Identifying Alternate Optimal Solutions to the Design Approximation Problem in Stock Cutting

By: J. Bhadury And R. Chandrasekaran B

Bhadury, J. and Chandrasekaran, R.(1999) 'Identifying Alternate Optimal Solutions to the Design Approximation Problem in Stock Cutting', *Engineering Optimization*, 31: 3, 369 — 392 DOI: 10.1080/03052159908941378

Made available courtesy of Taylor and Francis: http://dx.doi.org/10.1080/03052159908941378

Abstract:

The design approximation problem is a well known problem in stock cutting, where, in order to facilitate the optimization techniques used in the cutting process, it is required to approximate complex designs by simpler ones. Although there are algorithms available to solve this problem, they all suffer from an undesirable feature that they only produce one optimal solution to the problem, and do not identify the complete set of all optimal solutions. The focus of this paper is to study this hitherto unexplored aspect of the problem: specifically, the case is considered in which both the design and the parent material are convex shapes, and some essential properties of all optimal solutions to the design approximation problem are ascertained. These properties are then used to devise two efficient schemes to identify the set of all optimal solutions to the problem. Finally, the recovery of a desired optimal approximation from the identified sets of optimal solutions, is discussed.

Keywords: Design approximation; stock cutting; minimal nested polygon problem

Article:

1. INTRODUCTION

Stock cutting problems refer to the broad class of problems in which it is desired to cut out a specific design from a given parent material in as efficient a manner as possible. They have been studied in the engineering community for some time now. Among the first such studies on the subject were those of Gilmore and Gomory [9, 10]. Since then, there has been a plethora of work on that subject; see for example [4— 6, 12].

The broad class of stock cutting problems includes many subproblems that are of interest from both practical and theoretical standpoints. One such sub-problem is that of *Design Approximation*— see [2]. The design approximation problem occurs when the designs or shapes required to be cut are extremely complex, making it difficult and error prone to cut them. In addition, the complexity of these shapes, particularly the number of edges needed to describe them, make it time consuming to apply any optimization techniques utilized in the cutting process. Therefore, as a first step, it is desired to approximate the complex designs by the simplest possible shapes, *i.e.*, by other designs that have the fewest possible edges; hence, the name.

The specific design approximation problem that is studied here can be stated as follows: given a piece of parent material (assume that it is polygonal, *i.e., its* boundary is described by straight line edges), it is required to cut out a given design from it. Because of the complexity of the design, it is required, as a first step, to approximate it with another shape that has the fewest number of edges. This can be seen to be mathematically equivalent to the following problem, which is known as the *Minimal Nested Polygon Problem:* given a larger polygon P_{out}, (which, in this case, represents the parent material) and a smaller polygon P_{in} (the design to be cut out) that is completely contained in P_{out}, find a nested polygon (one that is contained in the annulus between P_{in} and P_{out} and contains the inner polygon P_{in}) P^*, that has the fewest number of edges. Besides design approximation, other applications of this problem include facility location (where the annulus between P_{in} and P_{out} represents a geographical region and it is desired to locate the fewest number of facilities in this region such that line-of-

sight communication is maintained between adjacent facilities) and robotics (where its is required to find a path for a robot in the annulus with the minimum number of turns). Due to its wide applicability, the Minimal Nested Polygon problem has received wide attention in the literature; see for example [1, 3, 8, 11]. However, all the algorithms available in the literature to solve this problem have a common shortcoming: they are designed to identify one, and only one, optimal solution.

This shortcoming renders these algorithms inexpedient in practice, where it is frequently desirable to search for alternate optimal approximations, because one may be preferable to another due to secondary criteria. For example, it is sometimes desired that a certain point or part of the annulus should be cut out by the optimal approximation because it contains flaws, and hence it is required to know if there exists an optimal approximation, *i.e.*, a Minimal Nested Polygon, that will do this. Such a need for alternate optimal solutions also arises in other applications of the Minimal Nested Polygon problem. For example, in location problems, one set of locations might be preferable to another due to geographical factors. In the context of robotics applications, it is frequently desired to investigate the existence of minimal turns paths in the annulus with all its turns in the interior of the annulus, in order to avoid collisions. This need to characterize alternate optimal solutions is made more acute by the fact that the optimal approximation, *i.e.*, the Minimal Nested Polygon is frequently non-unique — in fact there may be an infinite number of them. Consider for example the case where P_{out} and P_{in} are described by a large and small triangle respectively. The optimal solution in this case is given by any triangle in the annulus; and there are an infinite number of them. Thus the need arises to devise schemes to identify all optimal solutions to the design approximation problem, rather than prescribing only one.

As a first step in addressing this issue, the special case that is considered is the one in which both the parent material (P_{ont}) and the design (P_{in}) are convex shapes and hence, so is the optimal approximation (r) - see Bhadury and Chandrasekaran [2] for a proof of this assertion. Two indices are first defined for every point in the annulus — the Polygon Index and the Turn Index respectively. The former refers to the number of edges in a nested polygon that passes through a given point and has the minimum number of edges. The latter refers to a similar value, when it is required that the nested polygon also have a vertex at the given point. Two efficient algorithms are then given that identify all points in the annulus that can either be assumed to be the *vertex* of some r or *lying* on some r, based on these two indices.

This remainder of the paper is divided into four sections. The next section discusses the preliminary concepts and notation needed for the paper. Section three gives a partitioning algorithm for the boundary of P_{out}, which forms the basis for the two primary schemes that are prescribed for identifying all optimal approximations. These two schemes are themselves described in the next section, which ends with a discussion on the recovery procedures in order to use an alternate optimal solution. Finally, the fifth section summarizes the results of the paper and discusses topics for future research. Proofs of all major lemmas are given in the Appendix.

2. PRELIMINARIES

The analysis in the paper requires several terms and notation, which are now defined. To begin with, a convex polygon is one whose internal angles are all less than 1800; in other words, a convex polygon has the property that for any two points inside the polygon, the straight line segment joining them is completely contained inside the polygon itself.

As mentioned before, P_{out}, and P_{in} are assumed to be convex polygons, with $P_{in} \subset P_{out}$, and the Minimal Nested Polygon for P_{out} and P_{in} is assumed to be P*. n is assumed to represent the total number of edges of Pk, and P_{ont} and ϕ, the total number of edges in P*. The entire annular region between P.m and Pi_n is referred to as the annulus and designated by [P_{out}—P_{in}). The boundary of P_{out} (respectively, P_{in}) is referred to as bd(P_{out}) (respectively, bd(P_{in}) and IT_{out} ¯ P_{in}) refers to all the points that are in the interior of the annulus — *i.e.*, all points in [P_{out} − P_{in}] except those on bd(P_{out}) and bd(P_{in}).

