A GEOMETRIC ALGORITHM FOR FINDING THE LARGEST MILLING CUTTER

Zhiyang Yao

Mechanical Engineering Department and Institute for Systems Research
University of Maryland
College Park,MD-20742
Email:yaodan@Glue.umd.edu

Satyandra K. Gupta*

Mechanical Engineering Department and Institute for Systems Research
University of Maryland
College Park,MD-20742
Email:skgupta@eng.umd.edu

Dana S. Nau

Computer Science Department and Institute for Systems Research
University of Maryland
College Park,MD-20742
Email:nau@cs.umd.edu

ABSTRACT: In this paper, we describe a new geometric algorithm to determine the largest feasible cutter size for 2-*D* milling operations to be performed using a single cutter. In particular:

1. We give a general definition of the problem as the task of covering a *target region* without interfering with an *obstruction region*. This definition encompasses the task of milling a general 2-*D* profile that includes both open and closed edges.
2. We discuss three alternative definitions of what it means for a cutter to be feasible, and explain which of these definitions is most appropriate for the above problem.
3. We present a geometric algorithm for finding the maximal cutter for 2-*D* milling operations, and we show that our algorithm is correct.

KEYWORDS: Computer-Aided Manufacturing, Process Planning, Cutter Selection for Milling

1 INTRODUCTION

NC machining is being used to create increasingly complex shapes. These complex shapes are used in a variety of defense, aerospace, and automotive applications to (1) provide performance improvements, and (2) create high performance tooling (e.g., molds for injection molding). The importance of the machining process is increasing due to latest advances in high speed machining that allows machining to create even more complex shapes. Complex machined parts require several rouging and finishing passes. Selection of the right sets of tools and the right type of cutter trajectories is extremely important in ensuring high production rate and meeting the required quality level. It is difficult for human planners to select the optimal or near optimal machining strategies due to complex interactions among tools size, part shapes, and tool trajectories.

Although many researchers have studied cutter selection problems for milling processes, there still exist significant problems to be solved. Below are two examples:

- Most existing algorithms only work on 2-*D* closed pockets (i.e., pockets that have no open edges), despite the fact that open edges are very important in 2-*D* milling operations.
- Since there are several different definitions of what it means for a cutter to be feasible for a region, different algorithms that purport to find the largest cutter may in fact find cutters of different sizes.

In this paper, we present an algorithm for finding maximal cutter for general 2-*D* milling operations to be performed using a single cutter.

- We formulate the general 2-*D* milling problem in terms of a *target region* and an *obstruction region*. This problem formulation encompasses the general problem of how to mill a 2-*D* region that has both open edges (edges that don't touch the obstruction region) and closed edges (edges that touch the obstruction region) (see

* Corresponding author.

Section 3 for definitions). Our formulation allows arbitrarily complex starting stocks. Therefore, it can be used to model machining of geometrically complex castings such as engine blocks.

- We analyze three different definitions of what it means for a cutter to be feasible, and explain why one of these definitions is more appropriate than the others.
- We describe a new "region covering" algorithm that finds the largest cutter that can cover the target region without interfering with the obstruction region, and we give the correctness proof for our algorithm.

In practice, quite often multiple cutters are used to machine a complex milling feature. Our region-covering algorithm is also useful as one of the stage in finding an optimal sequence of cutters for machining a complex feature (for our subsequent work on this subject, please see [12]).

2 RELATED WORK

Because of the wide range of the complexity of products, requirement for machine accuracy, different machining stages, selecting optimal cutter size is an active research area. Below we provide a summary of previous research in the area of milling cutter selection.

Bala and Chang presented an algorithm to select cutters for roughing and finishing milling operations [2]. Their work encompasses almost all the features found on prismatic parts, such as slots, steps. Their algorithm is based on geometric constraints. The basic idea is trying to fit the possible large circle into contours to select possible large cutter to save processing time. They take both cutter change time and geometric constraints into consideration. They stated the problem as follows. If there exists a set of cutters, and after machining with these cutters, only finishing machining is needed for fillet radii corners, so the problem is to determine the cutter with the largest radius in this set. The main concern is to make sure that the material left behind by the cutter at each of the convex vertices can be removed by one pass along the boundary of the finishing cutter. A convex vertex is defined as a vertex at which the interior angle is less than 180°. For each convex vertex, the radius of the circle touching the edges forming the vertex as well as the fillet circle is found. Then they check to see if this circle intersects any of the edges of the bounding polygon or any of the islands. If an edge that violates the circle is found, the circle has to be modified or the radius has to be reduced to remove this violation. The aim is to make the circle tangential to this edge. After that, the reflex vertices have to be handled. Reflex vertices are those vertices at which the interior angles are greater than 180°. To solve the problem caused by edges, perpendiculars are drawn from the reflex vertex to each of the edges. First of all, they check to see if the perpendicular hits the edge or not. If not, no action is taken. Otherwise they have to check to see if that particular edge is causing a valid constraint or not. To resolve the constraint caused by other reflex vertices, the distances to all the valid reflex vertices is determined. The smallest of all these distances gives the smallest constriction within the pocket. After determining the cutter size, feasible cutter motion region can be identified and the cutter movement within the region can be optimized. By using their algorithm, the maximal cutter that can cover target region will be the one that based on the Alternative 1 of cutter feasible definition (see Section 3.1 for details).

Veeramani and Gau have developed a two-phases method to select a set of cutters [10]. In the first phase, a concept called the Voronoi mountain is employed in order to calculate the material volume that can be removed by a specific cutting-tool size, the material volume remained to be machined subsequently and the cutter-paths for each cutting-tool. In the second phase, a dynamic programming approach is applied for optimal selection of cutting-tool sizes on the basis of the processing time. This algorithm considers geometric constrains as well as total processing time. It is possible to save processing and machining cost compared to using a single cutter to machine the entire pocket. However, the algorithm of using Voronoi mountain can only handle problems without open edges.

Yang and Han presented a systematic tool-path generation methodology in which they incorporate interference detection and optimal tool selection for machining free-form surfaces on 3-axis CNC machines using ball-end cutters [11]. To find the optimal tools, a comparison of all possible combination of tools are performed. The maximum number of selected cutters is restricted to 3. This optimal tool selection method is designed aimed for any type of parametric surfaces to be machined. This algorithm has the following limitations. First, because the algorithms are grid-line based, if very fine resolution of grid is imposed, high computational power is demanded to implement the algorithm. Second, if the number of available tools is large, the comparison of all possible combination of tools could be time consuming.

Mahadevan, Putta and Sarma have developed a feature-free approach to automatic CAD/CAM integration for 3-axis machining [8]. The algorithm is based on Voronoi diagram. The objective is to select tools for global roughing and generate tool paths directly from the shape of the workpiece. First, slices are generated as sequences of closed contours. Then a Voronoi diagram is employed to generate the path of the centerline of the tool and calculate the accessibility region on each slice. The criterion to select a cutter in this algorithm is to select the cutter that can sweep much of the region of the slice instead of selecting a largest possible cutter. Because large tools have less reachable region than small tools so they incur the penalty of tool changes, so the optimal tool sequence should be selected based on the total time which depends on the region that each tool can access in each slice. To calculate the accessible region, the overall geometry of the tool assembly including the tool holder and the spindle is considered in this algorithm.

Other main research in the area of cutter selection includes work by Arya, Cheng and Mount [1] on multiple tool selection, Dong, Li and Vicker [3, 6] on rough machining of sculptured parts, Lee and Chang [4, 5] on 2.5D and 3D NC surface machining and 5-axis sculptured surface machining, Lim, Corney and Clark [7] on tool sizing for feature accessibility, Sun, Wang, Wright and Sequin [9] on operation decomposition for freeform surface features, and You and Chu [13] on NC rough cut machining. These works though related to cutter selection but are significantly different in scope from the problem being addressed in this paper.

3 DEFINITIONS

3.1 Basics

The most common milling problem is the problem of cutting a given 2-*D* region at some constant depth using one or more milling tools. In addition to the region to be machined (which we call the *target region T*), there is also an *obstruction region O*, a region which the tool should not cut during machining. An example is shown in Figure 1.

