Stock cutting to minimize cutting length

J. Bhadury and R. Chandrasekaran

J. Bhadury and R. Chandrasekaran, (1996). "Stock Cutting to Minimize Cutting Length", *European Journal of Operational Research*, Vol. 88, pp 69-87. doi:10.1016/0377-2217(94)00160-X

Made available courtesy of Elsevier: http://www.elsevier.com/

Abstract:

In this paper we investigate the following problem: Given two convex P_{in} and P_{out}, where P_{in} is completely contained in P_{out}, we wish to find a sequence of 'guillotine cuts' to cut out P_{in}, from P_{out} such that the total length of the cutting sequence is minimized. This problem has applications in stock cutting where a particular shape or design (in this case the polygon P_{in}) needs to be cut out of a given piece of parent material (the polygon P_{out}) using only guillotine cuts and where it is desired to minimize the cutting sequence length to improve the cutting time required per piece. We first prove some properties of the optimal solution to the problem and then give an approximation scheme for the problem that, given an error range δ, produces a cutting sequence whose total length is at most δ more than that of the optimal cutting sequence. Then it is shown that this problem has optimal solutions that lie in the algebraic extension of the field that the input data belongs to — hence due to this algebraic nature of the problem, an approximation scheme is the best that can be achieved. Extensions of these results are also studied in the case where the polygons P_{in} and P_{out} are non-convex.

Keywords: Cutting stock; Production; Dynamic programming; Approximation scheme

1. Introduction

The general area of stock cutting, where a given shape or design is required to be cut out of parent material, has been the source of many problems of both theoretical and practical interest and hence the focus of considerable work done in both OR/OM and computer science communities. Most of the existing literature on stock cutting, has emphasised minimizing the amount of parent material wasted when such cutting is done — for example Gilmore and Gomory [7,8], Dori and Ben Assat [6], Aggarwal et al. [1], Christofides and Whitlock [5], and Venkateswarlu and Martyn [12]. However, another parameter of interest in these problems is the total length of the cutting sequence and cutting strategies that minimize this total length. The major implication of minimizing the cutting sequence length is in minimizing the total cutting time required, since the cutting sequence length is also a surrogate for the total time taken by the cutting tool to cut out the required shape or design. Furthermore, the total length of a cutting sequence is a measure of the wear and tear of the cutting tool, when the cutting tool is physical, and of the total cutting material used when the cutting tool is non-physical (as for example, a flame torch or laser beam). Thus minimization of the cutting sequence length might lead to cutting strategies that minimize tool wear in the first case and consumption in the second one. Potential benefits of minimizing cutting sequence length may therefore include, improving speed and throughput of the cutting process, minimizing tool wear and decreasing consumption. It is this problem of minimizing the cutting sequence length that we address in this paper.

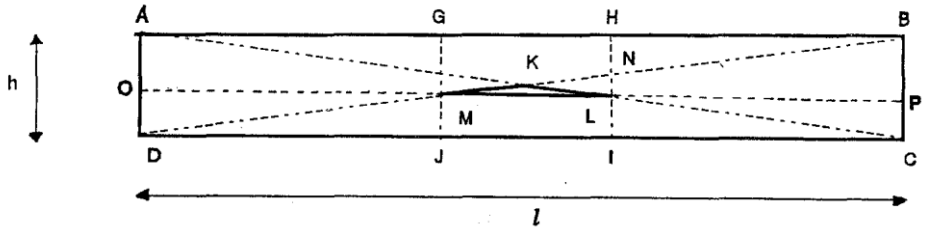

Fig. 1.

This paper is divided into five sections. In the present section, we have presented the problem, its motivation and applications, and discussed related literature. In the second section we state and prove some properties of the problem. Section 3 describes the polynomial approximation scheme suggested for the general problem. In Section 4 we prove some existence results and show the algebraic nature of the problem. We also show how to minimize the length of a given cutting sequence in a special case. Section 5 discusses the extensions of these results to the case where P_{in} and P_{out} are non-convex. The model of computation used in this paper is the usual random access machine (RAM) which can allow simple arithmetic operations like +, −, /, *, find roots of a polynomial with a fixed degree, etc. to be performed in unit time. We also assume that it can perform infinite precision arithmetic. These assumptions about our model of computation then allow us to claim that the various parameters required by the algorithms in this paper, namely the angles, slopes, trigonometric ratios, etc. can be computed with infinite precision and thus compared meaningfully.

2. Definitions and basic results

In this section we first define some terms and then state and prove some results about the nature of the optimal cutting sequence.

- It is assumed that P_{in} and P_{out} have n and m vertices respectively.
- It is assumed that the only cuts permitted are guillotine cuts. Two cuts are said to intersect each other *terminally* if they intersect on the boundary of P_{out}
- A *cutting sequence* refers to a particular sequence of cuts that cut out P_{in} from P_{out}. C is used to denote any given cutting sequence, and it is represented by $C = \{[1], [2], \ldots, [N]\}$ where [i] is the i-th cut in C. The optimal cutting sequence is denoted by C^* — a proof of its existence for any given instance of this problem is in Section 4 and all discussion until then will assume that C^* exists.
- S is used to denote any given set of cuts. It is obvious that any arbitrary instance of S may or may not contain a cutting sequence C, but we will assume in this paper that whenever a set S of cuts is given, it contains atleast one cutting sequence. S^* is used to denote the cutting sequence in S with the minimum total length. Hence, if we are restricted to cut out P_{in} only using the cuts in S, the optimal cutting sequence is S^*.

We also frequently use the term 'left' and 'right' of a cut; these are defined assuming that P_{in} lies below this cut.

2.1. *Properties of the optimal cutting sequence*

Lemma 1. *It can be assumed that all cuts in C^* touch P_{in}.*

Proof. The proof is by contradiction. Suppose that this is not the case and in the optimal cutting sequence C^*, there is a cut, which we denote by k, that does not touch P_{in}. Assume, without loss of generality, that P_{in} lies below k. We now establish that the cut k can be moved parallel to itself either towards P_{in} or away from it to produce another cutting sequence that is better that C^* − thus contradicting its optimality. This is proved by considering several cases obtained by conditioning on the number of vertices of P_{out} that k intersects: *Case (i):* the number of vertices is 0; *(ii):* the number of vertices is 1; and the number of vertices is 2.

Case (i): Consider cut k without any of the cuts that it may intersect terminally. Denote by $f(\varepsilon)$, the function that represents the change in the length of C^*, when cut k is moved by a small amount ε parallel to itself. Then it can be readily verified that in this case $f(\varepsilon)$ is a linear function of ε. In this proof we will distinguish between the two conditions when the slope of $f(\varepsilon)$ is non-zero (we refer to $f(\varepsilon)$ as *non-degenerate* if this is true) and when the slope of $f(\varepsilon)$ is zero (when we refer to $f(\varepsilon)$ as *degenerate*).

We begin by considering first the case when $f(\varepsilon)$ is non-degenerate. In this case, movement along one of the two directions (towards P_{in} or away from it) will decrease the total length of C^* — thus producing another cutting sequence that is strictly better than C^* and contradicting its optimality. If k intersects any cut terminally (say cut j) then it can be seen that the presence of cut j can have either one of the following two effects − either its presence adds to the savings obtained by moving cut k (for example, if cut j is below cut k and it saves to move cut k towards P_{in}) or its presence does not affect the savings obtained by moving cut k (for example, when cut j is below cut k and it saves to move cut k away from P_{in}). In either case, it saves to move cut k in the same direction as it would have had cut j been absent. Therefore, when case (i) occurs and the slope of $f(\varepsilon)$ is non-zero, if the cut k does not touch P_{in}, then the given cutting sequence C^* can not be optimal — a contradiction.

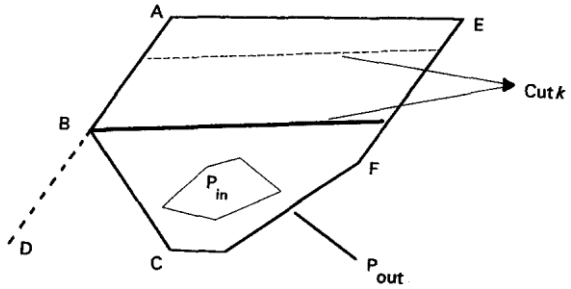

Fig. 2.

