Finding the Maximum Area Parallelogram in a Convex Polygon

Kai Jin* Kevin Matulef*

Abstract

We consider the problem of finding the maximum area parallelogram (MAP) inside a given convex polygon. Our main result is an algorithm for computing the MAP in an n-sided polygon in $O(n^2)$ time. Achieving this running time requires proving several new structural properties of the MAP, and combining them with a rotating technique of Toussaint [10].

We also discuss applications of our result to the problem of computing the maximum area centrally-symmetric convex body (MAC) inside a given convex polygon, and to a "fault tolerant area maximization" problem which we define.

1 Introduction

A common problem in computational geometry is that of finding the largest figure of one type contained in a given figure of another type. Over the last 30 years researchers have looked at several instances of this problem, such as finding the largest convex polygon contained in an arbitrary polygon [2], the largest axis-parallel rectangle in an arbitrary polygon [3] , the largest triangle inscribed in a convex polygon [4], the largest k-gon in a convex polygon [1], or the largest square in a convex polygon [7].

In this work, we consider the problem of finding the maximum area *parallelogram* (MAP) inside a convex polygon. Our main result is the following:

Theorem 1 *There is an algorithm for computing the MAP in a convex polygon with n sides in $O(n^2)$ time.*

As we shall see, achieving an $O(n^2)$ running time is not straightforward; it requires proving several structural properties of the MAP. We discuss the challenges involved, and our techniques for overcoming them, in Section 1.2.

1.1 Applications

The MAC. One reason why the parallelogram case is of special interest is because parallelograms are the simplest polygons that are "centrally-symmetric" (i.e. for which there exists a "center" such that every point on the figure, when reflected about the center, produces another point on the figure). It is natural to ask whether we can, in general, compute the Maximum Area Centrally-symmetric convex body (MAC) inside a given convex polygon or convex curve. Although it seems difficult to compute the area of the MAC exactly, it is known that the MAP serves as an approximation:[1]

Theorem 2 *[5, 8] For a convex curve Q, the area of the MAP inside it is always at least $\frac{2}{\pi} \approx 0.6366$ times the area of Q, Moreover, this bound is tight; the worst case is realized when the given convex curve is an ellipse.*

Theorem 2 follows from two results.[2] The first result of Dowker [5] says that for any centrally-symmetric convex body K in the plane, and any even $n \geq 4$, among the inscribed (or contained) convex n-gons of maximal area in K, there is one which is centrally-symmetric. The second result of Sas [8] says that for convex bodies in $\mathbb{R}^d$, the hardest to approximate with inscribed n-gons are exactly the ellipsoids.

By combining Theorem 2 with our Theorem 1, we get the following corollary.

Corollary 3 *There is a $\frac{2}{\pi}$-approximation algorithm for computing the area of the MAC in $O(n^2)$ time.*

Fault Tolerant Area Maximization. Consider the following general problem: you are allowed to place k points inside a polygon P, then an adversary removes j of them (where $j < k$). Your goal is to maximize the area of the convex hull of the remaining points. We call this the *Fault Tolerant Area (FTA) Maximization Problem.*

Let $FTA(k, j)$ be the maximum area you can achieve in the worst case. It is easy to see that $FTA(k, 0)$ is equivalent to finding the maximum area k-gon inside P. Boyce et. al. give a clever algorithm for solving this in $O(knlgn + nlg^2n)$ time [1]. However, when $j > 0$, the problem seems much less trivial. Perhaps the simplest

*IIIS, Tsinghua University, cscjjk@msn.com and matulef@gmail.com. Supported in part by the National Basic Research Program of China Grant 2007CB807900, 2007CB807901, and the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174.

[1] We may give the *simplest* credit to squares in some sense, but with only one constrain of been centrally-symmetric, parallelograms are *simpler* (more flexible) than squares in the less-constrains (flexible) sense. As a result, parallelograms are more suitable for approximating the MAC than squares.

[2] An earlier version of this paper contained an alternative proof of Theorem 2, see http://itcs.tsinghua.edu.cn/zh/kaijin/

non-trivial case is $FTA(4,1)$. In this case, we show that computing $FTA(4,1)$ reduces to the problem of computing both the maximum area *triangle* (which can be done using Boyce et. al.'s algorithm) and the MAP. Thus, we get the following corollary to our main theorem (due to space limitations, we present the proof of this corollary in the Appendix).