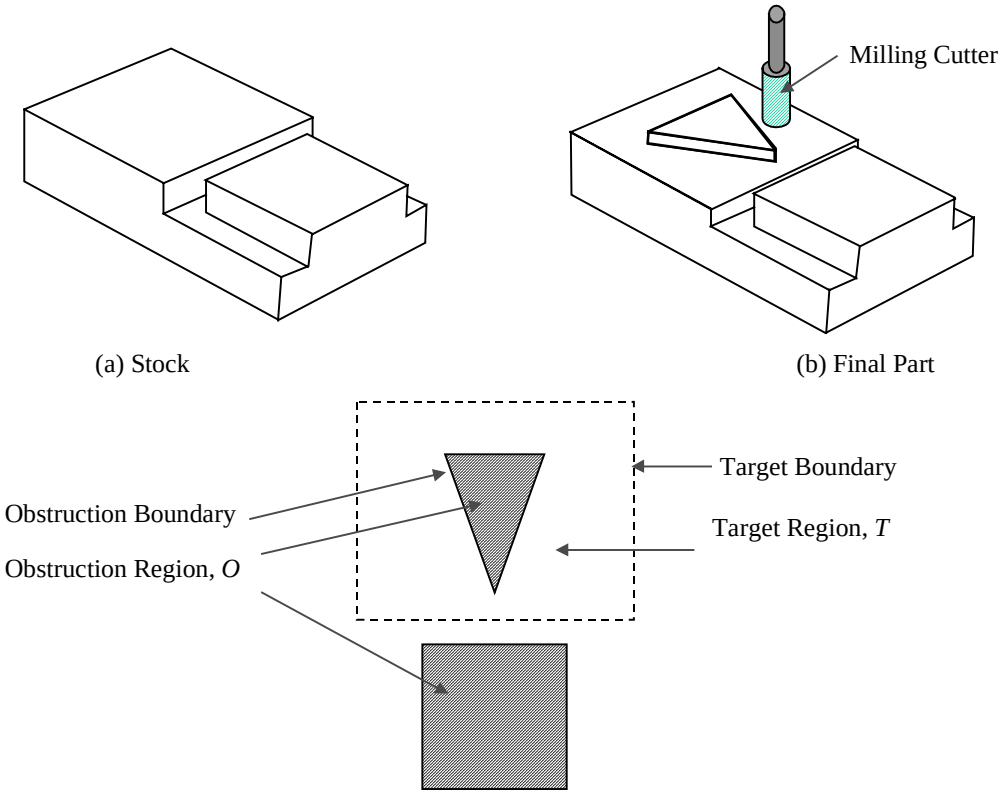


Figure 1: Examples of the stock, final part, target region, and obstruction region.

The target region and the obstruction region must both be regular sets, but may each consist of a number of non-adjacent sub-regions:

$$T = T_1 \cup \ldots \cup T_j;$$

$$O = O_1 \cup \ldots \cup O_k.$$

We assume that the boundary of each sub-region consists of only of line segments and segments of circles.

As shown in Figure 1(c), the *target boundary* B_T is the boundary of the target region, and the *obstruction boundary* B_O is the boundary of the obstruction region. The edges on the obstruction boundary are called *obstruction edges*. We call an edge of the target boundary a *closed edge* if it is coexists with an obstruction edge; otherwise we call it an *open edge*. (Note that 2-*D* closed pockets do not have any open edges.) We will use dashed lines to represent open edges, solid lines to represent closed edges, and diagonal stripes to represent the obstruction region.

Figure 2 shows examples of open and closed edges. Each closed edge *e* separates the *material* (i.e. the obstruction region) from part of the target region. The side of *e* on which the material lies is called *e*'s *material side* and the other side is called *e*'s *non-material side*.

Let $p = (x,y)$ be a point, and $r \geq 0$ Then *p*'s *r-offset region* is the set of all points within distance *r* of *p*:

$$\text{offset}(p,r) = \{(u,v) : \sqrt{(u-x)^2 + (v-y)^2} \leq r\}.$$

If *S* is a set of points, then its *r*-offset region is the union of the *r*-offset regions of the individual points:

$$\text{offset}(S,r) = \bigcup_{(x,y)\in S} \{(u,v) : \sqrt{(u-x)^2 + (v-y)^2} \leq r\}.$$

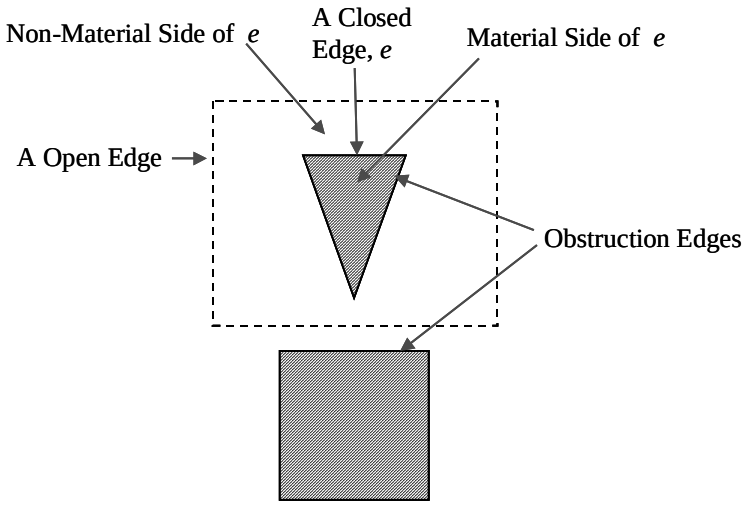


Figure 2: Examples of open, closed and obstruction edges.

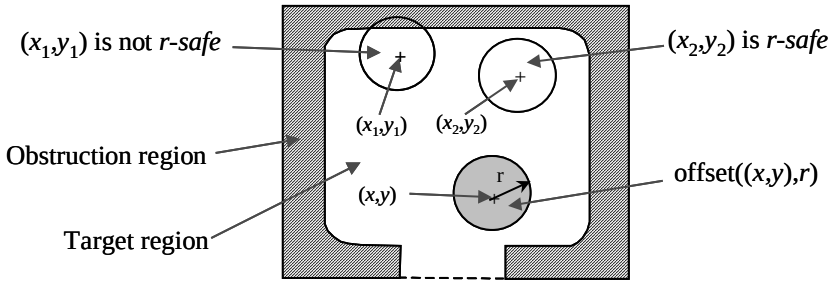


Figure 3: An *r*-offset region, and locations that are *r-safe* and not *r-safe*.

Intuitively, a point p is *r-safe* if a cutting tool of radius r can be placed at p without intersecting the obstruction (an example is shown in Figure 3). Mathematically, this is equivalent to any of the following two statements:

- p is *r-safe* if the distance between p and every point in the obstruction region is at least r;
- p is *r-safe* if offset(p,r) $\cap^*$ $O = \varnothing$.

Where $\cap^*$ is regularized intersection. Similarly we will use $-^*$ to denote regularized difference and use $\cup^*$ to denote regularized union.

A set of points S is r-safe if all of the individual points are r-safe, or equivalently, if $O \cap^*$ offset(S,r) = $\varnothing$. We define safe(S,r) as the *r-safe subset* of a set S. Therefore

$$\text{safe}(S,r) = S -^* \text{offset}(O,r).$$

Figure 4 shows an example of safe(T,r).

A point p is *r-cuttable* if there is an r-safe point q such that $p \in$ offset(q,r). Intuitively, this means p is r-cuttable if a cutting tool of radius r can be positioned to cover p without intersecting the obstruction region. A set of points S is r-cuttable if every point of S is r-cuttable.

Lemma 1 (*Cuttability of all safe points*). Any r-safe point is also r-cuttable.

Proof. Let p be any r-safe point. Then $p \in$ offset(p,r), so p is in the r-offset region of an r-safe point. □

Lemma 2 (*Non-cuttability of obstruction points*). No point in the interior of O is r-cuttable.

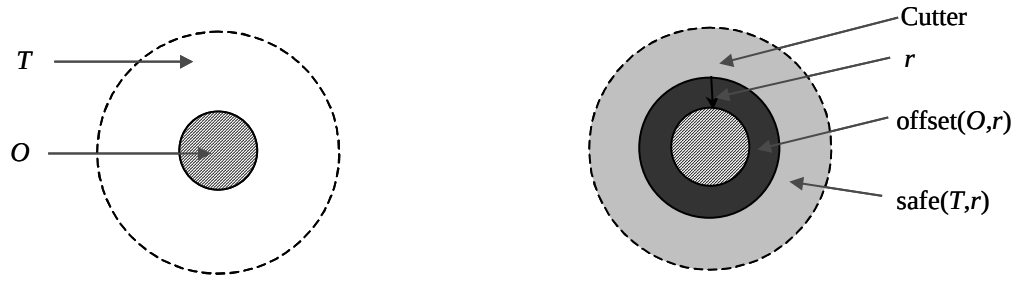
Proof. Suppose p is in the interior of O. Let q be any point such that $p \in$ offset(q,r). It suffices to show that q cannot be r-safe. To show this, we first note that the distance between p and q is $\leq r$. Since p is in the interior of O, it is easy to construct another point p' in the interior of O such that the distance between p and q is $< r$. Thus q is not r-safe. □

If it is obvious what r is, then we will say "offset" for "r-offset," "cuttable" for "r-cuttable," and so forth.

