Cutting out polygons with a circular saw*

Adrian Dumitrescu† Masud Hasan‡

April 30, 2012

Abstract

Given a simple (cuttable) polygon Q drawn on a piece of planar material R, we cut Q out of R by a (small) circular saw with a total number of cuts no more than twice the optimal. This improves the previous approximation ratio of 2.5 obtained by Demaine et al. in 2001.

Keywords: Approximation algorithm, saw cut, cuttable polygon, shortest path tree.

1 Introduction

The problem of efficiently cutting a polygon Q out of a piece of planar material R (Q is already drawn on R) is by now a well studied problem in computational geometry. This line of research was initiated by Overmars and Welzl in their seminal paper from 1985 [16]. There exist several variants of this problem, based on the cutting tools used and the measure of efficiency employed for analyzing them. The type of cutting tool also determines which polygon can be cut—a convex polygon or a non-convex polygon. The cutting tools that have been studied so far are line cuts, ray cuts, and saw cuts [1, 5–7, 9–12, 16, 17]. Since a saw can be abstracted as a moving line segment, one can refer to saw cuts also as segment cuts. Then the types of cutting tools can be expressed quite uniformly in geometric terms: line cuts, ray cuts, and segment cuts. The efficiency measures that have been considered are the total length of cuts and the total number of cuts. See also [2] for a survey on previous results on these variants.

A line cut is a line that does not go through Q and splits (cuts) R (or the current piece of material) into two pieces. For cutting Q out of R by line cuts, Q must be convex. The most studied efficiency measure for line cutting is the total length of the cuts, where several approximation algorithms have been obtained [1, 5–7, 9–12, 16, 17], including a PTAS [5].

In contrast, a ray cut comes from infinity and can stop at any point outside Q. The main focus of ray cutting problems is cutting non-convex polygons. However, not all non-convex polygons can be cut by ray cuts. Again, the most studied efficiency measure is the total length of the ray cuts. Here the results include deciding whether a polygon is ray cuttable and several approximation algorithms [7, 9, 10, 17].

Demaine et al. [10] studied the problem of cutting a polygon by a circular saw. A circular saw cut (also called a saw cut) is like a ray cut, but may not come from infinity. The saw is abstracted

*An extended abstract of this paper appeared in the *Proceedings of the the 22nd International Symposium on Algorithms and Computation* (ISAAC 2011), Yokohama, Japan, Vol. 7074 of LNCS, Springer, pp. 230–239.

†Department of Computer Science, University of Wisconsin–Milwaukee, WI 53201-0784, USA. Email: dumitres@uwm.edu. Supported in part by NSF grant CCF-0444188 and NSF grant DMS-1001667.

‡Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh. Email: masudhasan@cse.buet.ac.bd. Part of the research by this author was done while visiting University of Wisconsin-Milwaukee in Fall 2010 and with support from NSF grant CCF-0444188.

as a line segment, which cuts through the material if moved along its supporting line. If a small free space within R is available, a saw cut can be initiated there by maneuvering the saw. The space required for maneuvering the saw is proportional to the length of the saw (line segment). For convenience and ease of analysis the length of the saw was assumed in [10] to be arbitrarily small, and its width was assumed to be zero. This means that a saw cut can be initiated from an arbitrarily small free space.

Two efficiency measures were considered in [10]: the total length of the cuts and the total number of cuts. A smaller number of cuts implies a shorter maneuvering time. The authors gave an approximation algorithm for cutting Q out of its convex hull $\mathrm{conv}(Q)$, with both the total length of the cuts and the total number of cuts within 2.5 times their respective optimums. Here we improve the approximation guarantee in the latter measure of efficiency from 2.5 to 2. Moreover, our approximation guarantee is in a stronger sense than that achieved previously by Demaine et al. [10]. While theirs achieves ratio 2.5 for cutting out Q from $\mathrm{conv}(Q)$, ours achieves ratio 2 for cutting out Q from any enclosing polygon R. We summarize our result in the following theorem.

Theorem 1. *Given a simple polygon Q fixed inside a piece of planar material $R \supset Q$, Q can be cut out of R by a circular saw of arbitrarily small maneuvering space with a total number of cuts no more than twice the optimal. If R and Q have m and respectively n vertices, the cuts can be computed in $O(m + n \log n)$ time.*

Remark. More precisely we show that if Q has n vertices, it can be cut out using at most $2n - h$ cuts, where h is the size of $\mathrm{conv}(Q)$, i.e., the number of vertices of the convex hull of Q. In particular, if the average pocket complexity of Q is bounded from above by a constant, the approximation ratio of the algorithm drops below 2. Indeed, under the assumption we will make that no three vertices of Q are collinear, each edge of Q requires at least one cut, hence overall at least n cuts are needed.

