Cutting Out Polygons

Ramaswamy Chandrasekaran * Ovidiu Daescu*† Jun Luo*

Abstract

In this paper, we present approximation algorithms for the problem of cutting out a convex polygon P with n vertices from another convex polygon Q with m vertices by a sequence of guillotine cuts of smallest total length. Specifically, we give an $O(n^3 + m)$ running time, constant factor approximation algorithm, and an $O(n+m)$ running time, $O(\log n)$-factor approximation algorithm for cutting P out of Q. We also discuss some negative results for the case when guillotine cuts are replaced by ray cuts.

1 Introduction

Two decades ago Overmars and Welzl [5] studied the problem of cutting out a polygon P from another polygon Q in the cheapest possible way. The problem falls in the general area of stock cutting, where a given shape needs to be cut out from a parent piece of material, and it is defined as follows:

> *Given a polygonal piece of material Q with a polygon P drawn on it, cut P out of Q by a sequence of "guillotine cuts" in the cheapest possible way.*

A *guillotine cut* (also called *line cut*) is a line cut that does not cut through the interior of P and separates Q into a number of pieces, lying on both sides of the cut. A guillotine cut is an *edge cut* if it cuts along an edge of P. After a cut is made, Q is updated to that piece that still contains P. A *cutting sequence* is a sequence of cuts such that after the last cut in the sequence we have $P = Q$ (see Fig. 1). The cost of a cut is the length of the intersection of the cut with Q and the goal is to find a cutting sequence that minimizes the total cost. Clearly, the polygon P must be convex for a cutting sequence to exist.

Overmars and Welzl [5] proved a number of properties for the case when both P and Q are convex polygons, including the existence of a finite optimal cutting sequence with $O(n)$ cuts, all touching P. They further noted that when Q is not convex there are cases in which there is no optimal cutting sequence with all cuts touching P. When only edge cuts are allowed Overmars and Welzl proved an optimal cutting sequence can be computed in $O(m + n^3)$ time by a simple dynamic programming algorithm.

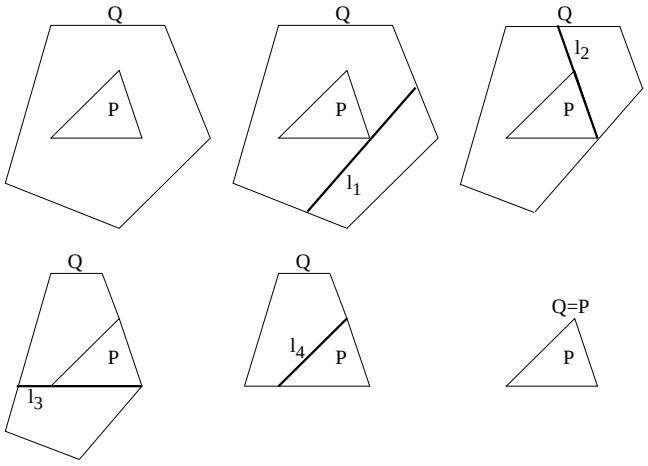

Figure 1: A cutting sequence (bold lines) $\{l_1, l_2, l_3, l_4\}$ for cutting P out of Q.

In [1], Bhadury and Chandrasekaran showed that the problem of cutting P out of Q has optimal solutions that lie in the algebraic extension of the field that the input data belongs to. They also provided an approximation scheme that, given an error range δ, is polynomial in δ and the encoding length of the input data in unary, and gives a cutting sequence $C(\delta)$ with total cost at most δ more than that of an optimal cutting sequence C^*.

In [4], Dumitrescu proved that there exists an $O(\log n)$-factor approximation algorithm, which runs in $O(mn + n \log n)$ time, for cutting out a convex polygon P from a convex polygon Q. He also raised the following questions: (1) If Q is the minimum axis-aligned rectangle enclosing P, is an optimal edge-cutting sequence a constant factor approximation of an optimal cutting sequence? (Answering this question in an affirmative sense results in a constant-factor approximation algorithm for cutting P out of Q.) (2) Is it possible to extend the results for guillotine cutting to ray cutting?

