The Largest Inscribed Triangle and the Smallest Circumscribed Triangle of a Convex Polygon: An Overview of Linear-Time Algorithms

Günter Rote

Extended Class Notes, June 3, 2019

Contents

1 Setup for the largest inscribed triangle

We are given a convex polygon P with n vertices in counterclockwise order. We look for a triangle ABC of largest area contained in P. It is obvious that the corners A, B, C must lie on the boundary of P, and hence we speak of an *inscribed* triangle.

Our approach is to solve a constrained problem where the direction of the edge BC is specified. More precisely, for a given direction vector $\mathbf{u} = \mathbf{u}(\theta) = \binom{\cos\theta}{\sin\theta}$, we look for the largest inscribed triangle among the triangles ABC for which $\mathbf{u}$ is the outer normal of the edge BC, see Figure 2a for an illustration. We call such triangles *ABC anchored at* $\mathbf{u}$, and we denote the *largest* such triangle by $A^*B^*C^* = A^*(\theta)B^*(\theta)C^*(\theta)$. We always label ABC in counterclockwise order.

The idea is now to sweep the direction θ through the full range of possible angles and maintain the triangle $A^*(\theta)B^*(\theta)C^*(\theta)$ along the way. The largest inscribed triangle must be encountered during this *Circular Sweep*. A nice animation of this process can be seen in [Kal, Figure 1].

It is clear that the corner A^* is the extreme vertex in direction $-\mathbf{u}$.[1] As we rotate the direction θ counterclockwise, the point $A^*(\theta)$ will jump from one vertex to the next in counterclockwise direction whenever $-\mathbf{u}(\theta)$ is the outer normal of a polygon edge. For the other two points, we have the following crucial properties.

1. The points $B^*(\theta)$ and $C^*(\theta)$ are unique (Lemma 1).
2. The points $B^*(\theta)$ and $C^*(\theta)$ move monotonically in counterclockwise direction on the boundary of the polygon as θ is increased. (Theorem 5.ii.)

We will see how to maintain $B^*(\theta)$ and $C^*(\theta)$ as θ ranges over the interval $[0° \mathrel{..} 360°]$. We have to process a linear number of events, and for each event, we can carry out the elementary steps and decisions of the process in constant time. Figure 1 shows an example how the area of $A^*(\theta)B^*(\theta)C^*(\theta)$ varies depending on θ.[2] By picking the maximum of this function, we find the largest inscribed triangle in linear time.

2 Finding the largest anchored triangle

We consider a fixed direction $\mathbf{u}$. We parameterize the triangle $A^*BC = A^*B(h)C(h)$ by the height h over the side BC, see Figure 2a. For a given height h, the segment $B(h)C(h)$ is determined as the intersection of the area of P with the line perpendicular to $\mathbf{u}$ at distance h from A^*. The variable h ranges between 0 and the width $w(\mathbf{u})$ of the polygon in direction $\mathbf{u}$. In particular, if P has an edge with outer normal $\mathbf{u}$, then $B(h)C(h)$ for $h = w(\mathbf{u})$ is equal to that edge, see Figure 5b. Since this case sometimes requires special arguments, we give it a name: We call an edge of P the **u**-*extreme edge* if its outer normal is $\mathbf{u}$. (For most directions $\mathbf{u}$, there is no **u**-extreme edge.)

It may happen that A^* is not unique, namely when the polygon has an edge with outer normal $-\mathbf{u}$, see Figure 2b. In this case, it does not matter which point A^* we pick from that edge: This choice affects neither the definition of $B(h)$ and $C(h)$ nor the area of the triangle $A^*B(h)C(h)$.

[1] In Kallus [Kal], anchored triangles with this corner A^* are called "candidate-anchored triangles". His "anchored triangles" are what we call *largest anchored triangles*.

[2] This polygon is instance number 18 in the test suite that Kallus [Kal] provided with the source files of his arXiv preprint and at `https://github.com/ykallus/max-triangle/releases/tag/v1.0`

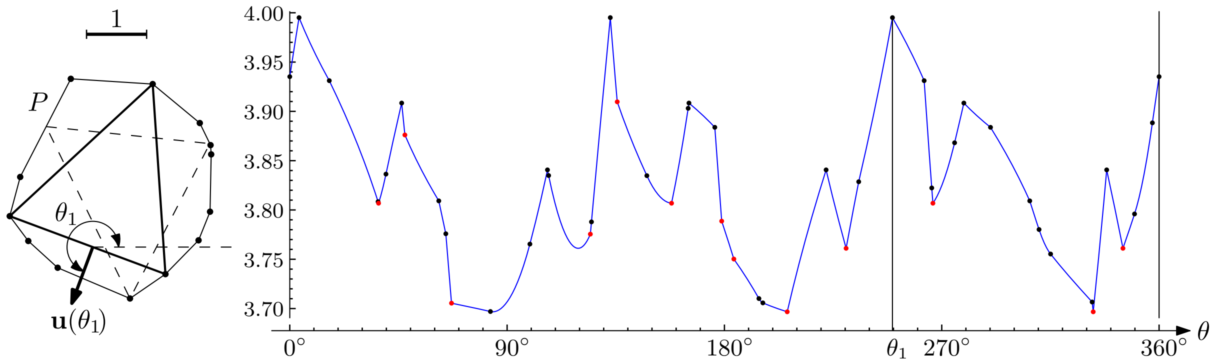


Figure 1: The area $F(\theta)$ of the largest θ-anchored triangle $A^*(\theta)B^*(\theta)C^*(\theta)$ as a function of the direction $\theta \in [0° \,..\, 360°]$, for the 13-gon P shown on the left. This function is piecewise smooth and continuous. The 47 dots on the graph, excluding the boundaries at 0° and 360°, are the breakpoints where the combinatorial type changes in the sense that a triangle corner moves to a different polygon edge or rests at a polygon vertex. (One black dot is almost hidden behind the first red dot.) The 13 red breakpoints correspond to the inner normals of the edges, where A^* jumps from one vertex to the next. The largest inscribed triangle in P is shown. It is encountered three times as a maximum of $F(\theta)$, namely whenever $\mathbf{u}(\theta)$ is one of the outer normals of this triangle. The direction θ_1 where this happens for the third time is indicated. The dashed triangle in P corresponds to the three minima of $F(\theta)$. We will see in Section 3 that it determines the smallest circumscribed triangle of P.

2.1 The largest anchored triangle is unique

Lemma 1. *The function* $f\colon [0 \,..\, w(\mathbf{u})] \to \mathbb{R}_{\ge 0}$ *defined by* $f(h) = \operatorname{area} A^*B(h)C(h)$ *is continuous and unimodal: It starts from* $f(0) = 0$ *with a strictly increasing part; it has a unique maximum; and this is followed by a strictly decreasing part. The decreasing part may be missing.*

Proof. [3] The area $f(h) = \frac{1}{2}h|B(h)C(h)|$ is $\frac{1}{2}$ times the product of the height h and the baseline $|B(h)C(h)|$ of the triangle. Since both factors are continuous between 0 and $w(\mathbf{u})$, f is continuous as well. Due to the convexity of P, the length $g(h) := |B(h)C(h)|$ is a concave function, and it consists of a weakly increasing part between $h = 0$ and some $h_{\max}$ where it achieves the maximum, and a decreasing part between $h_{\max}$ and $w(\mathbf{u})$. In the first part, $f(h) = \frac{1}{2}h \cdot g(h)$ is the product of h with a weakly increasing positive function, and is therefore strictly increasing. In the second part, we look at the derivative $f'(h) = \frac{1}{2}(g(h) + h \cdot g'(h))$. The function g is not differentiable everywhere, but we can take the right derivative in this equation. The function g is strictly decreasing, and the second term is the product of h with a negative piecewise constant decreasing function. Both terms are strictly decreasing. So the function f' is strictly decreasing, and the function f is strictly concave and therefore unimodal in the second part.

Since $f(0) = 0$, the increasing part is always present. The decreasing part may be missing when the polygon P has an edge with outer normal $\mathbf{u}$. □

2.2 The local problem with a triangular outer polygon

The range of the function f is decomposed into pieces. On each piece, $B(h)$ and $C(h)$ slide along two fixed edges b and c of P. In order to analyze the behavior of f on one of these pieces, we first consider the case that $B(h)$ and $C(h)$ range over two *lines* b and c.

[3] See [Kal, Lemma 2.2–3] for a different, less elementary proof of the unique maximum property. (The term "convex" should be read as "concave" or "downward convex".)

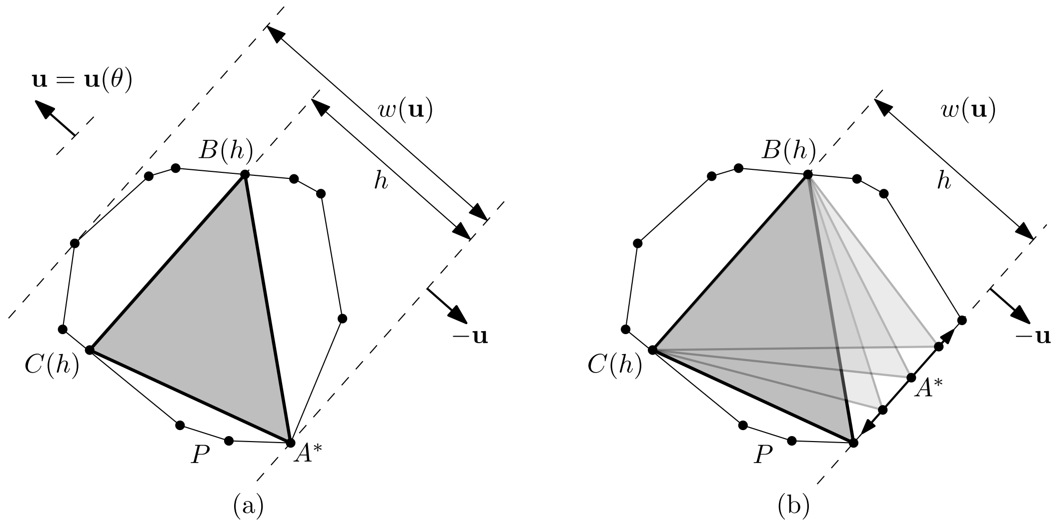


Figure 2: (a) Notations for anchored triangles $A^*B(h)C(h)$. (b) Moving A^* parallel to BC does not affect the area of A^*BC.

To facilitate the discussion, we assume in this section and whenever it is convenient that $\theta = 90°$ and $\mathbf{u}$ points in the upward direction. This allows us to use the words "above" and "below", "up" and "down" with reference to this situation. They have to interpreted appropriately when $\mathbf{u}$ is rotated.

Thus, we are looking for a triangle A^*BC with a horizontal edge BC that lies above A^*, where B and C are constrained to lie on two upward rays $\vec{b}$ and $\vec{c}$ and C should be to the left of B, see Figure 3.

Lemma 2. *The area of A^*BC is a quadratic function of h. If the rays $\vec{b}$ and $\vec{c}$ don't meet, then the area increases indefinitely with h, and there is no largest triangle. Otherwise, the area of A^*BC has a unique maximum, which is found as follows: let T be the intersection of $\vec{b}$ and $\vec{c}$. Then the edge B^*C^* of the largest triangle goes through the midpoint M of T and A^*.*

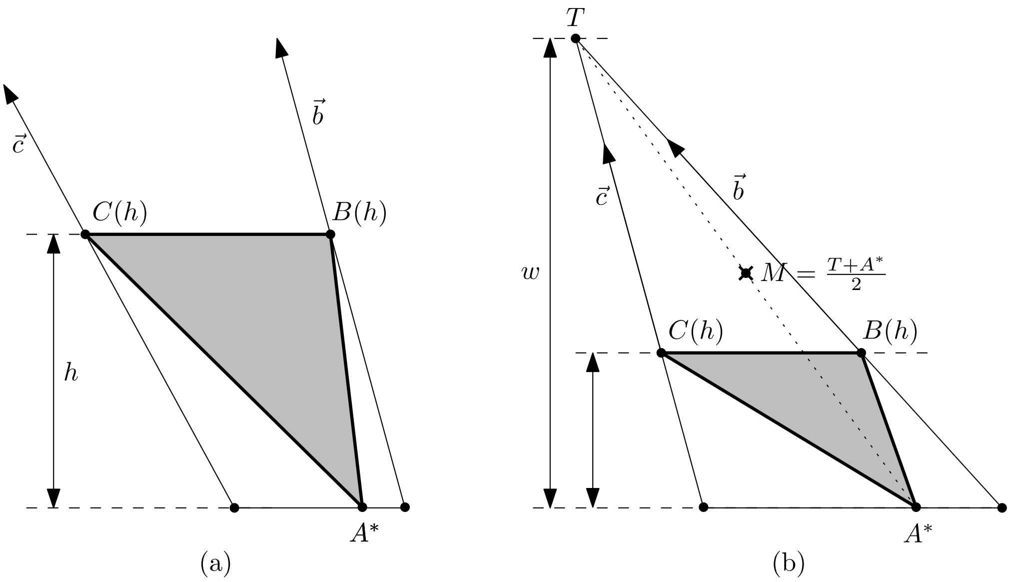


Figure 3: The largest anchored triangle restricted by two rays $\vec{b}$ and $\vec{c}$

Proof. The area $f(h) = \frac{1}{2}h|B(h)C(h)|$ is $\frac{1}{2}$ times the product of the height h and the baseline $|B(h)C(h)|$ of the triangle. If the rays $\vec{b}$ and $\vec{c}$ are parallel or diverge, then it is clear that the area increases without bounds, since h increases and the baseline $|B(h)C(h)|$ increases or remains constant, see Figure 3a.

