outside the polygon, we identify the ones that is probable to interfere with polygon edges.
To do so, the edges of the polygon are divided into 8 types. The type of each edge in the polygon is recognized by algorithm 1. In this algorithm the type of edge can be identified by walking on the edges of the polygon in counterclockwise order and examining the two end points of each edge. The classification is shown in Figure 7.

Algorithm 1: Examining the type of edges in a polygon

for edge=1:end
if $(x_{start} < x_{end} \;\; \&\& \;\; y_{end} > y_{start})$
then $edge - type \leftarrow 01$
else if $(x_{start} > x_{end} \;\; \&\& \;\; y_{end} > y_{start})$
then $edge - type \leftarrow 02$
else if $(x_{start} > x_{end} \;\; \&\& \;\; y_{end} < y_{start})$
then $edge - type \leftarrow 03$
else if $(x_{start} < x_{end} \;\; \&\& \;\; y_{end} < y_{start})$
then $edge - type \leftarrow 04$
else if $(x_{start} = x_{end} \;\; \&\& \;\; y_{end} > y_{start})$
then $edge - type \leftarrow 05$
else if $(x_{start} > x_{end} \;\; \&\& \;\; y_{end} = y_{start})$
then $edge - type \leftarrow 06$
else if $(x_{start} = x_{end} \;\; \&\& \;\; y_{end} < y_{start})$
then $edge - type \leftarrow 07$
else if $(x_{start} < x_{end} \;\; \&\& \;\; y_{end} = y_{start})$
then $edge - type \leftarrow 08$
end

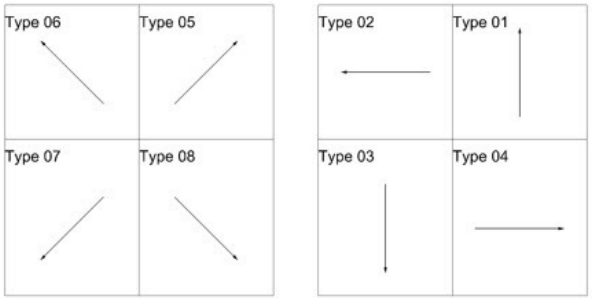

Figure 7: Types of polygon edges.

For instance, in Figure 8, the type of edges is shown for a polygon.
In the next step, for each edge of the rectangle that the given edge is one of its diagonals is considered and the rectangles that are within range are selected. For instance, the subareas of an edge are shown in Figure 9.

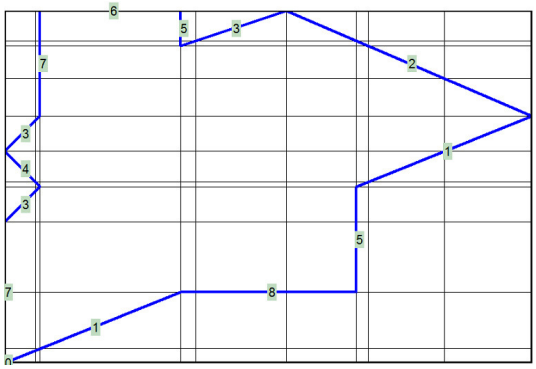

Figure 8: Identifying the type of edges in polygon.

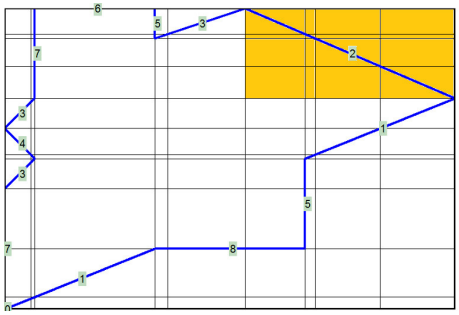

Figure 9: The subarea around the given edge.

The normal vector for each edge in the polygon (V_1) toward outside the area is calculated. Each edge is identified by using the internal or external rectangles around one edge. Algorithm 2 identifies the inside and outside rectangles using these vectors.