3.2 Cutter Feasibility

Most existing algorithms for cutter-tool-size selection just find the "largest feasible cutter" without clearly stating what it means for a cutter to be feasible. There are at least three different possible definitions, based on three different criteria for what kind of cutter path is acceptable.

Alternative 1: cutter feasibility based on Voronoi diagrams. It is easy to think of defining the largest feasible cutter to be the largest cutter that can go through all "bottlenecks" in the target region, as shown in Figure 5(a). This is equivalent to saying that a cutting tool C of radius r is feasible if T's Voronoi diagram [1] is r-safe. In Figure 5(a), C_1 is an example of the largest feasible cutter based on this definition.

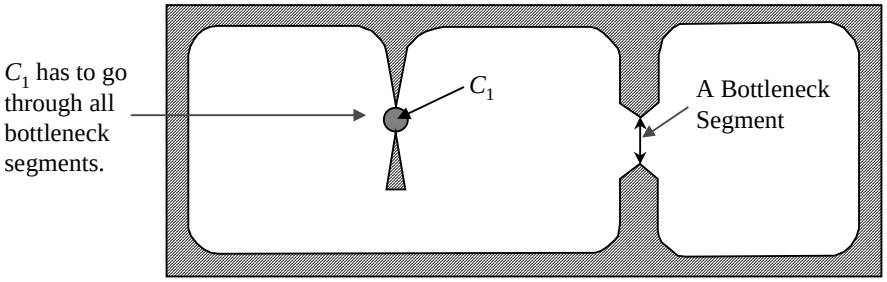


(a) Target and obstruction regions (b) Cutter and corresponding safe(T,r)

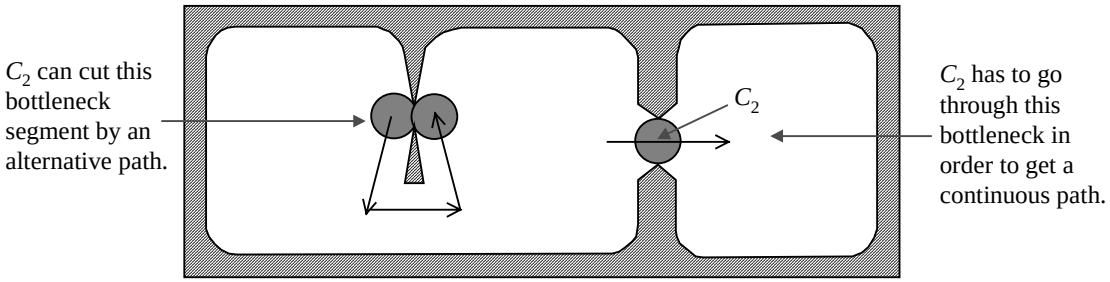
Figure 4: An Example of safe(T,r)

Alternative 2: cutter feasibility based on a continuous tool path. Rather than forcing the cutter to go through every bottleneck, we can allow the cutter to go around some of them instead (Figure 5(b)). Thus, one might want to say that a cutting tool of radius r is feasible if there is a continuous tool path H that is r-safe and whose r-offset region contains T. Intuitively, this means that C can machine all of T in one continuous pass. In Figure 5(b), C_2 is an example of the largest feasible cutter based on this definition.

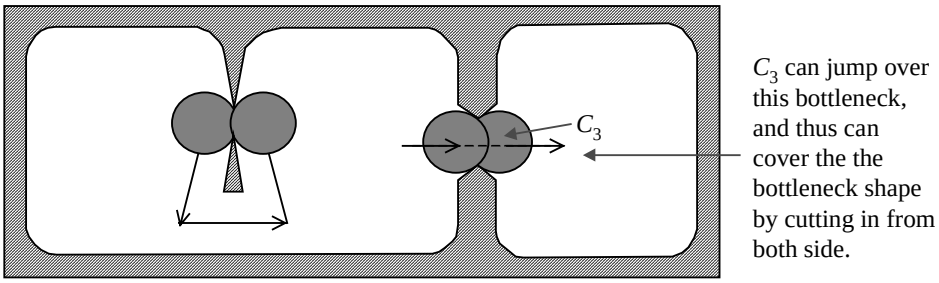
Alternative 3: cutter feasibility based on cuttability. If we are willing to interrupt the machining process briefly,



(a) Largest Feasible Cutter Based on Alternative 1



(b) Largest Feasible Cutter Based on Alternative 2



(c) Largest Feasible Cutter Based on Alternative 3

Figure 5: Different largest feasible cutters result from different definitions of cutter feasibility.

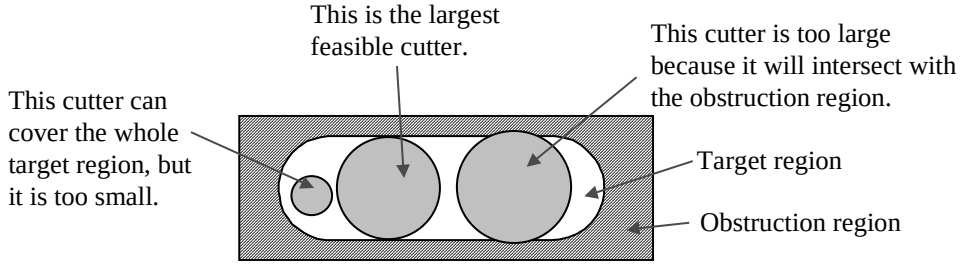


Figure 6: A simple example of the maximal cutting tool.

then we can jump over each bottleneck by lifting the cutter up and putting it down again on the other side of the bottleneck (Figure 5(c)). Thus, we might say that a cutting tool C of radius r is feasible if there is an r-safe set of points S whose r-offset region contains T (or equivalently, if T is r-cuttable). Intuitively, this means that C can machine T using one or more passes. In Figure 5(c), C_3 is an example of the maximal cutter based on this definition.

For simple cases such as the situation shown in Figure 6, all three alternatives will give the same answer. However, in more complicated situations such as the situation shown in Figure 5, Alternative 3 will give the largest cutter size and Alternative 1 will give the smallest cutter size.

The main goal of finding the maximal cutter is to reduce the manufacturing time and thereby reduce the manufacturing cost. Generally, using a small cutter requires much more time than would be needed by a larger cutter, even if the larger cutter needs to be lifted up and set down again. Thus, since Alternative 3 gives us the largest cutter, it is the definition of cutter feasibility that we think is preferable in many machining problems. Thus, it is the definition that we use in this paper.

Problem Formulation: With the above definition in mind, we define the *cutter selection problem* as follows: given a target region T and an obstruction region O, find $r^* = \max \{r : \text{a cutter of radius } r \text{ is feasible}\}$ where feasibility is as defined in Alternative 3.

Note that if any two closed edges meet at a convex corner (see Figure 7 for an example), then the corner point is not r-cuttable for any $r>0$. In this case, we say that the cutter selection problem is *unsolvable*. In all other cases, it is *solvable*.

3.3 Critical and Non-Critical Points

Intuitively, the *edge region of radius r* for a closed edge e is the region $E(e,r)$ formed by sweeping a cutter of radius r along the non-material side of e. Mathematically speaking, a point p is in $E(e,r)$ if p lies within a circle of radius r that is tangent to e_i on the non-material side of e. Figure 8(c) shows an example of an edge region. The *cumulative edge region* $E(r)$ is the union of the edge regions of all closed edges. For an example, see Figure 8(c). From this definition, the following lemma follows immediately:

Lemma 3.

$$\max \{r : E(r) \cap^* O = \varnothing\} = \min_e \max\{r : E(e,r) \cap^* O = \varnothing\},$$

where the minimum is taken over all closed edges e.

A point in T is *r-critical* if it is neither r-safe nor in an edge region of radius r. The *r-critical region* is the set $K(r)$ of all r-critical points of T. Note that

$$K(r) = T -^* \text{safe}(T,r) -^* E(r) = (T -^* E(r)) \cap^* \text{offset}(O,r).$$

The r-critical region may consist of several non-adjacent subregions:

$$K(r) = K_1(r) \cup \ldots \cup K_k(r).$$

Below we give several examples of what these subregions can look like:

- One kind of critical subregion can occur when two closed edges meet, as shown in Figure 9. However, this kind of critical region will not occur if the angle between the two closed edges is greater than 60°.

