Largest triangulation and minimal enclosing triangle of a convex polygon

Dirksen Maxime, Dupuis Emma, Renard Simon

November 17, 2021

1 Introduction

The largest triangle in a polygon problem consists in finding the maximum-area triangle that can be contained in a given convex polygon in the plane. In order to solve this problem, we have used the article written by *Keikha, Löffer, Urhausen and van der Hoog*[3]. The minimum-area enclosing triangle problem requires finding the smallest triangle that contains a given convex polygon. *Pârvu, Ovidiu and Gilbert, David* [4] present an algorithm to compute such a triangle.

To illustrate these problems and their corresponding algorithms, we made an interactive website to create geometric artworks. The user draws some points, then the convex hull is computed, thus we obtain a convex polygon. The biggest inscribed triangle in this polygon is computed to be colored. When this triangle is colored, we can get at most 3 convex polygons on which we apply the same steps on which we apply the same steps until we only get colored triangles. In addition, the smallest triangle that contains the original polygon is plotted. An example of the result is shown in Figure 1.

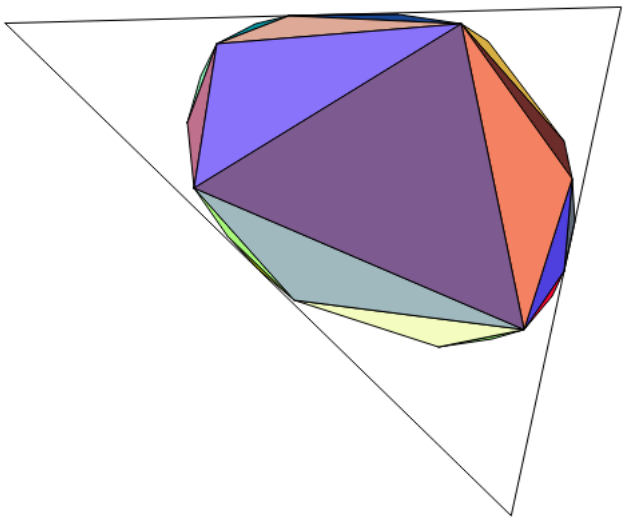


Figure 1: Artwork example

2 Maximum-area triangle in a convex polygon

2.1 Problem and main notions definitions

Let P be a convex polygon with n vertices. A triangle T is *P-aligned* if the vertices of T are a subset of P. All polygons have at least one P-aligned triangle because a polygon has at least 3 vertices. Let $a \in P$ be a root of T, every P-aligned triangle which contains this vertex a is said to be *rooted* on a. Two P-aligned triangles T1 and T2 are *interleaving* if between each vertex of T1,

we can find a vertex of T2 by x-coordinate and not radially (if they share a vertex, it does not break the interleaving) [1]. In order to find the largest triangle $T = \triangle abc$ inscribed in P, where a, b, c are three (different) points of P, we need to find a 3-stable triangle. A triangle is *3-stable*, if by moving only one vertex, we never get a bigger triangle. Thus $max\,(\triangle abc, \triangle a'bc, \triangle ab'c, \triangle abc') = \triangle abc$. Every 3-stable triangle is also *2-stable* which mean that if we fix a root a of the triangle $\triangle abc$, then there are no bigger triangle $\triangle ab'c$ or $\triangle abc'$. We can say that b and c are stable [2]. By definition, the largest triangle inscribed in a convex polygon is 3-stable, thus also 2-stable.

2.2 Dobkin and Snyder's algorithm

Dobkin and Snyder gave an algorithm to find the maximum-area triangle inscribed in a convex polygon in $O(n)$ [2]. The algorithm 1 resumed Dobkin and Snyder's solution.

Algorithm 1: $O(n)$ Dobkin and Snyder's algorithm

```
Input  : P: convex polygon,r: a vertex of P
Output: T: a triangle
Legend: Operation next means the next vertex in clockwise order of P
a ← r
b ← next(a)
c ← next(a)
m ← △abc
while True do
    while True do
        if △abnext(c) ≥ △abc then
            c ← next(c)
        end
        else if △anext(b)c ≥ △abc then
            b ← next(b)
        end
        else
            break
        end
    end
    if △abc ≥ m then
        m ← △abc
    end
    a ← next(a) if a = r then
        return m
    end
    if b = a then
        b ← next(a)
    end
    if c = b then
        c ← next(b)
    end
end
```

The algorithm correctness assumes the correctness of several statements.