Daescu and Luo [2], affirmatively answered the first question and presented an $O(n^3 + (n+m)\log(n+m))$, constant factor approximation algorithm, for cutting P out of Q when both P and Q are convex polygons. They also presented an $O((n+m)\log(n+m))$, $O(\log n)$-factor

*Department of Computer Science, University of Texas at Dallas, {chandra,daescu,ljroger}@utdallas.edu

†Daescu's research is supported by NSF grant CCF-0430366.

approximation algorithm for the same problem.

The related problem in which guillotine cuts are replaced by *ray cuts*, where a ray cut runs from infinity to some point in Q, has been less studied. In this case, some non-convex polygons are *ray-cuttable*. These polygons have the property that every edge of the polygon must be extendible to a ray. As observed in [3] by Demaine *et al.*, this condition is more restrictive than *(weakly) external visibility*, which requires that every point on the boundary of P can "see" to infinity. A linear time algorithm to test the ray cuttability of a polygon P is given in [3]. In [2], they show how to extend the line cutting solution to construct an approximate ray cutting sequence, thus answering the second question from [4]. Specifically, they present an $O(\log^2 n)$-factor approximation algorithm for cutting P out of a convex polygon Q, when P is ray cuttable, with running time $O(m + n \log n)$. The key idea is a recursive procedure for cutting out a ray cuttable polygon P from its convex hull in $O(\log n)$ steps, in each step constructing a ray cutting sequence based on some line cutting sequence.

Our results. To cut a convex polygon P with n vertices from a convex polygon Q with m vertices the algorithms in [2, 4] involve two phases: a separation phase, that cuts out a triangle enclosing P and of size about the size of P, and a carving phase, in which P is cut out of that triangle. The carving phase can be carried out in $O(n^3)$ time or in $O(n)$ time, depending whether an $O(1)$-factor or an $O(\log n)$-factor approximation is sought, respectively. The separation phase consists of computing two cuts of small total length, after which a third small length cut to form the triangle can by made in constant time.

In [2] they showed how to find the first two cuts in $O((n+m)\log(n+m)))$ time. Here, we give an $O(n+m)$ time algorithm to find these two cuts. In turn, this leads to $O(m + n^3)$ time or $O(m + n)$ time algorithms to cut out a convex polygon P from a convex polygon Q, depending whether an $O(1)$-factor or an $O(\log n)$-factor approximation is sought, respectively. Thus, our algorithms improve over those in [2] by a logarithmic factor in time.

We next present some negative results with respect to improving the approximation bound of the ray cutting algorithm in [2]. That algorithm involves three phases: (1) cutting out a triangle containing P of about the same size as P, (2) cutting out the convex hull $CH(P)$ of P from that triangle and (3) cutting out the *pockets* of P from its convex hull. The key phase is the third phase. In this phase, a pocket of P is cut out from $CH(P)$ in $O(\log n)$ recursive steps. Each step relies on constructing a ray cutting sequence from a line cutting sequence; the line cutting sequence is found by the same algorithm for the carving phase of the line cutting problem, that cuts out P from the smallest axis aligned rectangle enclosing P.

Consider the stronger assumption that we have an algorithm to efficiently compute an optimal (and for that matter an approximate) ray cutting sequence for cutting out P from the smallest axis aligned rectangle enclosing P. Then, we prove that in general one cannot use optimal ray cutting sequences (or approximate ray cutting sequences) computed by such an algorithm to replace the ray cutting sequences constructed in the third phase of the ray cutting algorithm in [2]. This suggests that in order to reduce the $O(\log^2 n)$ approximation factor one should use an approach different from the one in [2].

2 Cutting out polygons with guillotine cuts

Our approach for solving the problem resembles that in [2] (see above). Since our solution differs from the one in [2] in the separation phase, we only focus on this phase. As mentioned above, in the separation phase, three line cuts are used to obtain a triangle that encloses P and has perimeter roughly the same as that of P .

Our solution for the separation phase involves a solution for the following subproblem. Consider a bounded x-monotone polygonal chain Π and a real value c. For a point $q = (q_x, q_y)$ on Π we define the *window* $W(q)$ of q as the vertical strip bounded by the lines of x-coordinates $q_x - c$ and $q_x + c$, that is, the vertical strip centered at q and of width $2c$. We use $W_o(q)$ when $W(q)$ is open at $q_x - c$ and $q_x + c$ and $W_c(q)$ when it is closed. The goal is to find two points $a = (a_x, a_y)$ and $b = (b_x, b_y)$ such that $a_y + b_y$ is minimized and $b \notin W_o(a)$.