Corollary 4 (Reduction) *Computing $FTA(4,1)$ in a convex polygon P can be done in $O(n^2)$ time.*

1.2 Techniques

We start by proving the relatively simple fact that the MAP inside a convex polygon P must have all of its corners on the perimeter of P. This suggests the possibility of an algorithm that works by enumerating all 4-tuples of edges of P, and for each 4-tuple finding the largest parallelogram with one corner on each edge. Such an algorithm would, at best, run in $O(n^4)$ time.

To reduce the search space, we further prove that the MAP must be *anchored* on P. In other words, it must have at least one corner on a vertex of P. We prove this via a lemma we call the "hyperbola lemma" which may be of independent interest (see Section 2.2). We then divide the computation of the MAP into two cases: one where the MAP has two opposite, non-anchored corners, and one where it has two adjacent, anchored corners.

For the first case, we prove that for every pair of edges of P, finding the MAP with opposite non-anchored corners on those edges involves checking only $O(n)$ possibilities for the placement of the other corners. As there are $O(n^2)$ pairs of edges, in total this yields an $O(n^3)$ algorithm. In order to speed it up further, we employ a rotating technique of Toussaint [Tou83]. The main idea is to show that if the pairs of edges are processed in the right order, the amortized cost of computing the best placement for the other corners is only $O(1)$. Proving this requires proving additional structural properties of the MAP (see Section 3.1).

For the second case, when the MAP has two adjacent corners anchored on P, the algorithm is slightly more complicated, but uses similar ideas and still has running time $O(n^2)$ (see Section 3.2).

1.3 Related Work

In [2], Chang and Yap gave an algorithm for the "potato peeling" problem, or the problem of finding the largest convex polygon Q inside a given simple polygon P with n sides. They showed that this problem is computable in polynomial time, by giving algorithms computing the maximum area Q in $O(n^7)$, and the maximum perimeter Q in $O(n^6)$. Their investigation led them to define the general notion of "inclusion" problems for arbitrary classes of polygons $\mathcal{P}$ and $\mathcal{Q}$. The goal of the inclusion problem on $\mathcal{P}$ and $\mathcal{Q}$ is to find the largest polygon from $\mathcal{Q}$ inside a given polygon from $\mathcal{P}$ (here "largest" can be with respect to area, perimeter, or other measures). Chang and Yap surveyed several results on the inclusion problem for specific $\mathcal{P}$ and $\mathcal{Q}$, although to date no unified solution exists. For different $\mathcal{P}$ and $\mathcal{Q}$, it seems different techniques must be employed. The problem we solve in this work is the specific case where $\mathcal{P}$ is the set of convex polygons, and $\mathcal{Q}$ is the set of parallelograms.

For the case where $\mathcal{P}$ is the set of convex polygons, the inclusion problem has been studied for several different $\mathcal{Q}$. For example, given a convex polygon P with n vertices, Shamos [9] gave an algorithm for finding the diameter of P in linear time (this corresponds to $\mathcal{Q}$ being the set of "one-edge" polygons). Dobkin and Snyder [4] gave a linear time algorithm for finding the maximum area triangle; Boyce, Dobkin, Drysdale and Guibas [1] gave an algorithm for finding maximum area/perimeter k-gons in time $O(knlgn + nlg^2n)$. De Pano Ke and O'Rourke [7] gave an algorithm for finding the largest inscribed square in time $O(n^2)$; Fekete [6] gave an algorithm for finding all anchored squaresin $O(n \log^2 n)$ time; For the case where $\mathcal{P}$ is the set of all simple polygons, Daniels, Milenkovic and Roth [3] gave an algorithm for finding the maximum area axis-parallel rectangle in time $O(n \log^2 n)$.

2 Preliminaries

2.1 Basic notations and lemmas

We will use symbols A, B, A', B' to denote the four corners of a parallelogram $Q = ABA'B'$ (the pairs A, A' and B, B' denote opposite corners). We will use the symbol E to denote the center of Q.