Outline. The idea of our algorithm is as follows. The difference between Q and its convex hull $\mathrm{conv}(Q)$ consists of disjoint pockets. Each pocket is defined by an edge of the convex hull, which we call the *lid* of the pocket. For each edge of the convex hull we make a cut along the edge. In addition, if there is a pocket corresponding to this edge, we cut it out. We cut out each pocket of Q by using a number of cuts at most twice the number of edges of the respective pocket (excluding the lid) minus 1. To this end, for each pocket P, we shall compute the shortest path tree T of P from one of the vertices of the lid. Then we shall partition the edges of P into two types of polygonal chains. In the first type, each chain consists of only reflex edges of P and contains an internal vertex of T. In the first phase of our algorithm, we find those chains by traversing T. Along this traversal, we cut those chains by two cuts for each edge. The other type of chains are formed from the remaining edges. In the second phase of our algorithm, we cut the edges of those chains by using no more than two cuts per edge on average.

The paper is organized as follows. In Section 2, we give some preliminaries, including the setting for saw cuts and preliminaries on shortest path trees. Then in Section 3, we present the algorithm and analyze its approximation ratio and running time.

2 Preliminaries

Let Q be the polygon to be cut out from an arbitrary enclosing polygon R. We assume that the vertices of Q are in general position, i.e., no three vertices of Q are collinear. Let $n = |Q|$ and

$h = |\text{conv}(Q)|$ be the number of vertices of Q and $\text{conv}(Q)$, respectively. Let (s, r) be an arbitrary edge of $\text{conv}(Q)$. We first make one long cut along (s, r). This is similar to a line cut, and removes the part of R on the other side of the supporting line of (s, r). If there is a pocket behind the lid (s, r), we proceed to cut it out. The pocket is an open polygonal chain, say U, with endpoints s and r. $U \cup (s, r)$ defines a simple polygon, and we shall denote this polygon by P.

We say that a vertex v of P is *reflex* (resp. *convex*) if its angle within P is larger than π (resp. smaller than π). This definition of reflex and convex for v is opposite with respect to the interior of Q. An edge e of P is *reflex* if both its endpoints are reflex. After the first cut for P is made along the lid (s, r), the vertices s and r become reflex in P.

A region F in the plane outside Q but contained in R is called *free* if the material has been removed from F by making cuts along F's boundary. As cuts are executed, pieces of material that become free are automatically removed. A point x on the boundary of P is called *free* if there exists a free half-disc centered at x. A point x in the interior of P is called *free* if there exists a free disc centered at x. A *saw cut* is a line segment that starts from a free region and does not enter Q. An edge e of P is called *free* if every point of e except its convex endpoint (if any) is free.

When a saw cut is applied along an edge of Q, the edge becomes *disconnected* from Q. Clearly, a free edge is also disconnected, but the reverse is not true. We assume that after a saw cut is applied, any region of R that becomes disconnected from the remaining portion of R is automatically removed. Then that region of R becomes free. The removing procedure for a disconnected region does not require a planar motion; it can be done by lifting it up. Once the lid of a pocket P has been cut along, P is considered to have been *cut* when all the edges of the polygonal chain $U = U(P)$ have been disconnected.

A convex vertex of P can never be free, since it cannot be the center of a free half-disk. Therefore, a saw cut cannot be initiated from a convex vertex. However, by definition, if a free edge e has a convex endpoint v, then a cut can be initiated from a point arbitrarily close to v. We call a convex vertex to have been *disconnected* when both its adjacent edges are disconnected. For each pocket $P = U \cup (s, r)$, we cut P using a number of cuts no more than $2|U| - 1$, where $|U|$ is the number of edges of U. Since the polygonal chains U of the pockets are disjoint, summing over all pockets of $\text{conv}(Q) \setminus Q$ will result in cutting the entire polygon Q by at most $2n - h$ cuts.

Not every polygon can be cut by a circular saw, even with a saw of arbitrarily small maneuvering space. There may be an edge of the polygon whose two endpoints are reflex within the polygon (thus convex within a pocket). For such an edge, no cut can be initiated. Moreover, Demaine et al. [10] showed that a polygon Q is cuttable by a small circular saw if and only if Q does not have two consecutive reflex vertices. For brevity we call such a polygon *cuttable*. Equivalently, a polygon Q is cuttable if and only if each pocket of Q does not have two consecutive convex vertices. Again for brevity we refer to any such pocket as *cuttable*. Interestingly enough however, any simple polygon can be be modified very slightly, for instance by adding very flat reflex vertices in between any two consecutive convex vertices, so that the resulting polygon becomes cuttable; see [10] for more details.