Otherwise, the length of the baseline $B(h)C(h)$ is proportional to $w - h$, where w is the vertical distance between T and A^*, see Figure 3b. It follows that $f(h) = \frac{1}{2}h|B(h)C(h)|$ has the form $f(h) = \alpha h(w - h)$ for some constant α, and this is maximized for $h = w/2$. This is precisely the value h where the segment $B(h)C(h)$ goes through the midpoint $(T + A^*)/2$. □

Definition. We call $M = (T + A^*)/2$ the *critical pivot point* or simply the *critical point.*

The usefulness of the above lemma results from the way in which the optimality criterion is phrased: When $\mathbf{u}$ is rotated, the critical point remains fixed as long as A^* remains fixed, whereas w and h change.

2.3 The direction of improvement for the largest anchored triangle

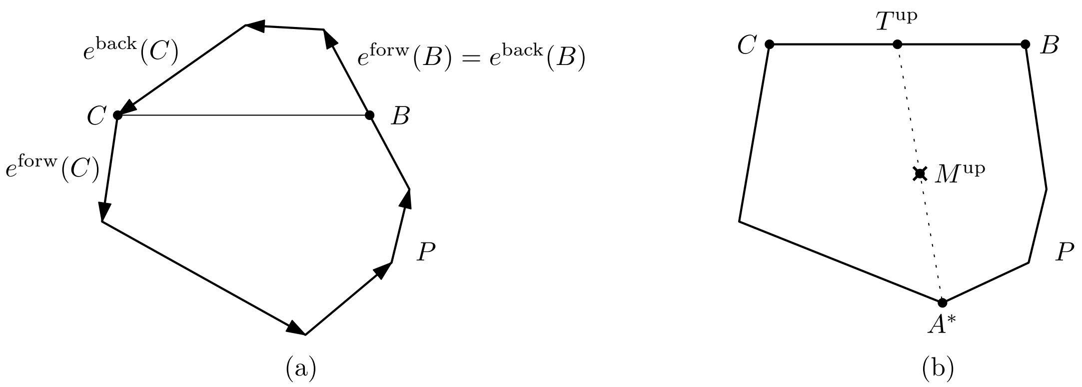


Figure 4: (a) The forward and backward incident edges of a point on the boundary of P. (b) A possible definition of M^{up} when BC is an edge of P

We now return to the situation when B and C are restricted to the original polygon P. To check whether the triangle ABC is largest, we use Lemma 2. If B or C is at a vertex of P, the function $f(h)$ is not differentiable at this point, and we have to look at its one-sided derivatives. For a point B (or C) that is a vertex of P, we call its two incident edges the *forward edge* $e^{\text{forw}}(B)$ and the *backward edge* $e^{\text{back}}(B)$, according to the counterclockwise orientation of P, see Figure 4a. If B lies in the interior of an edge e of P, we define $e^{\text{forw}}(B)$ and $e^{\text{back}}(B)$ to be that same edge e.

If we consider the behavior of $f(h)$ when h is increased, we have to look at the upward rays through the two *upper* incident edges $e^{\text{forw}}(B)$ and $e^{\text{back}}(C)$. We denote their intersection by T^{up}, if it exists, and the midpoint between this point and A^* is the *upward critical pivot point* M^{up}, see Figure 5a. Accordingly we define the *downward critical pivot point* M^{down} by the upward rays through the two *lower* incident edges $e^{\text{back}}(B)$ and $e^{\text{forw}}(C)$. If neither B nor C is a vertex of P, then M^{up} and M^{down} coincide. Otherwise, M^{up} lies *below* M^{down}, despite what the name suggests!

More generally, we will repeatedly compare critical points that are defined by pairs of edges. It will be good to remember that exchanging one of the defining edges by another edge further *down* causes the defining ray to bend *outward.* Hence the critical point will move *upward,* or cease to exist.

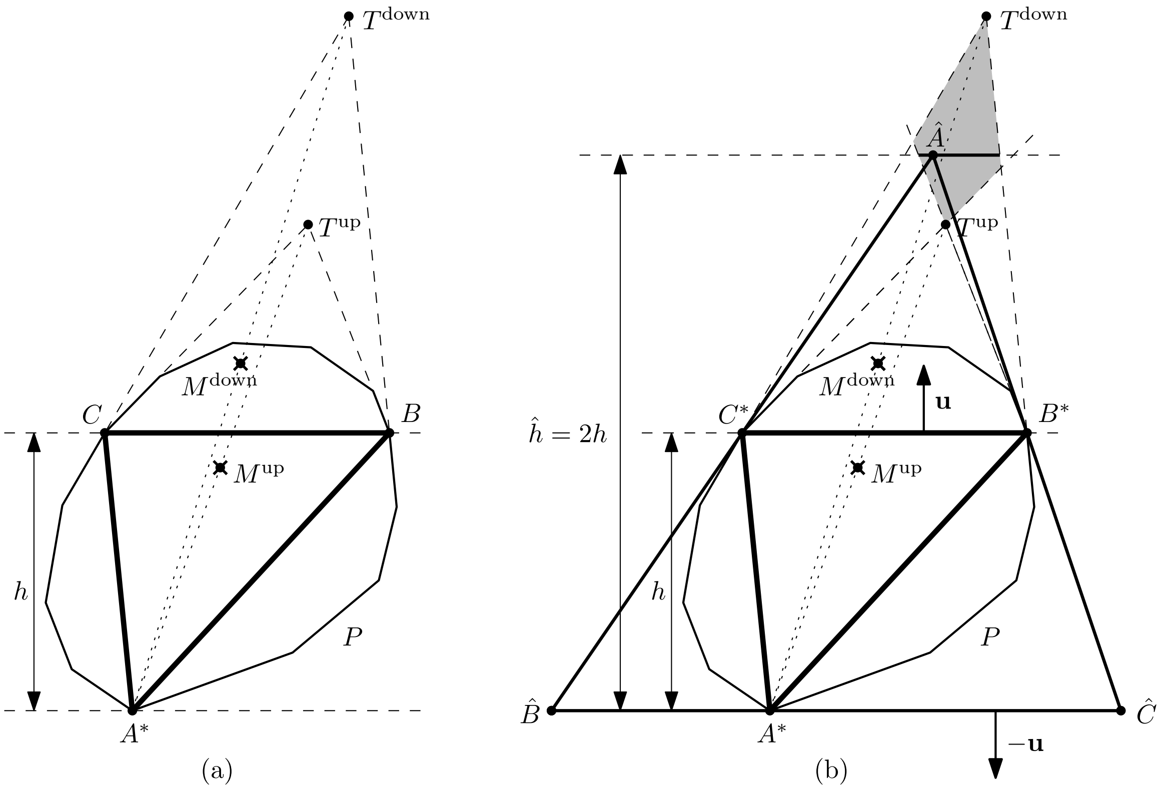


Figure 5: (a) The optimality criterion for a largest anchored triangle A^*BC. (b) An anchored circumscribed triangle $\hat{A}\hat{B}\hat{C}$ corresponding to a largest anchored inscribed triangle A^*BC. More typically, the gray feasible ares for $\hat{A}$ degenerates to a line segment or ray or to a single point.

By Lemma 2, the critical point M, if it exists, gives the direction in which BC has to move in order to increase the area, according to the following *Improvement Test*:

$$\text{We should increase } h \text{ if and only if } M^{\text{up}} \text{ lies above } BC. \tag{1}$$

$$\text{We should decrease } h \text{ if and only if } M^{\text{down}} \text{ lies below } BC. \tag{2}$$

As a memory aid, one can remember that BC wants to move *close* to the critical point.

The intersection T, and hence the critical point M, may not exist, and in that case, h should be increased if we want to increase the area. We can remember that, if the rays don't intersect, and hence the critical point does not exist, it is always consistent with (1) to treat this case *as if the critical point would lie above BC*. Nevertheless, we will always explicitly mention the case of nonexistence in the statements of the lemmas, at least parenthetically.

Lemma 3 (Optimality criterion for an anchored triangle)**.** *The inscribed triangle A^*BC with height h and side BC perpendicular to* $\mathbf{u}$ *is the optimum anchored triangle if and only if $h > 0$ and the following two conditions are satisfied:*

a) The downward critical point M^{down} lies on or above BC (or does not exist).

b) If h does not lie at the maximum of its range,
the upward critical point M^{up} lies on or below BC.

Proof. The conditions are the necessary conditions for a local maximum of $f(h)$: Condition (a) looks at the left derivative, and Condition (b) looks at the right derivative. The case when h is at the maximum of its range is treated specially in Condition (b) because there is no right derivative. When the critical point lies *on* the segment BC, the derivative is 0. Nevertheless, this is sufficient to conclude that the area cannot be increased by moving h in that direction, since the quadratic function $f(h)$ has then a critical point at h, and this critical point is a maximum.

For $h \to 0$, the area decreases to 0, and hence the optimum must occur at a positive height. By Lemma 1, the maximum is unique, and therefore the conditions are also sufficient. □

We mention that an alternative proof of Lemma 1 (uniqueness of B^* and C^*) can be obtained by arguing directly that the necessary conditions (a) and (b) can have at most one solution.[4]

3 The smallest anchored circumscribed triangle

We relate the largest inscribed triangle anchored at $\mathbf{u}$ to the *smallest-area circumscribed triangle* $\hat{A}\hat{B}\hat{C}$ among the triangles *anchored at* $-\mathbf{u}$, in the sense that $-\mathbf{u}$ is the outer normal of the edge $\hat{B}\hat{C}$.[5]

Lemma 4. *i) Let $A^*B^*C^*$ be a largest inscribed triangle anchored at $\mathbf{u}$, of height h. Then the smallest circumscribed triangle $\hat{A}\hat{B}\hat{C}$* anchored at $-\mathbf{u}$ *has height $\hat{h} = 2h$, and the length of its baseline is $\hat{B}\hat{C} = 2 \cdot B^*C^*$, and hence its area is 4 times the area of $A^*B^*C^*$.*

ii) There is always a smallest anchored circumscribed triangle $\hat{A}\hat{B}\hat{C}$ such that the side $\hat{A}\hat{B}$ or the side $\hat{A}\hat{C}$ touches a whole edge of P.[6]

Proof. Figure 5b shows how $\hat{A}\hat{B}\hat{C}$ is constructed. Again we assume without loss of generality that $\mathbf{u}$ points vertically upward. From an appropriate point $\hat{A}$ at height $2h$ above A^*, we put tangents to P through the points B^* and C^* and we extend these tangents until they meet the horizontal line through A^* in the points $\hat{C}$ and $\hat{B}$, respectively. Then $\hat{B}\hat{C} = 2 \cdot B^*C^*$, because the triangles $\hat{A}\hat{B}\hat{C}$ and $\hat{A}C^*B^*$ are similar and the ratio of their heights is 2.

We must show that a point $\hat{A}$ with the desired properties exists. The requirement that the tangents from $\hat{A}$ should touch P in the points B^* and C^* restricts $\hat{A}$ to the intersection of two wedges (the shaded area in Figure 5b). Its boundary is formed by at most four edges. By definition, the lowest point of the region is T^{up}, and from Condition (b) of Lemma 3, this point exists and lies below the line at height $2h$. The highest point is T^{down}, if that point exists, or otherwise the region is unbounded. Thus, by Condition (a) of Lemma 3, the region extends above the line at height $2h$. Thus, a point $\hat{A}$ at height $2h$ in this region can be found. In Figure 5b, The possible choices for $\hat{A}$ are highlighted.

Choosing $\hat{A}$ at the boundary of the allowed region ensures that one side of the triangle touches a whole edge of P, thus proving the second statement of the lemma.

We still need to show that there is no smaller anchored triangle containing P. In fact, there is not even a smaller anchored triangle that contains just the triangle $A^*B^*C^*$: This statement is dual to Lemma 2, and its proof is just as easy, see Figure 6. If we choose the point $\hat{A}$ at some height $\hat{h}$, the smallest anchored circumscribed triangle must contain the projection of $\hat{B}\hat{C}$ of the segment C^*B^* from $\hat{A}$ to the horizontal line through A^*, and by similar triangles, the base $\hat{B}\hat{C}$

[4]Consider the points M^{up} and M^{down} as h increases from 0 to the maximum value. After an initial period where the points don't exist and therefore M^{up} and M^{down} "lie above" BC, the critical points move downwards because the edges incident to B and C turn more and more inwards. At the same time the edge BC moves upwards. Thus, there can be only one point where (a) and (b) are fulfilled and the interval between M^{up} and M^{down} straddles the segment BC. The precise argument is a bit delicate because of the jumps of M^{up} and M^{down}.