Algorithm 2: Identifying the internal subareas

for edge=1:end
  for rectangle=1:end
    $v_1 \leftarrow$ Normal from polygon edge
    $v_2 \leftarrow$ Normal from rectangle vertex
    if $v_1.v_2 \geqslant 0$ **then**
      $rectangle - inside \leftarrow$ Yes
    else
      $rectangle - inside \leftarrow$ No
    end if
  end
end

In this algorithm, for every edge of the vertices of rectangles in the range, they are numbered like in the Figure 10 in counter clockwise order and it is drawn from the vertex that its number is equal to the edge and a vertical vector is drawn on the edge (V_2).
If the vectors V_1 and V_2 are in the same direction, the rectangle is inside the polygon and otherwise, it is outside (Figure 11). For instance, the internal subareas for a polygon is

Figure 10: Numbering the vertices of rectangular subareas.

shown in Figure 12.

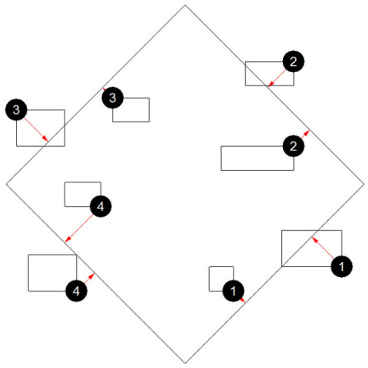

Figure 11: Distinguishing internal subareas from external ones.

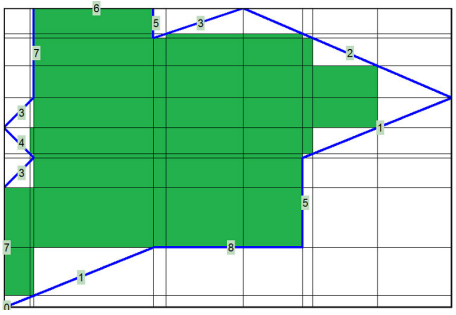

Figure 12: Internal subareas.

2.3 Constructing RA graph.

In order to simplify the process and easier identification the rectangular regions, a graph is constructed to show the rectangular subareas and their adjacencies. There is a vertex for each subarea and two subareas are adjacent in the graph is their share an edge. We call this graph, the RA graph. The subareas outside the polygon and identified in the previous step, form as isolated nodes without any edges connected to them. The vertices of RA graph are the vertices of a complete mesh but some edges are missing. Figure 13 shows the RA graph of polygon in Figure 12.

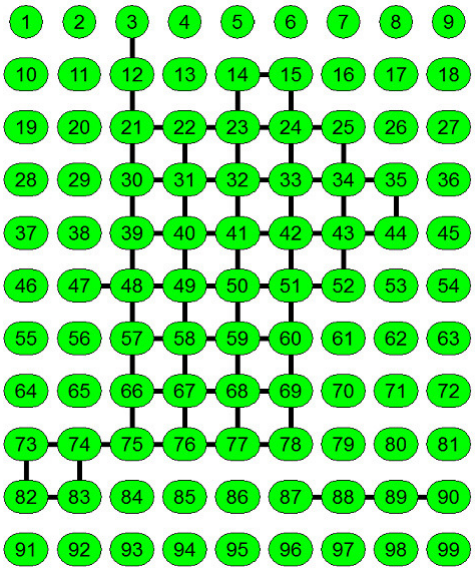

Figure 13: Equal graph in the rectangular subareas.

2.4 Identifying cycles and routes

In RA graph, each path of vertical edges, each path of horizontal edges and each cycle that is a rectangle in the graph show a rectangle in the polygon. To find the largest rectangle, we need to find the longest paths of these types and the biggest cycle.

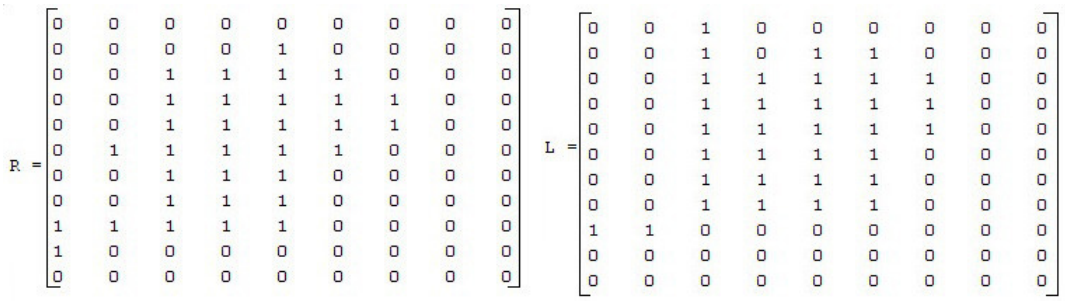

Figure 14: Matrices of L and R.