If $f(s)$ is degenerate, its slope is zero; for example, this may happen when the two edges of P_{out} that cut k intersects are parallel; this is illustrated in Fig. 2 where AB and EF are the two parallel edges and the initial position of k is shown by the dotted line. Consider moving k towards P_{in} until case (ii) happens (i.e., it encounters a vertex of P_{out} on one of its ends) or case (iii) happens (i.e., it encounters two vertices of P_{out} − one on each of its ends). Fig. 2 illustrates this situation when k is moved towards P_{in} from its initial position to its final position, and case (ii) occurs with it encountering vertex B on the left. Since the slope of $f(\varepsilon)$ is zero, this produces another cutting sequence whose total length is no more than that of C^*. Now if k is moved by a small amount further towards P_{in}, its own length, and therefore the total length of C^*, is guaranteed to decrease. This is so because of the convexity of P_{out}, which guarantees that the internal angle at B is less than π. It is apparent that this decrease in the length of k is even greater if cut k intersects other cuts − terminally or otherwise. Hence with k moved slightly beyond its final position, we can get another cutting sequence whose total length is strictly less than that of C^* − again a contradiction.

Case (ii): Without loss of generality, assume that k intersects one vertex of P_{out} on its left — as illustrated in Fig. 2, where we now assume that the initial position of k is shown by the thick line where it encounters B on its left. To motivate the proof, consider the situation where the edge BC of P_{out} is collinear with the edge AB, as shown by the line BD in this figure — it is clear that then we have case (i). Define $f(\varepsilon)$ as the change in the length of C^* when k is moved parallel to itself by a small amount s, assuming that BC is collinear with AB. If $f(\varepsilon)$ is non-degenerate, then it saves to move k either towards P_{in} or away from it and thus produces a strictly better cutting sequence than C *. If moving towards P_{in} saves when BC is collinear with AB, it will save even more when BC is not collinear with AB; since, by convexity of P_{out}, the internal angle at B is less than π. If, however, it saves to move away from P_{in} with BC collinear with AB, then the presence of BC does not affect the savings and hence it would not matter even if BC were not collinear with AB. If k intersects any cut terminally then an argument similar to that used in case (i) above will show that this result still holds as long as the slope of $f(\varepsilon)$ is non-zero. Thus, when case (ii) occurs and $f(\varepsilon)$ is non-degenerate, by moving k slightly from

its initial position, it is possible to produce another cutting sequence whose total length is less than that of C^* – a contradiction.

Now consider the case when $f(\varepsilon)$ is degenerate — in this case, the same proof as that in case (i), that exploits the convexity of P_{out}, will show that moving k slightly towards P_{in} is guaranteed to provide another cutting sequence whose total length is less than that of C^* – another contradiction.

The proof for case (iii) can be made along the same lines as that of case (ii) by defining $f(\varepsilon)$ similarly and conditioning on whether it is degenerate or not. ■

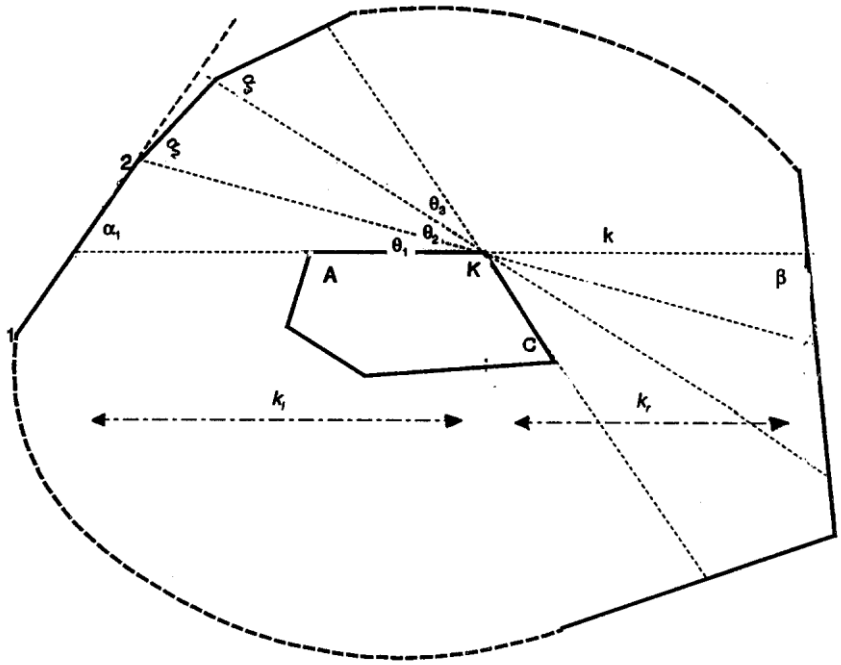

Fig. 3.

Lemma 2. *It can be assumed that C^* has $O(n)$ cuts.*

Proof. This has been shown in Overmars and Welzl [10]. There it is shown that no more that $5n$ cuts are necessary in any cutting sequence to cut out P_{in} from P_{out}, and hence it is true of C^* too. ■

The statement of Lemma 1 and a brief sketch of a proof also appears in Overmars and Welzl [10], but we have chosen to prove it here in a different way for completeness (the proof there does not discuss terminal cuts and cases (ii) and (iii)) and also because the proof here can be very easily modified to a corresponding result (which is Lemma 7 in this paper) in the non-convex case. Based on Lemmas 1 and 2 we will assume, for the rest of the paper that every cut in C^* touches P_{in} and that it has at most $5n$ cuts.

2.2. Rotation of one cut

As shown in Fig. 1, C^* may have one or more cuts that are not along the sides of P_{in}. Hence we now study the behaviour of cuts as they rotate around one of the (at most) two vertices of P_{out} they intersect. Refer to Fig. 3 where a cut k is assumed to be rotated clockwise about a vertex K from its initial position when it is flush with the edge AK of P_{in} to the final position where it is flush with the edge KC. Denote by θ the angle of rotation of k and let θ_1, θ_2, etc. be the angles at which k intersects the successive vertices of P_{out}. The section of k to the left of K and its length as a function of rotation of k by θ is denoted by k_l and $k_l(\theta)$ respectively; the corresponding notation for the section of k to the right of K are k_r and $k_r(\theta)$ respectively.

Lemma 3. *$k(\theta)$ is a piecewise convex function of θ and at the angles where a cut intersects a vertex of P_{out}, the right derivative of $k(\theta)$ is less than its left derivative.*

Proof. It can be verified from trigonometry that the function $k(\theta)$, when $0 \leq \theta \leq \theta_1$, is

$$k_l(\theta) = k_l(\theta) + k_r(\theta) = \frac{k_l(0)\sin\alpha_1}{\sin(\alpha_1+\theta)} + \frac{k_r(0)\sin\beta}{\sin(\beta+\theta)} \qquad (1)$$

$\sin\alpha_1$, $\sin\beta$ and the values of $k_l(0)$ and $k_r(0)$ are constants that can be calculated from the input in polynomial time under the assumed model of computation. Because $\alpha_1 + \theta$ and $\beta + \theta$ are always less than π in our range of interest, for a given value of α_1 and β, cosec($\alpha_1 + \theta$) and cosec($\beta + \theta$) are convex functions of θ. Hence $k_l(\theta)$ and $k_r(\theta)$ and therefore $k(\theta)$ are all convex functions of θ, proving the first property.