[5]The strong connection between the two problems was first explicitly noted and exploited by Chandran and Mount, see in particular [ChMo, Lemma 2.4 in connection with Lemma 2.5]. The statement of our Lemma 4.i is discussed after the proof of Lemma 2.4.

[6][KlLa, Theorem 2.1.iv].

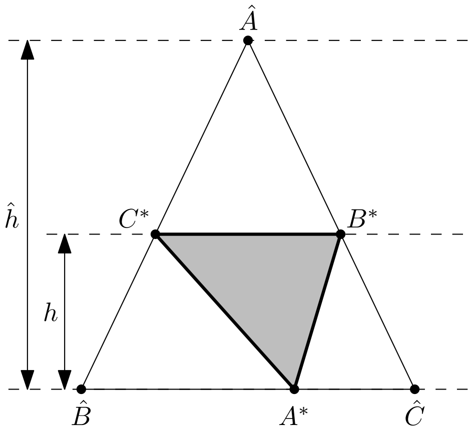


Figure 6: An anchored triangle containing $A^*B^*C^*$

is $\hat{h}/(\hat{h}-h)$ times as long as the segment C^*B^*, and hence the area of $\hat{A}\hat{B}\hat{C}$ is $\frac{1}{2}\cdot\hat{h}\cdot\frac{\hat{h}}{\hat{h}-h}B^*C^*$. The minimum of this expression is achieved for $\hat{h}=2h$. □

This lemma has a converse[7]: From a smallest circumscribed triangle $\hat{A}\hat{B}\hat{C}$ anchored at $-\mathbf{u}$, one can recover a largest inscribed triangle $A^*B^*C^*$ anchored at $\mathbf{u}$. We don't need this direction, but for completeness, it is proved in Appendix C (Lemma 14).

The lemma shows that, by computing the area $A^*(\theta)B^*(\theta)C^*(\theta)$ for all directions θ, we can simultaneously find the smallest circumscribed triangle: Instead of looking for the largest area among these triangles, we just look for the smallest area, and we multiply the result by 4. (It is a bit paradoxical that we should look for *largest* inscribed anchored triangles in order to find the circumscribed triangle with *smallest* area.)

4 How B^* and C^* move when the direction is rotated

We define the *combinatorial type* of an inscribed triangle ABC as the specification that tells for each of the three corners A, B, C on which vertex of P or in the interior of which edge of P it lies.

Theorem 5. *i) The domain of angles θ is partitioned into intervals at breakpoints $0^\circ=\theta_0<\theta_1<\cdots<\theta_i<\theta_{i+1}<\cdots<\theta_k=360^\circ$, such that in each open interval $(\theta_i \,..\, \theta_{i+1})$, all triangles $A^*(\theta)B^*(\theta)C^*(\theta)$ have the same combinatorial type. Moreover, in each closed interval $[\theta_i \,..\, \theta_{i+1}]$, the edge $B^*(\theta)C^*(\theta)$ pivots around a point M on this edge.[8] There are three mutually exclusive possibilities, which are illustrated in Figure 7.*

I. $M=B^*(\theta)$ is stationary at a vertex of P and $C^*(\theta)$ slides on a fixed edge of P.

II. $M=C^*(\theta)$ is stationary at a vertex of P and $B^*(\theta)$ slides on a fixed edge of P.

III. The critical pivot points M^{up} and M^{down} coincide, and $M^{\mathrm{up}}=M^{\mathrm{down}}=: M$ lies on the segment B^*C^*; the segment $B^*(\theta)C^*(\theta)$ rotates around M, and $B^*(\theta)$ and $C^*(\theta)$ slide on two fixed edges of P.[9]

[7] cf. [ChMo, Lemma 2.4]

[8] In the animation shown in [Kal, Figure 1], it is apparent that the optimal edges B^*C^* go through a common point when $\mathbf{u}$ is rotated in some range.

[9] See [ChMo, Figure 5], covering the case where the smallest anchored circumscribed triangle has "two flush legs". The pivot is the point x in that figure, and it is constructed by considering the local optimality condition of the circumscribed triangle.

ii) Moreover, $B^*(\theta)$ *and* $C^*(\theta)$ *move continuously and monotonically*[10] *in counterclockwise direction on the boundary of the polygon* P *as* θ *is increased. They make a full turn around* P *as* θ *ranges over the interval* $[0° \,..\, 360°]$.

iii) The number k *of intervals is at most* $5n+1$.[11]

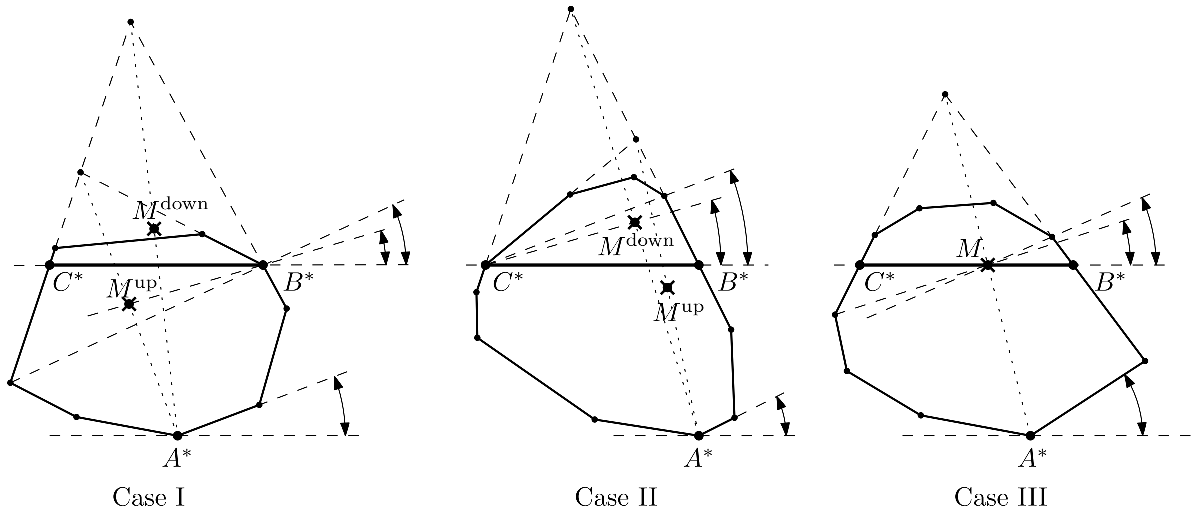


Figure 7: How the segment B^*C^* can rotate

In Case III, it may happen that the rotation center lies on an edge of P and hence coincides with B^* or C^*, see Figure 8. Then this corner of the triangle remains stationary.

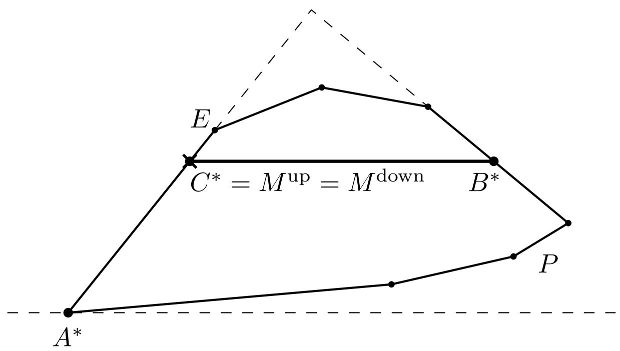


Figure 8: The pivot point M can lie on the boundary. (If this example is modified by shortening the edge A^*E so that E coincides with C^*, then the pivot around which the rotation occurs is still the point C^*, but the characteristic property $M^{\text{up}} = M^{\text{down}}$ of case III is lost, and we are in Case II.)

Proof. Consider a generic direction θ and the largest triangle according to Lemma 3. Two cases can arise:

- If the two points B^* and C^* lie in the interior of two edges of P, then $M^{\text{up}} = M^{\text{down}}$, and B^*C^* must go through this point; this condition does not change as long as B^* and C^* remain in the interior of the edges on which they move.

[10] cf. [OAMB, Lemma 2]. See Appendix D for another proof.

[11] cf. [ChMo, Lemma 3.1].

- If one point, B^* or C^*, lies on a vertex of P and the other one lies in the interior of an edge, then M^{up} and M^{down} are different, and they remain different provided that the point B^* or C^* which lies at a vertex stays there. The optimality condition remains satisfied as long as the segment B^*C^* does not cross M^{up} or M^{down} and as long as the moving point stays on the same edge.

There are degenerate situations which are not covered by these two cases: Both B^* and C^* can lie on vertices of P; or B^*C^* goes through a critical point M *and* a vertex of P that is different from M. (Or both of these situations happen simultaneously.) However, there are only finitely many potential pivot points and finitely many vertices. Thus, there are only finitely many directions θ which are not covered by the two cases.

We have therefore proved the first claim of the theorem: The open intervals with the same combinatorial type cover all angles except for a finite set of breakpoints.

Let us now look at these breakpoints. Figure 7 shows, for each case, the (at most) three events that compete for terminating the motion or validity of the optimality conditions when θ increases. One of the moving endpoints B^* or C^* might hit the endpoint of its edge, or the rotating segment might hit one of the pivot points M^{up} or M^{down}. In addition, the point A^* might jump to the next vertex. Of course, analogous events happen when θ is *decreased.*

When θ reaches such a breakpoint, the optimality conditions continue to hold. This is obvious if the rotating segment hits M^{up} or M^{down}. If one of the moving endpoints arrives at a vertex, then M^{up} or M^{down} may jump. However, such a jump is always in the good direction which makes the optimality conditions more liberal: M^{up} will jump to a lower position, and M^{down} will jump higher. Thus, the rotating segment will remain optimal at the boundaries of the intervals.

The rotation induces a continuous counterclockwise motion of $B^*(\theta)$ and $C^*(\theta)$ inside each interval. The only conceivably discontinuity is when B^*C^* coincides with the **u**-extreme edge of P, as in Figure 4b. However, in this case, it is easy to see that the segment will pivot around B^* when θ is increased (see Lemma 7 below), and hence the motion of $B^*(\theta)$ is continuous also here.

Since the closed intervals $[\theta_i \mathinner{..} \theta_{i+1}]$ overlap, the motion is continuous and monotone throughout. Since the points $B^*(\theta)$ and $C^*(\theta)$ cannot overtake $A^*(\theta)$ or be overtaken by $A^*(\theta)$, they have to make one complete turn.

Finally, we bound number of breakpoints. We will justify below that at each breakpoint θ_i, one or more of the following happen:

a) A^* jumps.
b) B^* or C^* arrives at a vertex p_j as θ approaches θ_i from the left.
c) B^* or C^* moves away from a vertex p_j as θ increases from θ_i to the right.

The breakpoints where A^* jumps are easy to count: There are exactly n of them. Each of the four types of events where B^* or C^* arrives or moves away from a vertex p_j can happen at most once per vertex, for a total of $4n$ events of these types. The extra $+1$ in the overall bound $5n+1$ on the number of intervals is for the artificial cut at $0°/360°$.

To justify the claim, consider an endpoint θ_i of an interval in the circular sweep. If A^* jumps, or if B^* or C^* was moving and arrives at a vertex, the claim is fulfilled. The only remaining case is when B^*C^* rotates around B^* at a polygon vertex (Case I) and hits the critical point M^{up}, or symmetrically, when it rotates around C^* at a polygon vertex (Case II) and hits M^{down}. Consider without loss of generality the latter case, see the middle picture of Figure 7. Then, if θ is further increased, the segment B^*C^* will start to pivot around M^{down} and C^* will move away from the vertex while B^* continues to move on its edge. This situation is optimal because M^{down} does not change, and M^{up} to jumps to M^{down}. (We are thus now in Case III. This analysis is a special case of the Movement Rule that will be stated later in Lemma 7.) □

The bound $5n + 1$ is usually an overestimate. Even in a generic situation, an event of type (b) and an event of type (c) can occur at the same breakpoint. Moreover, a breakpoint need not manifest itself in the shape of the function $F(\theta)$. There are even polygons where $F(\theta)$ is the constant function. One such example, from [BRS, Fig. 3], is the hexagon P with vertices $(3, 0), (3, 3), (0, 3), (-1, \frac{5}{3}), (-1, 0), (0, -1)$.

5 How the area changes when the direction is rotated

Lemma 6. *In each closed interval $[\theta_i \mathinner{..} \theta_{i+1}]$ where $B^*(\theta)$ and $C^*(\theta)$ lie on fixed edges, the area function $F(\theta)$ has at most one local minimum.*

It has no local maximum in the interior of the interval, unless $F(\theta)$ is constant in that interval.

Proof. The statement is clear if one endpoint is stationary (Cases I and II of Theorem 5): The point A^* is also stationary, and the third point moves monotonically on an edge. Hence $F(\theta)$ is either constant, or strictly increasing, or strictly decreasing.

The more interesting case is Case III, when the segment rotates around M. First of all, we note that area $A^*B^*C^* =$ area TC^*B^*, see Figure 9a: Indeed, the segment B^*C^* bisects both the triangle A^*TB^* and the triangle A^*TC^*, as is easily seen.