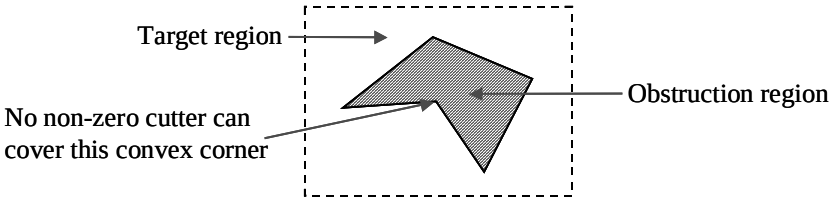
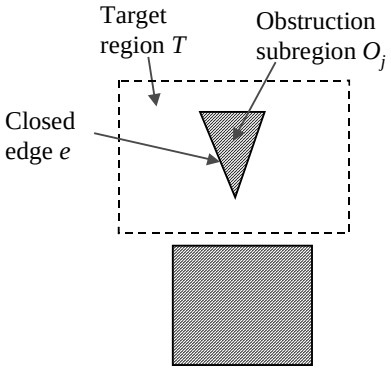


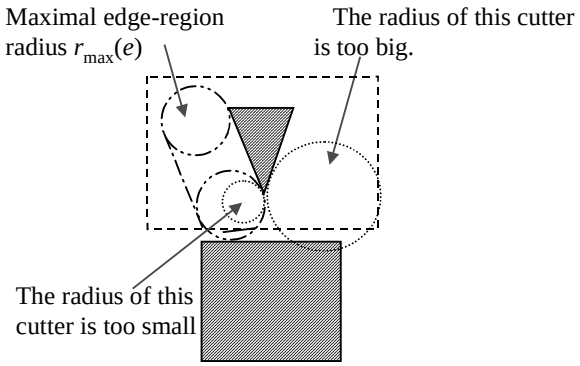
Figure 7: An example of unsolvable cutter-selection problem.

- Another kind of critical subregion can occur when an edge of an obstruction region occurs slightly outside the target region, as shown in Figure 9. However, this will not happen if the distance between the edge and the target region is greater than the cutting-tool radius *r*.

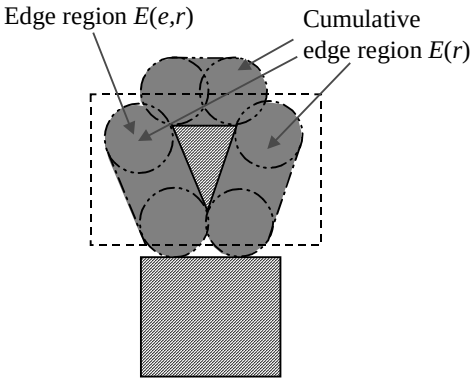
The following theorem says that a cutting tool can cut every non-critical point if and only if the cutting tool's radius is small enough that none of the edge regions intersect the obstruction region. Since the above examples suggest



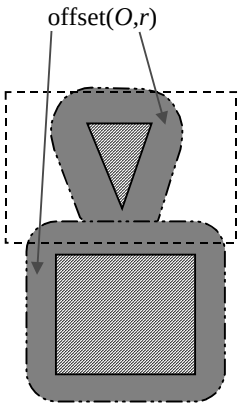
(a) Obstruction and target regions.



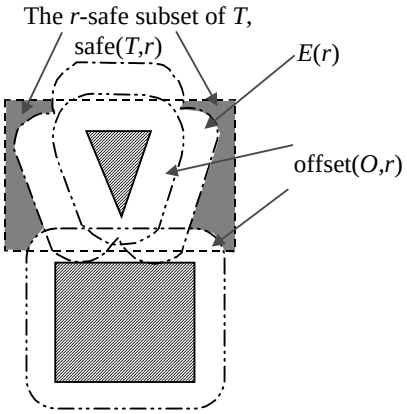
(b) For closed edges, we can find the maximal edge-region radius.



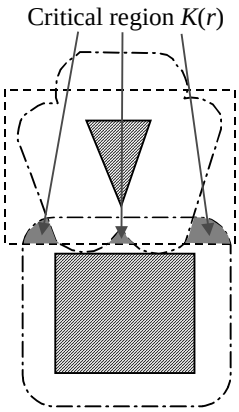
(c) For closed edges, we can find the edge region for each edge thus find the cumulative edge region.



(d) By given a radius r, we can find the offset of the obstruction region.



(e) We can get safe(T,r) after we find the offset region and the cumulative edge region.



(f) After we get the safe region, we can get the critical region

Figure 8: Examples of edge region, safe region, and critical region.

that most designs are unlikely to contain critical points, this means that in most cases it is easy to compute the maximal cutting-tool radius: just find the largest radius for which no edge region intersects the obstruction region.

Theorem 1: $E(r) \cap^* O = \emptyset$ if and only if every point in $T -^* K(r)$ is r-cuttable.

Proof.

We will first prove that if $E(r) \cap^* O = \emptyset$ then every point in $T -^* K(r)$ is r-cuttable. Let p be any non-critical target point, i.e., any point in $T -^* K(r)$. From the definition of $K(r)$, there are two cases.

Case 1: p is r-safe. Then from Lemma 1, p is r-cuttable.

Case 2: p is in some edge region $E(e,r)$. Then p is contained in a circular region R of radius r that is tangent to e on e's non-material side. Let q be the center of R. Then R = offset(q,r). Every point of offset(q,r) is in $E(r)$, so offset(q,r) $\cap^* O = \emptyset$, whence q is r-safe. Thus p is in the r-offset region of an r-safe point, so p is r-cuttable.

Now, we will prove that if $E(r) \cap^* O \neq \emptyset$ then some point in $T -^* K(r)$ is not r-cuttable. Suppose $E(r) \cap^* O \neq \emptyset$. Then for some closed edge e, $E(e,r) \cap^* O \neq \emptyset$, so there is a point $p \in E(e,r)$ such that $p \in O$. From this it is easy to construct a point $p' \in E(e,r)$ such that p' is in the interior of O. Let q be the point of e that is closest to p'. The only location where the cutter can cut q is the point c for which offset(c,r) is tangent to e at q. It is easy to show that

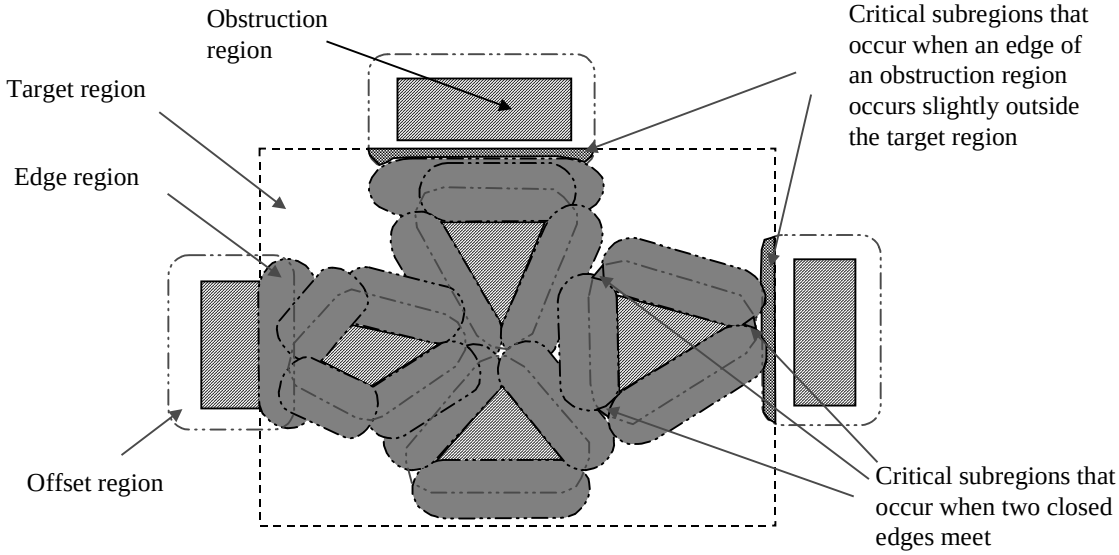


Figure 9: Some examples of critical regions.