A line segment [*x*,y] is assumed to be a closed interval containing its endpoints whereas the segment (x, y) (respectively, [*x*,*y*)) is assumed to contain all points on the line between x and *y* except x (respectively, *y*).

Furthermore, for two points x, y in [P_{out}— P_{in}], [$x \rightarrow y$] is assumed to represent a ray from x in the direction of *y*.

Let x and y be two points on bd(P_{out}), with the property that P_{in} is completely on one side of the line segment [*x*, y] and let *z* be any point on bd(P_{out}) that is on the same side of [x, y] as P. Then, in a clockwise traversal of bd(P_{out}) that begins at *z*, if *x* is encountered after *y*, then *x* is said to be **clockwise** of *y*. Further, *x* is said to be **at least clockwise** of *y* if either *x* and *y* are coincident or if *x* is clockwise of *y*. For any point *v* in [P_{out}—Pi_n], P(*v*) is defined as a nested polygon that passes through *v* and has the minimum number of edges and T(*v*) as a nested polygon that has v as a vertex and has the minimum number of edges. Two associated indices are also 'defined −|P(v)| (called the **Polygon Index)** and |T(*v*)| (the **Turn Index)** which are the number of edges in P(*v*) and T(*v*) respectively. Obviously, for any point v, |T(*v*)| > |P(*v*)|. See Figure I where P^* has been drawn for a given pair of polygons P_{in} and P_{out}, and for the two points v and *x*, the nested polygons T(*v*) (it will be proved later that the polygon shown is T(*v*)) and P(*x*) are also shown. Hence in this case, $\phi = 3$, |T(v)| = 5 and. |P(x)| = 3.

For any point v E [$P_o$ut — P_{in}] a clockwise **greedy structure G(v)** is defined, that is obtained as follows: (Fig. 2 shows G(a) for point a E bd($P_o$ut)): from v the clockwise tangent to Pi_n is drawn — this tangent is assumed to intersect bd(P_{in}) (or, in other words, is tangential to P_{in}) at a vertex denoted by **Tgt(v)** (in case it intersects two vertices, Tgt(v) is defined as the more clockwise of these two vertices). The point of intersection of the ray [v Tgt(v)] with bd(Pout) is defined as v_i. From v_1 this process is continued and successive points v_2, v_3 *etc.* (called the vertices of G(v)) are defined on bd($P_o$ut) similarly until the vertex of G(v) is reached where v becomes visible for the first time — it will be assumed throughout the paper that for any point v, this occurs on the kth vertex of G(v). Then another clockwise tangent is drawn from v_k to P_{in} to obtain the next vertex of G(v) *(i.e.,* the point v_{k+1}) in a manner similar to the other vertices. The line segments [v, v1], [v1, *[vk,* v_{k+1}] constitute G(v) — and these line segments are referred to as the edges of G(v). IG(v)I is used to denote the number of, edges in G(v). If the sequence of *anticlockwise* tangents is taken from v, the resulting greedy structure is denoted by G_a(v). The point of intersection of the line segment [*v*, v_i] and [v_k, v_{k+1}] is denoted by **Int(*v*)**. A point $v \in$ bd(P_{out}) is defined as a **tight point** if $v = v_\phi$. For such a tight point, G(*v*) is a closed polygon and is referred to as a **tight greedy polygon** for the tight point *v*.

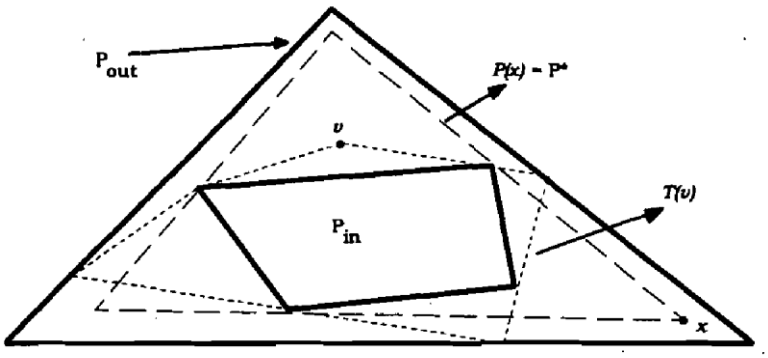

FIGURE 1

For any point $v \in$ [P_{out} − P_{in}], consider the anticlockwise tangent from *v* to P_{in} − the vertex of P_{in} that this anticlockwise tangent from v intersects is denoted by **Atgt(*v*)** — if it intersects two vertices, the more clockwise of these two is chosen as Atgt(*v*). Then the point of intersection of the ray [$v \rightarrow$ Atgt(*v*)] with bd(P_{out}) is denoted as **Anti**(v) — and since $P_{in} \subset P_{out}$, if $v \in$ bd(P_{out}), *v* and Anti(*v*) will occur on different edges of bd(P_{out}).

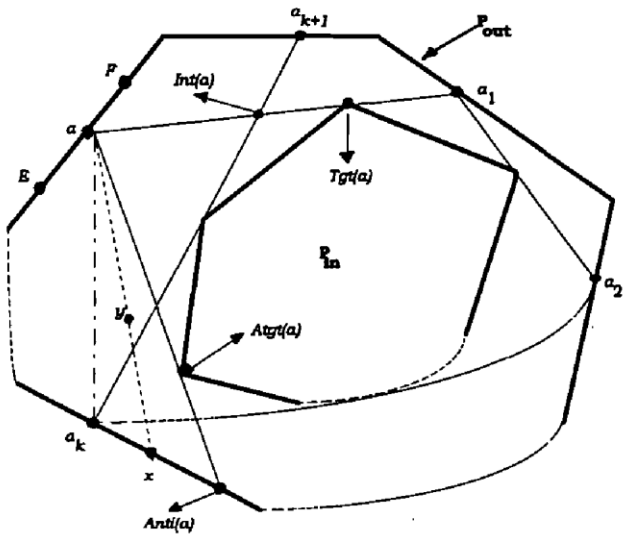

FIGURE 2

For any point $v \in bd(P_{out})$ with $|G(v)| = \phi$, the **slack cone** of v is defined as the entire region of the annulus bounded by the line segments $[v, v_{\phi-1}]$, Anti(v)] and the section of $bd(P_{out})$ between Anti(v) and $v_{\phi-1}$ (including these boundaries themselves). For example in Figure 2, if $|G(a)| = \phi$ (and hence $a_k = a_{\phi-1}$), then the slack cone of a is the triangle [a, Anti(a),a_k]. Note that slack cone for a point v is only defined if $v \in bd(P_{out})$ and $|G(v)| = \phi$.