Statement 1. *If $\triangle abc > \triangle ab\boldsymbol{next}(c)$ and $\triangle abc > \triangle a\boldsymbol{next}(b)c$, then $\triangle abc$ is 2-stable.*

Statement 2. *For a given root a, there exists only one 2-stable triangle $\triangle abc$.*

Proof 1 (proof of Statement 1)**.** *Initially, we have $T = \triangle abc$ with $b = \boldsymbol{next}(a)$, $c = \boldsymbol{next}(b)$.*
If Ts' area increases when we move the vertex c forward, the distance between the segment ab and the vertex c increases. After a few steps, we will get $c = c^*$ such that c^* is the vertex that maximizes the height. As we are in a convex polygon, we know that $\triangle abc^* > \triangle ab\boldsymbol{next}(c^*)$.
After finding c^*, we can apply the same reasoning to find b^*. By applying these steps several times, we get a triangle $T = \triangle abc$ such that $b = \boldsymbol{next}(a)$, $c = \boldsymbol{next}(b)$. By construction, this triangle rooted on a is 2-stable.

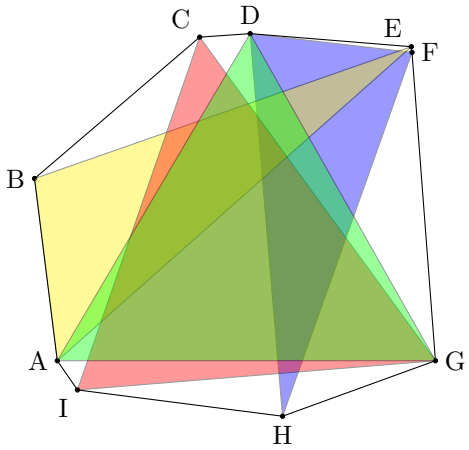


Figure 2: Counter example of Dobkin and Snyder's algorithm

We can demonstrate that the Statement 1 is true, but unfortunately Statement 2 is false. To prove this, we give in Section 2.2.1 a convex polygon such that, for a rooted vertex a, there exists two 2-stable triangles. More generally, the number of 2-stable triangles with a fixed root is bounded by $O(n)$ [3].

2.2.1 Counter example

Vahideh Keikha et al.[3] found a convex polygon such that the Algorithm 1 doesn't return the biggest inscribed triangle. The polygon is shown in Figure 2. Its vertices coordinates are: $A = (1000, 1000)$, $B = (759, 2927)$, $C = (2506, 4423)$, $D = (3040, 4460)$, $E = (4745, 4322)$, $F = (4752, 4262)$, $G = (5000, 1000)$, $H = (3383, 413)$, $I = (1213, 691)$.

With this polygon, Algorithm 1 found that the $\triangle CGI$ is the biggest. In reality, the biggest one is the triangle $\triangle ADG$. As it's the biggest, $\triangle ADG$ is 3-stable and so it's 2-stable whatever the root is.

Even if $\triangle ADG$ is 2-stable, Algorithm 1 will never find it. This mistake appears because for each vertex v of the real biggest triangle, there are two 2-stable triangles rooted on v and Algorithm 1 finds a smaller one than $\triangle ADG$. These 2-stable triangles rooted on the same vertices as $\triangle ADG$ are:

- When the root is A, $\triangle ABE$ (yellow)
- When the root is D, $\triangle DFH$ (blue)
- When the root is G, $\triangle GIC$ (red)

So, Algorithm 1 defined by *Dobkin and Snyder* is incorrect. In the next section will study another algorithm that is correct.

2.3 $O(n^2)$ algorithm

The idea is to find for each vertex $p_i \in P$ all 2-stable triangles, where p_i is the root. Moreover, we must find the largest 3-stable triangle among these 2-stable triangles. In contrast to the Dobkin and Snyder's algorithm, each time the root is moved, the two other points are reset [3].