We can thus look at the area of TC^*B^*. If we rotate the segment by a small amount $\Delta\theta$, Figure 9b shows how the triangle area changes: It grows on the left side and shrinks on the right side, by a triangular region in each case. We approximate these regions by circular sectors, leaving an error of small order (the blue regions in the figure):

$$F(\theta + \Delta\theta) - F(\theta) = \Delta(\text{area}\, TBC) = \tfrac{1}{2} \cdot \Delta\theta \cdot \big(|C^*M|^2 - |B^*M|^2\big) + O(\Delta\theta^2)$$

Letting $\Delta\theta \to 0$, one sees that the comparison between $|C^*M|$ and $|B^*M|$ decides about the sign of the derivative of F. The stationary situation is attained when $|C^*M| = |B^*M|$. Figure 9c shows that the unique segment B_0C_0 through M with this property can be obtained through symmetry, by reflecting the rays TB^* and TC^* at M and intersecting them with the original rays.

As the segment B^*C^* rotates counterclockwise around M and the points B^*, C^* move on the rays TB^* and TC^*, respectively, we initially have $|C^*M| < |B^*M|$, and $F(\theta)$ is strictly decreasing, until we reach B_0C_0. After this point, $|C^*M| > |B^*M|$ and $F(\theta)$ is strictly increasing. □

One might be tempted to prove Lemma 6 by showing that the pieces of $F(\theta)$ are convex functions. However, this is not the case, at least in terms of the parameterization by the angle θ. This can for example be observed (not very conspicuously) at the third piece from the left in Figure 1.

6 How the motion continues after a breakpoint

There is an easy rule that tells how the motion continues when θ is increased. This rule works irrespective of whether θ is at a breakpoint or not. Suppose we have determined the largest anchored triangle $A^*(\theta)B^*(\theta)C^*(\theta)$, and we want to increase θ. Assume again for simplicity that $\mathbf{u}(\theta)$ points vertically upwards. If A^* is not unique, we select the rightmost possibility, in preparation for the increase of θ. Now we construct the intersection T^{forw} of the upward rays through $e^{\text{forw}}(B^*)$ and $e^{\text{forw}}(C^*)$, and the *forward critical pivot point* $M^{\text{forw}} = (T^{\text{forw}} + A^*)/2$.

Lemma 7 (The Movement Rule)**.** *If θ is increased, the segment B^*C^* moves as follows, see Figure 10:*

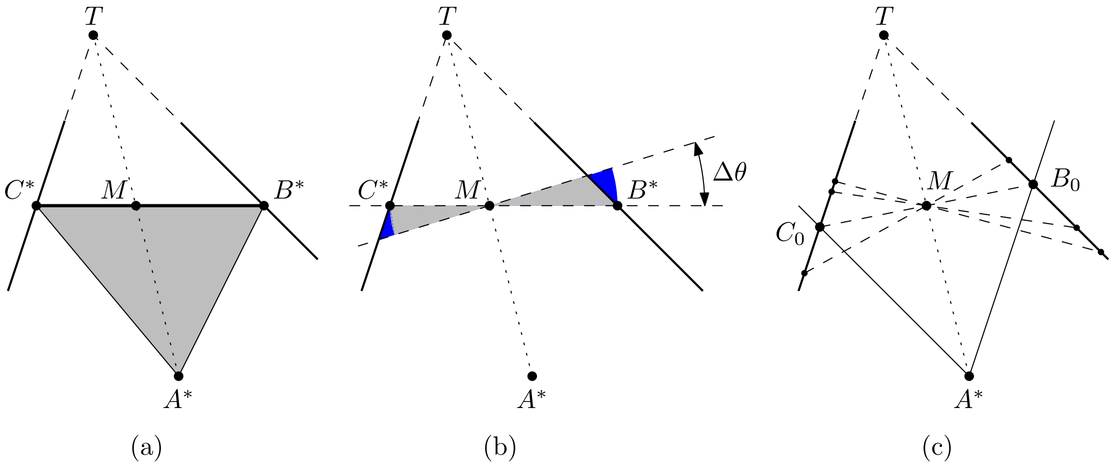


Figure 9: (a) area $A^*B^*C^* = \text{area}\, TC^*B^*$. (b) The area change under rotation of the segment B^*C^*. (c) The balanced segment B_0C_0 with $|B_0M| = |C_0M|$.

a) If M^{forw} lies on B^*C^*, then B^*C^* will rotate around this point.

b) If M^{forw} lies below B^*C^*, then B^*C^* will rotate around B^*.

c) If M^{forw} lies above B^*C^*, then B^*C^* will rotate around C^*. This includes the case that M^{forw} does not exist because the upward rays through $e^{\text{forw}}(B^*)$ and $e^{\text{forw}}(C^*)$ don't meet.

This rule is consistent with the tendency that B^*C^* wants to get (or stay) *close* to M^{forw}.

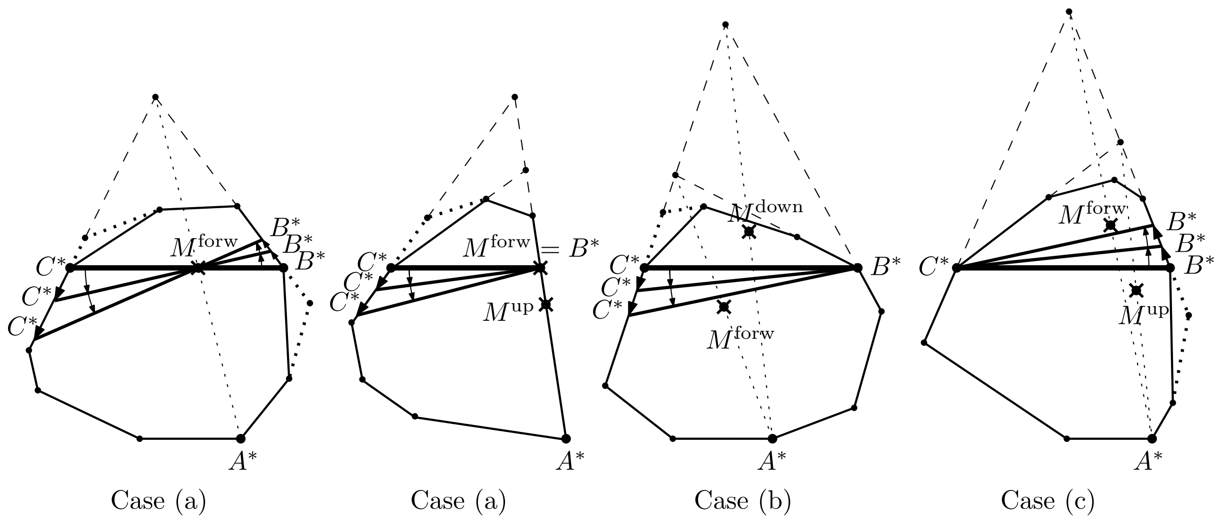


Figure 10: The pivot point around which the segment B^*C^* rotates. Case (a): an interior point or B^* or C^* (not shown); Case (b): B^*; Case (c): C^*. The labels M^{forw}, M^{up}, M^{down} refer to the situation before the motion starts. In some cases, it does not matter whether B^* or C^* lies on a vertex or not. This is indicated by dotted variations of the polygon P.

Proof. We prove that the described movement maintains optimality. If B^*C^* rotates around B^*, it can be for two reasons: Either we are in Case (b), or M^{forw} coincides with B^* in Case (a). In both cases, C^* will be interior to $e^{\text{forw}}(C^*)$ after the rotation starts, $e^{\text{back}}(C^*)$ will coincide with this edge $e^{\text{forw}}(C^*)$, and M^{forw} becomes M^{up}. Thus, M^{up} will be on B^*C^*, in Case (a), or below B^*C^*, in Case (b). M^{down} stays the same as before. Since B^*C^* was assumed to be optimal, M^{down} lies on or above B^*C^*, and it remains so since B^*C^* rotates downwards. Thus the optimality conditions are preserved.

If B^*C^* rotates around C^*, the argument holds *mutatis mutandis*.

Finally, if M^{forw} lies in the interior of B^*C^* in Case (a) and B^*C^* rotates around this point, then $M^{\text{up}} = M^{\text{down}} = M^{\text{forw}}$ after the rotation starts, and optimality is clear. □

We mention that the Movement Rule gives the right movement when B^*C^* coincides with the **u**-extreme edge of P: Then $T^{\text{forw}} = C^*$, and M^{forw} lies below B^*C^*. Hence the segment will rotate around B^*.

7 A linear-time algorithm for the Circular Sweep

It is straightforward to distill a linear-time algorithm for finding the largest anchored triangles for all directions θ from Lemmas 1 and 16:

We first compute the largest anchored triangle $A^*(\theta_0)B^*(\theta_0)C^*(\theta_0)$ for the starting direction $\theta_0 = 0°$. This triangle can be found in $O(\log^2 n)$ time[12] by nested binary search on the left and right boundary of P for the optimal height h, using the local optimality criteria of Lemma 3. Since we are going to spend linear time anyway, and since we need to do this only once for the initialization, we can instead perform a simple linear scan in linear time.[13]

We increase θ continuously to 360° and move the three corners A^*, B^*, C^* along.[14] We imagine this as a continuous process. We have to watch for three types of events, as described in the proof of Theorem 5, see Figure 7:

1. A^* jumps to the next corner.
2. A sliding corner B^* or C^* arrives at a vertex.
3. The segment B^*C^* hits a critical point M^{up} or M^{down}.

Whenever this happens, we are at a breakpoint, and we determine how the motion continues with the help of the Movement Rule of Lemma 7. By Theorem 5.iii, there are $O(n)$ events, and an event can be processed in $O(1)$ time. Thus, the overall effort is linear.

If we are looking for a largest inscribed triangle, Lemma 6 implies that it is sufficient to evaluate the area at the breakpoints and take the maximum. If we are looking for a smallest circumscribed triangle, we additionally have to consider the possibility of an interior local minimum, which is constructed according to Figure 9c for those intervals where B^*C^* rotates around an interior point M.[15]

Thus, we have achieved a linear-time algorithm, both for the largest inscribed triangle and the smallest circumscribed triangle. As we will see in the subsequent sections, there are special properties of the two problems that allow the algorithm to be simplified.

We can even construct the complete function $F(\theta)$, as in Figure 1. It is a continuous piecewise smooth function with at most $5n+1$ pieces. It is not hard to see from Figure 9b that each piece can be written in the form $F(\theta) = \alpha + \beta_1 \tan(\theta + \gamma_1) + \beta_2 \tan(\theta + \gamma_2)$ for some constants $\alpha, \beta_1, \gamma_1, \beta_2, \gamma_2$.

[12] See [KlLa, Section 2]. Footnote 21 sketches a faster method with only $O(\log n)$ runtime.

[13] We can either search from bottom to top or from top to bottom. A different initialization procedure, which searches B^* and C^* by moving a tentative point B^* from bottom to top while advancing C^* downwards from top to bottom, is described in Section 9, see Footnote 23.

[14] It is actually sufficient to sweep up to 180°: The largest or smallest triangle ABC will be discovered whenever $\mathbf{u}(\theta)$ is the outer normal of one of the three sides of ABC.

[15] In the beginning of Section 9, we will see that this extra effort for looking in the interior of intervals can be saved.

8 Speed-up for the largest inscribed triangle

It is well-known that the largest triangle has its corners at vertices of P:

Lemma 8. *The largest inscribed triangle ABC in a polygon P can be found among the triangles whose corners A, B, C are among the vertices of P.*

Proof. If a corner lies in the interior of an edge, then it can slide to one of the two endvertices of this edge without decreasing the area, keeping the other two corners fixed. □

Keeping this property in mind, we restrict our attention to points $A(\theta), B(\theta), C(\theta)$ that lie on vertices of P. We can formulate the following

> *Skipping Principle.* When, at any time during the Circular Sweep, it becomes known that $B^*(\theta)$ lies on a point B in the interior of an edge $p_i p_{i+1}$ of the polygon, or that it must lie ahead of such a point B, then it is not necessary to increase θ continuously. We can immediately advance B^* to the forward endpoint p_{i+1} of this edge, and adjust θ accordingly.
>
> The same statement holds for C^*.

8.1 The Skipping Algorithm

This results in the algorithm shown in Figure 11. The algorithm maintains three points A, B, C that move counterclockwise through the vertices of P. When we say we *advance* A or B or C we mean that we move it to the next vertex of P. The next vertex after A is denoted by $\mathit{next}(A)$. The direction θ does not explicitly appear in the algorithm but we can think of $\mathbf{u}(\theta)$ as attached to BC as its normal vector.