For a point $v \in [P_{out} - P_{in}]$, the **projector** of v, denoted by **Proj(v)** is defined as the point of intersection of the ray $[v_1 \rightarrow v]$ with $bd(P_{out})$ – for example in Figure 2, the point a is the projector of the point Int(a). If $v \in bd(P_{out})$, then it is defined as its own projector, hence, for all $v \in bd(P_{out})$, Proj(v) = v.

Finally, the computational notations of the paper are defined. A set is said to have $O(n)$ (respectively, $O(n\phi)$) elements when its cardinality is guaranteed to be less than a constant multiple of n (respectively, $n\phi$). Along the same vein, an algorithm is said to take $O(n)$ (respectively, $O(n)$) time if the total time taken by it is guaranteed to be less than a constant multiple of n (respectively, $O(n\phi)$). Such algorithms where the total time taken is a polynomial in the input parameters of the problem itself are considered efficient. As for the assumed model of computation in the paper, it is similar to the ones used in the literature, such as in [l]; the usual random access machine (RAM) that allows operations like +, –, /, *, and finding roots of a quadratic equation to be performed in unit time and is capable of performing infinite precision arithmetic.

Based on the definitions and notations above, the following results are either known in the literature or are easy to verify:

(i) It has been shown in [1], that for any point $v \in bd(P_{out})$, $\phi \leq |G(v)| \leq \phi + 1$.
(ii) If v is a tight point then $|G(v)| = \phi$. Furthermore, since $bd(P_{in})$ does not intersect $bd(P_{out})$, then for a tight point v, $v_k = v_{\phi-1}$ = Anti(v) and $v = v_{k+1} = v_\phi$.
(iii) For any point $v \in bd(P^o ut)$, $|G(v)| = \phi + 1$ iff Anti(v) (respectively, v) is clockwise of v_ϕ (respectively, v_ϕ).
(iv) For any point v E $[P_{out} - P_{in}]$, the edges $[v, v_1]$, $[v_1, v_2]$, . , $[v_k$, v] represents T(v). Hence $|T(v)| = |G(v)|$ – this also proves that in Figure 1, the polygon G(v) shown is also T(v).

With these preliminaries, the following results, are presented.

LEMMA 1 *For* any *point* $v \in [P_{out} — P_{in}]$, $\phi \leq |G(v)| \leq \phi + 2$ *(and hence,* $\phi \leq |T(v)| \leq \phi + 2$).

LEMMA 2 *For any point* $v \in [P_{out} — P_{in}]$, $\phi \leq |P(v)| \leq \phi + 1$.

LEMMA 3 *For any point* $v \in [P_{out} — P_{in}]$, $|P(v)| = \phi$ *iff v occurs in the slack cone of a point* $x \in bd(P_{out})$.

3. PARTITIONING $bd(P_{out})$ INTO CRITICAL INTERVALS

Both the schemes that are presented in the paper to identify the complete set of optimal approximations depend on one common procedure. This procedure, which is the focus of the present section, partitions $bd(P_{out})$ into disjoint intervals such that for any two points inside an interval the Turn Index is the same, *i.e.*, it is either ϕ or $\phi + 1$.

In order to understand the basic motivation of this procedure, consider a point v on $bd(P_{out})$ such that $|G(v)| = \phi + 1$ (and hence v is clockwise of v_ϕ). As v is moved clockwise on $bd(P_{out})$, along the edge of P_{out} that it lies on, all vertices and edges of $G(v)$ move clockwise too — a direct consequence of the fact that $bd(P_{out})$ and $bd(P_{in})$ are continuous, and $P_{in} \subset P_{out}$. During this movement of v, the following four events (hereinafter referred to as events I through IV) that can occur will be of interest to us: (Event I): An edge of $G(v)$ can encounter a new vertex of P_{in}. (Event II): A vertex of $G(v)$ can encounter a new vertex of P_{out}. (Event III): v can encounter the next vertex of P_{out} and (Event IV): v can encounter a tight point on $bd(P_{out})$.

Now suppose that v has been moved clockwise from its initial position by a distance d — where d is small enough so that none of events I, II or III occur *i.e.*, all of the vertices (respectively, edges) of $G(v)$ intersect the same edges (respectively, vertices) of P_{out} (respectively, P_{in}) and v itself remains on the same edge of P_{out}. Let $\delta v_j(d)$ denote the associated movement of the v_j along the edge of P_{out} that it lies on the $v_j(d)$ the vector that represents its final position. Then it is shown in [1], that

$$\delta Vj(d) = (c_j,1\ d + c_{j,2})/(c_{j,3}\ d + c_{j,4}), 1 \leq j \leq k + 1 \qquad (1)$$

where $c_{j,1}$, $c_{j,2}$, $c_{j,3}$, $c_{j,4}$ are fixed constants for v_j. As mentioned before, since $bd(P_{out})$ and $bd(P_{in})$ are continuous and $P_{in} \subset P_{out}$, $v_\phi(d)$ is a continuous function of $v(d)$. Therefore if in this movement of v, $|G(v)|$ changes to (and hence v, becomes clockwise of v), there must be a point in between on $bd(P_{out})$ where $v = v_\phi$, namely a tight point — *i.e.*, during this movement, event IV must have occurred. Note that in order to check for the existence of a tight point, equating $v(d)$ to $v_\phi(d)$ gives a quadratic in d, which has at most 2 distinct real roots.

Based on the above idea, an algorithm is now given to partition $bd(P_{out})$ into intervals that are "small" enough such that if the point v is restricted to move inside an interval, none of events I through IV will occur.

Algorithm Partition — $bd(P_{out})$

begin

Step 1 Every edge of P_{in} is extended to intersect with $bd(P_{out})$ and the two points of intersection are considered *critical points.* Every vertex of P_{out} is also considered a critical point.

Step 2 For every, critical point v found in Step 1, find the greedy structures $G(v)$ and $G_a(v)$ and the vertices of these two structures for this critical point are also included as critical points — for each critical point v store the following: the value of k; the points Anti(v), Tgt(v) and Atgt(v); all the vertices of $G(v)$ (including v_{k+1}); the points Tgt(v), Atgt(v) and functions $\delta v_j(d)$, for $1 \leq j \leq k + 1$.