Lemma 1 *Given a bounded x-monotone polygonal chain Π with n vertices and a real value c, a pair of points $a = (a_x, a_y)$, $b = (b_x, b_y)$ such that $a_y + b_y$ is minimized and $b \notin W_o(a)$ can be found in linear time.*

To prove Lemma 1 we first make the following observation. Let $a_0 = (x_0, y_0) \in \Pi$ be the point of smallest y-coordinate and let $a_1 = (x_1, y_1) \in \Pi$ be the point of second smallest y-coordinate.

Observation 1 *If $a_1 \in W(a_0)$ then the points $a = (a_x, a_y)$, $b = (b_x, b_y)$ such that $a_y + b_y$ is minimized and $b \notin W_o(a)$ are within $W_c(a_0)$.*

Proof. If one of the two points, say b, is outside the window $W_c(a_0)$ then we can use a_0 instead of a to obtain a better cut. □

Proof. (Lemma 1) Let $a_0 = (x_0, y_0) \in \Pi$ be the point of smallest y-coordinate and let $a_1 = (x_1, y_1) \in \Pi$ be the point of second smallest y-coordinate. Clearly, a_0 and a_1 can be found in linear time. If $a_1 \notin W(a_0)$ we set $a = a_0$ and $b = a_1$ and we are done. Then, consider the case when $a_1 \in W(a_0)$. We set $a = a_0$ and set b to

the point of Π with x-coordinate $x_0 + c$. Starting with a window bounded by the vertical line $l_l = x_0$ to the left and the vertical line $l_r = x_0 + c$ to the right, slide this window to the left until l_l becomes the left bounding line of $W_c(a_0)$ (that is, perform a line sweep of $W(a_0)$ until l_l becomes the left bounding line of $W_c(a_0)$). The window stops at each vertex of $\Pi \cap W_c(a_0)$. During this sweep, we maintain the smallest y-coordinate y_{min} of Π swept by l_r. If we get a smaller value for the sum of y_{min} and the y-coordinate of the point $\Pi \cap l_l$, we update a and b accordingly. Clearly, this sweep can be done in linear time. □

Our algorithms use Lemma 1 above with some small changes. First, between any two consecutive vertices of Π we have a convex curve rather than a line segment. Second, the line sweep should be "wrapped around" at the end points of Π if necessary. It is easy to see these changes do not affect the time complexity to find the points a and b and we ignore the wrap around in the description below.

We now adapt the algorithm in [2] to find the two cuts of total smallest length in linear time.

For a line cut l, the length of l is denoted as $|l|$. The angle of the line cut that is parallel to the x-axis and tangent to P from below is $0°$. The angle increases gradually as the line cut rotates counter-clockwise along the boundary of P while being tangent to P. Thus, we have that $\theta \in [0°, 360°)$.

The problem is to find two cuts l_1 and l_2 such that (1) l_1 and l_2 are tangent to P, (2) the angle between l_1 and l_2 is such that $|\theta_1 - \theta_2| \notin [0°, 20°)$ and (3) $|l_1| + |l_2|$ is minimized.

Let l be a line tangent to P and along the edge $v_{i-1}v_i$ of P, with $1 \leq i \leq n$ and $v_0 = v_n$, and consider rotating l around v_i until it overlaps with the edge v_iv_{i+1} or it touches a vertex of Q. Then, in this angle interval $\theta \in [\theta_i^1, \theta_i^2]$, the length $|l| = l(\theta)$ of l is a convex function [1].

The functions $l(\theta)$ defining the cut length $|l|$ for all the $O(n+m)$ angle intervals can be computed in $O(n+m)$ time by rotating l along the boundary of P (similar to the *rotating calipers* technique [6]). The diagram of $l(\theta)$ is a continuous function that consists of $O(n+m)$ convex curves, and it has $O(n+m)$ local minima.