The Algorithm 2 shows the pseudocode of the program presented in [3]. An error has been made in the second "while". Instead of having the condition **while** $c \neq a$ **do**, it should be **while next**$(c) \neq a$ **do**. Algorithm 2 contains the correction. Indeed, if the condition is $c = a$ in the most nested loop, c will never move as the area of $\triangle ab$**next**(c) will at some point take the value of $\triangle aba$ for which the area is equal to 0. Thus, the condition $\triangle aba \geq \triangle abc$ cannot be true for any value of b, thus the code will loop infinitely.

Algorithm 2: $O(n^2)$ algorithm

Input : $P = p_1, p_2, ..., p_n$: convex polygon
Output: $T = p_a, p_b, p_c$: biggest P-aligned triangle
Legend: Operation **next** means the next vertex in clockwise order of P

```
a ← p_0
b ← next(a)
c ← next(b)
m ← △abc
while True do
    while next(c) ≠ a do
        while △abnext(c) ≥ △abc do
            c ← next(c)
        end
        if △abc ≥ m then
            m ← △abc
        end
        b ← next(b)
    end
    a ← next(a)
    if a = p_0 then
        return m
    end
    b ← next(b)
    c ← next(b)
end
```

We can analyze the Algorithm 2 by breaking down the three while loops :

- **while** $\triangle ab\textbf{next}(c) \geq \triangle abc$ **do**: tries to move c forward as much as possible to increase the area.
- **while next**$(c) \neq a$ **do**: will stop when the area is going to be 0. We are sure that all 2-stable triangles have been found for the root a as we cannot move c in order to find a larger triangle. After each c augmenting loop, b moves to try to move c further successfully. — these two loops have a total complexity of $O(n)$ because we can shift c of at most $n-2$ positions and b can be moved more than c.
- **while** $True$ **do**: set iteratively each vertex as the root of a 2-stable $\triangle abc$ triangle, where a is the root. — $O(n)$, total : $O(n^2)$

In this way, all 2-stable triangles are found due to the fact that all 3-stable triangle are 2-stable. We can therefore find the largest triangle contained in a convex polygon.

2.4 Largest triangulation

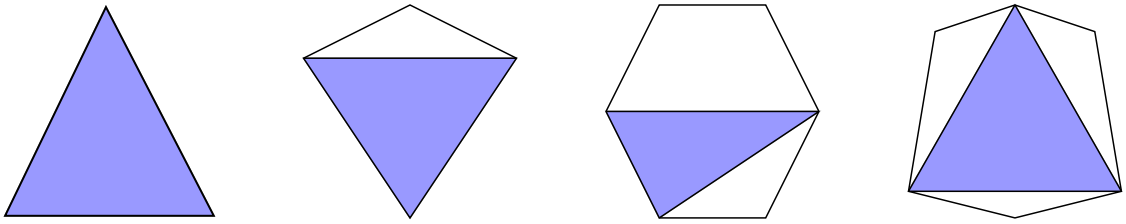
Our aim is to perform the largest triangulation, we just need to recursively run the Algorithm 2 on each new polygon created by removing the largest triangle found in the actual polygon. There are at most 3 recursions, the figure **??** takes up the 4 possible cases after the algorithm has found the largest triangle.

2.5 $O(n \log n)$ algorithm

to do

2.6 Can we do better ? $O(n)$

if it can be interesting



(a) 0 new polygon created (b) 1 new polygon created (c) 2 new polygons created (d) 3 new polygons created

Figure 3: The 4 cases of polygons created by removing the largest triangle. In blue the largest triangle to remove from the polygon, in white the polygons that will be created (not necessarily triangles).

3 Minimum-area enclosing triangle of a convex polygon

3.1 Problem and main notions and definitions

Let P be a convex polygon with n vertices. A triangle T is a minimum-area enclosing triangle if T corresponds to the smallest area triangle that can contain every vertex of P. This triangle T is represented by three vertices and three sides. vertexA, vertexB, vertexC $\in$ vertices of T and A, B, C $\in$ sides of T. A side S of T is said to be flush with an edge E of P if $S \supseteq E$. A side S of T is said to be tangent to the polygon P on vertex V if $S \supseteq V$.