Step 3 The critical points obtained above in Steps 1 and 2 partition $bd(P_{out})$ into disjoint intervals — for each interval do the following: check if there exists any tight point within this interval by checking if there is a solution to the quadratic $v(d) = v_\phi(d)$ in this interval (there can be atmost 2 tight points per interval). If a tight point exists, then for this point v, draw the tight greedy polygon and this tight point and the vertices of its

associated tight greedy polygon are also included as critical points. For each such tight point v, store all the parameters mentioned in Step 2 above.
end

At the end Step 1 of the' above algorithm, there are $O(n)$ critical points — since there is (respectively, are) one (respectively, two) for every vertex of P_{out} (respectively, edge of P_{in}). In Step 2, for each of the critical points found in Step 1, the greedy and anticlockwise greedy structures are found and all vertices of both the greedy structures are considered as critical points too — since there are atmost $O(\phi)$ vertices of any greedy structure, at the end of Step 2, there will be $O(n\phi)$ critical points. These $O(n\phi)$ critical points that are obtained after the completion of Step 2, divide $bd(P_{out})$ into intervals with the following property — if a point v in one of these intervals is moved clockwise within that interval itself, all edges (respectively, vertices) of $G(v)$ will intersect the same vertex of P_{in} (respectively, edges of P_{ont}) — *i.e.,* none of events I through III will occur. To prove this claim, consider two adjacent critical points E and F, after Step 2 is complete. Suppose that a point v is moved clockwise to another v', where v and $v' \in (E, F)$ and event H occurs at v' (*i.e.,* a vertex of $G(v')$ encounters a new vertex of P_{ont}) — assume that this new vertex of P_{out} is x. However, the vertex x is critical point after Step 1 and hence the anticlockwise greedy structure at x, namely $G_a(x)$, will therefore have a vertex at v' which would make v' is a critical point at the end of Step 2 — but this contradicts the original assumption that E and F are adjacent critical points after Step 2 is complete. A similar argument will show that event I cannot occur at v' either. Finally, Step 1 guarantees that event III cannot occur at v', thus proving the original claim.

After the completion of Steps 2-3 considers each of the $O(n\phi)$ intervals from Step 2, and checks if there exists a tight point in the interval — if so, this point and all the vertices of its tight greedy polygon are considered as critical points too. It is now shown that this step adds $O(n\phi)$ new critical points.

LEMMA 4 *Given that both* P_{in} *and* P_{out} *are convex, there are no more than* $O(n\phi)$ *tight polygons. Thus Step* 3 *of Algorithm Partition* $-$ $bd(P_{out})$ *can introduce atmost* $O(n\phi)$ *new critical points.*

Therefore, once the entire algorithm is over, there will be $O(n\phi)$ critical points that will partition $bd(P_{out})$ into as many intervals (heretofore referred to as *critical intervals)* with the property that if a point v is moved within a critical interval, none of changes I through III will occur (because of Steps 1 and 2) and nor will event IV occur (because of Step 3).

To calculate the time complexity of, *i.e.,* the total time taken by, Algorithm Partition — $bd(P_{out})$, it is clear that Steps I and 2 can be performed in $O(n\phi)$ time, since they are essentially equivalent to the algorithm given in Aggarwal *et al.* [1], which is also known to take as much time. At the end of Step 2, there are $O(n\phi)$ intervals — because the functions $\delta v_\phi(d)$ are known for each interval, the existence of a tight point can be checked in constant time and if one exists, all the vertices of the associated tight greedy polygon can be found in $O(\phi)$ time. Since, by Lemma 4, there are atmost $O(n)$ distinct tight polygons, Step 3 will take $O(n\phi)$ time — thus leading to an overall time complexity of $O(n\phi)$ for the above algorithm. The above discussion thus leads to the following observation and together, they also serve as a proof of the correctness of this algorithm.

LEMMA *5 After* $O(n\phi)$ *critical intervals have been identified by Algorithm Partition* $-$ $bd(P_{out})$ *in* $O(n\phi)$ *time, for any two points on* $bd(P_{out})$ *that lie in the interior of a critical interval, the Turn Indices are same.*

Since a tight point v $Ebd(P_{out})$ represents a 'crossing over' of the two points v and v_ϕ, if there exist two points x, $y \in bd(P_{out})$ such that $|G(x)|$ and $|G(y)|$ are not equal, then there must exist at least one tight point on the section of $bd(P_{out})$ between x and y. This leads to the following two observations, that will be used later.

COROLLARY 6 *After* $O(n\phi)$ *critical intervals have been identified by algorithm Partition* $-$ $bd(P_{out})$, *the following two statements are true for every critical interval* $[E, F]$.

(i) *If E is not a tight point then Turn Index in* (*E, F*), *i.e., Turn Index inside the interval* [*E, F*], *is equal to* |*G(E)*|.
(ii) *If Turn Index in the interval* (*E, F*) *is* (*1*) *then it is guaranteed that* $|G(E)| = |G(F)| = \phi$

If at a tight point *v*, the slope of the function $\delta v_\phi(d)$ is less than unity, then as *v* is moved clockwise, it will become more clockwise of the point v_ϕ, and |G(*v*)| will become $\phi + 1$ (the reverse is true if the slope is more than unity). It can be verified that there can be at most two distinct values of *d* for which this slope equals unity. This ensures that at a tight point *v*, by studying the first two derivatives of the function $\delta_\phi(d)$, it can be determined whether |G(*v*)| is ϕ or $\phi + 1$ when v is moved clockwise – this will be used in the two algorithms given for partitioning the annulus.

4. SCHEMES FOR IDENTIFICATION AND RECOVERY OF ALL OPTIMAL APPROXIMATIONS

In this section, two schemes are given that will help identify all points in the annulus [$P_{out} - P_{in}$] that belong to the set of optimal approximations. This is followed by a discussion of how to recover a desired optimal solution, *i.e.*, construct a desired optimal approximate polygon, that passes through a given point.