We assign two nm matrices L (and R) that shows the edges between each node in RA graph with the node below (right) it. If a node has edge to below (the right) node, the corresponding element in L is 1 otherwise it is 0. For example, Matrices R and L for the above mentioned graph is as follows in Figure 14:

Algorithm 3: identifying cycles and routes.

for i=1:N

  for j=1:M

    if $L(i,j) > 0$ && $R(i,j) > 0$**then**

```
        create H and V matrix
        if H(1), V(1) ≠ 0 then
            for k = 2 : V(1) + 1
                if H(k) ⩾ 1 then
                    for l = 2 : H(k) + 1
                        if V(l) ⩾ i − 1 then
                            create cycle
                        endif
                    end
                endif
            end
        endif
    endif
  end
end
```

In this algorithm, form every node in R and L which is more than zero and it is possible to form a cycle, the whole rectangular cycle is calculated from that node. For every node two vectors of V and H are formed. These two vectors show how much one can move from node to the right and below in the graph and find another vertex diagonally in front of that node to form the rectangular cycle. In order to form these two vectors, the longest horizontal and vertical paths are identified from each node to right and below. The length of vector H is equal to the length of vertical path plus 1 and the length of vector V equals to the length of horizontal path plus 1. Vector H is computed using R and vector V is computed using L. For example, an arbitrary node is shown for forming vectors V and H in Figure 15. Every cycle shows a rectangle made of attaching the forming subareas. The passing areas from every node are identified and their algorithm is very simple and does not need more explanation. A cycle or a path with maximum total area shows the largest inscribed rectangle (Figure 16).

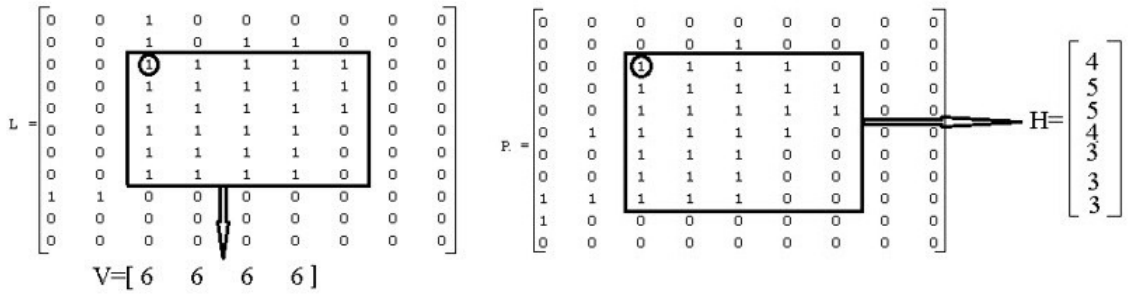

Figure 15: Forming vectors V and H.

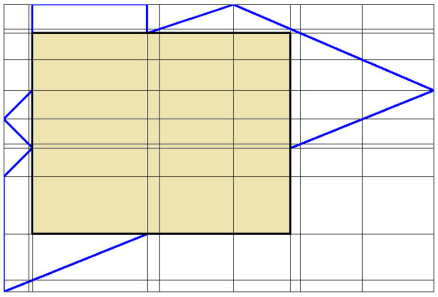

Figure 16: The largest inscribed rectangle.