Definition 5 (Inscribed) *We say a parallelogram Q is **inscribed** on a polygon P if and only if all four corners of Q are on the boundary of P.*

Definition 6 (Anchored) *We say a parallelogram Q is **anchored** on a polygon P if it is inscribed on P and at least one of its corners lies on a vertex of P.*

Definition 7 (Narrow side & Broad side) *Suppose b, b' are two nonparallel edges of P. They divide the other edges of P into two sets, the edges in the **Narrow side** (where the extended lines of b and b' intersect) and the edges in the **Broad side** (where b and b' are further apart), illustrated in Figure 1.*

Lemma 8 *The parallelogram inside P with the maximum area must be inscribed on P.*

Proof. We prove this by contradiction. Suppose $Q = ABA'B'$ is a parallelogram which has maximal area in

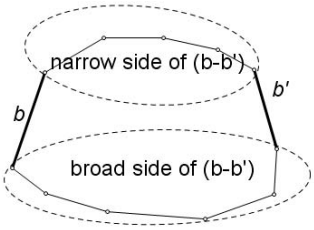

Figure 1: Broad side and Narrow side

P but is not inscribed in P (See Figure 2). Without loss of generality, assume A is a vertex which is not on the boundary of P. First, we slide segment AB along direction $\vec{BA}$ for a sufficiently small distance to create A_1B_1. Next we slide it along direction $\vec{B'A_1}$ for a sufficiently small distance to create A_2B_2 where A_2 and B_2 are still inside P. It's easy to see that $Area(ABA'B') < Area(A_2B_2A'B')$. □

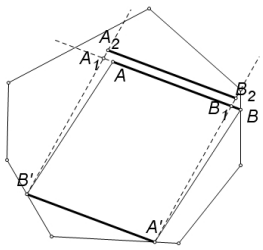

Figure 2: Illustration of Theorem 8

Lemma 8 says if a parallelogram $Q = ABA'B'$ is the MAP of a polygon P, then all its corner must lie on the boundary of P. For points A, B, A', B' that do not lie on vertices of P, we will use the lowercase letters a, b, a', b' respectively to denote the edges of P they lie on.

Lemma 9 (Two Lines Lemma) *Given two nonparallel lines b, b' and one point E not on them, there is exactly one segment connecting b and b' with midpoint E. Moreover, the endpoints of this segment, denoted as B and B', can be computed in constant time.*

This lemma is simple; we omit its proof. Next we introduce the segment version of Lemma 9, it will be used many times in our algorithm.

Suppose there are two segments $b = B_1B_2$ and $b' = B_3B_4$ which, when extended, intersect at point O. For $1 \le i \le 4$, let M_i be the midpoint of OB_i (see Figure 3). We draw a parallelogram $P(b, b')$ such that one pair of sides is parallel to b and crossing M_3 and M_4, and the other pair of sides is parallel to b' and crossing M_1 and M_2.

Lemma 10 (Two Segments Lemma) *There is a segment connecting b and b' with midpoint E, if and only if E is inside of $P(b, b')$.*

The proof of Lemma 10 is also simple; due to space constraints we omit it.

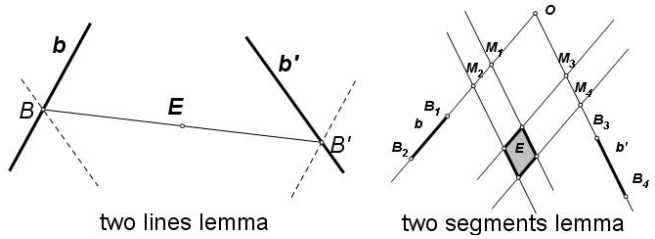

Figure 3: Two lines lemma and two segments lemma

Note that for a parallelogram Q with center E and opposite corners B and B' located on b and b' respectively, we know $E \in P(b, b')$ because E is the midpoint of the diagonal BB'.