To prove the second property, consider k_l at the angles θ_1 and θ_2 as shown in Fig. 3. It can be verified that for $0 \leq \theta \leq \theta_1$,

$k_l(\theta) = k_l(\theta)\sin(\alpha_1)/\sin(\alpha_1+\theta)$

and for $\theta_1 \leq \theta \leq \theta_2$,

$$k_l(\theta) = \frac{k_l(0)\sin\alpha_1}{\sin(\alpha_1+\theta)} + \frac{k_r(0)\sin\beta}{\sin(\beta+\theta)}.$$

where the last expression is obtained by noting that $k_l(\theta_1) = [k_l(0)\sin\alpha_1]/\sin(\alpha_1+\theta_1)$. To study the derivatives of the function $k_l(\theta)$, note that

$$k_l'(\theta) = \begin{cases} -[k_l\sin\alpha_1\cos(\alpha_1+\theta)]/\sin^2(\alpha_1+\theta_1) & for\ 0 \leq \theta \leq \theta_1, \\ -\left[\frac{k_l(0)\sin\alpha_1}{\sin(\alpha_2+\theta-\theta_1}\right][\sin\alpha_2\cos(\alpha_2+\theta-\theta_1)]/\sin^2(\alpha_2+\theta-\theta_1) & for\ \theta_1 \leq \theta \leq \theta_2. \end{cases}$$

At $\theta = \theta_1$, the values of the two derivatives are

$$k_l'(\theta_1^-) = -k_l(0)\sin\alpha_1\cos(\alpha_1+\theta_1)/\sin^2(\alpha_1+\theta_1)$$

and

$$k_l'(\theta_1^+) = -[k_l(0)\sin\alpha_1\cos\alpha_2]/[\sin(\alpha_1+\theta_1)\sin\alpha_2].$$

$k_l'(\theta_1^-) - k_l'(\theta_1^+)$ is equal to

$$\frac{k_1(0)\ \sin\alpha_1}{\sin(\alpha_1+\theta_1)} * [\cot\alpha_2 - \cot(\alpha_1+\theta_1)] \qquad (2)$$

Since α_2 and $\alpha_1 + \theta_1$ are less than π in our range of interest, cotangent is a monotonically decreasing function in this range. By convexity of P_{out} we have $\alpha_2 \leq \alpha_1 + \theta_1$ which implies that for our range of interest $\cot(\alpha_2) \geq \cot(\alpha_1+\theta_1)$ and all the other terms in (2) above are non-negative. This implies that (2) is non-negative. The second property is then proved by noting that the same property is true for $k_r(\theta)$ too. ■

This behaviour of the derivatives at the critical angles can also be explained geometrically. Referring to Fig. 3, we see that had vertex 2 been absent, the edge (1, 2) of P_{out} would have been extended as shown along the line. But the existence of this vertex and the convexity of P_{out} ensures that the length of k_l immediately after θ_1 will be less than what it would have been had vertex 2 been absent. Note that $k(\theta)$ is a piece-wise convex function when α_1 and β are constants − it can be verified that it may not be convex in α_1 or β or piecewise convex in all three variables.

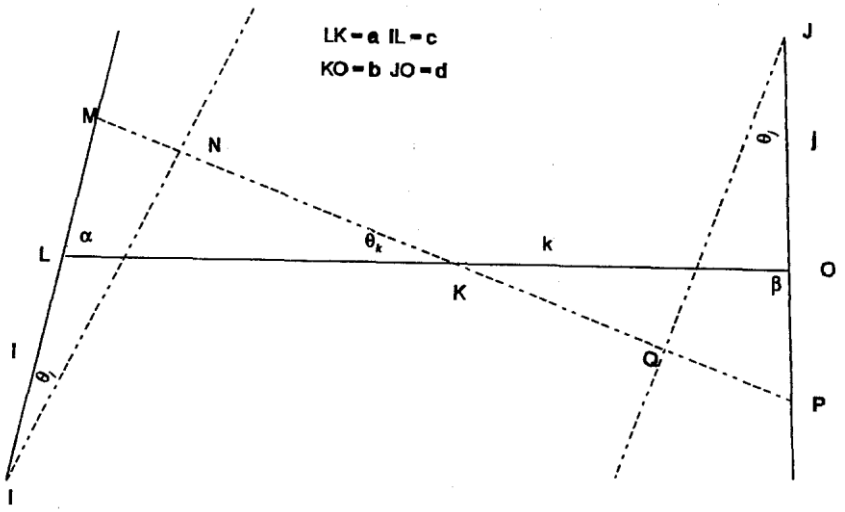

Fig. 4.

2.3. Rotation of multiple cuts

We will now study the change in the length of the cut k when it is rotated simultaneously with two other cuts that it intersects, one on the left of K (denoted by cut i) and one on the right (denoted by cut j). Some more notation needs to be defined as follows.

- Denote by I, J and K the vertices of P_{in} about which cuts i, j and k are rotated.
- Let α denote the distance of K from the intersection of cuts k and i and b, the distance of the point K from the intersection of cuts k and j; similarly, let c denote the distance of I from the intersection of cuts k and i and d, the distance of the point J from the intersection of cuts k and j.
- Let the cuts i, j and k be rotated by be rotated by θ_i, θ_j and θ_k, respectively, and let $\in_i$, $\in_j$ and $\in_k$ be the respective maximum rotations possible (in a clockwise direction) before it encounters a vertex of P_{out} or becomes flush with another edge of P_{in}.

The vertices I and J can each be either above or below K — which gives rise to four cases: Case I: I below K; case II: I above K; case III: J below K, and case IV: J above K. Shown in Fig. 4 is an instance where Case I is applicable for k_l and Case IV for k_r. The change in the length of k_l (respectively, k_r) as a function of θ_i, θ_k (respectively, θ_p θ_k) in cases I and II (respectively, cases III and IV) is denoted by $k_l(\theta_i, \theta_k)$ (respectively, $k_r(\theta_j, \theta_k)$) and given in Table 1.

Table 1

Case I: $a \sin\alpha/\sin(\alpha+\theta_k) - (\sin\theta_i/\sin(\alpha+\theta_k-\theta_i)[c + a\sin\theta_k/\sin(\alpha+\theta_k)])$	(3)
Case II: $a \sin\alpha/\sin(\alpha+\theta_k) + (\sin\theta_i/\sin(\alpha+\theta_k-\theta_i)[c - a\sin\theta_k/\sin(\alpha+\theta_k)])$	(4)
Case III: $b \sin\beta/\sin(\beta+\theta_k) + (\sin\theta_j/\sin(\beta+\theta_k-\theta_j)[d - b\sin\theta_k/\sin(\beta+\theta_k)])$	(5)
Case IV: $b \sin\beta/\sin(\beta+\theta_k) - (\sin\theta_j/\sin(\beta+\theta_k-\theta_j)[d + b\sin\theta_k/\sin(\beta+\theta_k)])$	(6)

We will only show the derivation of Eq. (3) for $k_l(\theta_i, \theta_k)$ assuming case I is applicable, as shown in Fig. 4. Here the length of k_l after rotation by θ_k is shown in the figure to be equal to KM − MN. Note that $\angle\text{LMK} = \pi - (\alpha + \theta_k)$ and that $\angle\text{MNI} = \pi - (\theta_i + \angle\text{LMK}) = (\alpha + \theta_k - \theta_i)$. Applying the law of sines to the triangle KLM we obtain

$$\frac{\text{LM}}{\sin\angle\text{LKM}} = \frac{\text{KL}}{\sin\angle\text{LMK}} \Rightarrow \text{LM} = \alpha * \frac{\sin\theta_k}{\sin(\alpha+\theta_k)}$$

Applying the law of sines similarly to triangle MNI, we obtain

$$\mathrm{MN} = \frac{\sin\theta_i}{\sin(\alpha + \theta_k - \theta_i)}\left[\mathrm{c} + \frac{\alpha \sin\theta_k}{\sin(\alpha + \theta_k)}\right].$$

From Eq. (1) applied to cut *k*, it can be seen that

$$\mathrm{KM} = \alpha * [\sin\alpha / \sin(\alpha + \theta_k)].$$

Using the two we obtain that KM - MN is

$$k_l(\theta_i, \theta_k) = \frac{\alpha \sin\alpha}{\sin(\alpha + \theta_k)} - \frac{\sin\theta_i}{\sin(\alpha + \theta_k - \theta_i)}\left[c + \frac{\alpha \sin\theta_k}{\sin(\alpha + \theta_k)}\right]$$

The expressions for cases II, III and IV can be derived similarly. It can then be verified that these equations are not guaranteed to be convex/concave for all values of θ_i, θ_j and θ_k in our range of interest.

Letting $\theta_k = 0$ in Eqs. (3)-(6) gives the expressions for the change in length of cut k_l when cut *i* is rotated by θ_i and the change in k_r when cut *j* is rotated by θ_j. The expressions are given in Table 2.

It can be verified that the equations (8)-(11) in Table 2, for $k_l(\theta_i)$ (respectively, $k_r(\theta_j)$) are not guaranteed to be convex or concave functions of θ_i (respectively, θ_j) for all values of θ_i (respectively, θ_j) in our range of interest.