We next give some preliminaries on shortest paths trees inside a simple polygon P, that are key to our algorithm. Let s be a point contained in P and v be a vertex of P (in our case s is an endpoint of the pocket lid of P, hence also a vertex of P). The Euclidean shortest path from s to v inside P (according to total Euclidean length) is a unique polygonal chain (path) from s to v which can make turns only at reflex vertices of P [14, 15]. This path is denoted by $\pi(s, v)$. The union of the paths $\pi(s, v)$ over all vertices v of P forms a plane tree, called the *shortest path tree* of P from s [14, 15]. We denote this tree by T. The reflex vertices of P at which paths of T make turns are exactly the internal vertices of T. On the other hand, the vertices of P at which paths of T do not make turns are the leaves of T. While all the internal vertices of T are reflex, a leaf

vertex of T can be reflex or convex. An edge connecting two internal vertices in T is referred to as an *internal edge* of T. An internal edge of T may be an edge of P or not. If it is an edge of P, we call it a *boundary edge* of T, otherwise we call it a *bridging edge* of T. An internal edge of T is called *reflex* if both its endpoints are reflex.

Lemma 1. *Let (u, v) be a (reflex/non-reflex) edge of P or a reflex edge of T, with u being a reflex vertex. Let $(p, v) \supseteq (u, v)$ be a line segment such that p is free. Then it is possible to make (u, v) free by using two cuts initiated from the neighborhood of p. If v is also reflex, this cut makes v free too.*

Proof. See Fig. 1 for an illustration of how the cuts are executed in each case. □

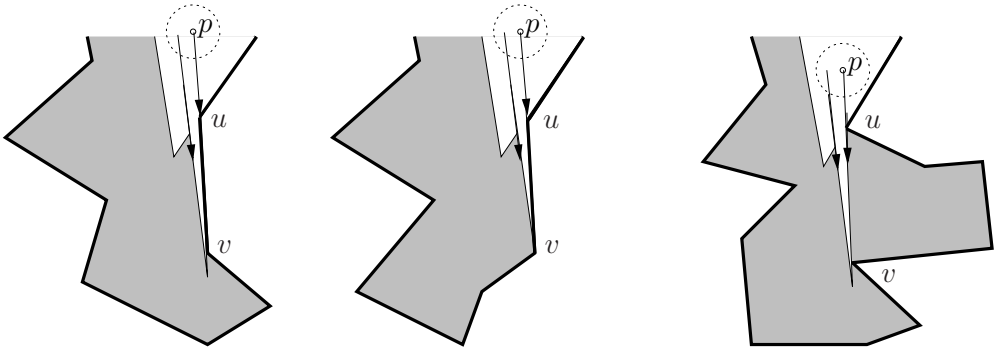

Figure 1: Two cuts are sufficient two make free a reflex or non-reflex edge of P (left & middle), or a reflex edge of T (right), if there is a free space to initiate the cuts.

We compute the shortest path tree T of P from s. We identify two types of polygonal chains on the boundary of P: reachable chains and critical chains. These chains are edge-disjoint. We also categorize the vertices of P into three types. Let v be an internal vertex of T. If v is an endpoint of a reflex edge e, then let C_R be the maximal polygonal chain that consists of only reflex edges of P including e. We call C_R a *reachable chain* of P. The vertices of reachable chains are called *type-1* vertices. On the other hand, the two adjacent vertices of v can be convex, and in that case, we call v a *type-2* vertex. (A type-2 vertex can be considered as a reachable chain with zero edges.) Critical chains are defined by type-2 vertices and the endpoints of reachable chains. More precisely, a *critical chain* is a polygonal chain whose two endpoints are type-2 vertices or endpoints of reachable chains. Reachable chains contain all internal vertices of T and possibly some leaves of T. Other vertices of P are the internal vertices of critical chains and we call them *type-3* vertices. Type-3 vertices are leaves of T; these vertices can be reflex or convex. See Fig. 2.