Compute $A^*(\theta_0)$, $B^*(\theta_0)$, and $C^*(\theta_0)$ for $\theta_0 = 0°$. (Initialization)
set A to the forward endpoint of $e^{\text{back}}(A^*(\theta_0))$
set B to the forward endpoint of $e^{\text{back}}(B^*(\theta_0))$
set C to the forward endpoint of $e^{\text{back}}(C^*(\theta_0))$
$\mathit{maxarea} := 0$
(∗) **while** B is not to the left of C: (θ has not completed a half-turn)
(i) **if** area $\mathit{next}(A)BC \geq$ area ABC:
advance A. (Move towards the extreme point $A^*(\theta)$ in direction $-\mathbf{u}(\theta)$)
(ii) **else if** decreasing h would increase the area:
advance C. (Move towards $C^*(\theta)$)
(iii) **else if** increasing h is possible and would increase the area:
advance B. (Move towards $B^*(\theta)$)
else: (Now $BC = B^*(\theta)C^*(\theta)$, and ABC is a candidate for the largest triangle.)
$\mathit{maxarea} := \max(\mathit{maxarea}, \text{area } ABC)$
(iv) Determine how the edge B^*C^* will rotate when θ continues to increase.
It rotates either
around B^* or
around C^* or
around a critical pivot point M in the interior of the edge B^*C^*.
Accordingly, either C^*, or B^*, or both points move.
Advance the corresponding point C, or B, or both B and C

Figure 11: The Skipping Algorithm for the largest inscribed triangle

The initialization makes sure that A lies at a vertex, and it advances B and C to the next vertex if $B^*(\theta_0)$ or $C^*(\theta_0)$ lies in the middle of an edge, following the Skipping Principle.

The test (i) ensures that the rest of the loop is not entered before A is at the point $A^*(\theta)$ for the current direction θ. In case of a tie, we advance A in order to be prepared for increasing θ in step (iv).

The advancements in steps (ii)–(iv) are justified by the Skipping Principle. The conditions in steps (ii) and (iii) are checked according to the Improvement Test (conditions (1)–(2)) and the criteria (a) and (b) of Lemma 3. The test in step (iv) is carried out according to the Movement Rule (Lemma 7).

The termination condition $(*)$ will be discussed in Section 8.3.

8.2 Simplifying the test: Jin's Algorithm

The tests (ii)–(iv) can be subsumed in one simple common test:

    Construct the point M^{forw}
    if M^{forw} lies below BC:
        advance C
    else if M^{forw} lies on or above BC or M^{forw} does not exist:
        advance B

Indeed, by construction, M^{forw} lies higher than M^{up} and lower than M^{down}. Thus, if the test (ii) succeeds because M^{down} lies below BC, then M^{forw} lies below BC and the simplified algorithm will do the right thing. If the test (iii) succeeds because M^{up} lies on or above BC, the analogous argument leads to the same conclusion. (If M^{up} does not exist then M^{forw} does not exist.)

Finally, let us consider the test (iv). It is carried out when $BC = B^*C^*$, and hence the Movement Rule (Lemma 7) applies. If M^{forw} does not lie on B^*C^* (Cases (b) and (c) of Lemma 7), the segment rotates around one endpoint, and the other endpoint can be advanced. The simplified algorithm makes the right choice. Finally, if M^{forw} lies on B^*C^*, the simplified algorithm always advances B, whereas the original Skipping Algorithm would sometimes advance C or both points. If M^{forw} lies in the interior of B^*C^*, the original Skipping Algorithm advances both points. Here, the simplified algorithm behaves differently. However, advancing only B is still correct since it is justified by the Skipping Principle. (It is simpler to avoid an extra test and miss a few opportunities of advancing a point.)

The only case when there would be a discrepancy between the Skipping Algorithm and the simplified test is when $M^{\text{forw}} = B^*$ and therefore C should be advanced, see the second example in Figure 10. However, M^{forw} can coincide with B^* only if the edge $e^{\text{forw}}(B^*)$ extends all the way down to A^*. Since $B^* = B$ is a vertex of P, this case is excluded.

The whole loop, together with the advancement of A, becomes extremely simple:

    while B is not to the left of C:
        while area $\mathit{next}(A)BC \geq$ area ABC:
            advance A
        $\mathit{maxarea} := \max(\mathit{maxarea}, \text{area}\, ABC)$
        if M^{forw} exists and lies below BC:
            advance C
        else:
            advance B

Since we don't distinguish whether $BC = B^*C^*$, we simply take all triangles ABC that we encounter after the loop (i) as candidates for the largest triangle.[16] We call this *Jin's Algorithm.*

[16] On a superficial level, the algorithm resembles the incorrect algorithm of Dobkin and Snyder [DS]. However, that algorithm controls the advancement of B and C by a different criterion, namely the comparison of areas.

The algorithm for the largest inscribed triangle as described in Jin [Jin] uses a different initialization[17] and termination condition, but apart from that, it differs only in minor details.[18] Jin did not derive his algorithm as a simplification of the circular sweep over all anchored triangles, but with a different approach, the *Rotate-and-Kill* method.

A nice feature of this algorithm, besides a potential speedup, is that the only points A, B, and C that are ever considered are vertices of the polygon. Even in the initialization step, when B^* and C^* are found by scanning the left and right side of P simultaneously, it is never necessary to explicitly handle any boundary points B^* and C^* that are not vertices. All that is necessary is a comparison of critical points M with vertices.

8.3 Correctness, termination, and running time

The Skipping Algorithm starts with $\theta = 0°$ and rotates the direction until the condition $(*)$ indicates termination. This happens when the normal direction of BC falls in the range $180° < \theta < 360°$. The Skipping Algorithm is guaranteed to visit at least all triangles $A^*(\theta)B^*(\theta)C^*(\theta)$ for which both $B^*(\theta)$ and $C^*(\theta)$ lie at vertices of P (in addition to $A^*(\theta)$). The largest inscribed triangle has these properties, and, like every triangle, it has some normal $\mathbf{u}(\theta)$ in the range $0° \le \theta < 180°$. Thus, it is ensured that the largest inscribed triangle is found before the algorithm terminates.

The termination argument is a little subtle because the three points A, B, C are not always distinct.

Lemma 9. *Assume that P has at least 3 vertices. In the Skipping Algorithm, both in the original and the simplified version, collisions between the points A, B, C are subject to the following constraints:*

a) The points B and C are always distinct.

b) As the points are advanced, C can catch up with A, and A can catch up with B, but no point overtakes another point.

c) Consequently, the points A, B, C are always in counterclockwise order whenever they are distinct.

Proof. We have seen after Lemma 7 that B is not advanced when $C = \mathit{next}(B)$, because this is the case when BC is the $\mathbf{u}$-extreme edge. It is possible that C catches up with A (even right after initialization), but then A will immediately advance. So C cannot overtake A.

The point A can only catch up with B if $B = \mathit{next}(C)$. This can indeed happen, for example when P is a triangle. In this situation, the next step will advance B. Thus, B and C remain always distinct, and A cannot overtake B.

In the original version of the Skipping Algorithm, there is a case when both B and C move simultaneously, but then the only collision that can happen is that C runs into A, and this case has been treated above. □

Since B and C are always distinct, the segment BC has a well-defined direction.

---

[17] Jin initializes his algorithm with a "3-stable" triangle. One can easily see that a 3-stable triangle is the largest anchored triangle for all three directions to which it is anchored. Jin shows how to find such a triangle in linear time by a simple algorithm, which considers only triangles with vertices from the polygon [Jin, Section 2].

[18] The main difference is that the algorithm in [Jin] does not advance A in case of equality. Our choice of advancing A was necessary for the original Circular Sweep Algorithm of Section 7, but one can easily check that in the present simplified algorithm, it makes no difference whether we advance A or not in case of ties. The other difference is that the test is not expressed in term of the critical point M^{forw} but in another, equivalent way, see Appendix B.2.2.

Lemma 10. *The counterclockwise change of direction of the segment BC in one step of the algorithm is less than* $180°$.

Proof. The points B and C can advance only one vertex at a time (perhaps simultaneously, in the original Skipping Algorithm). Now, consider moving two points B and C forward on the boundary of a convex region from some starting position B_0C_0, without B moving past C_0 or C moving past B_0, see Figure 12. Then one can turn the segment BC by at most $180°$, and the only way to reach $180°$ is for B and C to swap places, but this is impossible in one step in a polygon with more than 2 vertices. ☐

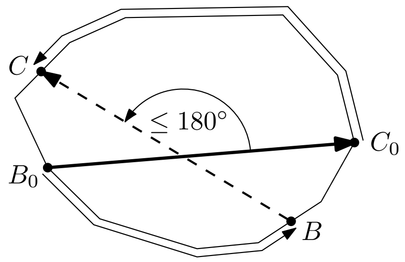


Figure 12: How much BC can rotate in one step

So we know that the direction θ increases from the initial value $0°$ in steps less than $180°$. Thus it cannot jump over the terminating interval $180° < \theta < 360°$ in one step. Consequently, the total counterclockwise turn of the segment BC is less than $360°$.

Termination in linear time is now guaranteed by the fact that each loop iteration advances one or several of the points A, B, C, and the points cannot overtake each other. ☐

Exercise. 1. True or false:

The loop can be stopped already as soon as A is to the right of B,

(a) because the sequence of triangles starts repeating from this point, with rotated labels A, B, C;

(b) for a different reason.

2. In case this improved termination condition works: At what point in the algorithm should it be tested?

9 Speed-up for the smallest circumscribed triangle

We will now specialize the Circular Sweep algorithm of Section 7 to smallest circumscribed triangles. The following basic observation allows to simplify the algorithm in this case.

Lemma 11. *There is a smallest circumscribed triangle that touches a polygon edge.*[19]

Proof. The smallest circumscribed triangle is anchored at some direction $\mathbf{u}$. According to Lemma 4.ii, there is a smallest circumscribed triangle anchored at that direction with the claimed property. ☐

A circumscribed triangle that touches a polygon edge is anchored at the outer normal direction of that edge. Thus, *it suffices to look at* $F(\theta)$ *for those breakpoints which are* inner *normals*

[19] In fact, *every* smallest circumscribed triangle has this property. This follows from [KlLa, Lemma 1.3], see also [KlLa, Theorem 2.1.iv]. Also, there is a smallest circumscribed triangle that touches at least two polygon edges, cf. [OAMB, Lemma 1].

of polygon edges (where A^* jumps). In particular, this implies that it is not necessary to look at the local minima in the interior of the intervals: The same minima will also be discovered at breakpoints.

One can use this observation to shortcut the sweep more aggressively. In the algorithm of O'Rourke, Aggarwal, Maddila, and Baldwin [OAMB], θ jumps from one inner normal direction of P to the next. Like in the Skipping Algorithm in Figure 11, two points B and C are maintained. After increasing θ, we will approach $B^*(\theta)$ and $C^*(\theta)$ step by step by moving either B or C to the next vertex.

A typical situation is shown in Figure 13a. We again assume w. l. o. g. that $\mathbf{u}$ points vertically upward. We have the points $B = B^*(\theta^{\text{old}})$ and $C = C^*(\theta^{\text{old}})$ from the previous direction θ^{old}, and we have advanced θ to the next edge, on which A^* now lies. By Theorem 5.ii, we know that the points $B^* = B^*(\theta)$ and $C^* = C^*(\theta)$ can only lie ahead of B and C. Statements 1 and 2 of the following lemma allows us to advance B or C while maintaining this property.

The boundary of P has a *left side* and a *right side*, relative to the current direction $\mathbf{u}$. (Edges perpendicular to $\mathbf{u}$ belong to neither side.) We denote by $h(X)$ the *height* of the point X over A^* in direction $\mathbf{u}$.

Lemma 12. [20] *Let $(B, next(B))$ be an edge on the right side and let $(C, next(C))$ be an edge on the left side of a convex polygon P. Let M^{forw} be the critical point computed from these edges, as illustrated in Figure 13a.*

1. If $h(M^{\text{forw}}) \le h(next(C))$ and $h(next(C)) \ge h(B)$, then

$$h(B^*) = h(C^*) \le h(next(C)).$$

2. If $h(M^{\text{forw}}) \ge h(next(B))$ or M^{forw} does not exist, and $h(next(B)) \le h(C)$, then

$$h(B^*) = h(C^*) \ge h(next(B)).$$

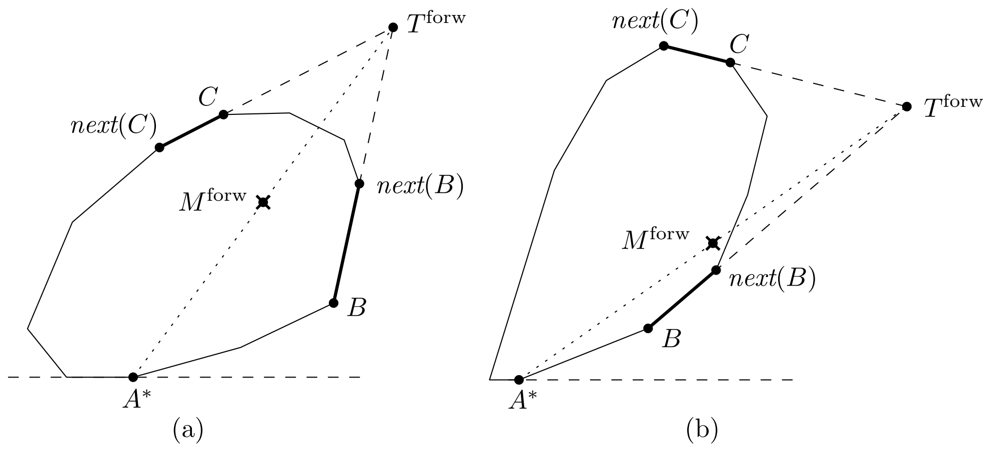


Figure 13: (a) The criterion for advancing either B or C. (b) The situation when C is not on the left side.