3.2 Important theorems

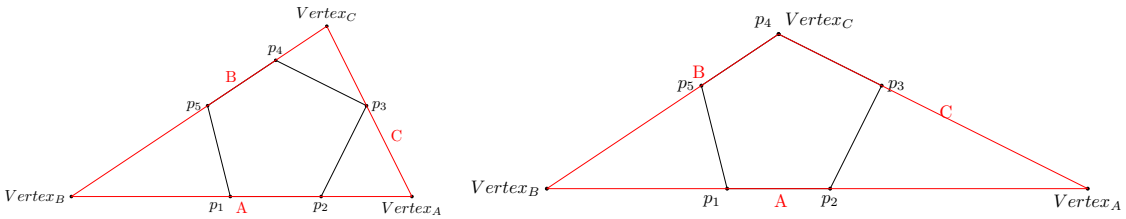
We base our algorithms on several important theorems introduced in [4].

Theorem 1. *The midpoint M of each side of a minimum-area enclosing triangle must touch the polygon. $P \supseteq M$.*

Theorem 2. *For any P, a minimum-area enclosing triangle has at least two sides are flush and a third side which can be either flush or tangent to P.*

3.3 Brute force

One way of finding this minimum-area enclosing triangle T is by brute force, seen in Algorithm 3. To do this, we analyze each triangle possible following theorems 1 and 2. Each one of these triangles are found by first choosing two vertices i and j of the polygon to form the two first flush sides $A = [i, i+1]$ and $B = [j, j+1]$ of the polygon. We then choose the last vertex k of the polygon to be either the midpoint of the tangent side or the flush side of the triangle to the polygon, depending on the area of these triangles.



(a) Example of triangle with tangent side (b) Example of triangle with 3 flush sides

Figure 4: 2 types of possible enclosing triangles

Finding the tangent side

Let A and B be two flush sides with known equations. We must find the third side C of the triangle tangent to P's vertex k as seen in Figure 4a. Theorem 1 indicates that k must be the midpoint of C, as it is the only intersection between C and P. We then know that the distance between the x coordinate of $vertex_A$ and the x coordinate of p_3 is the same as the distance between the x coordinate of $vertex_C$ and the x coordinate of p_3. This is the same for the y coordinate. We can now solve this problem with a two equations system.

Algorithm 3: Brute force algorithm

Input : $P = p_1, p_2, ..., p_n$: convex polygon
Output: T: the minimum enclosing triangle
Legend: Operation **next** means the next vertex in clockwise order of P

```
for 1 ≤ i, j, k ≤ n do
    A ← [p_i, next(p_i)]
    B ← [p_j, next(p_j)]
    if A ≠ B then
        if k ∉ A ∧ k ∉ B then
            flushTriangle ← getTriangleAllFlushSides(A, B, k)
            tangentTriangle ← getTriangleWithTangentSide(A, B, k)
            minEnclosingTriangle ←
              min(minEnclosingTriangle, flushTriangle, tangentTriangle)
        end
    end
end
return minEnclosingTriangle;
```

Algorithm 4: getTriangleAllFlushSides

Input : A, B, k: Flush sides A & B, vertex k of polygon
Output: T: the enclosing triangle
Legend: Operation **next** means the next vertex in clockwise order of P

```
C ← [k, next(k)]
vertexA ← intersection(A, B)
vertexB ← intersection(B, C)
vertexC ← intersection(C, A)
return Triangle(vertexA, vertexB, vertexC)
```

4 Implementation

An implementation of our work is available online here `https://sirenard.github.io/maximalTriangulationArt/index.html`. On the main page, you can draw your own artwork from your polygon. You also have access to another webpage, on which you can see each steps of Algorithm 1 on a custom polygon or on the polygon presented in Section 2.2.1 to convince you of the non correctness of the algorithm.

References

[1] J. E. Boyce, D. P. Dobkin, R. L. S. Drysdale, and L. J. Guibas. Finding extermal polygons. *SIAM Journal on Computing*, 14(1):134–147, 1985.

[2] David P. Dobkin and Lawrence Snyder. On a general method for maximizing and minimizing among certain geometric problems. pages 9–17, 1979.

[3] Vahideh Keikha, Maarten Löffler, Jérôme Urhausen, and Ivor van der Hoog. Maximum-area triangle in a convex polygon, revisited. *CoRR*, abs/1705.11035, 2017.

[4] Ovidiu Pârvu and David Gilbert. Implementation of linear minimum area enclosing triangle algorithm. *Computational and Applied Mathematics*, 35(2):423–438, 2016.