Some properties of reachable chains and critical chains are easy to follow. A reachable chain has at least one edge and does not have any convex vertex. Due to the maximality property, two reachable chains cannot be adjacent, i.e., two reachable chains are vertex disjoint. However, two critical chains may be adjacent in a type-2 vertex. A reachable chain is adjacent to a critical chain in a type-1 vertex. A critical chain has at least one convex vertex. The two adjacent edges of this convex vertex imply that a critical chain has at least two edges.

3 Algorithm

We first compute K, a minimum axis-aligned rectangle containing R. For convenience, we start all cuts that originate outside $\text{conv}(Q)$ on the boundary of K. Alternatively, these cuts can be

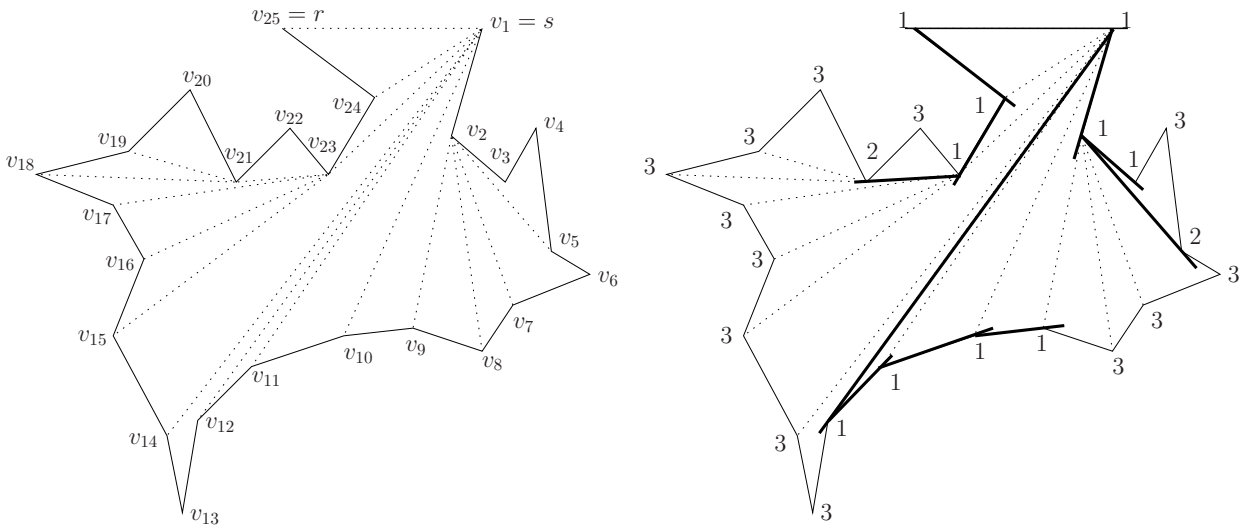

Figure 2: Left: A pocket P with its shortest path tree T from s. The bridging edges of T are shown in dotted lines. The reachable chains of P are $\{v_1, v_2, v_3\}$, $\{v_9, v_{10}, v_{11}, v_{12}\}$ and $\{v_{23}, v_{24}, v_{25}\}$. Type-1 vertices of P are all vertices in these three chains. Type-2 vertices of P are v_5, and v_{21}. The rest of vertices of P are of type-3. The critical chains of P are $\{v_3, v_4, v_5\}$, $\{v_5, v_6, v_7, v_8, v_9\}$, $\{v_{12}, v_{13}, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20}, v_{21}\}$, and $\{v_{21}, v_{22}, v_{23}\}$. Right: double cuts made in Phase 1 are drawn in bold lines; the cut along the pocket lid (in Phase 0) is a single cut.

initiated near their intersection points with the boundary of R.

Let (s, r) be an arbitrary edge of $\mathrm{conv}(Q)$. We first make one long cut along (s, r); we will refer to this single cut as Phase 0. If there is no pocket of Q behind this edge, we proceed to the next edge of $\mathrm{conv}(Q)$. If there is, i.e., (s, r) is the lid of a pocket $P = U \cup (s, r)$, we proceed to cut it out. After the cut along (s, r) the vertices s and r become reflex. Denote by t_1, t_2, and t_3 the number of type-1, type-2 and type-3 vertices of P. Vertices s and r are of type-1, hence $t_1 \geq 2$ in particular. Clearly we have $t_1 + t_2 + t_3 = t$, where $t \geq 2$ is the number of vertices in P, including the two endpoints of the pocket lid.

In Phase 1 of our algorithm, we shall cut along all internal edges of T and along the edges of reachable chains by at most $2(t_1 + t_2) - 4$ cuts. This will make free all type-1 and type-2 vertices and the edges of all reachable chains. In Phase 2 of our algorithm, we shall cut along the edges of critical chains by using at most $2t_3$ cuts. Completion of these two phases will give the result.