The algorithm is as follows:

1: Let $\mathcal{M} = \{m_1, m_2, \ldots, m_p\}$ be the list of local minima, where $p = O(m+n)$ is the number of local minima. Let the corresponding angles and cuts be $\mathcal{A} = \{a_1, a_2, \ldots, a_p\}$ and $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$, respectively. Note that the cuts are such that $a_1 < a_2 < \ldots < a_p$.
2: Find the two cuts $l_1, l_2 \in \mathcal{C}$ of smallest length, with $|l_1| \leq |l_2|$. Let θ_1 and θ_2 be the angles of l_1 and l_2, respectively.
3: Set $l = |l_1| + |l_2|$.
4: **if** $|\theta_1 - \theta_2| \in [0°, 20°)$ **then**
5: Set $l = |l_1| + min\{|c_{-20°}|, |c_{+20°}|\}$ where $c_{-20°}$ and $c_{+20°}$ are the two cuts of angle $\theta_1 - 20°$ and $\theta_1 - 20°$ respectively. Then, perform a sweep as in Lemma 1 within the angle interval $(\theta_1 - 20°, \theta_1 + 20°)$, starting with a sliding window that has the left line $l_l = \theta_1$ and the right line $l_r = \theta_1 + 20°$. During this sweep, if we find two cuts of smaller total length we update l accordingly.
6: **end if**

Lemma 2 *Given two convex polygons P and Q, with $P \subset Q$, we can find the two cuts l_1 and l_2 of minimum total length $l_{min} = |l_1| + |l_2|$ in $O(n+m)$ time.*

Proof. Use the algorithm above. The total running time is $O(n+m)$ since finding the two smallest cuts in $\mathcal{M}$ takes $O(n+m)$ time and the cost of the sweep is $O(n+m)$, as only insertions of local minima are required. We next argue that $l_{min} = |l_1| + |l_2|$, as computed by the algorithm above, is of minimum length. Let θ_1 and θ_2 be the angle of the cuts l_1 and l_2, respectively. Let θ_{min} be the angle corresponding to the cut of smallest length. We have $\theta_1 = \theta_{min}$ and $\theta_2 \notin (\theta_1 - 20°, \theta_1 + 20°)$ or $\theta_1 \in (\theta_{min} - 20°, \theta_{min})$ and $\theta_2 \in [\theta_1 + 20°, \theta_{min} + 20°)$ or $\theta_1 \in (\theta_{min}, \theta_{min} + 20°)$ and $\theta_2 \in (\theta_{min} - 20°, \theta_1 - 20°]$. We check all possible pairs (l_1, l_2) during the window sweep and maintain the smallest cut length for $|l_1| + |l_2|$ over all such pairs. □

From Lemma 2 it follows that the separation phase can be done in $O(n+m)$ time. Recall from Section 1 that the carving phase can be carried out in $O(n^3)$ time or in $O(n)$ time, depending whether an $O(1)$-factor or an $O(\log n)$-factor approximation is sought, respectively. Putting these together we have:

Theorem 3 *Given two convex polygons P and Q, $P \subset Q$, with n and m vertices, respectively, an $O(1)$-factor approximation of an optimal cutting sequence for cutting P out of Q can be computed in $O(n^3 + m)$ time. An $O(\log n)$-factor approximation of an optimal cutting sequence can be found in $O(n+m)$ time.*

3 The Ray Cutting Problem

In this section we present some negative results with respect to improving the approximation bound of the ray cutting algorithm in [2]. Here, P is a ray cuttable polygon and Q is a convex polygon, with $P \subset Q$.

The algorithm in [2] involves three phases. The first phase cuts out a triangle containing P of about the same size as P, in $O(m+n)$ time. The cost of this cutting sequence is an $O(1)$-factor more than the cost of an optimal ray cutting sequence for P. The second phase cuts out the convex hull $CH(P)$ of P from that triangle, in $O(n)$ time. It uses the $O(n)$ time algorithm for the carving phase of the line cutting problem, producing

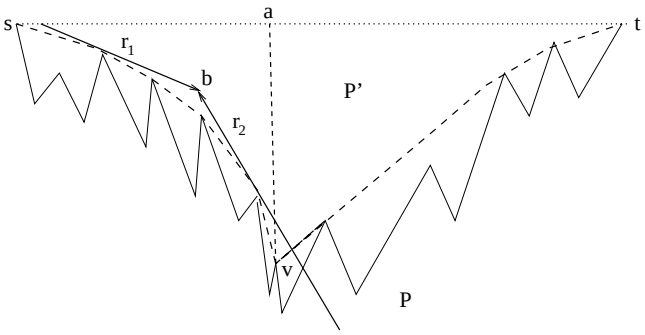

Figure 2: Two rays r_1 and r_2 for cutting out CH_{vs}; the ray r_2 cuts through P.