2.2 The Hyperbola Lemma

Definition 11 *For two nonparallel lines b_1, b_2 intersecting at O, and a point A strictly in-between them, there is a unique hyperbola asymptotic to b_1 and b_2 and intersecting A, denoted as $h_A^{b_1,b_2}$ (or h_A for short). Let $C_A^{b_1,b_2}$ (or C_A for short) denote the distance from O to the nearest point on h_A.*

Lemma 12 (Hyperbola Lemma) *Suppose b, b' are two nonparallel lines which intersect at origin O. Let h_1 and h_2 be two hyperbolas which are both asymptotic to b and b'. Then all parallelograms $Q = ABA'B'$, where A, B, A', B' lie on h_1, b, h_2, b' respectively, have the same area.*

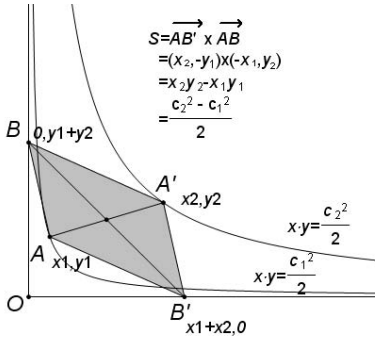

Figure 4: Hyperbola Lemma (orthogonal case)

Proof. We will prove the lemma in the case when b and b' are orthogonal. The general case follows from a linear transformation.

Build a Cartesian coordinate system with origin O and let b', b be the x-axis and y-axis (see Figure 4). Suppose the coordinates of A and A' are (x_1, y_1) and (x_2, y_2) respectively. The center E is the midpoint of AA', and thus has coordinates $(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$. By Lemma 9, the coordinates of B and B' are uniquely determined, and are easily verified to have coordinates $(0, y_1 + y_2)$ and $(x_1 + x_2, 0)$. Thus, we can compute $area(Q) = x_2y_2 - x_1y_1$. Since h_1 and h_2 are hyperbolas asymptotic to b and b', this means $x_1y_1 = C_A^2/2$ and $x_2y_2 = C_{A'}^2/2$. Hence, $area(Q) = C_{A'}^2/2 - C_A^2/2$, which is invariant. □

We will apply the hyperbola lemma with b and b' equal to *extensions* of edges of the original polygon P. Note that to find the maximum area parallelogram in P with one vertex on b and another on b', we should choose A and A' so as to maximize $C_{A'}$ and minimize C_A. However, this is trickier than it seems, since if we are allowed to choose A and A' arbitrarily, the resulting B and B' may not actually lie on the original edges of the polygon (which are just *segments*, not lines). We discuss this complication further in Section 3.1.

Definition 13 *Let b, b' be two nonparallel edges, and let c be an edge in the broad side. Then we use X_c to denote the intersection point of b and the extended line of c, Y_c to denote the intersection point of b' and the extended line of c, and Z_c to denote the midpoint of X_cY_c.*

Lemma 14 *Suppose D is a point on segment X_cY_c. Then C_D increases while D goes from X_c to Z_c, and while D goes from Y_c to Z_c.*

Proof. We only need to prove it in the case when b and b' are orthogonal, for the same reason used in Lemma 12. Without loss of generality, assume b, b' are on the x, y-axis respectively, $X_c = (x_0, 0), Y_c = (0, y_0)$. Assume $D = (x, y_0 - x(y_0/x_0))$. It's not hard to show that $C_D = \sqrt{2 * x * [y_0 - x(y_0/x_0)]}$. Note $x * [y_0 - x(y_0/x_0)]$ is a quadratic equation maximized when $x = \frac{x_0}{2}$. □

2.3 The Anchor Theorem

Theorem 15 (Anchor Theorem) *The MAP in P must be anchored on P.*

Proof. Suppose $Q = ABA'B'$ is a parallelogram inscribed but not anchored on P. We will show that Q is not the MAP in P. First, assume neither pair a, a' nor b, b' is parallel to each other, otherwise the theorem is trivial to prove. Assume a is in the narrow side. We can construct a new parallelogram as follows. Since A is not on an endpoint of a, we can move A a little bit along a so that C_A decreases (see Lemma 14). We keep the position of A' so that $C_{A'}$ doesn't change. Afterward we replace the new center E by the midpoint of segment AA'. Then according to Lemma 9, B and B' can be computed since b', b', and E are all fixed. We can make sure that B, B' will still be inside segments b, b' respectively by only moving A for a sufficiently small distance. We know that the area of this new parallelogram is larger than the area of Q according to Lemma 12. Hence Q is not the MAP in P. □

Theorem 15 leads one to wonder whether the MAP must always be *double-anchored* on P (that is, whether the MAP must have *two* of its corners on vertices of P). Unfortunately, this is not the case. Figure 9 in the Appendix illustrates an example where the double-anchored MAP is smaller than the actual MAP.