2.4. An algorithm for a special case of the problem and its extension

Consider a special case of the problem where we are given a set *S*, consisting of *n* guillotine cuts, each of which is along an edge of P_{in}, and it is desired to find S^*. This special case of the problem is considered in Overmars and Welzl [10] and the algorithm we describe is due to them. It will become apparent that it is possible to extend this algorithm in a simple way to solve the general problem. Assume that the cuts in *S* (or, equivalently, the edges of P_{in} extended to intersect P_{out}) are numbered clockwise 1 through *n*. Suppose that the cuts *i* and *j* (where $i < j$) are made at any two successive steps of the cutting sequence. This cuts P_{in} into two sub-polygons; $P_{in}^{(i,j)}$ and $P_{in}^{(j,i)}$. The former comprises of all edges from *i* to *j* and the latter of all edges from *j* to *i*. Similarly these cuts also divide P_{out} into two sub-polygons, which we denote by $P_{out}^{(i,j)}$ and $P_{out}^{(j,i)}$ respectively. These sub-polygons are shown in Fig. 5. This results in two independent subproblems of the original problem; namely, cutting out $P_{in}^{(i,j)}$ from $P_{out}^{(i,j)}$ and cutting out $P_{in}^{(j,i)}$ from $P_{out}^{(j,i)}$, respectively. Furthermore, these two problems are defined completely by the cuts *i* and *j*. The length of the optimal cutting sequence for the original is equal to the sum of the lengths of the optimal cutting sequences for cutting out $P_{in}^{(i,j)}$ and $P_{in}^{(j,i)}$ from $P_{out}^{(i,j)}$ and $P_{out}^{(j,i)}$, respectively, plus the sum of the lengths of cuts *i* and *j*. Moreover, the optimal answer to these subproblems is independent of the order of cuts made prior to *i* and *j*. This makes the problem amenable to dynamic programming.

Table 2

Case I: $k_l(\theta_i) = a - [c \sin\theta_i / \sin(\alpha - \theta_i)]$	(8)
Case II: $k_l(\theta_i) = a + [c \sin\theta_i / \sin(\alpha - \theta_i)]$	(9)
Case III: $k_r(\theta_j) = b + [d \sin\theta_j / \sin(\beta - \theta_j)]$	(10)
Case IV: $k_r(\theta_j) = b - [d \sin\theta_j / \sin(\beta - \theta_j)]$	(11)

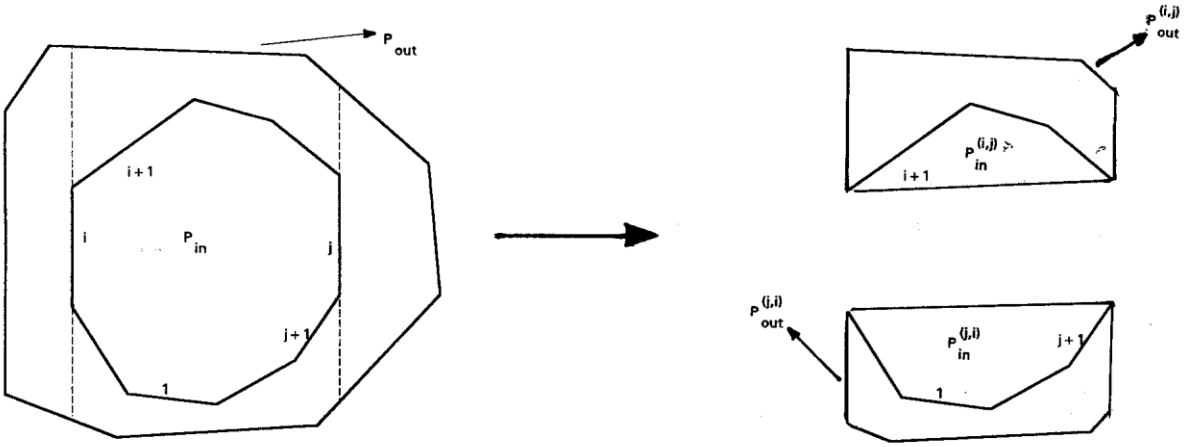

Fig. 5.

First we calculate the best way to cut out $P_{\text{in}}^{(i,j)}$ in from $P_{\text{out}}^{(i,j)}$ when $j = i + 1$, $\forall 1 \leq i \leq n$, which is simple. Assume, inductively, that we have calculated the best way to cut out $P_{\text{in}}^{(i,j)}$ from $P_{\text{out}}^{(i,j)}$ when $j = i + p$, $\forall 1 \leq i \leq n$. Then the optimal answer to a subproblem with $j = i + p + 1$ for any given i is found by deciding which of the cuts between i and j should be made first and the optimal answer to the resulting subproblems would already have been computed at the previous step and need not be computed again. This implies that $O(n)$ work is required at every stage of the algorithm. The number of stages is $O(n^2)$ since a stage (or equivalently, a polygon pair) is completely defined by a pair of cuts. Therefore the overall complexity of the algorithm is $O(n^3)$. A mathematical representation of the algorithm is provided in the Appendix.

Having described the dynamic programming algorithm due to Overmars and Welzl for the special case, it is worthwhile to note that cutting only along edges of P_{in} can be very expensive. For example in Fig. 1, any cutting sequence that cuts only along edges of P_{in} will have to make either one of the cuts DB, AC or OP first. If P_{out} is 'long and thin', then the length of this cutting sequence will be much more than that of the cutting sequence C2 in Section 1 (and hence much more than the length of C^*). Hence there is a need to compute the optimal cutting sequence C^*.

It will now be shown that it is possible to generalise the dynamic programming algorithm discussed above to the case where not all the cuts are along edges of P_{in} in a very simple way. The reason why the algorithm takes $O(n^3)$ time is because after two cuts i and j are made the problem of cutting $P_{\text{in}}^{(i,j)}$ from $P_{\text{out}}^{(i,j)}$ and $P_{\text{in}}^{(j,i)}$ from $P_{\text{out}}^{(j,i)}$ are independent of each other and the order of the cuts made prior to i and j. Hence *this algorithm will work even if the prespecified set of cuts are not restricted to be along the sides of* P_{in} *but only required to touch it* as this property still holds. Hence, the next lemma:

Lemma 4. *Given a set of S of p cuts containing at least one cutting sequence and such that all cuts in S touch* P_{in}*, it is possible to find* S^* *by a dynamic programming algorithm in* $O(p^3)$ *time.*

Proof. Obvious from the above discussion. ■

Hence by Lemma 4, if one gives a prespecified set of cuts all of which touch P_{in} (even though some may not be along edges of P_{in}) and the set contains at least one cutting sequence, it is possible to apply the same dynamic programming algorithm to get the best cutting sequence from this set of cuts. This enables us to use this algorithm for an approximation scheme for the general case of the problem.

Table 3

Algorithm-*Compute* $C(\delta)$
Input: Error range δ, and the polygons P_{in} and P_{out}.
Output: A cutting sequence $C(\delta)$ whose total length is at most δ more than that of C^*.

begin
Step 1. **for** each edge of P_{in} **do**
{
Extend this edge of P_{in} to intersect P_{out} on both ends, thus giving a cut that is along this edge of P_{in}.
}
Step 2. **for** each vertex of P_{out} **do**
{
Draw the two tangents (the clockwise and the anticlockwise one) from this vertex of P_{out} to P_{in}, thus adding two cuts for this vertex.
}
Step 3. **for** each cut obtained by Steps 1 and 2 above, **do**
{
Calculate the angle θ_k by the procedure discussed below. Between cut k and the next cut after cut k (clockwise) introduce the smallest number of cuts necessary so that no two cuts are separated by more than θ_k.
}
Step 4. With the cuts introduced after Step 3, apply the dynamic programming algorithm (referred to in Lemma 4) on the problem. The optimal cutting sequence produced by it is the required cutting sequence $C(\delta)$ for the given problem.
end

3. An approximation scheme for the general case of the problem

In this section we describe a method in which, given any arbitrary error range $\delta > 0$, we will give a cutting sequence $C(\delta)$ (in which all cuts touch P_{in} which is guaranteed to have a length that is at most δ more than that of C^*. Furthermore the time taken by this approximation scheme is polynomial in δ and the encoding length of the input data in unary. It has already been shown that given S, a set of cuts where all cuts touch P_{in}, the dynamic programming algorithm of the previous section can be used to obtain S^*. It is intuitively clear that as more cuts are allowed in S at every vertex of P_{in}, the algorithm produces answers that are closer to the optimal answer C^*. The algorithm, *Algorithm-Compute* $C(\delta)$, is given in Table 3.