The number of cuts made for cutting out one pocket (including the initial cut along the convex hull edge) is at most $1 + 2(t_1 + t_2) - 4 + 2t_3 = 2(t_1 + t_2 + t_3) - 3 = 2t - 3$. Overall, Q is cut out of R by at most $\sum_P (2t - 3) = \sum_P (2|U| - 1) = 2n - h$ cuts. Since n is a trivial lower bound on the number of cuts needed, the approximation ratio of 2 immediately follows; see also [10].

Phase 1: Cutting reachable chains. We shall traverse T in level order. At each level $i \geq 1$, we assume that all reflex vertices in level $i - 1$ are already free. In addition we assume that all reachable chains incident to reflex vertices in level $i - 1$ are also free. For level 0 we start with two reflex vertices, s and r, which are already free. In addition, if w is free and (w, v) is a reflex edge, we make free (w, v) by two cuts by Lemma 1. We repeat this step to make free all reflex edges that are reachable in this way, i.e., part of reachable chains. So the above assumptions are satisfied for level zero.

Let $i \geq 1$. We look into each reflex vertex v in level i. If v is not free, let w be the parent of

v in level $i-1$. Observe that (w,v) is a bridging edge. Because, if (w,v) were a reflex edge of P, then it would belong to the reachable chain that contains w, and therefore we would have made free (w,v) as well as v in the previous level. Since w is already free, we make free (w,v) by two cuts by Lemma 1. Then if v belongs to a reachable chain, say C_R, then for each reflex edge e of C_R, if e is not free, we make it free by two cuts by Lemma 1.

The following lemma proves that we have made free the edges of all reachable chains and quantifies the number of cuts used for that.

Lemma 2. *After Phase 1, all internal vertices of T and the edges of all reachable chains of P are free. The number of cuts used to make them free in Phase 1 is at most $2(t_1+t_2)-4$, where t_1 and t_2 are the total number of type-1 and type-2 internal vertices of P, respectively. These cuts also make free the endpoints of all reachable chains.*

Proof. By our algorithm, we have traversed all internal vertices of T and made them free, if they were not free already, from their parents in the respective previous levels. Since every reachable chain contains an internal vertex of T, this guarantees that for each reachable chain we have reached one of its vertices and from there we have made free all of its edges.

Let v be an internal vertex of T to which we have reached from its parent w in the previous level (or from r). Let C_R be the reachable chain that contains v. If v was not already free, we made it free by two cuts. Then we used two more cuts to free each edge, and corresponding new vertex of C_R. If t_R is the number of vertices in C_R, then there are t_R-1 edges in C_R. Therefore, we have used at most $2+2(t_R-1)=2t_R$ cuts for C_R. Since two reachable chains are disjoint, over all reachable chains we have used at most $\sum_{c_R} 2t_R = 2t_1$ cuts.

If v is a type-2 vertex, then we have used only two cuts to make it free and no cuts for a reachable chain (a reachable chain does not exist in this case). So for all these t_2 vertices, we have used a total of $2t_2$ cuts.

It follows that so far at most $2(t_1+t_2)$ cuts have been made in Phase 1. Recall that s and r were made free by one cut in Phase 0. Observe that each pair of cuts executed in Phase 1 made free a new vertex. Thus the above expression over-counts by 4, hence we have used in fact at most $2(t_1+t_2)-4$ cuts. Finally, since the endpoints of reachable chains are reflex vertices, by Lemma 1, the above cuts have made free those endpoints too. □

At the conclusion of Phase 1, all type-1 and type-2 vertices of P and the edges of all reachable chains have been made free. We are left with only the edges and internal vertices (of type-3) of critical chains. In what follows, we shall see how to make them free.

Phase 2: Cutting critical chains. Let $P=\{s=v_1,v_2,\ldots,v_n=r\}$ be the clockwise vertex sequence of P. Consider a path $\pi(s,v_i)$ in T. We denote by $C_{right}(v_i)=\{v_1,v_2,\ldots,v_i\}$ the polygonal chain consisting of the vertices from v_1 to v_i and by $C_{left}(v_i)=\{v_i,v_{i+1},\ldots,v_n\}$ the polygonal chain consisting of the vertices from v_i to v_n.