To design our algorithm for finding the MAP, we divide anchored parallelograms into two cases, described by the following definitions:

Definition 16 *We say that a parallelogram Q is **adjacent-double-anchored** on a polygon P if it is inscribed on P and two adjacent corners lie on the vertices of P.*

Definition 17 *We say that a parallelogram Q is **opposite-free-anchored** on a polygon P if it is anchored on P but has two opposite corners which are not anchored.*

Note that for a parallelogram Q anchored on P, it must either be adjacent-double-anchored, or opposite-free-anchored; it cannot be both.

3 The Algorithm

In this section we describe our algorithm for finding the MAP in a convex polygon. Our general algorithm will actually consist of two algorithms, one to handle the case when the MAP is opposite-free-anchored, and the other to handle the case when the MAP is adjacent-double-anchored. Both algorithms use similar ideas, and have running time $O(n^2)$.

3.1 The Opposite-Free-Anchored Case

First we give an algorithm for finding the MAP when the MAP is opposite-free-anchored. Without loss of generality, assume that B, B' are not anchored, and are inscribed on b, b' respectively. Let A be the vertex in the narrow side and A' in the broad side. Let M_{b_1,b_2} (M for short) be the point in P such that $C_M^{b_1,b_2} = max\{C_V^{b_1,b_2} | V \in P\}$. For fixed b and b', the following corollary of Lemma 14 helps us compute the optimal placement of M.

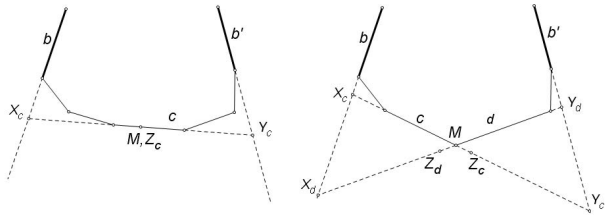

Figure 5: Either $M = Z_c$ (pictured at left), or M is on a vertex of P (pictured at right)

Corollary 18 *There is at most one edge c in the broad side such that Z_c actually lies on c. When there is such an edge, then $M = Z_c$. When there is not such an edge, then M is on a vertex formed by two edges denoted c and d. The point Z_c lies to the right of M, and the point Z_d lies to the left of M (see Figure 5).*

We are now ready to prove one of the main theorems behind our algorithm in the opposite-free-anchored case. Suppose B and B' are non-anchored vertices on fixed edges b and b'. The following theorem reduces the computation of the optimal placement of A and A' to a finite set of possibilities:

Theorem 19 *Suppose Q is the opposite-free-anchored MAP on P, where B and B' are located on (non-endpoints) of b and b', A is in the narrow side and A' is in the broad side. Then $A' = M$, and A is anchored on P.*

Proof. From Corollary 18, we know that if a dynamic point D goes from one end in the broad side to another, C_D will increase before D reaches M, and decrease after D reaches M. So if $A' \neq M$, there exists an $A^* \in P$ near A' such that $C_{A^*} > C_{A'}$. Then, as in the proof of Theorem 15, we should be able to slightly adjust B and B' (still on b and b') to construct a new parallelogram with vertices A^*, A, and two vertices on b, b', which has area bigger than that of Q. This is a contradiction.

Similarly, suppose A is not anchored on P. There exists a point A^* near A such that $C_{A^*} < C_A$. Again, this means we should be able to slightly adjust A, and then B, B', to construct a new parallelogram with vertices A^*, A' and two vertices on b, b' with area bigger than that of Q. This is also a contradiction. □

Theorem 19 suggests a simple enumerative algorithm in the opposite-free-anchored case: for each pair of edges (b, b'), compute the optimal A' and A by cycling through all possibilities. This is described in Algorithm 1.