Step 1 of **Algorithm**-*Compute* $C(\delta)$ extends all edges of P_{in} to P_{out}; hence we have $O(n)$ cuts after this step is over. Step 2 then adds two cuts for every vertex of P_{out}, as shown in Fig. 6. Thus there are $O(n + m)$ cuts at the end of Steps 1 and 2. At the end of Step 2, if any cut is rotated clockwise about a vertex of P_{in}, from its initial position to its final position (where it becomes flush with the next cut clockwise), it always intersects the same pair of edges of P_{out}, in between — hence by these two steps one of the combinatorial features of this problem, that arises due to cuts passing through different vertices of is removed.

Consider the set S of cuts obtained after Steps 1 and 2 are over. If C^* has any cut that is different from the cuts in S, then it can be assumed that all such cuts in C^* are wedged between two successive cuts in S. Any cut in C^* that is wedged between two such cuts can be assumed to have been obtained by clockwise rotation of one of the cuts in S around a particular vertex of P_{in}. Hence, to determine θ_k for a cut k note that if a cutting sequence $C(\delta)$ needs to be found, then the set S must have enough cuts to ensure that if any cutting sequence from S is considered, then the total length of this cutting sequence under consideration should not decrease by more than δ by rotation of the cuts in it. Given any triple of cuts (i, j, k) in S where cut k intersects cut i on the left and j on the right, consider the maximum reduction in length of k possible by the rotation of i, j and k. If it can be ensured that this change is less than $\delta(5n)$ for any triple (i, j, k), then S^* will be the cutting sequence $C(\delta)$. This will form the basis of computing θ_k.

Computing θ_k for cut k: For this cut k, consider two cuts i and j where cut k intersects cut i (respectively, j) on the left (respectively, right) of the vertex of P_{in} about which k can be rotated clockwise. Eqs. (3)—(6) give the expressions for the change in length of k_l and k_r when all three cuts are rotated clockwise around their respective vertices. We will only show how to calculate θ_k assuming case I is applicable for k_l (as shown in Fig. 4) — the other cases are similar. In case I, the function denoting the change in length of k_l is given by $k_l(\theta_1, \theta_k)$ in Eq. (3). Let $U[\partial k_l/\partial\theta_k(U[\partial k_l/\partial\theta_i])$ be numbers such that

$$\left|\max_{\theta_l\theta_k}\left[\frac{\partial k_l(\theta_i\theta_k)}{\partial\theta_k}\right]\right|\left(or\left|\max_{\theta_i\theta_k}\left[\frac{\partial k_l(\theta_i\theta_k)}{\partial\theta_i}\right]\right|\right) \le U\left[\frac{\partial k_l}{\partial\theta_k}\right]\left(or\ U\left[\frac{\partial k_l}{\partial\theta_i}\right]\right)\ \forall\theta_i(\theta_k)\in[0,\in_i(\in_k)] \tag{12}$$

Define $U[\partial k_r/\partial\theta_k]$ ($U[\partial k_r/\partial\theta_j]$) similarly. Also let

$$U(k_l)=\max\left[U\frac{\partial k_l}{\partial\theta_k},U\frac{\partial k_l}{\partial\theta_i}\right],U(k_r)=\max\left[U\frac{\partial k_r}{\partial\theta_k},U\frac{\partial k_r}{\partial\theta_j}\right],$$

and

$$U(k)=\max[U(k_l),U(k_r)].$$

To find $U[\partial k_l/\partial\theta_k]$ it can be shown that

$$[\partial k_l(\theta_i,\theta_k)/\partial\theta_i]=\text{cosec}^2(\alpha+\theta_k-\theta_i)[c\sin(\alpha+\theta_k)-\alpha\sin\theta_k] \tag{13}$$

and

$$\begin{aligned}[\partial k_l(\theta_i,\theta_k)/\partial\theta_k]=&\ \alpha\sin\theta_i\cos\theta_k\,\text{cosec}(\alpha+\theta_k)\,\text{cosec}(\alpha+\theta_k-\theta_i)\\&-a\cos(\alpha+\theta_k)\,\text{cosec}^2(\alpha+\theta_k)[\sin\alpha+\sin\theta_i\sin\theta_k\,\text{cosec}(\alpha+\theta_k-\theta_i)\\&-\sin\theta_i\cos(\alpha+\theta_k-\text{cosec}^2(\alpha+\theta_k-\theta_i)[c+\alpha\sin\theta_k\,\text{cosec}(\alpha+\theta_k)].\end{aligned} \tag{14}$$

The angles θ_k, θ_i, $\alpha+\theta_k$ and $a+\theta_k-\theta_i$ are all non-negative and less than π in our range of interest. $U[\partial k_l/\partial\theta_k]$, $U[\partial k_l/\partial\theta i]$ can be computed by finding an upper bound for the r.h.s. of Eqs. (13) and (14). An upper bound for the value of $\text{cosec}(\alpha+\theta_k-\theta_i)$ can be obtained by computing the value of this function at the four extreme points $\theta_k=\theta_i=0$, $\theta_k(\theta_i)=\in_k(\in i)$, $\theta_i(\theta_k)=0$, and $\theta_i=\in_i$, $\theta_k\in_k$, taking the maximum of these, and denoting it as $\text{cosec}^*(\alpha+\theta_k-\theta_i)$. $\text{cosec}^*(a+\theta_k)$ is similarly defined for $\text{cosec}(\alpha+\theta_k)$. Unity can be used as an upper bound for cosine and sine functions. Doing so it can be see that one choice for these upper bounds is the following:

$$U[\partial k_l/\partial\theta_i]=(c+a)[\text{cosec}^*(\alpha+\theta_k-\theta_i)]^2 \tag{15}$$

and

$$\begin{aligned}U[\partial k_l/\partial\theta_k]=&\ a\,\text{cosec}^*(\alpha+\theta_k)\,\text{cosec}^*(\alpha+\theta_k-\theta_i)\\&+a[\text{cosec}^*(\alpha+\theta_k)]^2[1+\text{cosec}^*(\alpha+\theta_k-\theta_i)]\\&+[\text{cosec}^*(\alpha+\theta_k-\theta_k)]^2[c+a\,\text{cosec}^*(\alpha+\theta_k)].\end{aligned} \tag{16}$$

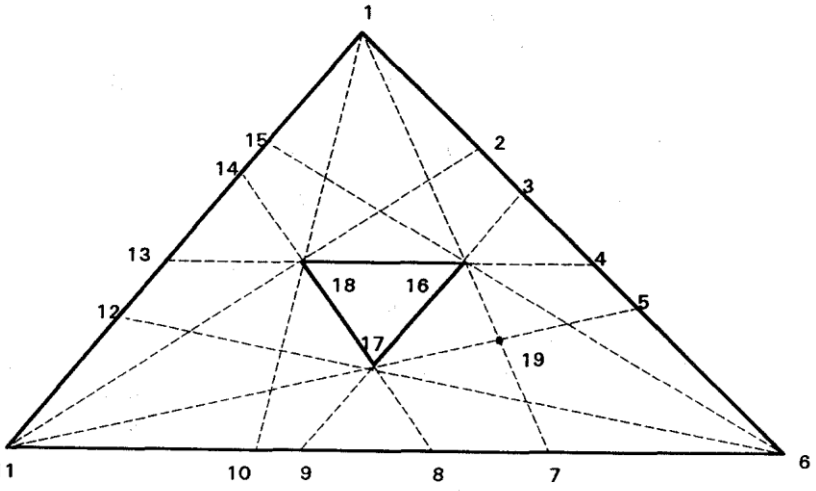

Fig. 6.

From (15) and (16) $U(k_l)$ can be found. By doing a similar calculation for k_r, $U(k_r)$ and hence $U(k)$ can be determined. Repeating this procedure to find $U(k)$ for each such pair of cuts i and j (where i intersects k on left and j on the right), and by taking their maximum, we can find the upper bound on the rate of change of the length of cut k; denote it as $U(k^*)$. Then, choosing $\theta_k = \delta/[5nU(k^*)]$ will suffice. That completes the procedure for calculating θ_k for this cut k. Note that for every cut k, $U(k)$ will always be finite (and hence θ_k will be non-zero) because for $U(k)$ to be infinite, two adjacent edges of P_{out} would have to be parallel, which is impossible. This guarantees that the approximation scheme will always work.