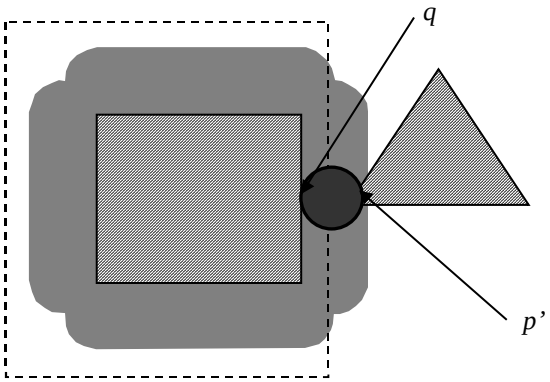


Figure 10: In this example, if $E(r) \cap^* O \neq \emptyset$ then some points in $T -^* K(r)$ are not r-cuttable.

offset(c,r) also contains p'. Thus c is not r-safe, so q is not r-cuttable. An example is shown in Figure 10. □

The above theorem says nothing about whether p is r-cuttable if p is in the critical region $K(r)$. In such cases, p may or may not be r-cuttable. If $p \in K(r)$, then $p \in$ offset(O_j,r) for some subregion O_j of O. If O_j is convex and if $p \notin$ offset(O_k,r) for all $k \neq j$, then p is r-cuttable (see Figure 11 for an example). However, if O_j is not convex or if $p \in$ offset(O_k,r) for some $k \neq j$, then sometimes p is r-cuttable and sometimes it is not.

4 ALGORITHM FOR FINDING THE MAXIMAL CUTTER

Our main algorithm for the maximal cutter selection problem is called ***Find_Maximal_Cutter_Radius*** (***FMCR*** for short)**.** For every closed edge a, this algorithm calls the subroutine ***Maximal_Edge_Region_Radius*** to find the maximal edge-region radius for a. Then it uses the smallest of those radii (denoted by r_E) to compute the cumulative edge region $E(r_E)$, the safe region $S(T, r_E)$, and the critical region $K(r_E)$. The algorithm then calls the subroutine ***Maximal_Critical_Region_Radius*** to find the largest r (denoted by r_K) such that $K(r_E)$ is r-cuttable. The final result is the minimum of that radius r_E and r_K.

Procedure *Find_Maximal_Cutter_Radius*(T, O)

//T is the target region, and O is the obstruction region.

1. $r_E = \infty$, $r_K = \infty$;
2. For each closed edge a and obstruction edge b, do
   - r = ***Maximal_Edge_Region_Radius***(a,b) // this subroutine is described in Section 5
   - $r_E = \min\{r, r_E\}$; // r_E is now the largest radius for which no edge region intersects the obstruction
3. $E = E(r_E)$; // the cumulative edge region
4. F = offset(O, r_E); // the offset of the obstruction region
5. $S = T -^* F$; // the "safe" region
6. $K = T -^* S -^* E$ // the critical region
7. If K is nonempty then

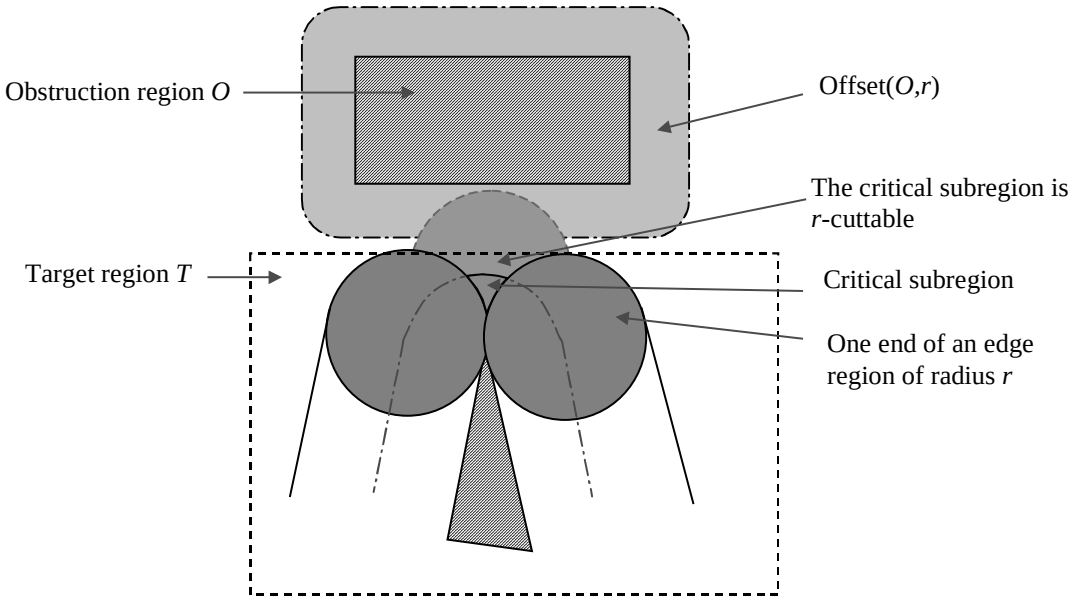


Figure 11: An example of a critical subregion that can be covered without reducing the tool radius.

- r_K = ***Maximal_Critical_Region_Radius***(T,O,K,r_E)

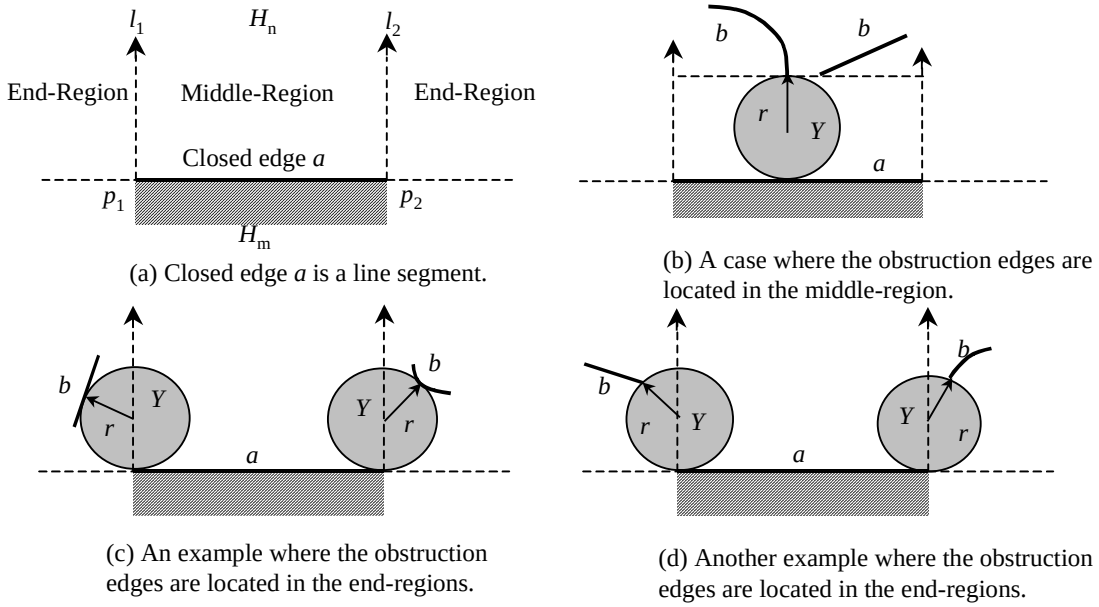
  // this subroutine is described in Section 6

  // r_K is now the largest r such that K is r-cuttable

- Return $r = \min\{ r_E, r_K \}$

8. Else return $r = r_E$;

5 FINDING THE MAXIMAL SWEPT CUTTER FOR A CLOSED EDGE



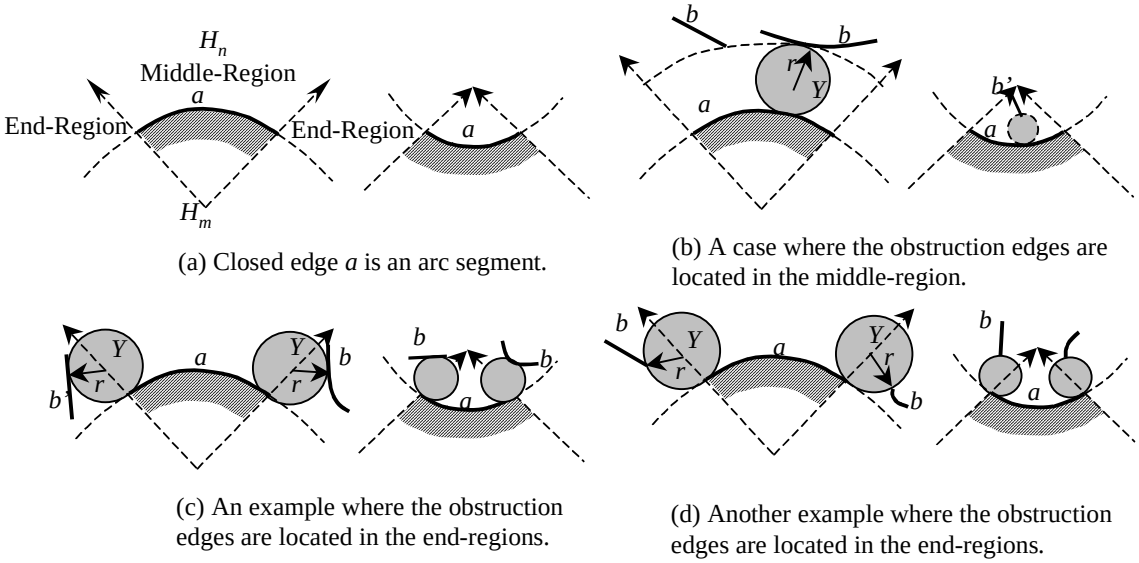
(a) Closed edge a is a line segment.