Let $e=(v_i,v_{i+1})$ be an edge of P where v_i and v_{i+1} are leaves of T. Let a be the lowest common ancestor of the paths $\pi(s,v_i)$ and $\pi(s,v_{i+1})$. Let $\pi'(a,v_i)$ and $\pi'(a,v_{i+1})$ be the sub-paths of $\pi(s,v_i)$ and $\pi(s,v_{i+1})$ from a to v_i and v_{i+1}, respectively.

It is known [14,15] that both $\pi'(a,v_i)$ and $\pi'(a,v_{i+1})$ are outward convex and oppositely oriented. Together with e they define a *funnel* with *tip* a, denoted by $\mathcal{F}(v_i,v_{i+1})$.

Lemma 3. *Let f be an edge of $\pi'(a,v_i)$. If f is an edge of P, then f is an edge of $C_{right}(v_i)$.*

Proof. If we traverse $C_{right}(v_i)$ from v_i to s, then the interior of P is on the left side. Similarly, if we traverse $C_{left}(v_i)$ from v_i to s, then the interior of P is on the right side. Since T is a plane

tree, T partitions P into disjoint plane regions. The interior of the funnel $\mathcal{F}(v_i, v_{i+1})$ is one of these regions. If we traverse $\pi(s, v_i)$ from v_i to s, this funnel is on the left side. Therefore, if f is in $\pi'(a, v_i)$, f is in $C_{right}(v_i)$. □

Lemma 4. *Consider a funnel $\mathcal{F}(v_i, v_{i+1})$ with tip a, and a point x on $\pi'(a, v_{i+1})$. Then any ray emanating from x towards the interior of $\mathcal{F}(v_i, v_{i+1})$ must hit $e = (v_i, v_{i+1})$ or an edge in $C_{right}(v_i)$.*

Proof. Let the ray emanating from x be ℓ. Since $\pi'(a, v_{i+1})$ is outward convex, ℓ cannot hit an edge of $\pi'(a, v_{i+1})$. So, ℓ hits either e or $\pi'(a, v_i)$. Assume that ℓ hits $\pi'(a, v_i)$ at the edge e'. If e' is an edge of P, then by Lemma 3, e' is an edge of $C_{right}(v_i)$ and we are done. Otherwise, $e' = (v_j, v_k)$, for some $j < k \leq i$, is a bridging edge of T. Then ℓ hits one of the edges of P in the chain $\{v_j, v_{j+1}, \ldots, v_k\}$, that is, an edge in $C_{right}(v_i)$. □

The following observation is very useful in cutting critical chains. We formulate it as a lemma.

Lemma 5. *Let $C = \{v_j, v_{j+1}, \ldots, v_k\}$, where $j < k$, be a chain of edges of P, and assume that the endpoints of the chain v_j and v_k are already free. Assume that all edges of this chain have been cut along and are thus disconnected. Then each edge in the chain $\{v_j, v_{j+1}, \ldots, v_k\}$ is free.*

Proof. Observe that there exist paths of free points (hence corridors of free space) connecting each of v_j and v_k with the free space outside $\text{conv}(Q)$. Since all edges of C have been disconnected, the part of R adjacent to any such edge can be removed and thus the edge is made free. □

Now, we will show how to cut along and thus make free the edges of the critical chains. We consider the critical chains one by one in clockwise sequence starting from s.

Lemma 6. *Let $C_C = \{v_j, v_{j+1}, \ldots, v_k\}$, for $k - j \geq 2$, be a critical chain in clockwise sequence. Then, there exist at most $2(k-j-1)$ cuts by which the $(k-j)$ edges of C_C can be made free. Over all critical chains of P, this takes at most $2t_3$ cuts.*

Proof. We move from v_j towards v_k. Recall that v_j and v_k are reflex and free, and v_{j+1} and v_{k-1} (possibly $v_{j+1} = v_{k-1}$) are convex. No other vertices in this chain are free. There may be other convex vertices in C_C. However, since P is cuttable, between any two convex vertices there is at least one reflex vertex. If $k-j = 2$ we apply two cuts, one from v_j and one from v_k and by Lemma 5 we are done with this chain.

If $k - j \geq 3$ there are at least two convex vertices in C_C, v_{j+1} and v_{k-1}, and at least one reflex vertex other than v_j and v_k. Hence $k - j \geq 4$. We apply two cuts from v_j to free (v_j, v_{j+1}) by Lemma 1. Then, for each reflex vertex in the sequence of C_C, we proceed as follows. Let v_i, for some $j + 2 \leq i \leq k - 2$, be the next reflex vertex. (In the first round $i = j + 2$ and v_{i-1} is convex.)