To show how θ_k is calculated for a given cut k, consider the case shown in Fig. 6, where we will assume that the x and y coordinates of the vertices 1, 6 and 11 of P_{out} are given by (9, 18), (16, 9) and (2, 9) respectively. The corresponding coordinates for the vertices 16, 17 and 18 of P_{in} are given by (11, 13), (9, 11) and (7, 13) respectively. Assume that all distances are in centimeters and angles in degrees. Consider the cut (5, 11) as shown in Fig. 6, which is rotated clockwise about vertex 17 of P_{in}. With the figure rotated so that P_{in}, is below the cut (5, 11), it can be verified that the other cuts that (5, 11) intersects are (1, 7), (6, 15) on the left and (2, 11), (1, 10) and (8, 14) on the right; it is obvious that cuts (3, 9) and (6, 12) need not be considered as their vertex of rotation is also 17. Consider the pair of cuts (5, 11) and (1, 7) with (5, 11) as cut k, rotated clockwise about vertex 17 and (1, 7) as cut i rotated about vertex 16. Then the point of intersection of these two cuts is the point 19 in the figure. It can be verified that a (given by line segment [17, 19]) is equal to 2.6 and c (given by [16, 19]) is 1.4. Furthermore a is given by the angle $\angle 7, 19, 11$ and is equal to 96.1°. When (5, 11) is rotated clockwise, the next clockwise cut that it becomes flush with is (6, 12) — and it can be verified that the angle between them, i.e. $\in_k$, is equal to 31°. Similarly, when (1, 7) is rotated clockwise the next cut it becomes flush with is (3, 9) and thus $\in_i$ is equal to 66.25°. From these it can be verified that cosec* $(\alpha + \theta_k)$ is 1.2538 and cosec * $(\alpha + \theta_k - \theta_i)$ is 2.0091; hence, by Eqs. (15) and (16), $U(\partial k_l/\partial\theta_i)$ is 0.2818 and $U(\partial k_l/\partial\theta_k)$ is 0.6572. Thus $U(k_l)$ is equal to 0.6572. By a similar analysis, it can be seen that the upper bounds (i.e., $U(k_l)$ or $U(k_r)$, as the case may be) obtained for intersections with the cuts (6, 15), (8, 14), (2, 11) and (1, 10) are 3.3856, 0.1954, 12.317 and 1.4409 respectively; thus $U(k^*)$ for the cut (5, 11) is the maximum of these, i.e., 12.317. Hence with δ equal to 10, we obtain θ_k equal to 0.054°. Table 4 contains details of these calculations and shows similar results for another cut, namely (1, 10).

To compute the time taken by Algorithm-Compute $C(\delta)$, it can be seen that Steps 1 and 2 can be performed in $O(n + m)$ time. In Step 3, for any cut k, θ_k can be found in polynomial time and its encoding length is also polynomial in the range of the input data and δ. This in turn guarantees that for every cut k, the number of additional cuts introduced at Step 3 is also polynomial in the same parameters. Since the dynamic programming algorithm that is performed in Step 4 takes time that is polynomial in the number of cuts allowed, the overall algorithm Algorithm-Compute $C(\delta)$ is polynomial in δ and the encoding length of the input data in unary encoding.

Table 4
Coordinates for the vertices of p_{in} and P_{out} in Fig. 6. All distances are in centimetres and angles in degrees.

Vertex No.	(x, y)
1	(9, 18)
6	(16, 9)
11	(2, 9)
16	(11, 13)
17	(9, 11)
18	(7, 13)

Detailed calculations for finding upper bounds

Cuts intersected by cut k above	Cut (5, 11) as cut k Rotated about vertex 17 of P_{in}					Cut (1, 10) as cut k Rotated about vertex 18 of P_{in}				
	(8, 14) as cut i. Rotated about vertex 18	(6, 15) as cut i. Rotated about vertex 16	(1, 7) as cut i. Rotated about vertex 16	(1,10) as cut j. Rotated about vertex 18	(2, 11) as cut j. Rotated about vertex 18	(5, 11) as cut i. Rotated about vertex 17	(6, 12) as cut i. Rotated about vertex 17	(4, 13) as cut i. Rotated about vertex 16	(6, 15) as cut j. Rotated about vertex 16	(1, 7) as cut j. Rotated about vertex 16
a/b	0	3.5	2.6	3.3	7.3	3.25	1.4	1.0	2.61	5.4
c/d	2.8	4.7	1.4	3.25	6.5	3.3	2.6	2.6	3.85	5.35
α/β	119°	126°	96.1°	52°	22.1°	128°	96.5°	112°	73.1°	43°
ϵ_i/ϵ_j	66.25°	29.5°	66.25°	29.8°	38°	31°	30°	38.5°	29.5°	66.25°
ϵ_k	31°	31°	31°	31°	31°	29.8°	29.8°	29.8°	29.8°	29.8°
$U(k_l)/U(k_r)$	0.1954	3.3856	0.6572	1.4409	12.317	3.301	0.2385	0.3575	0.4369	20.986
	$U(k^*) = 12.317$					$U(k^*) = 20.986$				

4. Existence and algebraic nature of C^* and minimizing a given cutting sequence

All results in the previous sections have assumed the existence of the optimal solution C^*; a proof of that will now be given in the next lemma. In this section we also show the algebraic nature of this problem and state and prove some more existence results. Finally, we give a polynomial procedure to minimize the length of a given cutting sequence by rotating one, or multiple but non-intersecting cuts.

4.1. Existence results

Lemma 5. *For any given configuration of P_{in} and P_{out} it can be assumed that C * exists. Furthermore, there exists an optimal solution to this problem that is algebraic.*

Proof. For a given pair of polygons P_{in} and P_{out}, consider the set S of all cuts after Step 1 of **Algorithm-***Compute* $C(\delta)$ is done. As discussed above, it can be assumed that if C * exists then it can be obtained by rotating clockwise one or more cuts of S around the respective vertices of P_{in}; i.e., by choosing a given cutting sequence in S and minimizing its length by rotating the cuts in this cutting sequence. S contains an exponential, but finite number of cutting sequences — let $C = \{[1],[2],\ldots,[N]\}$ be one such cutting sequence in S. To minimize the length of C, we need to rotate (assume, without loss of generality, clockwise) the cuts in C. For cut $[i]$ let $\theta_{[i]}$ denote the amount of rotation and $\in_{[i]}$ the maximum rotation of cut [i] before it becomes flush with the next clockwise cut in S. If $C(\theta_{[1]}, \theta_{[2]},\ldots,\theta_{[N]})$ represents the length of C. Then the problem of minimizing the length of C can be posed as the following constrained optimization problem:

$$\text{Min } C(\theta_{[1]},\theta_{[2]},\ldots, \theta_{[N]})$$
$$\text{s.t. } 0 \le \theta_{[i]} \in_{[i]}. \qquad (17)$$

$C(\theta_{[1]}, \theta_{[2]},\ldots, \theta_{[N]})$ is the algebraic sum of individual functions like (1) and (3) through (6) — this is because the length of a cut $[i]$ in C behaves according to (1) if it does not intersect any of the cuts [1], [2], $[i-1]$ and according to the sum of two of the other four functions if it intersects at most two of these. As each of these functions is continuous, so is $C(\theta_{[1]}, \theta_{[2]},\ldots, \theta_{[N]})$ (however, $C(\theta_{[1]}, \theta_{[2]}, , \theta_{[N]})$ is not guaranteed to be a convex/concave function since Eq. (1) is convex but the same is not true of (3)—(6)). The constraint set of this optimization problem is bounded and closed; i.e., compact. Hence, as we are minimizing a continuous function over a compact set, the optimal solution exists. The existence of C *, as claimed in the first statement, is then guaranteed by the fact that there are only finite number of such cutting sequences in S.