(b) A case where the obstruction edges are located in the middle-region.

(c) An example where the obstruction edges are located in the end-regions.

(d) Another example where the obstruction edges are located in the end-regions.

Figure 12: Finding maximal cutter radius for a linear closed edge in presence of different types of obstruction edges.



(a) Closed edge a is an arc segment.

(b) A case where the obstruction edges are located in the middle-region.

(c) An example where the obstruction edges are located in the end-regions.

(d) Another example where the obstruction edges are located in the end-regions.

Figure 13: Finding maximal cutter radius for a circular closed edge in presence of different types of obstruction edges.

In the algorithm ***Find_Maximal_Cutter_Radius*** described in Section 4, the purpose of the subroutine ***Maximal_Edge_Region_Radius*** is to solve the following problem: given a closed edge *a* and an obstruction edge *b*, find the largest *r* such that the region $E(a,r)$ does not intersect the edge *b*.

If the closed edge is an arc segment and its angle is greater than 180°, then we split this arc into two arc segments such that each arc segment's angle is less than or equal to 180°. For each closed edge *a*, we will do the following:

a. Extend closed edge *a* in each direction at its end points to infinity using rays that are tangent to end-points and going away from the edge.

b. The extended edge divides the space into two half-space, H_m and H_n. H_m is the half-space that contains material side of *a*. H_n is half-space that contains non-material side of *a*.

c. Use perpendicular lines at end points to split H_m into the following three regions: *one middle region* of *a* and *two end-regions* of *a*. Figures 12(a) shows examples of these regions when *a* is a line segment and Figure 13(a) shows examples of these regions when *a* is an arc segment.

Procedure *Maximal_Edge_Region_Radius* (*a*, *b*)

1. Split *b* into at most two segments such that each segment is completely contained in H_m or H_n.
2. $r = \infty$;
3. for every segment *b'* of *b* that in H_n, do the following:

   r' = ***Max_edge_region_radius_for_closed_edges*** (*a*, *b'*);

   $r = \min\{r, r'\}$;
4. return *r*.

Procedure *Max_edge_region_radius_for_closed_edges* (*a*, *b*)

1. Split *b* into at most three segments such that each segment is in the middle region of *a* or in one of the two end-regions of *a*.
2. If *b* is in middle region of *a*, then

   Return 1/2 of the distance between *a* and *b*.

   // This is the same as the maximum diameter of any circle that touches *b* and is tangent to *a*. Examples are shown in Figure 12(b) and Figure 13(b).
3. If *b* is in the end region of *a*, then
   - Let *e* be whichever end point of *a* is closet to *b*;
   - Let *p* be the point in *b* that is closest to *e*;
   - Let *Y* be the circle that is tangent to *a* at *e* and contains *p*;
   - Return the radius of *Y*.

   // In practice, if we can find a circle that is tangent to both a and b at p, then Y is that circle. Otherwise, we get Y by finding that minimal one of the circles that is tangent to a at p and pass through one end point of b. Examples are shown in Figure 12(c, d) and Figure 13(c, d).

6 FINDING THE MAXIMAL CUTTER FOR CRITICAL REGION

In Section 4, if the critical region in Step 6 of ***Find_Maximal_Cutter_Radius*** is not empty, then a cutter of radius r_E may be too large to cover all of the critical region. In this case, the subroutine ***Maximal_Critical_Region_Radius***, described below, will find a maximal cutter radius that can cover the critical region. In this subroutine, the number r_i is a constant set by the user. It should correspond to the increment in tool sizes that are available in a shop floor.

Procedure *Maximal_Critical_Region_Radius*(*T*, *O*, *K*, r_E)

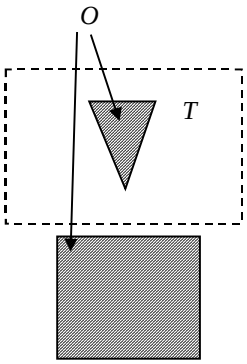
// *T* is the target region, *O* is the obstruction region, *K* is the critical region and r_E is the maximal edge-region radius.

Let $r_K = r_E$.

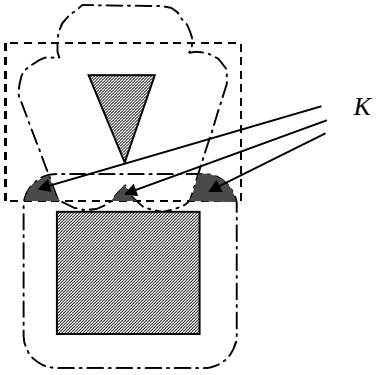
1. Let U be a finite region that encloses the obstruction and target regions.

   // In practice we compute U by computing the bounding box of $T \cup^* O$.

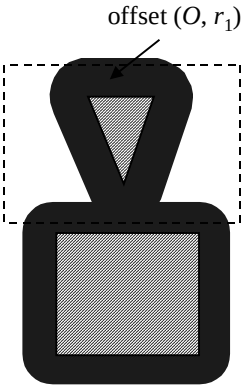
2. loop



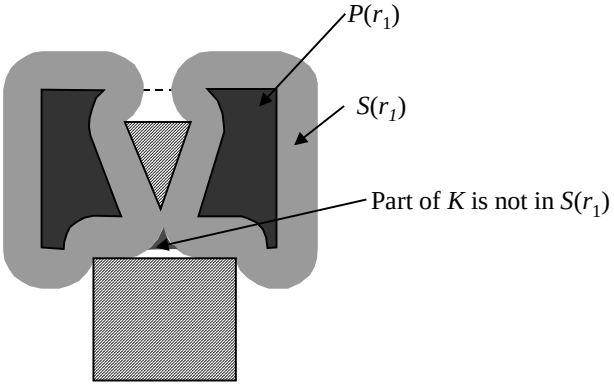
(a) Obstruction and target regions.



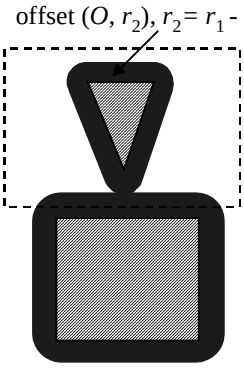
(b) The critical regions.



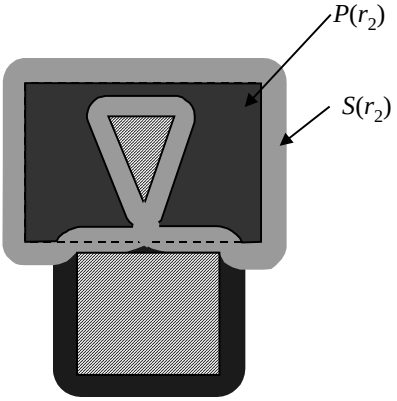
(c) Offset obstruction region using radius r_1.



(d) After offset $P(r_1)$ by using radius r_1, there exists some region that is nor safe.



(e) Reduce the radius from $r_1$to r_2 and offset obstruction region by using radius r_2.



(f) After offset $P(r_2)$ by using radius r_2, the whole target region is safe, therefore, $r_2$is the approximate maximal cutter.

Figure 14: Using approximation algorithm to find the near maximal cutter for critical region.

- $P = U -^{*}$ offset (O, r_E).

  // P is = {safe locations for a cutter of radius r_E}

- S = offset(U,r_E)

  // S = {points cuttable by a cutter of radius r_E}

- if $K \subseteq S$, then return r_K.

- else $r_K = r_K - r_i$

  // the constant r_i is described in the text.