If v_i is already free, make cuts along its incident edges if not yet done previously: (i) if v_{i-1} is convex apply one cut along the edge (v_i, v_{i-1}), otherwise v_{i-1} is reflex and also already free. (ii) apply two cuts along the edge (v_i, v_{i+1}) by Lemma 1; if v_{i+1} is reflex, this will make free v_{i+1} too.

Assume now that v_i is not free. We look into the edge (v_i, v_{i+1}). We extend this edge from v_i until it hits the boundary of P. Let ℓ be this extended line segment and p be the point on the boundary of P where ℓ hits. At this moment, we make the following claims.

- *Claim 1.* The edges of $C_{right}(v_{i-1})$ are already free.
- *Claim 2.* ℓ hits an edge of $C_{right}(v_{i-1})$.

Assuming the above claims hold, p is free. We make free the edge (v_i, v_{i+1}) by two cuts from p by Lemma 1. This will make free v_i. If v_{i+1} is also reflex, this will make free v_{i+1} too. If v_{i-1} is convex, we apply a single cut along (v_i, v_{i-1}). By Claim 1, all edges of $C_{right}(v_{i-1})$ were disconnected, and due to the cut along (v_i, v_{i-1}), the edges of $C_{right}(v_i)$ are now disconnected. Since v_i is now free, by Lemma 5 all edges of $C_{right}(v_i)$ are now free.

We repeat the above procedure until all reflex vertices in C_C have been made free. Finally, we apply a single cut along (v_k, v_{k-1}) which disconnects the convex vertex v_{k-1}. By induction, it follows that all edges of $C_{right}(v_k)$ are now free. We now prove the claims.

Proof of Claim 1: The reachable chains of $C_{right}(v_{i-1})$ (preceding C_C) have already been made free in Phase 1 of our algorithm. The critical chains of $C_{right}(v_{i-1})$ (preceding C_C) are also free, because we process them in clockwise sequence. Finally, within C_C, we cut its edges in clockwise sequence. Therefore, all edges preceding v_{i-1} in clockwise order are free.

Proof of Claim 2: Recall that ℓ extends the edge (v_{i+1}, v_i) from v_i until it hits the boundary of P. Since v_i is reflex (as assumed), ℓ precedes (v_i, v_{i-1}) clockwise by at most π; see Fig. 3. To prove that ℓ hits an edge of $C_{right}(v_{i-1})$, we look into the funnel $\mathcal{F}(v_{i-1}, v_i)$. Note that $\mathcal{F}(v_{i-1}, v_i)$ is non-degenerate, in the sense that none of v_{i-1} and v_i is the parent of the other in T; recall that both are vertices of a critical chain that are leaves in T; see Fig. 3(left). If ℓ is on the left side of $\pi(a, v_i)$, then the path $\pi(s, v_{i+1})$ must make a turn at v_i. But that would imply that v_i is an internal vertex of T, which contradicts that v_i is a vertex of a critical chain; see Fig. 3(left). It follows that ℓ enters this funnel. Then by Lemma 4, ℓ hits $C_{right}(v_{i-1})$; see Fig. 3(right).

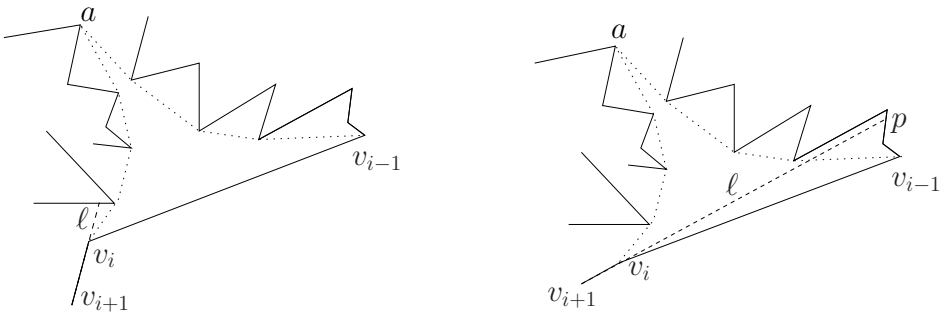

Figure 3: Cutting the edges of a critical chain.

We now count the total number of cuts used to make free the edges of C_C. Let x be the number of convex vertices in C_C. If $x = 1$, then we have used two cuts in total for two edges of C_C. Therefore, $k - j = 2$ and the lemma holds. For $x \geq 2$, we have used two cuts for the very first edge of C_C, which is (v_j, v_{j+1}). Then for each edge (v_i, v_{i+1}), for $j + 1 \leq i \leq k - 1$, we have used two cuts if v_i is reflex and one cut if v_i is convex. Since $x \geq 2$, we have used a total of $2((k - j) - x) + x = 2(k - j) - x \leq 2(k - j - 1)$ cuts.