In the constrained optimization problem (17), if we remove the trigonometric functions by substitution, it becomes apparent that we are minimizing an algebraic function over a compact set. Hence, from Tarski [6] it is clear that there exists an optimal solution to this problem which lies in the algebraic extension of the field that the input data belongs to. This proves the second statement. ■

Therefore by Lemma 5 above, due to the algebraic nature of the problem, only approximate answers to this problem can be found and the best that can be done is give an approximation scheme (similar to **Algorithm**-*Compute C*(δ) for the problem. This verifies the conjecture made in Overmars and Welzl [10] that obtaining the optimal solution to this problem is intrinsically difficult. There are similar problems in the literature; for example as shown in Chadrasekaran and Tamir [3,4], Quadratic Fractional Programs, Ratio Games and the Fermat—Weber Location problem are all problems of similar type where it is not possible to give the exact optimal solution and the authors pose the solution in the form of a polynomial and an interval such that the only solution to this polynomial in this interval is the optimal solution to the problem.

Lemma 6. *There exists a finite, non-zero error range δ^* such that when this is prescribed as the error range to* **Algorithm**-*Compute C*(δ)*, the cutting sequence produced is combinatorially the same as C^* – i.e. it has the same cuts in the same order.*

Proof. Consider the set S of cuts obtained after completing Step 1 of the **Algorithm**-*Compute C*(δ): it has an exponential but finite number of cutting sequences. Minimize each such sequence by solving the constrained optimization problem (17). By Lemma 5, each of these minima exists and if any two are not equal, they differ by a finite value. Choosing the minimum of these differences as δ^* suffices. ■

4.2. Minimizing the length of a cutting sequence by rotating one cut

Now we give a polynomial time procedure for minimizing the length of a given cutting sequence *C* with *N* cuts, when only one cut in it (assume cut *i*) is allowed to be rotated by θ (arbitrarily assume clockwise) about one of the vertices of P_{in}, that it intersects. We wish to find the optimal value of θ such that the total length of *C* is minimized. It will become apparent that the same scheme can also be applied to minimize the length of *C* by rotating multiple cuts, provided the cuts chosen for rotation do not intersect each other (but we do not discuss that). This procedure can be used in applications where it is not desired to produce the cutting sequence *C*(δ) either because it is cumbersome or because a high degree of approximation is not needed. In such cases one can perform Steps 1 and 2 of **Algorithm**-*Compute C*(δ), skip Step 3 and apply the algorithm in Step 4 to get the best possible cutting sequence. Thereafter, one cut (or multiple cuts, provided they do not intersect) in this sequence can be chosen for rotation and this sequence can be minimized by rotating this cut(s).

When cut *i* rotates by θ, the following quantities change: the length of cut *i* changes, in accordance with (1), and the length of all the cuts in *C* that occur after *i* and intersect cut *i* in accordance with (8)—(11). Therefore, the total length of C as a function of θ, denoted by $C(\theta)$, is given by sums of expressions similar to (1) and (8)—(11) and the problem reduces to a constrained optimization problem similar to

$$\text{Min } C(\theta) = \frac{a_1 \sin\alpha_1}{\sin(\alpha_1+\theta)} + \sum_{k=1}^{k=N} \frac{a_k \sin\alpha_k \sin\theta}{\sin(\alpha_k+\theta)}$$
$$s.t. 0 \le \theta \le \in, \qquad (18)$$

where a_i, α_i, $i = 1, 2, \ldots, N$, are input parameters (or can be calculated from it in polynomial time under the assumed model of computation) and $\in$ is the maximum rotation possible by cut *i*. Differentiating $C(\theta)$ w.r.t. θ and equating equal to 0, to get the extreme points, we get

$$-\frac{\alpha_1 \sin\alpha_1 \cos(\alpha_1+\theta)}{\sin^2(\alpha_1+\theta)} + \sum_{k=1}^{k=N} \frac{\alpha_k \sin\alpha_k \sin\theta}{\sin^2(\alpha_k+\theta)} = 0$$

$$\Rightarrow -a_1 \sin\alpha_1 \cos(\alpha_1+\theta)\prod_{k=1}^{k=N}\sin^2(\alpha_k+\theta)+\sum_{k=1}^{k=N}\left[\alpha_k \sin\alpha_k \prod_{\substack{j=1\\ j\neq k}}^{j=N}\sin^2(\alpha_j+\theta)\right]=0.$$

Putting Sin $\theta = y$ to get an algebraic expression, we can reduce it to the following polynomial:

$$-a_1 \sin\alpha_1 (\cos\alpha_1 \bullet \sqrt{1-y^2}-\sin\alpha_1 \bullet y)\prod_{k=1}^{k=N}(\sin\alpha_k \bullet \sqrt{1-y^2}+\cos\alpha_k \bullet y)^2$$
$$+\sum_{k=1}^{k=N}\left[a_k \sin\alpha_k \prod_{\substack{j=1\\ j\neq k}}^{j=N}(\sin\alpha_j \bullet \sqrt{1-y^2}+\cos\alpha_j \bullet y)^2\right]=0$$

The l.h.s. of Eq. (19) is a polynomial of degree no more than 2*N*. By Lemma 2, $N \leq 5n$; hence, (19) can have at most 10*n* distinct real roots. Assuming that the roots of a polynomial of any fixed degree can be found in constant time (and this is permissible under our assumed model of computation), all roots of (19) can be found (and hence (18) can also be solved) in polynomial time. If the cuts chosen for rotation from *C* intersect then it can be verified that the resulting polynomial have degrees that are exponential in *n*.

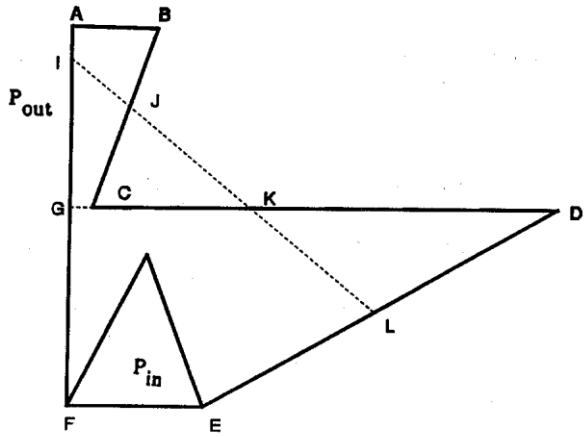

Fig. 7.

5. The non-convex case

In this section we study the extensions of the above results to the case where the two polygons P_{in} and P_{out} are non-convex. It is clear that using guillotine cuts it is not possible to cut out P_{in}, if it is non-convex — therefore it will be assumed that P_{in} is convex (and if not, then its convex hull will be considered as P_{in}). A guillotine cut also needs to be defined more clearly here since a cut can intersect P_{out} several times. Here we define a 'guillotine' cut as follows: if a cut intersects the boundary of P_{out} at several points, then the disjoint intervals of this cut that are fully contained in the *interior* of P_{out} (considering P_{out} as a closed set) are regarded as separate individual cuts. In practice this assumes a cutting tool that can get into every 'nook and cranny' of P_{out}. See for example Fig. 7 where the cut GD is replaced by GC and the cut IL by cuts IJ and KL. This definition is required because the existence proof of C^* in Lemma 5 requires continuity and as shown in Fig. 7, as a 'guillotine' cut moves downward, from AB parallel to GCD, its length is a discontinuous function of its movement at GCD unless we modify the definition as we have done. With our definition of a guillotine cut P_{out} can be replaced by the set of all points in it from which it is possible to draw at least one tangent to P_{in}. For example, in Fig. 8 the ordered set of vertices of P_{out} are [1, 2, 3,…, 14] and this set is represented by the polygon whose ordered set of vertices is [1, 2, 3', 4, 5', 5'', 7, 8, 9, 10, 11, 12, 12', 14]. Given P_{in} and P_{out}, this set can be calculated in polynomial time and we will assume that P_{out} has been preprocessed and replaced with this set. All the lemmas

of Sections 2 and 3 hold here except Lemma 1 and Lemma 2. Lemma 1 needs to be modified to the following in the non-convex case because here it is possible that C^* contains a cut that does not touch P_{in}.