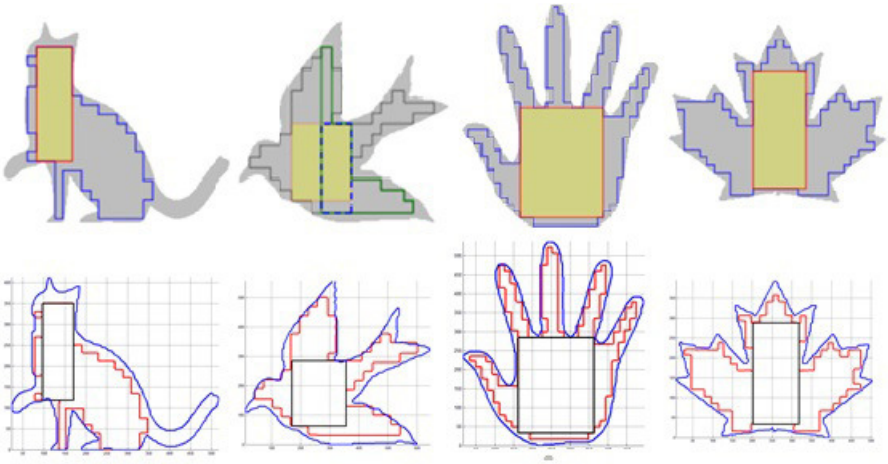

Figure 17: The largest inscribed rectangle in an IIC by: (first row) the results of Sarkar et al [4] (second row) the proposed method.

Table 1: A comparison of the largest inscribed rectangle for the original figure and IIC figure

Figure	**IIC (area)**	**Contour (area)**	**Area diff (%)**
Cat	17094	23085	26
Bird	41292	50794	19
Hand	50451	54600	8
Mapple Leaf	29580	39204	25

3 Experimental study

In this section, the results of the algorithm are examined for some different figures. The examined figures in this part of the article are extracted from Sarkar et al. [4]. They identified the largest inscribed rectangle in IIC by a specific algorithm after extracting IIC from the images. In this section, the results for the proposed algorithm are compared to other figures by Sarkar et al [4]. For this purpose, IIC and the contour of the original figure were each defined as the entering code and in both cases, the largest inscribed

rectangle was extracted. In Figure 17, the results of the IIC are compared to the results by Sarkar et al [4].

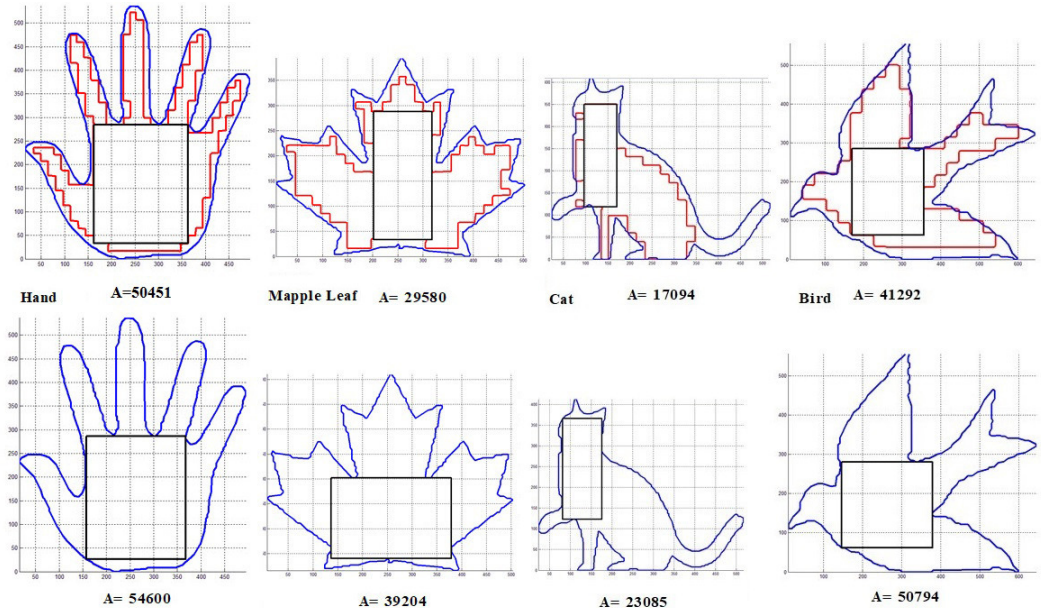

Figure 18: The largest inscribed rectangle (second row) for the original figure (first row) IIC.