If implemented naively, the time complexity of Algorithm 1 is $O(n^3)$. In order to speed up it further, we employ a rotating technique of Toussaint [10]. The main idea is to show that it isn't necessary to spend $O(n)$ time calculating A and A' for every pair of edges (b, b'). In fact, it can be done in *amortized* $O(1)$ time.

```
1  foreach edge b ∈ P,b' ∈ P do
2      foreach vertex V ∈ P, V not on b or b' do
3          A' ← V if C_V > C_A'.
4      end
5      foreach edge c ∈ P, c ≠ b, c ≠ b' do
6          A' ← Z_c if C_Zc > C_A' and Z_c ∈ P.
7      end
8      foreach vertex A ∈ P, A not on b or b' do
9          E ← the midpoint of AA'.
10         Compute B, B' by Lemma 9.
11         Q ← ABA'B' if Area(ABA'B') > Area(Q)
           and ABA'B' is inside P.
12     end
13 end
```

Algorithm 1: opposite-free-anchored

Suppose we fix an edge b, and consider the sequence of pairs $(b, b'_1), (b, b'_2), (b, b'_3), \ldots$, where the edge sequence $b'_1, b'_2, b'_3, \ldots$ is formed by walking counter-clockwise along the boundary of P. Let A_i and A'_i be the optimal values computed for the pair (b, b'_i). Then it is possible to show that the sequences $A_1, A_2, A_3, \ldots$, and $A'_1, A'_2, A'_3, \ldots$, also move counter-clockwise along the boundary of P. Thus, for a fixed b, we only spend $O(n)$ time calculating all values of A_i and A'_i. Repeating with all other edges in place of b yields an $O(n^2)$ time algorithm.

To prove this, there are two stages in Algorithm 1 that need to be analyzed carefully. For every pair (b, b'), the first stage (lines 2-7) takes $O(n)$ time to find A', the second stage (lines 8-12) also takes $O(n)$ time to enumerate all the vertices in the narrow side to find A.

To show that the first stage can be made to have small amortized cost, we cite the following lemma:

Lemma 20 (A monotone property of A') *Suppose b is fixed, and b' moves counter-clockwise along P. Then the distances between A' and b are non-decreasing. In other words, A' can also only move counter-clockwise around P (see Figure 7 for an example).*

The proof of Lemma 20 is simple. We omit it due to space limitations.

For the second stage, let $P_{A'}(b, b')$ be a 2-scaling of $P(b, b')$ around point A'. We claim that $A \in P_{A'}(b, b')$, because $E \in P(b, b')$ and E is the midpoint of AA' (see Figure 6).

Lemma 21 *The parallelograms $P_{A'_1}(b, b'_1)$, $P_{A'_2}(b, b'_2)$, $P_{A'_3}(b, b'_3), \ldots$ are all non-overlapping. Additionally, their distance to line b is decreasing (see Figure 7 for an example).*

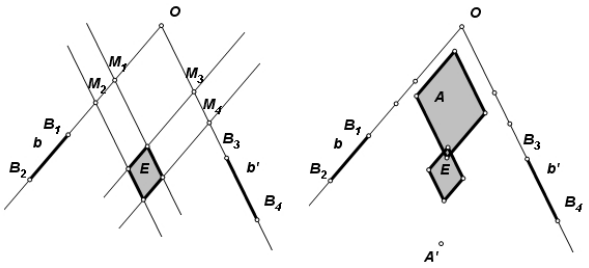

Figure 6: $P_{A'}(b, b')$ is the region where A might lie.

Proof. First, $P_{A'_i}(b, b'_i)$ are parallelograms, two sides of which are parallel to b. While b' is shifting, the distances between $P(b, b'_i)$ and line b is decreasing, and the distances between A'_i and line b is non-decreasing, so the distances between $P_{A'_i}(b, b'_i)$ and b is decreasing. Thus, they do not overlap with each other. □

In line 8 of Algorithm 1, if we replace "$A \in P$" with "$A \in P_{A'}(b, b')$", then for a fixed b each vertex A will be enumerated at most once. Hence the this stage can be reduced to $O(1)$ time on average, and therefore Algorithm 1 can be implemented in $O(n^2)$ time total.