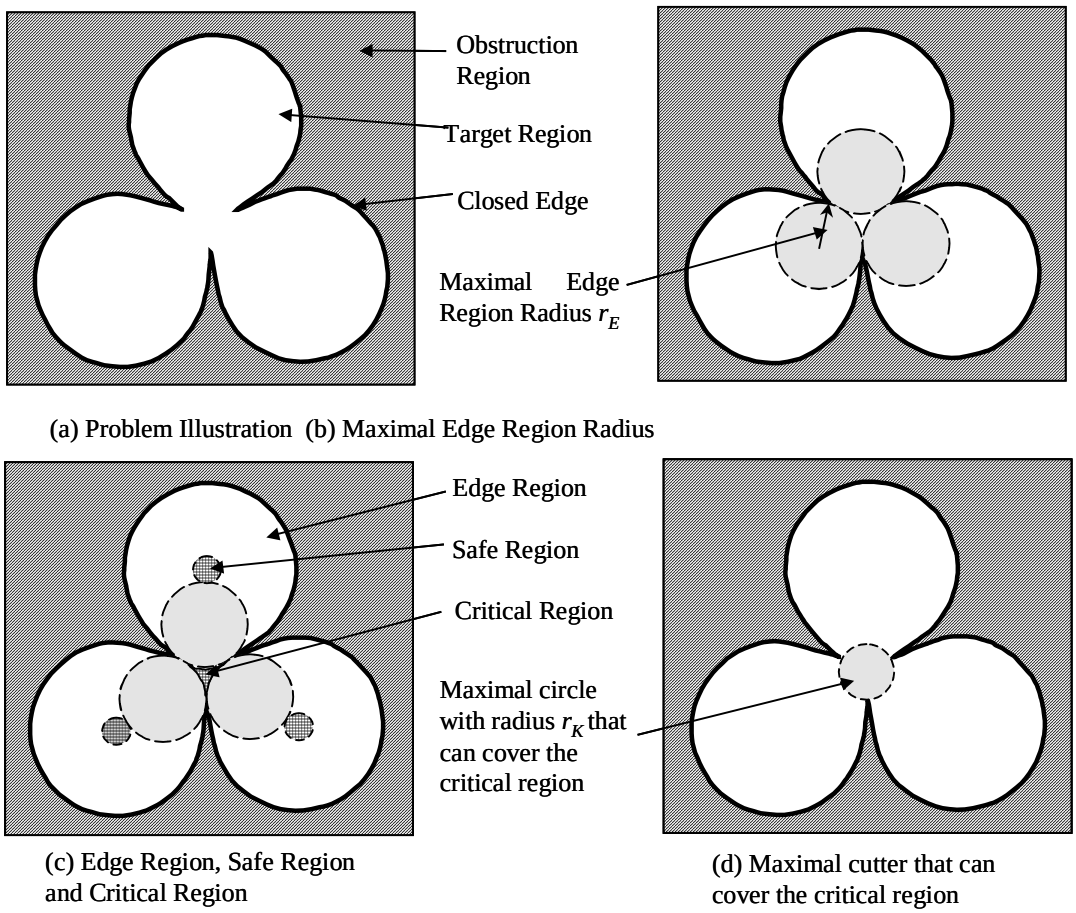
3. repeat

// Figure 14 shows an example of how this procedure works. In Figure 14 (c) and (d), r_K=r_E, and $K \not\subset S$. After one or more iterations, r_K=r_E - cr_i, where c is a constant, then we have $K \subseteq$ S shown in Figure 14 (e) and (f).

7 DISSCUSSION OF CORRECTNESS OF OUR ALGORITHM

For a solvable problem, our algorithm exhibits the following two properties:

Property 1. If the critical region in Step 6 of ***FMCR*** is empty, then ***FMCR*** returns r^{*} = max {r : a cutter of radius r



(a) Problem Illustration (b) Maximal Edge Region Radius

(c) Edge Region, Safe Region and Critical Region

(d) Maximal cutter that can cover the critical region

Figure 15: An example of the operation of our algorithm.

is feasible}.

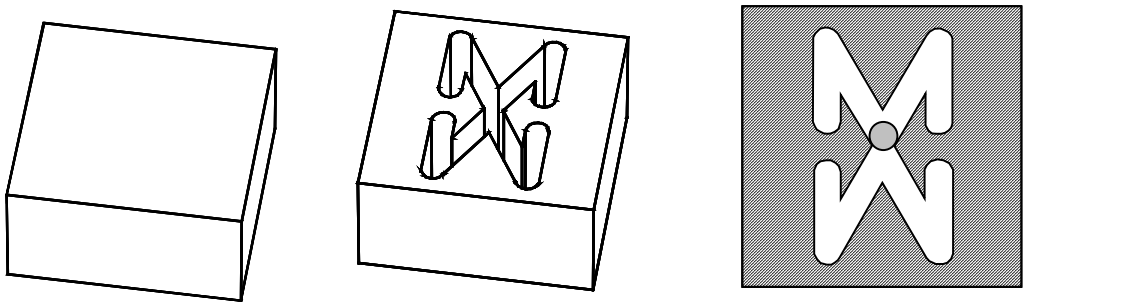
Proof. Suppose there is no critical region, and let r be the number returned by *FMCR*. Here $r = r_E$. Then r_E is the minimum, over all closed edges e, of max $\{r : E(e,r) \cap^* O = \varnothing\}$. Thus from Lemma 3, r_E is the largest number such that $E(r_E) \cap^* O = \varnothing$. Therefore, from Theorem 1, a cutter of radius r_E will be able to cut all of T. For any $r > r_E$, we would have $E(r) \cap^* O \neq \varnothing$. Thus from Theorem 1, T would not be r-cuttable. □

Property 2. If the leftover region in Step 6 of ***FMCR*** is not empty, then ***FMCR*** returns a number r such that a cutter of radius r is feasible, and $r^* - r < r_i$ (where r^* is the maximum feasible cutter radius).

Proof. Let r be the number returned by ***FMCR***. Here $r = \min\{ r_E, r_K \}$. Then $E(r) \cap^* O = \varnothing$, so from Theorem 1 we know that $T–K(r)$ is r-cuttable. Procedure ***Maximal_Critical_Region_Radius*** only allows cutters that can cover K. Therefore r can cover the target region and is feasible. If r_K is equal to r_E, then r is the exact solution and there is no difference between theoretical answer and the result found by ***FMCR***. If $r_K < r_E$, then $r + r_i$ cannot be a feasible solution. Otherwise, ***FMCR*** would have returned this solution. Therefore, in this case, theoretically maximal radius $r^* < r + r_i$. Therefore the difference between the theoretically maximal diameter and result returned by ***FMCR*** is smaller than r_i.

8 IMPLEMENTATION AND EXAMPLES

We use the example shown in the Figure 15 to illustrate the operation of our algorithm. The target region and obstruction regions are shown in Figure 15(a). The details are as follows: Our main algorithm for the maximal cutter selection problem is called ***Find_Maximal_Cutter_Radius*** (***FMCR*** for short)**.** For every closed edge a, this algorithm calls the subroutine ***Maximal_Edge_Region_Radius*** to find the maximal edge-region radius for a. Then it uses the smallest of those radii (denoted by r_E) to compute the cumulative edge region $E(r_E)$, the safe region $S(T, r_E)$, and the critical region $K(r_E)$. The algorithm then calls the subroutine ***Maximal_Critical_Region_Radius*** to find the

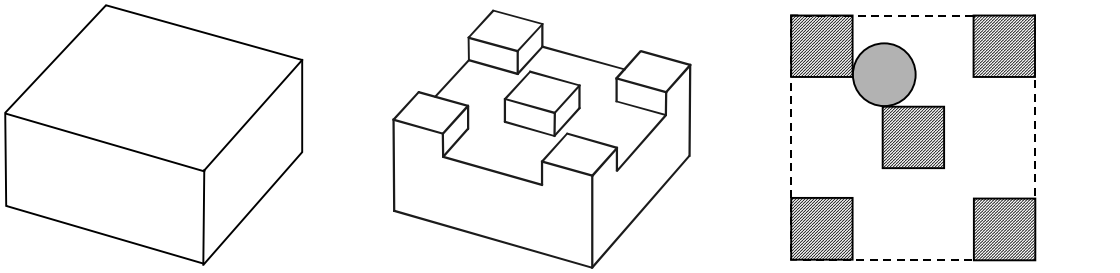


(a) The starting stock.

(b) The final part.

(c) The profile of the final part along with the maximal cutter found by our algorithm.

Figure 16: Example 1.

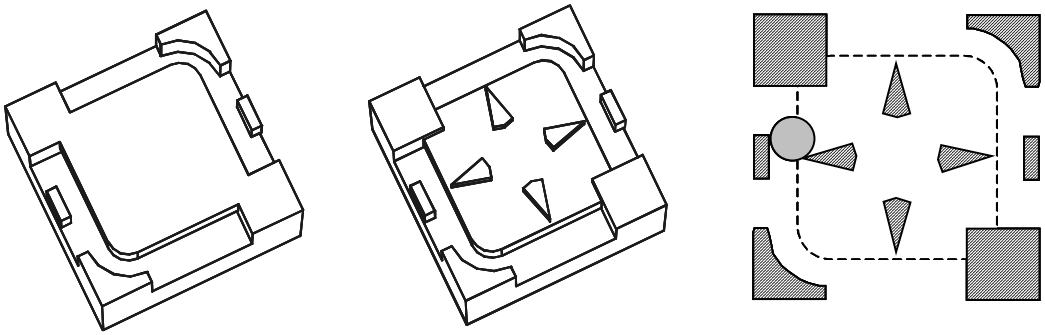


(a) The starting stock.