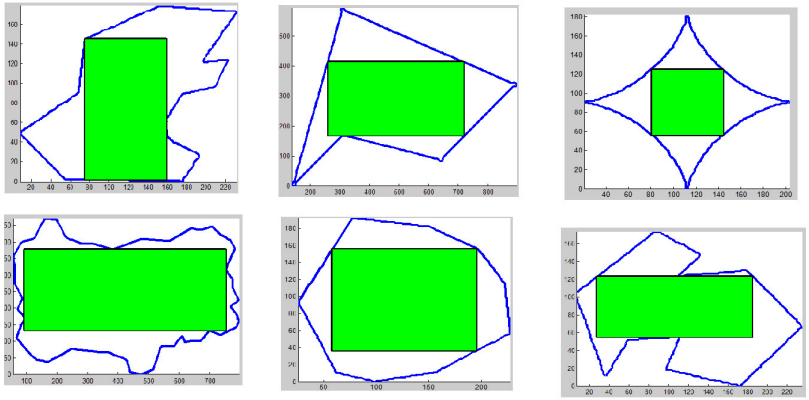

Figure 19: Some other examples.

As shown, the largest rectangle is the same in both methods and there is no difference between them. In Figure 18, the area of the largest inscribed rectangle is shown for the original figure and IIC and Table 1 compares the area for the largest inscribed rectangle with two methods. Some other examples can be seen in Figure 20. In all examples, the area of largest rectangle in original figure is greater than with IIC. So it means that,

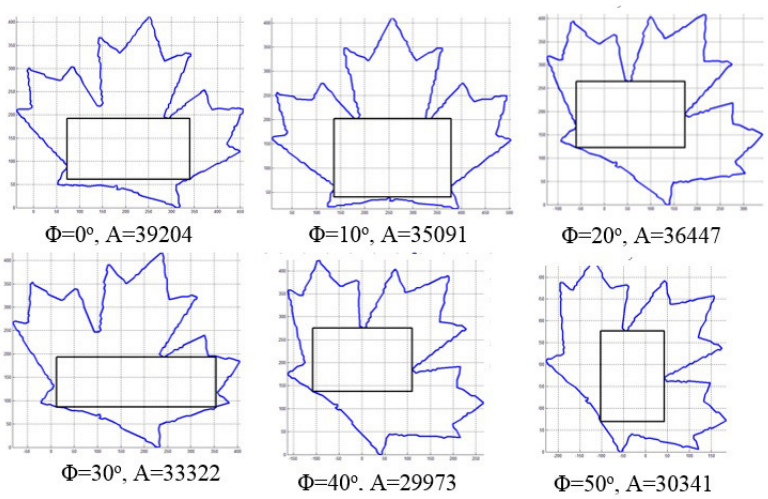

Figure 20: The largest rectangle in the desired direction.

the proposed method finds larger rectangles than Sarkar et al [4] . Their method was dependent on the size of the network for measuring IIC. The minimum percentage of area difference is 8 and the maximum is 26.

4 Conclusion

In this paper, an innovative method is presented for identifying the largest inscribed rectangle in a general polygon. The results are compared to another algorithm on some figures. For IIC, there was no difference between the results of this method and those of Sarkar et al [4] . However, comparing the areas of the largest inscribed rectangle, the areas were bigger by this algorithm because this algorithm directly calculates the polygon and does not need IIC. Therefore, the minimum difference in areas was 8 and the maximum was 26 percent.
The algorithm of Sarkar et al [4] based on IIC, the precision of the solution depends on the precision of IIC. In the proposed algorithm, there is no such dependence and the answer is better. Here's a question that might be posed: the largest rectangle found axis parallel. Therefore, the proposed algorithm is limited to calculating the largest axis parallel rectangle. To resolve this limitation, the solution is considered such that by rotating the boundary of the shape in different directions, in each case the largest rectangle is computed. Then, by comparing their area with each other, the largest rectangle is specified in the desired direction. For example, this is done for one of the above problems and the result is shown in Figure 20.

References

[1] Alt, H., Hsu, D., and Snoeyink, J., Computing the largest inscribed isothetic rectangle, In CCCG (1995), 67-72.

[2] Cabello, S., Cheong, O., Knauer, C., and Schlipf, L., Finding largest rectangles in convex polygons, Computational Geometry, 51 (2016), 67-74.

[3] Knauer, C., Schlipf, L., Schmidt, J. M., and Tiwary, H. R., Largest inscribed rectangles in convex polygons, Journal of discrete algorithms, 13 (2012), 78-85.

[4] Sarkar, A., Biswas, A., Dutt, M., Bhattacharya, A., Finding a largest rectangle inside a digital object and rectangularization, Journal of Computer and System Sciences, 95 (2018), 204217.