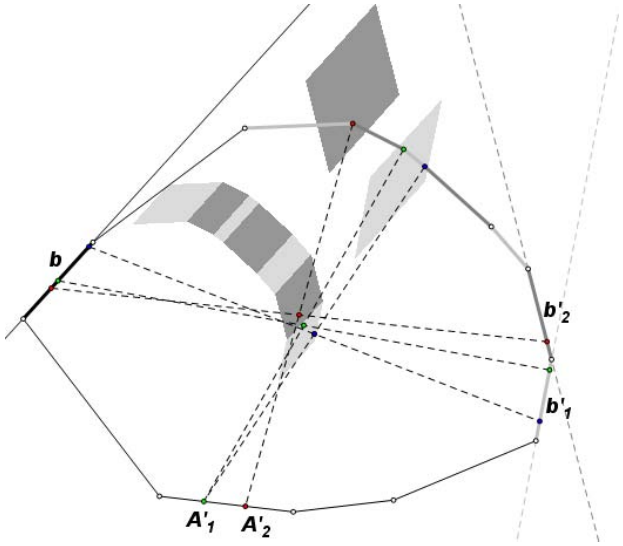

Figure 7: Illustration of Algorithm 1

3.2 The Adjacent-Double-Anchored Case

Next we give an algorithm for finding the MAP when the MAP is double-adjacent-anchored.

While it is tempting to just run Algorithm 1 and hope that it works in this case too, unfortunately it does not. The reason is that Theorem 19 crucially assumes that B and B' are flexible on b and b', in order to reduce the space of possible values for A and A'. Without the guarantee that B and B' are not anchored, it is possible that the best choice of A' is not equal to M, or that the best choice of A is not on a vertex of P.

Nevertheless, when we assume the MAP has two adjacent anchored vertices, we can still prove some constraints on the placement of the other vertices. This allows us to develop an algorithm similar to Algorithm 1 that enumerates over all choices of anchored vertex B and opposite edge b'. If implemented correctly, this algorithm can be made to run in $O(n^2)$ by showing an amortized analysis similar to the one we used before.

Due to space limitations, we present the full details of our algorithm in the adjacent-double-anchored case in the Appendix.

Acknowledgments

The authors are grateful to the anonymous reviewers of an earlier version of this paper for pointing out the references [5, 8] and for other helpful comments.

The authors would also like to thank Xiaoming Sun, Tiancheng Lou, and Zhiyi Huang for taking part in fruitful discussions.

References

[1] J. E. Boyce, D. P. Dobkin, R. L. Drysdale, III, and L. J. Guibas. Finding extremal polygons. In *Proc. of the 14th annu. ACM symp. on Theory of comp.*, STOC '82, pages 282–289, New York, NY, USA, 1982. ACM.

[2] J. Chang and C. Yap. A polynomial solution for the potato-peeling problem. *Discrete and Computational Geometry*, 1:155–182, 1986. 10.1007/BF02187692.

[3] M. Daniels and Roth. Finding the largest area axis-parallel rectangle in a polygon. *CGTA: Computational Geometry: Theory and Applications*, 7, 1997.

[4] D. Dobkin and L. Snyder. On a general method for maximizing and minimizing among certain geometric problems. In *Proceedings of the 20th Annual Symposium on FOCS*, pages 9–17. IEEE Computer Society, 1979.

[5] C. Dowker. On minimum circumscribed polygons. *Bull. Amer. Math. Soc*, 50:120–122, 1944.

[6] S. P. Fekete. Finding all anchored squares in a convex polygon in subquadratic time. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 71–76, 1992.

[7] J. O. N. Adlai De Pano, Yan Ke. Finding largest inscribed equilateral triangles and squares. In *Proc. Allerton Conf.*, pages 869–878, 1987.

[8] E. Sas. über ein extremumeigenschaft der ellipsen. *Compositio Math*, 6:468–470, 1939.

[9] M. Shamos. Computational geometry. 1978.

[10] G. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE Melecon*, volume 83, pages 1–4. Citeseer, 1983.