Proof. 1. We apply the Improvement Test to the upward direction at height $h(next(C))$. The point M^{up} is determined by the ray through $(C, next(C))$, and the ray through either $(B, next(B))$ or through a higher edge. Since the edges bend inwards when going from $(B, next(B))$

[20] [OAMB, Lemmas 3 and 4] obtain the same conclusions under the stronger assumption that $h(next(C)) > h(next(B))$.

to a higher edge, $h(M^{\mathrm{up}}) \le h(M^{\mathrm{forw}})$. With the assumption $h(M^{\mathrm{forw}}) \le h(\mathit{next}(C))$ we get $h(M^{\mathrm{up}}) \le h(\mathit{next}(C))$, which implies that B^*C^* cannot lie above $\mathit{next}(C)$.

2. The second statement is completely analogous. We apply the Improvement Test to the downward direction at height $h(\mathit{next}(B))$. The point M^{down} is determined by the ray through $(B, \mathit{next}(B))$, and the ray through either $(C, \mathit{next}(C))$ or through a lower edge. Since the edges bend outwards when going from $(C, \mathit{next}(C))$ to a lower edge, $h(M^{\mathrm{down}}) \ge h(M^{\mathrm{forw}})$ if M^{down} exists, and if M^{forw} does not exist, the M^{down} does not exist. With the assumption $h(M^{\mathrm{forw}}) \ge h(\mathit{next}(B))$ we get $h(M^{\mathrm{down}}) \ge h(\mathit{next}(B))$ if M^{down} exists at all, which implies that B^*C^* cannot lie below $\mathit{next}(B)$. □

```
(i)     B := p_1 and C := p_2
        minarea := ∞
        for i := 1 .. n:
            let u be the inner normal of the edge p_{i-1}p_i, and set A* := p_i
(ii)        while h(next(C)) ≥ h(C): (Ensure that C lies on the left side)
                advance C
            loop
(iii)           compute the critical point M^forw = (T^forw + A*)/2 from the intersection T^forw
                           of the forward extension of the edge (B, next(B))
                           and the backward extension of the edge (C, next(C)).
(iv)            if h(M^forw) ≤ h(next(C)):
                    if h(next(C)) ≥ h(B):
                        advance C downwards to next(C)
                    else: exit loop
(v)             else if h(M^forw) ≥ h(next(B)), or M^forw does not exist:
                    if h(next(B)) ≤ h(C):
                        advance B upwards to next(B)
                    else: exit loop
                else: exit loop
            We now know the edges (B, next(B)) and (C, next(C)) on which B* and C* lie.
            Construct the horizontal segment B*C* between these edges as follows:
(vi)        if h(M^forw) ≤ h(B):
                B* = B, and C* lies on (C, next(C)) at the same height as B
(vii)       else if h(M^forw) ≥ h(C) or M^forw does not exist:
                C* = C, and B* lies on (B, next(B)) at the same height as C
(viii)      else:
                B*C* lies at the height of M^forw
            minarea := min(minarea, 4 · area A*B*C*).
```

Figure 14: The linear-time algorithm specialized for the smallest circumscribed triangle

This lemma is important because it allows to draw conclusions about the height of B^*C^* from two points B and C which are not at the same height.[21] It is the basis for the algorithm

[21] Lemma 12 can be used to compute a largest anchored triangle by the prune-and-search technique in $O(\log n)$ time, as opposed to the nested binary search mentioned at the beginning of Section 7, which requires $O(\log^2 n)$ time. Lemma 12 is geared towards the case that $(C, \mathit{next}(C))$ is higher than $(B, \mathit{next}(B))$. It has a mirror-symmetric version in which the roles of $B, \mathit{next}(B), C, \mathit{next}(C)$ are swapped with $\mathit{next}(C), C, \mathit{next}(B), B$, and which applies in the opposite situation.

We maintain a chain of *left candidate edges* and a chain of *right candidate edges* on which the true points B^* and C^* can lie. Initially, all left and right edges are candidates. We pick the median B and C from each chain and compute M^{forw} for the edges $(B, \mathit{next}(B))$ and $(C, \mathit{next}(C))$. If the height ranges of $(B, \mathit{next}(B))$ and $(C, \mathit{next}(C))$

shown in Figure 14. There is an outer loop that cycles through all inner normal directions $\mathbf{u}$ of the edges and records the largest area. In the main inner loop (iii)–(v), the algorithm looks for the edges $(B, \mathit{next}(B))$ and $(C, \mathit{next}(C))$ that contain B^* and C^*. For this purpose, B and C are advanced towards B^* and C^*, maintaining the invariant that *the correct segment* B^*C^* *is sandwiched between* B *and* C:

$$h(B) \leq h(B^*) = h(C^*) \leq h(C) \tag{3}$$

Let us discuss how this invariant is maintained, ignoring for the time being the question of termination of the inner loop and the computation of B^*C^* in the second part (vi)–(viii). After the tests (iv) and (v), the advancement of B and C is justified by Lemma 12, provided that B lies on the right side and C lies on the left side. It can happen that C starts out on the right side after the direction $\mathbf{u}$ has been advanced, as shown in Figure 13b. The loop (ii) ensures that C is moved over to the left side before the main loop starts.[22] Another potentially dangerous situation is that B lies on the bottom horizontal edge, to the left of A^*. In this case, T^{forw} does not exist, and the algorithm automatically advances B to coincide with A^*, which is the right action. So, this special case is resolved without requiring special treatment.

When the point A^* together with the normal direction $\mathbf{u}$ is advanced, we leave the points B and C as they are. We don't initialize them to the previous points B^* and C^*, as suggested in our first informal description of the algorithm. The invariant is maintained according the monotonic movement of $B^*(\theta)$ and $C^*(\theta)$ (Theorem 5.ii) *unless* B or C are on the wrong side after the increase of θ. In this case, as we have seen, the algorithm will first move B and C to the first vertex on the correct side, either explicitly in (ii), for C, or implicitly, for B, and the invariant is trivially reestablished at this point.

To *initialize* the process for the first direction $\mathbf{u}$, we start with B at the lowest vertex on the right side and C at the highest vertex on the left side. Then the invariant holds trivially. In line (i), we have set C to the vertex after B, and the loop (ii) ensures that C crawls over to the top of the left side.[23]

It is easy to argue that the inner loop must *terminate*: We have proved that we maintain the invariant (3), and in particular, the relation $h(B) \leq h(C)$. Since B moves upwards and C moves downwards from vertex to vertex, the loop cannot run forever.

Let us now turn to the construction of B^*C^* in (vi)–(viii). The following auxiliary lemma will be used to justify the cases (vi) and (vii). In contrast to Lemma 12, this lemma applies when the vertical ranges of the edges $(B, \mathit{next}(B))$ and $(C, \mathit{next}(C))$ overlap.

Lemma 13. *In addition to the assumptions of Lemma 12, we assume the invariant* (3).

---

don't overlap, then Lemma 12 or its symmetric counterpart is guaranteed to give a conclusion about the relative height of B^* and C^* with respect to at least one of the four involved vertices. Thus, we can discard half of the edges of either the left chain or the right chain, give or take one or two edges for rounding effects.

If the height ranges of $(B, \mathit{next}(B))$ and $(C, \mathit{next}(C))$ overlap, we can perform the standard Improvement Test for one of the vertices that lie in the overlapping interval, and discard half of the edges from *both* chains.

One chain is reduced to a constant number of edges after $O(\log n)$ such tests, and the search can be finished with standard binary search in $O(\log n)$ time.

[22] The loop (ii) can actually be omitted because the second loop (iii)–(v) will do the right thing on its own: Our definition of M^{forw} from the *backward* extension of the edge $(C, \mathit{next}(C))$ in line (iii) is appropriate for this situation. This definition is consistent with our convention of using the "upward" ray extending $(C, \mathit{next}(C))$ when C lies on the left side, and it is also consistent with the way how T^{forw} is conveniently computed, see Appendix B.2.1. In Figure 13b, we see that, when C is on the right side, T^{forw} is below C and $\mathit{next}(C)$, and so is M^{forw}. Also, since B and C are on the right side, $h(\mathit{next}(C)) \geq h(C) \geq h(B)$, and thus the algorithm advances C. (For this case, the description of advancing C "downwards" in (iv) is not appropriate.) The same happens in the boundary case when the edge $(C, \mathit{next}(C))$ is a horizontal $\mathbf{u}$-extreme edge.

[23] This initialization together with a single iteration of the outer loop gives another way of computing a largest anchored triangle from scratch in linear time, and thus for initializing any of the other derivatives of the Circular Sweep algorithm.

1. If $h(M^{\text{forw}}) \le h(B)$ and $h(\mathit{next}(C)) \le h(B)$, then

$$h(B^*) = h(C^*) = h(B)$$

2. If $h(M^{\text{forw}}) \ge h(C)$ or M^{forw} does not exist, and $h(\mathit{next}(B)) \ge h(C)$, then

$$h(B^*) = h(C^*) = h(C).$$

The proof will be given below.

There are three ways of terminating the inner loop. Let us first consider the exit when neither of the conditions (iv) and (v) hold:

$$h(\mathit{next}(C)) < h(M^{\text{forw}}) < h(\mathit{next}(B)) \tag{4}$$

The easiest case is case (viii), when $h(B) < h(M^{\text{forw}}) < h(C)$: then both points B^*C^* at height $h(M^{\text{forw}})$ lie in the interior of the respective edges $(B, \mathit{next}(B))$ and $(C, \mathit{next}(C))$. The critical points $M^{\text{up}} = M^{\text{down}} = M^{\text{forw}}$ coincide, and the optimality condition of Lemma 3 is fulfilled.

In case (vi), $h(M^{\text{forw}}) \le h(B)$, and this together with (4) implies the second assumption of Lemma 13.1. In case (vii), $h(M^{\text{forw}}) \ge h(C)$, and this together with (4) implies the second assumption of Lemma 13.2. In either case, Lemma 13 justifies the decision of the algorithm.

If the exit of the loop was through (iv), then

$$h(M^{\text{forw}}) \le h(\mathit{next}(C)) < h(B).$$

The algorithm will thus take the branch (vi), and this is justified by Lemma 13.1.

If the exit of the loop was through (v), then

$$h(M^{\text{forw}}) \ge h(\mathit{next}(B))$$

if M^{forw} exists, and

$$h(\mathit{next}(B)) > h(C).$$

The algorithm will thus take the branch (vii), and this is justified by Lemma 13.2.

To conclude the correctness proof of the algorithm, we supply the easy proof of Lemma 13:

Proof of Lemma 13. 1. Let us tentatively put B^*C^* at the height of $h(B)$ and see why this is the correct solution. If $h(B) = h(C)$, the invariant (3) leaves no other choice for B^*C^*. Otherwise, $h(C) > h(B)$, and then the critical point M^{up} at the height of B is determined by the same edges as M^{forw}. Thus, $h(M^{\text{up}}) = h(M^{\text{forw}}) \le h(B)$. Then the optimality condition of Lemma 3b tells us that the optimal segment B^*C^* does not lie above B. By the invariant, we know that B^*C^* cannot lie below B. Thus, B^*C^* must go through B.

2. The second statement is analogous. □

It is easy to see that the algorithm takes only linear time. There are n iterations of the outer loop, advancing A^* one vertex at a time. Each iteration of an inner loop advances B or C, but B and C cannot overtake A^*.

This algorithm shares the nice feature with Jin's Algorithm of Section 8.2 that the points A^*, B, and C range only over vertices.

The shortcut in this section is similar in spirit to the shortcut introduced in Section 8 for the largest inscribed triangle by way of the Skipping Principle. The crucial difference is that, in Section 8, we advance B and C and let $\mathbf{u}$ follow. Here, we advance the direction $\mathbf{u}$, and B and C have to catch up.

A Literature

This note gives a self-contained development of linear-time algorithms for largest inscribed and the smallest circumscribed triangle, starting from scratch. The essential ideas and inspirations have been taken from the literature, but I have tried to streamline the presentation for simplicity. A distinguishing feature of my treatment is the central role that is given to the critical pivot point M. As discussed in Appendix B.2.2, the same optimality condition (Lemma 3) appears in various other guises in the literature. I hope that my presentation may contribute to the clarification of the ideas underlying the algorithms. I have sprinkled the text with footnotes that acknowledge sources or clarify clashes of terminology.

I give a brief account of the relevant literature in chronological order, together with the publication dates.