4.1. Partitioning of the Annulus Based on the Turn Index |T(*v*)|

The first partitioning scheme is a polynomial time algorithm based on Lemma 1, to partition the annulus according to |T(*v*)|. The algorithm is based on the following observation made in Lemma 1: consider any point $v \in (P_{out} - P_{in})$ and its projector Proj(*v*) (assume that it is denoted by *x*). Then, if *v* lies on the segment [*x*, Int(*x*)], |T(*v*)| = |T(*x*)|, else if v lies on (Int(*x*), Tgt(*x*)], then |T(*v*)| = |T(*x*)| + l. Therefore the main idea in this algorithm is to move a point v clockwise on bd(P_{out}) and trace the locus of the point Int(*v*).

Algorithm Partltlon — |T(*v*)|

begin
Step 1 Find a $\mathbf{P}^*$ and *F* for P_{in}, and P_{out} using the algorithm in [1].
Step 2 Partition bd(P_{out}) into critical intervals using *Partition* – bd(P_{out}).
Step 3 For each critical interval [*E, F*] (assume that ***F*** is clockwise of *E)* **do**
{
Step 3.1 Retrieve G(*E*) and G(*F*) and identify Int(*E*) and Int(*F*), Tgt(*E*) and Tgt(*F*).
Step 3.2 Retrieve the functions $\delta E_j(d)$, $j = 1, k, k + 1$. Using them, obtain the locus of Int(*v*) within this interval [E, *F*] as point *v* is moved from *E* to *F*.
Step 3.3 Partition the area of the annulus [$P_{out}-P_{in}$] in between the line segments [*E,* Tgt(*E*)], *[F,* Tgt(*E*)] and the intervale [*E, F*] into two sets — the 'outer' set that is bounded by the critical interval [*E, F*], the line segments [*E*,Int(*E*)] and [*F*,Int(*F*)], and the locus of Int(*v*) from Int(*E*) to Int(*F*). The 'inner' set is bounded by [Int(*E*), Tgt(*E*)] and [Int(*F*),Tgt(*E*)], and the locus of Int(*v*) from Int(*E*) to Int(*F*). If *E* Int(*E*), the two sets are defined similarly, see Figure 3.
Step 3.4 If $E_\phi = E$ **then**
{
Step 3.4.1 The following points are labelled with Turn Index = |G(*E*)|: all points in the interior of the outer set, all points on bd(P_{out}) in the interval [*E, F*], all points on the locus of Int(*v*) (except Int(*F*)), and all points on the line segment [*E,* Int(*E*)].
Step 3.4.2 The following points are labelled with Turn Index = |G(*E*)| + 1: all points in the interior of the inner set and all points on the line segment [Int(*E*), Tgt(*E*)].
} end if
Step 3.5 If $E_{ls} = E$ *(i.e., E is* a tight point) **then**
{
Step 3.5.1 Retrieve the function $\delta E_\phi(d)$. By examining its first two derivatives at $d = 0$, determine whether, for a point inside the interval, the Turn Index is ϕ or $\phi + 1$.

Step 3.5.2 The following points receive a label of Turn Index = Turn Index in (*E, F*): all points on bd(P_{out}) in the interval (*E, F*); all points on the locus of Int(*v*) (except Int(*F*)) and all points in interior of the outer set.
Step 3.5.3 The following points are labelled with Turn Index = Turn Index in (*E, F*) + 1: all points on the line segment *(E,* Tgt(*E*)); and all points in the interior of the inner set.
Step 3.5.4 The point *E* is labelled with a Turn Index equal to *F*.
}end if
}end for
end

The first two Steps of the above algorithm partition bd(P_{out}) into critical intervals that guarantee that within a critical interval the Turn Index of any point will remain the same. Then in Step 3.2, for each critical interval [*E, F*], the algorithm traces the locus of Int(*v*) as a point *v* on bd(P_{out}) is moved from *E* to *F*. This enables the identification of (at most) seven distinct region in Step 3.3: the two line segments [*E,* Int(*E*)] and [Int(*E*), Tgt(*E*)], the points in the interior of the outer set, the points in the interior of the inner set, all points on the locus of Int(*v*) (except Int(*F*)), the interval [*E, F*] on bd(P_{out}) and the point *E* itself (in Step 3.5.4 when *E* is a tight point). In Steps 3.4 and 3.5, each of these regions then receives a Turn Index value depending on the Turn Index in *(E, F)* — when *E* is not a tight point (Step 3.4) the algorithm uses |G(*E*)| instead of the Turn Index in (*E, F*) since, by Corollary 6, the two are equal. Since there are $O(n\phi)$ critical intervals produced by *Partition* — bd(P_{out}), there will be no more than $O(n\phi)$ disjoint regions produced by *Partition* − |IT(*v*)|.

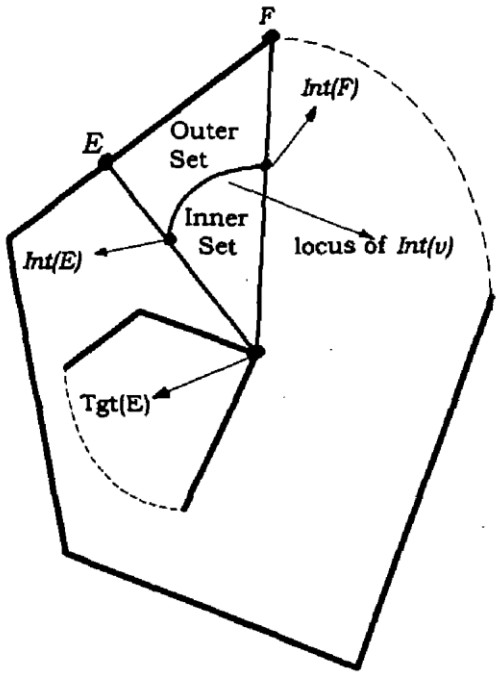

FIGURE 3

To see the time complexity of this algorithm, it is clear that Step 2 takes $O(n\phi)$ time and produces $O(n\phi)$ intervals for consideration by Step 3. Finding the locus of Int(*v*) in the critical interval [*E, F*] in Step 3.2, reduces to the problem of finding the solution to a pair of simultaneous linear equations (namely those of line segments [$v(d)$, $v_1(d)$] and [$v_k(d)$, $v_{k+1}(d)$]) and expressing it as a function of *d*. Given all the information, it is evident that this can be accomplished in constant time. Steps 3.3-3.5 can also be done in constant time (since all the information required is available from Step 2 and there are at most seven regions to label) and thus each iteration of Step 3 can be completed in constant time. Therefore, Step 3 takes $O(n\phi)$ time, leading to $O(n\phi)$ time complexity for the overall algorithm *Partition* − |T*(v)*|.