(b) The final part.

(c) The profile of the final part along with the maximal cutter found by our algorithm.

Figure 17: Example 2.

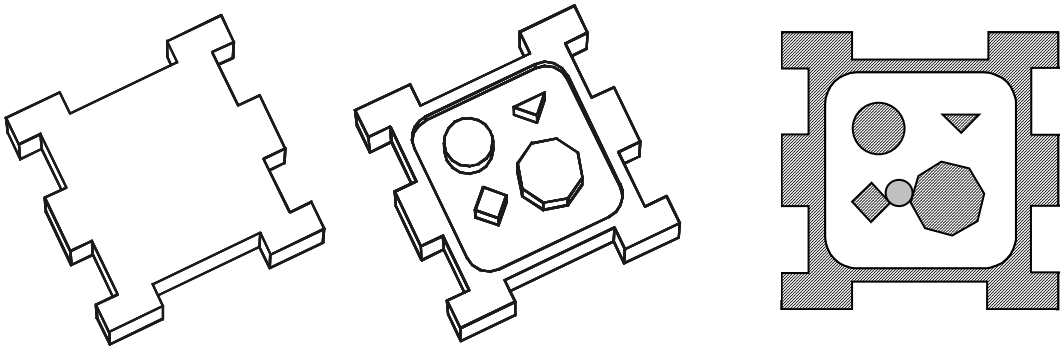


(a) The starting stock.

(b) The final part.

(c) The profile of the final part along with the maximal cutter found by our algorithm.

Figure 18: Example 3.



(a) The starting stock.

(b) The final part.

(c) The profile of the final part along with the maximal cutter found by our algorithm.

Figure 19: Example 4.

largest r (denoted by r_K) such that $K(r_E)$ is r-cuttable. The final result is the minimum of that radius r_E and r_K.

- First for every closed edge a, we find the maximal edge-region radius for a. (shown in Figure 15(b))
- We use the smallest of those radii, r_E, to compute the cumulative edge region $E(r_E)$ and safe region S. (shown in Figure 15(c)).
- We then get the critical region K. For the critical region, we will find the maximal cutter with radius r_K that covers it and does not interfere with obstruction region, as shown in Figure 15(d).
- The maximal cutter that covers the target region without interfering with the obstruction region will be the minimal one of r_E and r_K.

We have implemented our algorithm to find the maximal cutter for 2-D milling operation. The core programming work is done by using C++ on UNIX system. Meanwhile, we have linked our core code with the ACIS Toolkit® and JAVA 3D® such that by inputting a solid model of a milling problem, we can extract the profile of target and obstruction regions and then execute the core code to get the maximal cutter. Finally, the result can be shown in 3-*D* version.

Figure 16-19 show the results of the maximal cutters selected by our algorithm. Our algorithm solved every one of those examples in less than one second on an Ultra 10 computer.

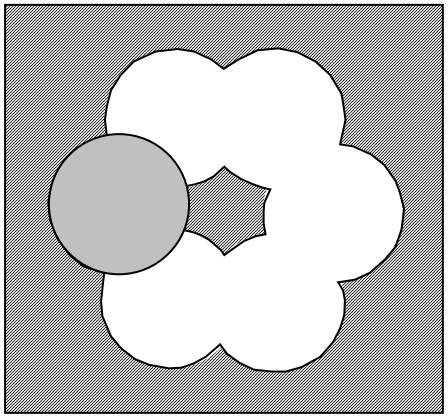


Figure 20: Manufacturing Consideration in Choosing Maximal Cutter

9 CONCLUSION AND DISCUSSION

In this paper, we have presented a geometric algorithm for finding the maximal cutter size for a 2-*D* milling process. Our algorithm has the following properties:

1. It finds the largest cutter that can cover the region to be machined without interfering with the obstruction region.
2. In addition to solving traditional pocket-milling problems, our algorithm can solve a wide variety of milling problems that involve open edges. Consideration of open edges is extremely important when near-net shape castings are used as starting stocks.
3. Our algorithm uses a cutter feasibility definition based on cutter's ability to cover the target region. Therefore, it can find larger cutters than the ones found by algorithms that are based on alternative definitions of feasibility (e.g., either based on covering every bottleneck in the target region or existence of continuous path between every pair of points in the target region).

The maximal cutter that we have found is based on the geometric constrains. In actual machining, we will need to consider several other cutting constrains. Here are some examples:

- There are several cutting parameters that may influence the selection of cutter size. We cannot use cutters that will conflict with the machining constraints. For example, we know that the material removal rate is proportional to the diameter of the cuter in milling operations. As a result, if we use a bigger cutter, then we can have higher material removal rate, thus we can save the cutting time. On the other hand, the maximum power of a machine is constant. The required cutting power is proportional to the metal removal rate. Therefore the maximal diameter is actually limited by the maximum machine power.
- Sometimes the cutter selected by region covering idea may not be the best one for manufacturing. For example, Figure 20 shows an example in which if we use the maximal cutter selected by our algorithm, we will have to lift up and put down the cutter several times. In this particular case, the maximal cutter selected by us may not save total cutting time and may get bad manufacturing surface. Beside the geometric constraints, we also need manufacturing knowledge to help us decide which is the best cutter size.
- In addition to geometric considerations described in this paper, several other machining considerations such as available fixturing options, surface finish requirements, available cutting tool geometries, and the resulting cutting forces play a role in cutter selection and should be considered in selecting milling cutters.

We are currently extending our algorithm to perform cutter selection optimization by considering multiple cutters. Our preliminary results are described in [12].

ACKNOWLEDGEMENT

This research has been supported by the NSF grants DMI9896255 and DMI9713718. Opinions expressed in this paper are those of authors and do not necessarily reflect opinion of the National Science Foundation.

REFERENCES

1. S. Arya, S. W. Cheng and D. M. Mount. Approximation algorithm for multiple-tool milling. *Proc. Of the 14th Annual ACM Symposium on Computational Geometry*, 1998, pp. 297-306.
2. M. Bala and T. C. Chang. Automatic cutter selection and optimal cutter-path generation for prismatic parts. *International Journal of Production Research*, 29(11), 1991, pp. 2163-2176.
3. Z. Dong, H. Li and G. W. Vicker. Optimal rough machining of sculptured parts on a CNC milling machine. *Transactions of ASME Journal of Engineering for Industry*, 115(64), 1993, pp. 424-431.
4. Y. S. Lee and T. C. Chang. Application of computational geometry in optimization 2.5D and 3D NC surface machining. *Computers in Industry*, 26(1), 1995, pp. 41-59.
5. Y. S. Lee and T. C. Chang. Automatic cutter selection for 5-axis sculptured surface machining. *International Journal of Production Research*, 34(4), 1996, pp. 977-998.
6. H. Li, Z. Dong and G. W. Vicker. Optimal toolpath pattern identification for single island, sculptured part rough machining using fuzzy pattern analysis. *Computer Aided Design*, 26(11), 1994, pp. 787-795.
7. T. Lim, J. Corney and D. E. R. Clark. Exact tool sizing for feature accessibility. *International Journal of Advanced Manufacturing Technology*, Vol.16, 2000, pp.791-802.
8. B. Mahadevan, L. Putta and S. Sarma. A feature free approach to tool selection and path planning in 3-axis rough cutting. *Proceedings of First International Conference on Responsive Manufacturing*, Nottingham, September 1997, pp.47-60.
9. G. Sun, F. Wang, P. Wright and C. Sequin. Operation decomposition for freeform surface features in process planning. In *Proc. DETC 1999: 1999 ASME Design Engineering Technical Conference*, Las Vegas, Nevada, September 12-15, 1999.
10. D. Veeramani and Y. S. Gau. Selection of an optimal set of cutting-tool sizes for 2.5D pocket machining. *Computer-Aided Design*, 29(12), 1997, pp.869-877.
11. D. C. H. Yang and Z. Han. Interference detection and optimal tool selection in 3-axis NC machining of free-form surface. *Computer-Aided Design*, 31(5), 1999, pp.303-315.
12. Z. Yao, S. K. Gupta and D. S. Nau. Selecting Flat End Mills for 2-1/2D Milling Operations. *ISR Technical Report, TR 2000-41*, University of Maryland, College Park, 2000.
13. C. F. You and C. H. Chu. An automatic path generation method of NC rough cut machining from solid models. *Computers in Industry*, 26(1), 1995, pp.161-173.