- Dobkin and Snyder [DS] in 1979 were the first to propose a linear-time algorithm for the largest inscribed triangle. In 2017, this algorithm found to be wrong, see below.
- In 1985, Klee and Laskowski [KlLa] developed an algorithm for computing the smallest circumscribed triangle in $O(n \log^2 n)$ time.[24]
- Building on this work, O'Rourke, Aggarwal, Maddila, and Baldwin [OAMB] improved this in 1986 to linear time.[25]
- In 1992, Chandran and Mount [ChMo] noted the strong connection between the largest inscribed triangle and smallest circumscribed triangle problems, and they succeeded to solve both problems simultaneously in linear time.[26] At the time, this did not offer any improvement over existing algorithms regarding the asymptotic running time. The selling point of this paper were fast parallel algorithms for the two problems.
- In 2017, Keikha, Löffler, Urhausen, and van der Hoog found out that the algorithm of Dobkin and Snyder [DS] does not work. Dobkin and Snyder [DS] had promised a proof of a crucial lemma in a subsequent full version, but this never appeared. Keikha et al. presented a counterexample in the first version of the arXiv preprint [K^+] in May 2017, and they were initially unaware of the previous linear-time solution of Chandran and Mount [ChMo]. As a replacement for the incorrect solution, they proposed a divide-and-conquer algorithm of running time $O(n \log n)$ for the largest inscribed triangle.
- The discovery of the mistake in [DS] prompted two linear-time algorithms that were again posted as arXiv preprints: By Kallus [Kal], posted in June 2017, and by
- Jin [Jin], whose first version was posted in July 2017. Both papers deal with the largest inscribed triangle problem.[27] In subsequent versions of [Jin], the smallest circumscribed triangle problem is also treated.

References

[BRS] Peter Braß, Günter Rote, and Konrad J. Swanepoel. Triangles of extremal area or perimeter in a finite planar point set. *Discrete and Computational Geometry*, 26:51–58, 2001. `doi:10.1007/s00454-001-0010-6`.

[ChMo] Sharat Chandran and David M. Mount. A parallel algorithm for enclosed and enclosing triangles. *International Journal of Computational Geometry & Applications*, 2(2):191–214, 1992. `doi:10.1142/S0218195992000123`.

[24] This algorithm computes the largest triangle anchored to the each inner edge normal. Each triangle is computed from scratch in $O(\log^2 n)$ time, as sketched at the beginning of Section 7.

[25] This is essentially the algorithm in Section 9.

[26] This is the Circular Sweep algorithm in Section 7.

[27] The algorithm of Kallus is a rediscovery of the Circular Sweep algorithm of Chandran and Mount [ChMo], restricted to the case of the largest inscribed triangle problem. The algorithm of Jin is described in Section 8.2.

[DS] David P. Dobkin and Lawrence Snyder. On a general method for maximizing and minimizing among certain geometric problems. In *20th Annual Symposium on Foundations of Computer Science*, pages 9–17. IEEE, 1979. `doi:10.1109/SFCS.1979.28`.

[Jin] Kai Jin. Maximal area triangles in a convex polygon. Preprint, September 2017. `arXiv:1707.04071v3`.

[Kal] Yoav Kallus. A linear-time algorithm for the maximum-area inscribed triangle in a convex polygon. Preprint, June 2017. `arXiv:1706.03049`.

[K+] Vahideh Keikha, Maarten Löffler, Ali Mohades, Jérôme Urhausen, and Ivor van der Hoog. Maximum-area triangle in a convex polygon, revisited. Preprint, 2017. `arXiv:1705.11035v2`.

[KlLa] Victor Klee and Michael C. Laskowski. Finding the smallest triangles containing a given convex polygon. *Journal of Algorithms*, 6(3):359–375, 1985. `doi:10.1016/0196-6774(85)90005-7`.

[OAMB] Joseph O'Rourke, Alok Aggarwal, Sanjeev Maddila, and Michael Baldwin. An optimal algorithm for finding minimal enclosing triangles. *Journal of Algorithms*, 7(2):258–269, 1986. `doi:10.1016/0196-6774(86)90007-6`.

B Primitive operations

There are two basic operations in the algorithms, besides the calculation and comparison of triangle areas:

1. Constructing the critical pivot point M and comparing it to the edge BC, in order to decide in which direction the "current triangle" should be improved.
2. Finding the next breakpoint when θ is rotated.

Since the first operation is tied to the optimality condition of anchored triangles, this test occurs in every algorithm that is based on anchored triangles. As we have seen in Section 8.2, it also appears in Jin's Algorithm, even though Jin's own derivation [Jin] does not refer to anchored triangles at all.

The second operation has to consider the up to three candidates for terminating the current interval (see Figure 7) and compare them to see which one is next. This operation also involves a critical point M, either as a pivot point or as an obstacle that might be hit by a rotating segment. This operation appears only in the original Circular Sweep Algorithm and not in the simplified versions that are specialized for the largest inscribed triangle or smallest circumscribed triangle.

After introducing the wedge product as a basic operation in Section B.1, we consider the two basic operations in Sections B.2 and B.3. In Section B.4, we discuss the degree of the algebraic expressions that arise when carrying out the primitive operations on a computer. Finally, Section B.5 investigates the degree of the area computation for circumscribed triangles.

B.1 The area of the parallelogram spanned by two vectors

For two vectors or points $\vec{a}_1 = \binom{x_1}{y_1}$ and $\vec{a}_2 = \binom{x_2}{y_2}$ in the plane, we use the wedge product notation for the signed area of the parallelogram spanned by $\vec{a}_1$ and $\vec{a}_2$:

$$\vec{a}_1 \wedge \vec{a}_2 = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1.$$

B.2 The improvement test for anchored triangles

We first develop the algebra for the Improvement Test (Section B.2.1). In Section B.2.2, we compare how this test is expressed geometrically in different papers.

B.2.1 Algebraic calculation of the sign of the derivative of $f(h)$

We are given the vertices $p_1, p_2, \dots, p_n$ of the convex n-gon P in counterclockwise order. Indices are modulo n.

We want to calculate the sign of the one-sided derivative of $f(h)$ at some point h. According to the Improvement Test (conditions (1)–(2)) and the criteria of Lemma 3, this boils down to constructing the points T and M and testing the position of M with respect to BC.

We specify the test by five parameters: three indices i, j, k, the vector $\mathbf{u}$, and the point A^*. Their meaning is as follows: B moves on the line through the edge p_i, p_{i+1} and C moves on the line through the edge p_j, p_{j+1}. The current location of the segment BC is specified by one point p_k through which it goes and by the normal direction $\mathbf{u}$ (pointing to the right of BC). When the test is called, the point p_k is always one of $p_i, p_{i+1}, p_j, p_{j+1}$.

We start by computing the upward vectors $\vec{b} = p_{i+1} - p_i$ and $\vec{c} = p_j - p_{j+1}$. We compute the wedge product

$$\Delta = \vec{c} \wedge \vec{b}.$$

If $\Delta \le 0$, the forward extensions of $\vec{b}$ and $\vec{c}$ diverge, and the derivative of f is positive. Otherwise, their intersection point is

$$T = \frac{\hat{T}}{\Delta} \tag{5}$$

with

$$\hat{T} = (p_j \wedge p_{j+1}) \cdot \vec{b} + (p_i \wedge p_{i+1}) \cdot \vec{c}.$$

This formula can be worked out by solving the system of linear equations, or it can be checked by computing the products $T \wedge \vec{b}$ and $T \wedge \vec{c}$ and comparing them to what they should be.

To test whether $\frac{T+A^*}{2}$ is above or below BC, we have to check the sign of the scalar product $\left(\frac{T+A^*}{2} - p_k\right) \cdot \mathbf{u}$, which, after multiplying the denominator, becomes

$$S := \Big(\hat{T} + (A^* - 2p_k)\Delta\Big) \cdot \mathbf{u}. \tag{6}$$

The sign of this expression is the sign of the derivative of $f(h)$.

We can assume that both vectors $\vec{b}$ and $\vec{c}$ have a nonnegative scalar product with $\mathbf{u}$. Let us first make the additional assumption that at least one vector has a positive scalar product with $\mathbf{u}$. Then, if $\Delta < 0$, the computed intersection point T lies below A^*, and so does M, but the multiplication by Δ reverses the sign, leading to the correct (positive) sign of S. One can check that S is positive also for $\Delta = 0$. Thus, (6) can be used in all cases, and the sign test of Δ is not necessary. The test covers even the critical point M^{up} when $i = j$ and BC is the $\mathbf{u}$-extreme edge of P, which does not fall under the initial assumption: In this case, $\vec{c} = -\vec{b}$, $\hat{T} = \binom{0}{0}$, $\Delta = 0$, and $S = 0$, correctly indicating that no improvement is possible by increasing h.

The components of the vector in the large parentheses in (6) have degree 3 in the input variables. When the test is used in the algorithm, the vector $\mathbf{u}$ is typically the perpendicular vector of the next edge incident to A^* or of the vector BC between two vertices of P. The expression (6) has thus degree 4 in the input coordinates.

B.2.2 Geometric constructions of the improvement test

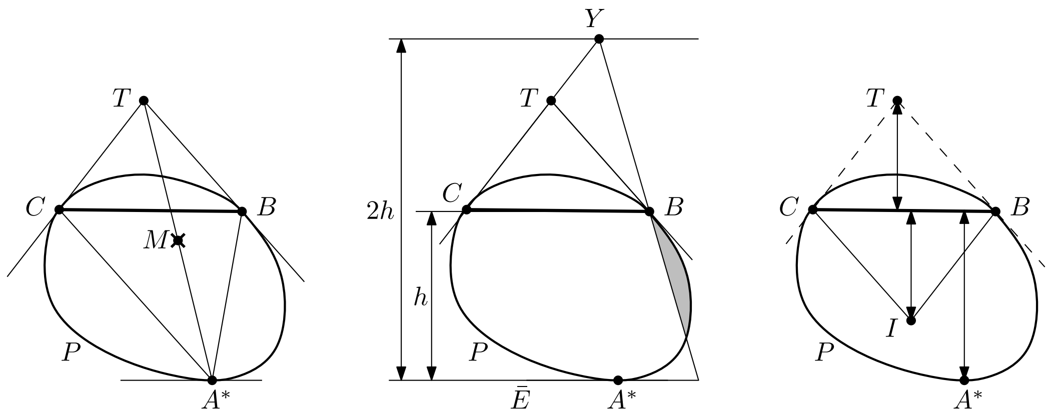
It is interesting to see how this test can be expressed geometrically in different ways. In the algorithms, the test is variously applied to the forward or backward edges incident to B and C. To abstract from these details the tests are illustrated with a smooth convex body P that has unique tangents everywhere. We have also unified the notation, and we don't necessarily use the same wording as in the original sources. Figure 15a shows the test as expressed in this note: Is the critical point $M = (A^* + T)/2$ below or above BC?

Figure 15b shows the criterion of Klee and Laskowski [KlLa, Figure 11], see also [OAMB, Figure 1]: Let h be the height of BC over the tangent $\bar{E}$ at A^* (which is parallel to BC). Now the tangent at C is extended to a point Y that has height $2h$. Then the line YB is formed, and the question is: Does YB intersect the polygon P below B or above B?

This example is particularly instructive: Our test starts with the given vertices and edges and proceeds by intersecting certain lines or drawing lines through certain points, and in the end, certain distances or locations are compared. In the critical situation, when the outcome of the test changes, there will be some extra incidence. In Figure 15a, the point T would have height $2h$ in the critical situation. Figure 15b performs the construction backwards and makes the comparison at an intermediate stage: It constructs what the tangent at B should be in the critical situation, namely the line YB, and compares it to the actual tangent at B.

This form of the test has the nice feature that it works regardless of whether the upward tangent rays through B and C meet. This is in accordance with the observation from the algebraic calculations in Section B.2.1 that it is not necessary to check whether the rays meet.

Figure 15c shows the criterion used by Jin [Jin]. It takes the fourth point I of the parallelogram $BTCI$ (without constructing T), and compares the distances of I and A^* from BC. This is obviously equivalent to the test in Figure 15a.



(a) this note (b) [KlLa] (c) [Jin]

Figure 15: The different geometric ways of expressing the direction of improvement

B.3 Finding the next breakpoint

B.3.1 Algebraic computation of the next breakpoint

For determining the next breakpoint, the three cases with the three candidate directions each are shown in Figure 7. For example, in Case II, we have to compare the directions $\mathit{next}(A) - A$, $M^{\text{down}} - C^*$, and $B^{\text{forw}} - C^*$, where B^{forw} is the endpoint of $e^{\text{forw}}(B)$. Two such vectors are compared by looking at the sign of their wedge product. Two of these directions are directions between two polygon vertices, and hace they come directly from the input data. The challenging case is the vector that involves a critical point. For example, to compare which of the vectors

$M^{\mathrm{down}} - C^*$ and $next(A) - A$ comes first, we look at the sign of the wedge product $(M^{\mathrm{down}} - C^*) \wedge (next(A) - A)$. This is a quadratic expression in the coordinates of the four involved points, but since M^{down} involves T^{down}, which is given by the rational expression (5), we multiply by Δ. As in (6), the result is a degree-4 polynomial.

Case I is similar. In Case III, it seems that we have to compare two directions that involve M, namely $B^{\mathrm{forw}} - M$ and $M - C^{\mathrm{forw}}$, which would lead to a degree-6 predicate. However, this comparison is equivalent to an orientation test for the triangle $B^{\mathrm{forw}} M C^{\mathrm{forw}}$. Thus we can compare, for example, $B^{\mathrm{forw}} - C^{\mathrm{forw}}$ against $B^{\mathrm{forw}} - M$ to get a degree-4 test with an equivalent outcome.