To see why the above algorithm works correctly, consider any point $v \in [P_{out} - P_{in}]$. It is clear that for any point in the annulus, the projector is unique – assume that x, the projector of the point v, lies in the critical interval $[E, F)$. Since $bd(P_{in})$ and $bd(P_{out})$ are continuous and $P_{in} \subset P_{out}$, it is assured that the points $Int(E)$ and $Int(F)$ are distinct. This leaves only two cases: either **Case (i)**: $E_\phi \neq E$ *(i.e.,* $Int(E) \neq E$*)* or **Case (ii)**: $E_\phi = E$ *(i.e.,* $Int(E) = E$). Assume, without loss of generality, that case (i) holds, *i.e.*, $E_\phi = E$. Then if v is in the interior of the inner set produced by this interval, or if it lies in the interval $(Int(E), Tgt(E)]$ (in which case $E = x$), then it follows that $v \in (Int(x), Tgt(x)]$ and hence, by Lemma l, the Turn Index for v should be equal to $|T(x)| + 1$ – which is the value given to $|T(v)|$ by the algorithm. If, however, v is in the interior of the outer set or any of the boundaries of the outer set (except the line segment $[F, Int(F)]$), then it can be claimed that $v \in [x, Int(x)]$ and hence Turn Index of v should be equal to $|T(x)|$, as given by the algorithm. It should be evident that the argument is identical for case (ii) where $E_\phi = E$, since their distinctness is not used anywhere in the proof. This, along with the fact that the algorithm takes polynomial time and produces polynomial number of disjoint regions, serves as a proof for the correctness of the algorithm.

In closing, it can therefore be claimed that after algorithm *Partition* — $|T(v)|$ is over, for any point $v \in [P_{out} - P_{in}]$, there exists a P^* passing through v with v as its vertex iff its Turn Index has been labelled as being equal to ϕ by the algorithm. Hence this partitioning catalogs all points in the annulus that can be assumed to be the vertex of some Minimal Nested Polygon for P_{out} and P_{in}, thus identifying the set of all optimal approximations.

4.2. Identifying Points in the Annulus with $|P(v)| = \phi$

In the previous sub-section, all points we reidentified in the annulus with the property that there was at least one Minimal Nested Polygon with a vertex there. Now the requirement of having a vertex at that point is dropped the following general question is addressed: given any point x in the annulus, does there exist a Minimal Nested Polygon passing through x, with or without a vertex at x; and if so, produce it. In this section a scheme is developed that answers this question. This scheme, which is based on Lemma 3, accomplishes this by finding all points in the annulus for which $|P(v)| = \phi$.

After partitioning $bd(P_{out})$ using *Partition* – $bd(P_{out})$, for every critical interval $[E, F]$ with Turn Index equal to ϕ, a point v is moved for E to F and the entire region swept out by the slack cone of the point v is found (see Fig. 4) — and because v remains within a critical interval, $v_{\phi\text{-}1}$ remains at least clockwise of $Anti(v)$ and the slack cone exists for each point in the interval. The region swept out is bounded by the following — the interval $[E, F]$, the section of $bd(P_{out})$ between $Anti(E)$ and $F_{\phi\text{-}1}$ and the two envelopes formed by the line segments [v, $Anti(v)$] and $[v, v_{\phi\text{-}1}]$ as v moves from E to F — called the "inner" and the "outer" envelopes respectively. The inner envelope is given by the pair of straight lines $[Anti(E), Atgt(E)]$, $[Atgt(E), F]$. The outer envelope can be found on a case by case basis. Because E and F are adjacent critical points, they will lie on the same edge of P_{out} and the same is therefore true of the pair of points $E_{\phi\text{-}1}$ and $F_{\phi\text{-}1}$ – however these two pairs of points may or may not all be collinear and this gives rise to the following two cases:

Case (*A*) When $E_{\phi\text{-}1}$ and $F_{\phi\text{-}1}$ are not on the same edge as E and F — this is shown in Figure 4. Here the upper envelope is the pointwise maximum of the line segment $[v, v_{\phi\text{-}1}]$ as v is moved from E to F. Given E, F, $Atgt(E)$, $E_{\phi\text{-}1}$ $Anti(E)$, $F_{\phi\text{-}1}$ and $Anti(F)$, it can be verified that this outer envelope can be found algebraically in constant time, since all equations and inequalities involved are of a fixed degree.

Case (*B*) When the points $E_{\phi\text{-}1}$, E and F are collinear. Here the upper envelope is the section of $bd(P_{out})$ between $Anti(E)$ and F.

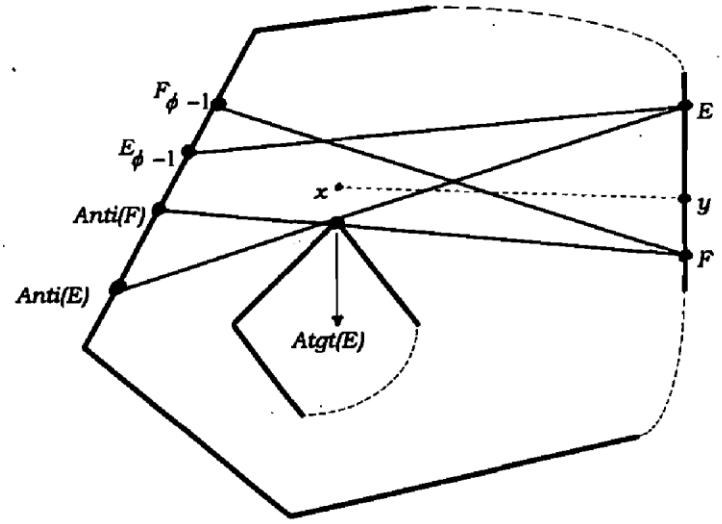

FIGURE 4

Given E, F, G(E) and G(F), checking for the collinearity of the two pairs of points, namely $E_{\phi\text{-}1}$, $F_{\phi\text{-}1}$ and E, F, can be done in constant time. It can therefore be assumed that given a critical interval [E, F] and the associated greedy structures, the two envelopes and therefore the entire region that is swept by the slack cone can be found in constant time. Based on this the algorithm is given below.