Lemma 7. *If P_{out} is non-convex it can be assumed that for every cut in C^* at least one of the following is true:* (i) *the cut touches P_{in}*; (ii) *the cut intersects a reflex vertex of P_{out} at one of its ends.*

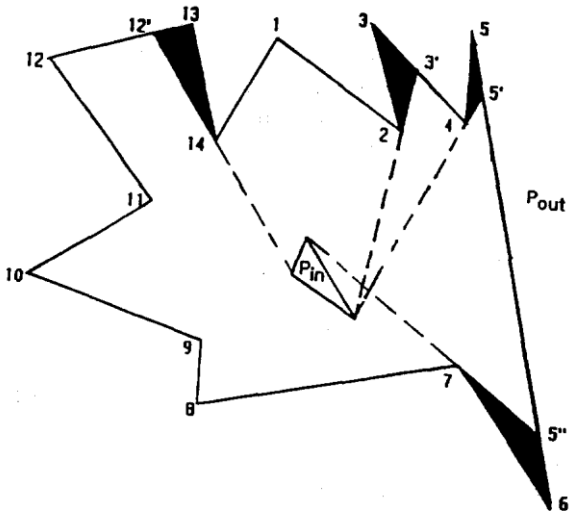

Fig. 8.

Proof. If there is any cut in C^* that does not touch P_{in} then from Lemma 1 it can be assumed that for this cut neither of the following can be true: (i) the cut does not intersect any vertex of P_{out} and (ii) the cut intersects two non-reflex vertices of P_{out} on its two ends. Therefore this cut must intersect at least one reflex vertex of P_{out}. ■

The first statement of Lemma 3 is still true; however the second statement about the relationship between the two derivatives of $k(\theta)$ is not true since it assumed convexity of P_{out}. Lemmas 5 and 6 did not require any assumption of convexity and hold for this case, too. Therefore one can modify **Algorithm**-*Compute* $C(\delta)$ for this case by only changing Step 1 (which is used to remove one part of the combinatorial nature of the problem). The modified algorithm is given in Table 5.

As before, Step 1 of the algorithm in Table 5 adds $O(n)$ cuts. In Step 2.1, there are atmost two tangents added for each vertex of P_{out} and hence this step adds $O(m)$ more cuts. Step 2.2 then adds cuts between each reflex vertex and all other vertices visible to it — thus introducing another $O(m^2)$ cuts. Therefore the total number of cuts after Steps 1 and 2 are over is $O(n + m^2)$. In this case the Dynamic Programming algorithm in Step 4 takes exponential time because of the cuts that do not touch P_{in}, since there is an exponential number of subproblems that need to be considered. Hence the time taken by the approximation algorithm is exponential in the input data and δ.

Table 5

Algorithm-*Compute* $C(\delta)$ (for non-convex case)
begin
Step 1. Same as before.
Step 2. for each vertex of P_{out} **do**
{
Step 2.1. Draw tangents from this vertex of P_{out} to P_{in}. In this case, it may be possible to draw only one tangent.
Step 2.2. **if** this vertex of P_{out} is reflex **then**
{
Draw cuts from this vertex to all other vertices of P_{out} that are visible to it.
} **end if**
}
Step 3. Same as before.
Step 4. same as before.
end

6. Conclusions and future directions

In this paper we have considered the problem of cutting out a given shape/design from another given piece of parent material using a cutting tool that can make only 'guillotine cuts' where it is required to minimize the tool wear caused by the cutting. It has been shown that cutting non-optimally can be very expensive. The problem is formally posed in terms of a pair of nested polygons P_{in} and P_{out} and then the following conclusions can be drawn about the problem:

1. It is shown that an optimal cutting sequence to cut out P_{in} always exists (although in case P_{out} is non-convex, a guillotine cut needs to be defined more rigorously for this purpose).
2. The problem is inherently algebraic in nature because the optimal cutting sequence may lie in the algebraic extension of the field that the input data belongs to; hence, the best that can be done is to provide approximately optimal solutions.
3. We give an approximation scheme to solve this problem. This scheme, which is based on Dynamic Programming, takes time that is polynomial in the encoding length of the input data in unary, when P_{out} is convex, and exponential time, if not.

There are many different open problems in this area. For example it has been conjectured that when P_{out} and P_{in} are convex 'most' cuts in any optimal cutting sequence will be along the edges of P_{in}; i.e., no more than a few non-edge cuts may be required by any optimal cutting sequence. It will be interesting to prove or disprove this conjecture. One can also consider problems where different types of cuts are allowed (for example half rays). The problem where P_{out} contains 'obstacles' that the cutting tool cannot traverse is also unknown. No results are known for higher-dimensional versions of this problem where, we might, for example, be interested in minimizing the total area of the cutting sequence.

Appendix

A mathematical representation of the dynamic programming algorithm

Here we give a mathematical representation of the dynamic programming algorithm in the special case where all cuts are along edges of P_{in}. To do so the following notations are introduced: Let $C^*(i, j)$ be the optimal cutting sequence (from the given set of cuts) to cut out $P_{\text{in}}^{(i,j)}$ from $P_{\text{out}}^{(i,j)}$. Denote by $\|k^{(i,j)}\|$ the length of cut k if it is the first cut after cuts i and j have been made and let $\|k\|$ denotes the length of the same cut before any cuts are made.

Let e_k denote the length of the k-th edge of P_{in}. The functional equations of the dynamic programing are as follows. At the first and the second stages, respectively,

$$C^*(i, j) = 0 \ \forall i, j, 1 \le i \le j = (i+1) \bmod n$$

and

$$C^*(i, j) = e_{(i+1) \bmod n} \ \forall i, j, 1 \le i \le n, (i+2) \bmod n.$$

Inductively, at the p-th stage ($p \ge 3$) we would have

$$C^*(i, j) \ \forall i, 1 \le i \le n \text{ and } j = (i + p) \bmod n.$$

and at the $(p + 1)$st stage we compute

$$C^*(i,j) = \min_{k:\,(i+1)\bmod n,\ldots,(i+p)\bmod n} \left[\left\|k^{(i,j)}\right\| + C^*(i,k) + C^*(k,j)\right]$$

Finally we obtain

$$C^*(i,i) = \min_{k:\,(i+1)\bmod n,\ldots,(i+(n-1))\bmod n} \left[\left\|k^{(i,i)}\right\| + C^*(i,k) + C^*(k,i)\right]$$

Once $C^*(i, i)$ has been found the algorithm finds the optimal cutting sequence from this set of allowable cuts by computing

$$\min_{1\le i\le n} [\|i\| + C^*(i,i)].$$

Acknowledgments

The work of the first author was supported by NSERC Grant #OGP 012 1689 and University of New Brunswick Grant UNB 23-80. This support is gratefully acknowledged.

The work of the second author was supported, in part, by The Morris Hite Center at The University of Texas at Dallas. This support is gratefully acknowledged.

References:

[1] Aggarwa], A., Chang, J.S., and Yap, C.K., "Minimum area circumscribing polygons", *Visual Computer* 1 (1985) 112-117.

[2] A]bano, A., and Orsini, R., "A heuristic solution of the rectangular cutting stock problem", *The Computer Journal* 23 (1979) 338-343.

[3] Chandrasekaran, R., and Tamir, A., "Optimization problems with algebraic solutions: Quadratic fractiona] programs and ratio games", *Mathematical Programming* 30 (1984) 326-339.

[4] Chandrasekaran, R., and Tamir, A., "Algebraic optimization: The Fermat—Weber Location Problem", *Mathematical Programming* 46 (1990) 219-224.

[5] Christofides, N., and Whitlock, C., "An algorithm for two dimensional cutting problems", *Operations Research* 25 (1977) 30-44.

[6] Dori, B., and Ben-Assat, M., "Circumscribing a convex polygon by a polygon of fewer sides with minimum area addition", *Computer Vision, Graphics and Image Processing* 24 (1983) 131-159.

[7] Gi]more, P.C., and Gomory, R.E., "A Linear Programming approach to the Cutting Stock Problem", *Operations Research* 9/6 (1961) 849-859.

[8] Gilmore, P.C., and Gomory, R.E., "A Linear Programming approach to the Cutting Stock Problem — Part II", *Operations Research* 11/6 (1963) 863-888.

[9] Gi]more, P.C., and Gomory, R.E., "Multistage Cutting Stock Problems of two and more dimensions", *Operations Research* 13/1 (1965) 94-120.

[10] Overmars, M.H., and Welzl, E., "The complexity of cutting paper" Extended Abstract, in: *Proceedings of the ACM Annual Symposium on Computational Geometry,* 1985.

[11] Tarski, A., A *Decision Method for Elementary Algebra and Geometry,* 2nd revised ed.: University of California Press, Berkeley, CA, 1951.

[12] Venkateswarlu, P., and Martyn, C.W., "The trim-loss problem in a wooden drum industry", in: *OR-92: Proceedings of the Convention of Operational Research Society of India,* December 1992.