Summarizing, we have shown that the selection of the next event can be done by evaluating the signs of polynomials of degree at most 4 in the input coordinates.

B.3.2 Computation and construction of the next breakpoint in the literature

As in Section B.2.2, we want to compare how these tests are expressed in the literature. Two papers have suggested the conceptual sweep through all angles θ: Chandran and Mount [ChMo], and Kallus [Kal].

Kallus [Kal, Theorem 6] describes the necessary tests purely in algebraic terms, by setting derivatives to 0, without distilling the geometric content. [Kal, Listing 3] spells out the formulas for all primitives. Some of these expressions are rational expressions with numerator and denominator of degree 4, and they are compared against other rational expressions with numerator and denominator of degree 2. The comparison amounts to computing the sign of a degree-6 polynomial. So it appears that these expressions are not optimized.

Chandran and Mount's algorithm [ChMo], by contrast, is described in geometric terms, and we can compare their description to ours. Indeed, when both B^* and C^* move (Case III), they construct the point M around which B^*C^* rotates. This is [ChMo, Figure 5], covering the case where the outer triangle has "two flush legs". (The pivot is the point x in that figure, and it is constructed by considering the local optimality condition of the corresponding circumscribed triangle.)

The case of "one flush leg" covers Case I and II of Figure 7, where one point B^* or C^* remains at a vertex. The interesting event is that B^*C^* hits a critical point M. Figure 16 shows the test of [ChMo, Figure 6] for Case II, converted to our notation and our conventions. The construction extends the line from the intersection point T^{down} of the tangents towards the point C^*, going twice the distance $T^{\mathrm{down}}C^*$, and arriving at the point y. The direction where the critical event happens is determined by the line A^*y. Again, this criterion is found from the local optimality condition of the circumscribed triangle. Clearly, as the triangles $T^{\mathrm{down}}C^*M^{\mathrm{down}}$ and $T^{\mathrm{down}}yA^*$ are similar, this is the same direction as C^*M^{down}.

B.4 The degree of the predicates

As we have seen, the Improvement Test boils down to a sign test for a degree-4 polynomial. The degree is important when predicates are evaluated exactly, because it determines the blow-up of the involved numbers. The problem *statement* of the largest inscribed triangle, however, refers only the computation and comparison of triangle areas, which is an easy degree-2 operation.

All known linear-time algorithms require the Improvement Test in one form or another. There is an algorithm to compute the largest inscribed triangle in $O(n \log n)$ time, which only compares triangle areas [K$^+$]. Is there a linear-time algorithm that avoids degree-4 predicates?

B.5 The area of the circumscribed triangle

As for circumscribed triangles, Klee and Laskowski [KlLa] advertise their algorithm for finding all local minima of circumscribed triangles with the following words: "It does not compute any

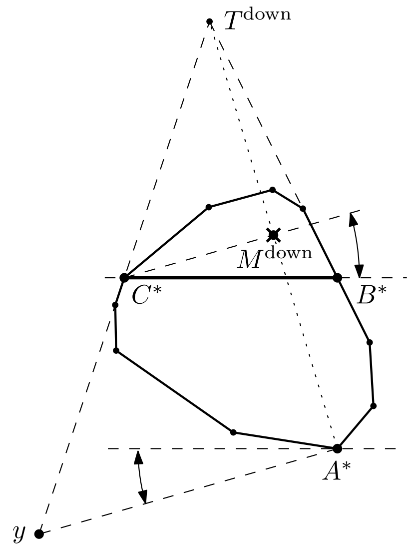


Figure 16: Two different geometric ways of finding the next event in Case II, when B^*C^* rotates around C^*

areas, but relies on a geometric characterization of the local minima and on simple computational steps such as finding intersections of lines." Actually, for circumscribed triangles, this is a justified remark, because the area of a triangle that is given by edges is not so nice to compute as when the vertices are given, and this is reflected in the algebraic degree. Consider a triangle where each side is specified by two points (x_i, y_i) and (u_i, v_i) through which it goes, for $i = 1, 2, 3$. Such a triangle, touching three edges of the input polygon P, can arise as a smallest circumscribed triangle. Its area is the following rational expression whose numerator has degree 8 and whose denominator has degree 6:

$$\frac{\pm \begin{vmatrix} x_1v_1 - y_1u_1 & x_1 - u_1 & y_1 - v_1 \\ x_2v_2 - y_2u_2 & x_2 - u_2 & y_2 - v_2 \\ x_3v_3 - y_3u_3 & x_3 - u_3 & y_3 - v_3 \end{vmatrix}^2}{2 \cdot \begin{vmatrix} x_1 - u_1 & y_1 - v_1 \\ x_2 - u_2 & y_2 - v_2 \end{vmatrix} \cdot \begin{vmatrix} x_2 - u_2 & y_2 - v_2 \\ x_3 - u_3 & y_3 - v_3 \end{vmatrix} \cdot \begin{vmatrix} x_3 - u_3 & y_3 - v_3 \\ x_1 - u_1 & y_1 - v_1 \end{vmatrix}}$$

This formula was calculated with the help of a computer algebra system. To compare two such areas exactly requires the evaluation of the sign of a degree-14 polynomial in the input variables.

There is another case, when the smallest circumscribed triangle has just two "flush sides". The definition of such a triangle involves only 5 input points, and it can be worked out by hand. If the third side goes through the point (x_3, y_3) and the other two sides are specified as before, numerator of the area has degree 4 and the denominator has degree 2:

$$\frac{\pm 2 \cdot \begin{vmatrix} x_2 - x_3 & y_2 - y_3 \\ x_2 - u_2 & y_2 - v_2 \end{vmatrix} \cdot \begin{vmatrix} x_1 - x_3 & y_1 - y_3 \\ x_1 - u_1 & y_1 - v_1 \end{vmatrix}}{\begin{vmatrix} x_1 - u_1 & y_1 - v_1 \\ x_2 - u_2 & y_2 - v_2 \end{vmatrix}}$$

C Constructing the largest anchored inscribed triangle from the smallest anchored circumscribed triangle

Here is the converse statement to Lemma 4.i.

Lemma 14. *Let $\hat{A}\hat{B}\hat{C}$ be a smallest circumscribed triangle anchored at $-\mathbf{u}$, of height $\hat{h}$. Then the largest inscribed triangle $A^*B^*C^*$ anchored at $\mathbf{u}$ has vertices $B^* = (\hat{A}+\hat{C})/2$ and $C^* = (\hat{A}+\hat{B})/2$, and the vertex A^* lies on the side $\hat{B}\hat{C}$ (see Figure 5b). Hence it has height $h = \hat{h}/2$, and the length of its baseline is $B^*C^* = \hat{B}\hat{C}/2$, and its area is $1/4$ of the area of $\hat{A}\hat{B}\hat{C}$.*[28]

The proof hinges on the well-known optimality condition for circumscribed triangles:

Lemma 15. *Let $\hat{O}\hat{X}\hat{Y}$ be a smallest triangle containing a convex polygon P under the constraint that $\hat{O}$ is fixed and $\hat{X}$ and $\hat{Y}$ lie on two given rays emanating from $\hat{O}$. Then the midpoint $(\hat{X}+\hat{Y})/2$ touches P.*[29]

Proof. Clearly, the side $\hat{X}\hat{Y}$ must touch P. If it does not touch P at the midpoint $(\hat{X}+\hat{Y})/2$, then the area can be decreased by tilting the side $\hat{X}\hat{Y}$ around the vertex where it touches P. This has been implicitly shown in the proof of Lemma 6, see Figure 9b with TC^*B^* in the role of $\hat{O}\hat{X}\hat{Y}$. If the side $\hat{X}\hat{Y}$ touches an edge of P, we tilt it around the endpoint closer to the midpoint. □

Proof of Lemma 14. It is obvious that A^* lies on the side $\hat{B}\hat{C}$. By Lemma 15, applied to $\hat{O}\hat{X}\hat{Y} = \hat{B}\hat{C}\hat{A}$ and $\hat{O}\hat{X}\hat{Y} = \hat{C}\hat{A}\hat{B}$, the midpoints $B^* = (\hat{A}+\hat{C})/2$ and $C^* = (\hat{A}+\hat{B})/2$ of the two "legs" $\hat{A}\hat{C}$ and $\hat{A}\hat{B}$ lie in P.

Optimality of $A^*B^*C^*$ within P follows easily by Lemma 2: An anchored triangle larger than $A^*B^*C^*$ cannot even be found in the circumscribed triangle $\hat{A}\hat{B}\hat{C} \supseteq P$. □

D An alternative proof that B^* and C^* move monotonically

We have proved the monotone movement of the points $B^*(\theta)$ and $C^*(\theta)$ as a consequence of the analysis of the possible local movements at each direction in Theorem 5. We will give an independent self-contained proof.[30]

Lemma 16. *As θ increases, each of the points $B^*(\theta)$ and $C^*(\theta)$ moves only in the forward direction (or stays where it is).*

Proof. It is enough to prove monotonicity for some range of directions θ where A^* is constant.

It is impossible that none of B^* and C^* moves forward, because then the segment B^*C^* would stay the same or turn clockwise while its supposed normal direction $\mathbf{u}(\theta)$ turns counterclockwise.

Thus, we are left to exclude the case that one of the points B^* and C^* moves backward and the other moves forward. If this happens, then there are two values $\theta_1 \neq \theta_2$ such that the four points $B_1 = B^*(\theta_1), C_1 = C^*(\theta_1), B_2 = B^*(\theta_2), C_2 = C^*(\theta_2)$ are distinct and occur in the clockwise order $B_1B_2C_2C_1$ on the boundary, see Figure 17.

Let us look at the edges $e^{\text{forw}}(B_1)$ and $e^{\text{back}}(C_1)$. By the optimality criterion, their upward extensions intersect in some point T_1^{up}, and the critical pivot point $M_1^{\text{up}} = (T_1^{\text{up}} + A^*)/2$ lies on or below the line B_1C_1. The edges $e^{\text{back}}(B_2)$ and $e^{\text{forw}}(C_2)$ lie between $e^{\text{forw}}(B_1)$ and $e^{\text{back}}(C_1)$ in the cyclic order, with equality permitted. Hence, their intersection point T_2^{down} lies in the triangle $B_1C_1T_1^{\text{up}}$. This restricts the critical pivot point $M_2^{\text{down}} = (T_2^{\text{down}} + A^*)/2$ of $A^*B_2C_2$ to a smaller triangle Δ that is dilated from the center A^* with a factor $\frac{1}{2}$. The triangle Δ has its top vertex at M_1^{up}, and its lower edge is parallel to B_1C_1. It follows that M_2^{down} lies on or below B_1C_1, and therefore strictly below B_2C_2, and hence B_2C_2 is not optimal. □

---

[28] Chandran and Mount [ChMo, Lemma 2.4] proved that there is always an "inner triangle" $A^*B^*C^*$ that satisfies all the geometric relations stated in Lemma 14, without noting (or caring to state) that $A^*B^*C^*$ is the *largest* anchored inscribed triangle. In a separate lemma [ChMo, Lemma 2.5(ii)], they proved only that the *overall* largest inscribed triangle arises as the inner triangle of some (special) smallest anchored circumscribed triangle. In this case, the inner triangle is even a homothetic copy of the circumscribed triangle, scaled with the factor $-\frac{1}{2}$.

[29] This condition is also sufficient for optimality in the setting of this lemma, see [KlLa, Lemma 1.2].

[30] See also the "interspersing property" in [OAMB, Lemma 2]. The "interleaving property" in [K$^+$, Lemma 5] is similar, but it holds for a different class of triangles, the so-called "3-stable" triangles.

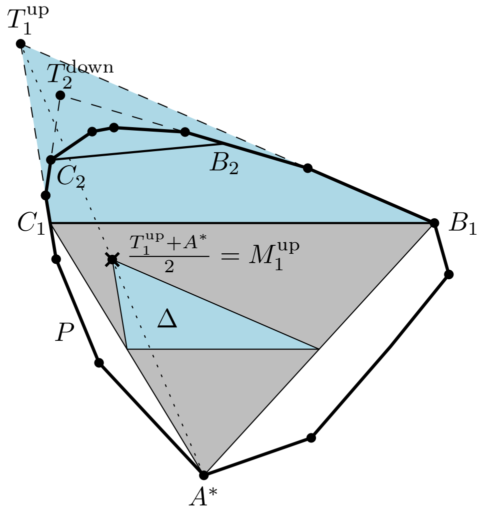


Figure 17: Proof of Lemma 16. In this example, $\theta_1 < \theta_2$. The proof works equally when the opposite relation holds.

As a consequence of this lemma, one can conclude that the motion of $B^*(\theta)$ and $C^*(\theta)$ is continuous, because a discontinuity would be inconsistent with monotonicity, given that the direction changes continuously. The case when B^*C^* is the $\mathbf{u}$-extreme edge of P must be considered separately for this argument.

Continuity can also be established directly from basic properties of the underlying optimization problem [Kal, Lemma 3.2].

We have used continuity as part of Theorem 5 only to establish monotonicity, but otherwise, the algorithm does not depend on continuity. However, if continuity can be assumed, this would simplify some arguments in the proof of Theorem 5.