Algorithm Partition – |P(v)|

begin
Step 1 Find a P^* and ϕ for P_{in} and P_{out} using the algorithm in [1].
Step 2 Partition bd(P_{out}) into critical intervals using *Algorithm Partition* — bd(P_{out}).
Step 3 For each critical interval [E, F] (assume that F is clockwise of E) **do begin**
{
Step 3.1 Retrieve $G(E)$, $G(F)$, Anti(E), Anti(F) and identify Int(E) and Atgt(E).
Step 3.2 If ($E_\phi \neq E$) **AND** (|G(E)| = F) **then**
{
Step 3.2.1 All points on bd(P_{out}) in the intervals [E, F] and [Anti(E), $F_{F\text{-}1}$] are labelled with a Polygon Index = ϕ.
Step 3.2.2 Determine whether Cases A or B is applicable and compute the outer and the inner envelopes of the region swept out by slack cone. All points in this region, including the ones on the envelopes are labelled with a Polygon Index = ϕ.
} **end if**
Step 3.3 If *(E_ϕ = E), (i.e., E is* a tight point) **then**
{
Step 3.3.1 Retrieve the function $\delta E_\phi(d)$. By examining its first two derivatives *at* d = 0, determine whether, for a point inside the interval, the Turn Index is ϕ or ϕ + I.
Step 3.3.2 If (Turn Index in (E, F) *is* ϕ + I) **then**
{
All points in the line segment [Anti(E), E] are labelled with a Polygon Index equal to ϕ.
} **end if**
Step 3.33 If the Turn Index in the interval *(E, F) is* ϕ **then**
{
All points on bd(P_{out}) in intervals [E, F] and [Anti(E), $F_{\phi\text{-}i}$] are labelled with a Polygon Index = ϕ.

Determine whether Case A or B is applicable and Compute the outer and the inner envelopes of the region swept out by slack cone. All points in this region, including the ones on the envelopes are labelled with a Polygon Index = ϕ.
} **end if**
} **end if**
} **end for**
end

The first two Steps of the algorithm use *Partition* — bd(P_{out}) to partition bd(P_{out}) into critical intervals such that the Turn Index within an interval remains unchanged. Then in Steps 3.2 and 3.3.3, the algorithm does the following for each critical interval [*E, F*] for which the Turn Index inside the interval is equal to ϕ: it algebraically determines the entire region swept out by the slack cone of a point *v* as it is moved clockwise from *E* to *F*. All points in this region swept out (including the ones on the boundaries of this region) are labelled with a Polygon Index equal to ϕ. The reason why even the boundaries are included is because, by Corollary 6, if the Turn Index in (*E, F*) *is* ϕ, then it is guaranteed that |G(*E*)| = |G(*F*)| = ϕ, and all points on these boundaries are in the slack cone of some point in the interval [*E, F*]. Thus the only remaining case is where *E* is a tight point but Turn Index in (*E, F*) *is* $\phi + 1$ — Step 3.3.2 takes care of this case by assigning all points in the segment [Anti(*E*), *E*] a Polygon Index of ϕ.

After *Algorithm* — |*P*(*v*)| is over, all points in the annulus that do not have a label of |P(*v*)| = ϕ are the ones that have their Polygon Index equal to ϕ + l. Although it has not been done above for simplicity, every time that a critical interval *[E, F]* with Turn Index equal to ϕ identifies a region swept out by the slack cone, this region can be marked with a secondary label to indicate that it was generated by the interval [*E, F*]. This will be useful in the recovery of optimal solutions after the algorithm is over. As there are $O(n\phi)$ critical intervals produced by *Partition* — bd(P_{out}), this algorithm produces as many regions. Note that in this case however, these regions may not be disjoint, as it is possible that the same point may be within the slack cone of several points on bd(P_{out}). An argument similar to the one used for *Partition* — |T(*v*)| will show that even in the above algorithm Steps 2 and 3 take $O(n\phi)$ time each, leading to an overall time complexity of $O(n\phi)$ for *Algorithm* — |P(*v*)|.

To see why the algorithm is correct, consider any point, say *w*, in the annulus for which the Polygon Index is ϕ — then by Lemma 3, *w* occurs in the slack cone 'of at least one other point on bd(Pout), for which the Turn Index is ϕ. Assume that this point occurs in the critical interval [*E, F*], where *F* is clockwise of *E*. As the Turn Index in the interval [*E, F*] is ϕ, it is guaranteed that as a point *v* is moved clockwise from *E* to *F*, $v_{\phi-1}$ remains at least clockwise of Anti(*v*) and the slack cone is defined for every point on bd(P_{out}) within this interval. As the algorithm finds the inner and outer envelopes of all the slack cones in this interval, w is guaranteed to be within these envelopes and hence receive a label of Polygon Index equal to ϕ. This, along with the fact that the algorithm takes polynomial time and produces a polynomial number of regions, completes a proof for the correctness of the algorithm. Hence it can be claimed that for any point $v \in [P_{out} - P_{in}]$, there exists a Minimal Nested Polygon passing through *v* iff |P(*v*)| = ϕ and hence this scheme identifies all points in the annulus that can be assumed to lie on some Minimal Nested Polygon for P_{out} and P_{in}.

4.3. Recovery of Any Optimal Approximation

The previous two sub-sections described two different schemes, which together enable the identification of the set of all optimal approximations of P_{in}, *i.e.*, all the Minimal Nested Polygons. Now it remains to be demonstrated how to recover a desired optimal approximation after the two schemes are over.

The key issue in recovering an optimal solution, is to be able to answer the following question: once the two algorithms are over, if any point *x* in the annulus is given, how do we determine if there exists an optimal approximation, *i.e.*, a Minimal Nested Polygon that passes through *x*? If one does exist a procedure to construct it, is needed.

In order to accomplish this, first check if x lies in any of the different regions produced by *Algorithm Partition* — $|T(v)|$ whose label is ϕ. Since each such region is specified by a fixed number of inequalities, this checking can be performed in constant time for one region and hence, $O(n\phi)$ time for the entire annulus. If x does belong to one such region, then, by drawing $G(x)$, the required P^* can be found. If not, then it can be concluded that there is no P^* with a vertex at x however, there may be one with an edge passing through x. To verify that, now check if x belongs to any one of the $O(n\phi)$ regions produced by *Algorithm* — $|P(v)|$. Suppose it is found that x belongs to the region swept out by the slack cone in the interval $[E, F]$, as shown in Figure 4. Then choose any point in $[E, F]$ that is visible to x – say y as shown in the figure. By drawing the $G(y)$ a P^* that passes through x can be found. If x does not belong to the any of the $O(n\phi)$ different regions produced by *Algorithm* — $|P(v)|$, then it can be claimed that there is no P^* passing through x. An argument similar to the one above will show that the checking based on $|P(v)|$ can also be accomplished in $O(n\phi)$ time for the entire annulus, thus leading to an overall time complexity of $O(n\phi)$ for the entire recovery procedure.