a cutting sequence of cost $O(|P| \log n)$, where $|P|$ is the perimeter of P. Thus, the cost of this cutting sequence is an $O(\log n)$-factor approximation of an optimal ray cutting sequence for P. The third phase cuts out the *pockets* of P from its convex hull, $CH(P)$. In this phase, a pocket of P is cut out from $CH(P)$ in $O(\log n)$ recursive steps. Each step relies on constructing a ray cutting sequence from a line cutting sequence; the line cutting sequence is found by the same algorithm for phase two above. This phase takes $O(n \log n)$ time and produces a cutting sequence that is an $O(\log^2 n)$-factor approximation of an optimal ray cutting sequence for P.

Naturally, one could ask the following question: if an algorithm to efficiently compute an optimal (and for that matter an approximate) ray cutting sequence for cutting out P from the smallest axis aligned rectangle enclosing P is known, would that improve the approximation bound in phase three?

Here, we prove that in general one cannot use optimal ray cutting sequences (or approximate ray cutting sequences) computed by such an algorithm to replace the ray cutting sequences constructed in the third phase of the ray cutting algorithm in [2]. Our result suggests that in order to reduce the $O(\log^2 n)$ approximation factor one should use an approach different from the one in [2].

Lemma 4 *In general, optimal ray cutting sequences (or approximate ray cutting sequences) cannot be used to replace the ray cutting sequences constructed in the third phase of the ray cutting algorithm in [2].*

Proof. The key subproblem within some step of the third phase is illustrated in Fig. 2. Here, P' is a pocket of P (the portion of Q between P and an edge st of $CH(P) \setminus P$). The goal for this subproblem is to cut out CH_{vs}, representing the shortest path in $Q \setminus P$ from v to s. The example in Fig. 2 shows that in general one cannot use regular ray cuts to cut out CH_{vs}, and thus the pocket P' of P. The reason is that in general an optimal (or approximate) ray cutting sequence for cutting out CH_{vs} from its smallest enclosing rectangle, computed with no consideration to the context of the larger problem, can have rays that cut through P, as does the ray r_2 in Fig. 2. There, r_2 ends at point b on another ray cut r_1 and thus it must extend to infinity at the other end, thus cutting through P. In contrast, the algorithm in [2] constructs a particular ray cutting sequence in which all rays start at infinity, cross st, and end on a special ray cut (cut av in Fig. 2). In general, the cost of this ray cutting sequence could be much higher than the cost of an optimal ray cutting sequence (but not more than an $O(\log n)$-factor). □

4 Conclusion

In this paper we discussed approximation algorithms for the problem of cutting out a convex polygon P from another convex polygon Q by a sequence of guillotine cuts of smallest total length. We presented an $O(n^3 + m)$ running time, constant-factor approximation algorithm, and an $O(n + m)$ running time, $O(\log n)$-factor approximation algorithm for cutting P out of Q. We also discussed negative results for the case when guillotine cuts are replaced by ray cuts within the context of the ray cutting algorithm in [2]. These results give more insight into the complexity of the ray cutting problem.

References

[1] Bhadury, J. and Chandrasekaran, R.: Stock cutting to minimize cutting length. European Journal of Operational Research. **88** (1996) 69–87.

[2] Daescu, O., Luo, J.: Cutting out polygons with lines and rays. Proc. 15th Annual International Symposium on Algorithms and Computation (ISAAC'04), LNCS 3341, (2004), 669–680.

[3] Demaine, E.D., Demaine, M.L. and Kaplan,C.S.: Polygons cuttable by a circular saw. Computational Geometry: Theory and Applications. **20** (2001) 69–84.

[4] Dumitrescu, A.: An approximation algorithm for cutting out convex polygons. Computational Geometry: Theory and Applications. **29(3)** (2004) 223–231.

[5] Overmars, M.H. and Welzl, E.: The complexity of cutting paper. Procs. of the 1st Annual ACM Symposium on Computational Geometry. (1985) 316–321.

[6] Toussaint, G.T.: Solving geometric problems with the 'rotating calipers'. Procs. MELECON, Athens, Greece, 1983.