The number of internal vertices in C_C is $k - j - 1$. Since two critical chains are edge-disjoint, they do not share any internal vertex. So we have $\sum_{C_C} 2(k - j - 1) = 2t_3$. Therefore, over all critical chains, we need at most $2t_3$ cuts. □

Algorithm analysis. The minimum enclosing rectangle K can be easily computed in $O(m)$ time. If P is a pocket with t vertices, a triangulation of P is first computed in $O(t)$ time [8]. Then the shortest path tree T can be computed in $O(t)$ time [14]; see also [3, 13]. In preparation for computing the cuts, P is preprocessed in $O(t)$ time, so that ray shooting queries can be answered

in $O(\log t)$ time per query [13, p. 267]. Obviously the union of the edges of P and T make a planar straight line graph G with no more than $2t$ edges (segments). The Voronoi diagram of G, denoted $\mathrm{Vor}(G)$, can be computed $O(t \log t)$ time [4]. As a planar graph, its complexity is linear in t. When making a pair of cuts along the edge (u, v) as in Lemma 1, the Voronoi diagram is used for choosing suitable points where the two cuts are initiated and where they meet. The goal is to ensure that the cuts do not intersect the interior of Q.

After the above preprocessing, Phase 1 can be executed in $O(t)$ time by traversing T and $\mathrm{Vor}(G)$ and scanning (in part) the boundary of P. Similarly, computing edge extensions from reflex vertices of critical chains in Phase 2 requires answering not more than t ray shooting queries and traversing $\mathrm{Vor}(P)$. Hence Phase 2 can be also executed in $O(t \log t)$ time. It follows that cutting out P takes $O(t \log t)$ time. By summing up over all pockets P of Q, and adding the time to compute K, we conclude that Q can be cut out from R in $O(m + \sum_P t \log t) = O(m + n \log n)$ total time.

Remark. The algorithm we have presented can be slightly modified so that the number of cuts made is reduced to $2n - h - p$ cuts, where p is the number of pockets of P. To this end we eliminate the cuts along the convex hull edges representing pocket lids. If (s, r) is such an edge, to make s and r free (which were earlier made free by the (s, r)-cut), we extend the two cuts along the two edges adjacent to s and r respectively, up to the boundary of K. That is, for each pocket, these two cuts start from the boundary of K.

References

[1] S. I. Ahmed, M. A. Islam, and M. Hasan. Cutting a cornered convex polygon out of a circle. *Journal of Computers*, 5(1):4–11, 2010.

[2] S. I. Ahmed, M. Hasan, and M. A. Islam. Cutting a convex polyhedron out of a sphere. *Graphs and Combinatorics*, 27(3):307–319, 2011.

[3] T. Asano, S. Ghosh, and T. Shermer. Visibility in the plane. In *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia (eds.), pp. 829–876, Elsevier Science, Amsterdam, 2000.

[4] F. Aurenhammer and R. Klein. Voronoi diagrams. In *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia (eds.), pp. 201–290, Elsevier Science, Amsterdam, 2000.

[5] S. Bereg, O. Dăescu, and M. Jiang. A PTAS for cutting out polygons with lines. *Algorithmica*, 53:157–171, 2009.

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

[7] R. Chandrasekaran, O. Dăescu, and J. Luo. Cutting out polygons. In *Proceedings of the 17th Canadian Conference on Computational Geometry*, pp. 183–186, 2005.

[8] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6:485–524, 1991.

[9] O. Dăescu and J. Luo. Cutting out polygons with lines and rays. *International Journal of Computational Geometry and Applications*, 16:227–248, 2006.

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

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

[12] A. Dumitrescu. The cost of cutting out convex n-gons. *Discrete Applied Mathematics*, 143:353–358, 2004.

[13] S. K. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.

[14] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[15] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.

[16] M. H. Overmars and E. Welzl. The complexity of cutting paper. In *Proceedings of the 1st Annual ACM Symposium on Computational Geometry*, pp. 316–321, 1985.

[17] X. Tan. Approximation algorithms for cutting out polygons with lines and rays. In *Proceedings of the 11th Annual International Computing and Combinatorics Conference*, Vol. 3595 of *LNCS*, pp. 534–543, Springer Verlag, 2005.