5. CONCLUSIONS AND FUTURE WORK

This paper, has tried to address a gap in the current literature on the design approximation problem in stock cutting. Presently, all known algorithms for this problem are only capable of generating one optimal solution, rendering them inexpedient in practice, where alternate optimal solutions are frequently desired. This issue has been addressed by considering the case where both the parent material and the design are convex shapes and two schemes have been presented that help identify the complete set of optimal solutions to the design approximation problem. In addition, a recovery procedure was discussed to construct a desired optimal solution form the identified sets. All the schemes were shown to be efficient in terms of the time taken for their respective executions.

This paper opens up a whole avenue of interesting unexplored problems. Among the most immediate extensions of the present work would be to explore the possibility of extending the algorithms to arbitrary, non-convex shapes of the parent material and/or design. Another strand of future research may be to investigate the possibility of using advanced data structures to improve the time and space complexity of the two algorithms and the recovery procedures.

Acknowledgements

The first author was supported by Grant # OGP 012 1689 from the Natural Sciences and Engineering Research Council of Canada. The second author was supported by the Morris Hite Center at the University of Texas at Dallas. Both supports are gratefully acknowledged. An extended abstract of this paper appeared in the proceedings of the Canadian Conference on Computation Geometry held at Carleton University, Ottawa, in August 1996.

References:

[1] Aggarwal, A., Booth, H., O'Rourke, J., Suri, S. and Yap, C. K. (1989). Finding Minimal Convex Nested *Polygons. Information and Computation,* 83(1), 98-110.

[2] Bhadury, J. and Chandrasekaran, R. (1995). Stock cutting of complicated designs by computing Minimal Nested Polygons. *Engineering Optimization,* 25, 165-178.

[3] Chandra, V., Ghosh, S. K., Maheshwari, A., Rajan, V. T. and Saluja, S. (1995). NC-algorithms for minimum link path and related problems. *Journal of Algorithms,* 19(2), 173-205.

[4] Chauny, F. and Loulou, R. (1994). LP-based method for the muti-sheet cutting stock problem. *INFOR,* 32(4), 253-264.

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

[6] De Carvalho, J. M. V. and Rodrigues, A. J. G. (1994). A computer based interactive approach to a two-stage cutting stock problem. *INFOR,* 32(4), 243-252.

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

[8] Ghosh, S. K. and Maheshwari, A. (1990). Optimal algorithm for computing Minimal Nested Non-convex Polygon. *Information Processing Letters,* 36, 277-280.
[9] Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting stock problem. *Operations Research,* 9(6), 849-859.
[10] Gílmore, P. C. and Gomory, R. E. (1963). A linear programming approach to the cutting stock problem — Part [I. *Operations Research,* 11(6), 863-888.
[11] Sufi, S. and O'Rourke, J. (1985). Finding Minimal Nested Polygons. *Tech. Report,* The Johns Hopkins University.
[12] Venkateswarlu, P. and Martyn, C. W. (1992). The trim-loss problem in a wooden .drum industry. *OR-92: Proceedings of the Convention of Operational Research Society of India.*

Appendix

The appendix contains proofs of the various lemmas in the paper.

Proof of Lemma 1 The lower bound on $|G(v)|$ is obvious by the minimality of P^*. To show the upper bound, consider any point $v \in (P_{out} - P_{in})$, and let x be the projector of v. By the choice of v and x, it is assured that all vertices of $G(v)$ and $G(x)$ are coincident. Consider the two disjoint line intervals $[x, \text{Int}(x)]$ and $(\text{Int}(x), \text{Tgt}(v)]$. It is obvious that if $v \in [x, \text{Int}(x)]$ then $|G(v)| = |G(x)|$; else if $v \in (\text{Int}(x), \text{Tgt}(v)]$, then $|G(v)|$ $|G(x)| + l$. Since $|G(x)| \leq \phi + 1$, the lemma follows. ■

Proof of Lemma 2 Let x be the projector of a point $v \in [P_{out} - P_{in}]$ Consider the greedy structure $G(x)$ – the line segments $[x, v_1]$, $[v_1, v2]$,…,$[vk, x]$ form a nested polygon that passes through v. Further, because $x \in \text{bd}(P_{out})$, $\phi \leq |G(x)| \leq \phi + 1$, ensuring that this nested polygon has $\phi + 1$ or fewer edges. ■

Proof of Lemma 3 This begins by showing that if a point occurs in the slack cone of another point on $\text{bd}(P_{out})$, then its Polygon index is ϕ). To see this assume that in Figure 2, $|G(a)| = \phi$ and hence the point y occurs in the slack cone of a. Let x be the point of intersection of the ray $[a \rightarrow y]$ with $\text{bd}(P_{out})$, as shown. Then it can be verified that the line segments $[a, a_1]$, $[a_l, a_2]$,…,$[a_{\phi-2}, x]$, $[x, a]$ form a nested polygon with edges that passes through y.

To prove the converse, consider all nested polygons through v with 4 edges - *i.e.;* the set of all $P(v)$. Choose one such polygon $P(v)$ and an edge of $P(v)$ that v lies on. Extend this edge to intersect $\text{bd}(P_{out})$ at the points x and y, where x is clockwise of y. The it is clear that v occurs in the slack cone of x. ■

Proof of Lemma 4 After Steps 1 and 2 of Algorithm *Partition* – $\text{bd}(P_{out})$ are over, $O(n\phi)$ critical points are identified on $\text{bd}(P_{out})$, which partition it into as many intervals. Consider one such interval, say Interval i — this interval i may have at most two tight points, which may contribute exactly 2ϕ new critical points. However, once these two tight points in interval i have been identified, and the resulting new critical points have been added, intervals need not be examined again — these are those intervals that contain the vertices of the two tight greedy polygons that were identified in interval i. This guarantees that every time an interval adds or 2ϕ new critical points in any particular iteration of Step 3, it also enables the removal of exactly *4)* intervals from further consideration in future iterations. Since there are only $O(n\phi)$ critical intervals to start with at the beginning of Step 3, this process can be repeated at most $O(n)$ times. Thus at most $O(n)$ distinct tight greedy polygons can be identified in Step 3, and since each has ϕ vertices, this step can add at most $O(n\phi)$ new critical points. ■