Journal of Manufacturing Processes
Vol. 3/No. 1
2001

A Geometric Algorithm for Finding the Largest Milling Cutter

Zhiyang Yao (*yaodan@Glue.umd.edu*) and **Satyandra K. Gupta** (*skgupta@eng.umd.edu*), Mechanical Engineering Dept. and Institute for Systems Research, and **Dana S. Nau** (*nau@cs.umd.edu*), Computer Science Dept. and Institute for Systems Research, University of Maryland, College Park, Maryland, USA

Abstract

This paper describes a new geometric algorithm to determine the largest feasible cutter size for 2-D milling operations to be performed using a single cutter. First is given 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. Discussed next are three alternative definitions of what it means for a cutter to be feasible, with explanations of which of these definitions is most appropriate for the above problem. Then, a geometric algorithm is presented for finding the maximal cutter for 2-D milling operations, and the algorithm is shown to be correct.

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

1. Introduction

Numerical control (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 (such as molds for injection molding). The importance of the machining process is increasing due to the latest advances in high-speed machining that allows machining to create even more complex shapes. Complex machined parts require several roughing and finishing passes. Selection of the right sets of tools and the right type of cutter trajectories is extremely important in ensuring high production rates 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 tool 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 (that is, pockets that have no open edges), despite the fact that open edges are very important in 2-D milling operations.
- Because 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.

This paper presents an algorithm for finding the maximal cutter for general 2-D milling operations to be performed using a single cutter.

- The general 2-D milling problem is formulated 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 Section 3 for definitions). The formulation allows arbitrarily complex starting stocks. Therefore, it can be used to model machining of geometrically complex castings such as engine blocks.
- Three different definitions are given of what it means for a cutter to be feasible, and it is explained why one of these definitions is more appropriate than the others.
- A new "region covering" algorithm is described that finds the largest cutter that can cover the target region without interfering with the obstruction region, and the correctness proof is given for the algorithm.

In practice, quite often multiple cutters are used to machine a complex milling feature. The region-covering algorithm is also useful as one of the stages in finding an optimal sequence of cutters for

machining a complex feature (for subsequent work on this subject, please see Yao, Gupta, and Nau[1]).

2. Related Work

Because of the wide range of the complexity of products, requirements for machine accuracy, different machining stages, selecting optimal cutter size is an active research area. Below is 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 of the features found on prismatic parts, such as slots and 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 cutters to save processing time. Bala and Chang take both cutter change time and geometric constraints into consideration. They state the problem as follows. If there exists a set of cutters and, after machining with these cutters, only finish machining is needed for fillet radii corners, 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 Bala and Chang 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, Bala and Chang 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 are determined. The smallest of all these distances gives the smallest constriction within the pocket. After determining the cutter size, the feasible cutter motion region can be identified and the cutter movement within the region can be optimized. By using the Bala and Chang algorithm, the maximal cutter that can cover the target region will be the one that is based on Alternative 1 of cutter feasible definition (see Section 3.1 for details).

Veeramani and Gau have developed a two-phase method to select a set of cutters.[3] In the first phase, a concept called the Voronoi mountain is employed to calculate the material volume that can be removed by a specific cutting tool size, the material volume remaining 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 the 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 three-axis CNC machines using ball-end cutters.[4] To find the optimal tools, a comparison of all possible combinations of tools is performed. The maximum number of selected cutters is restricted to three. This optimal tool selection method is designed 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 the 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 three-axis machining.[5] The algorithm is based on the 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 a less reachable region than small tools, they incur the penalty of tool changes, and 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 research in the area of cutter selection includes work by Arya, Cheng, and Mount[6] on multiple tool selection; Dong, Li, and Vicker[7] and Li, Dong, and Vicker[8] on rough machining of sculptured parts; Lee and Chang[9,10] on 2.5-D and 3-D NC surface machining and five-axis sculptured surface machining; Lim, Corney, and Clark[11] on tool sizing for feature accessibility; Sun et al.[12] on operation decomposition for freeform surface features; and You and Chu[13] on NC rough cut machining. These works are 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 (called the *target region, T*), there is also an *obstruction region, O*, a region that the tool should not cut during machining. An example is shown in *Figure 1*. The target region and the obstruc-

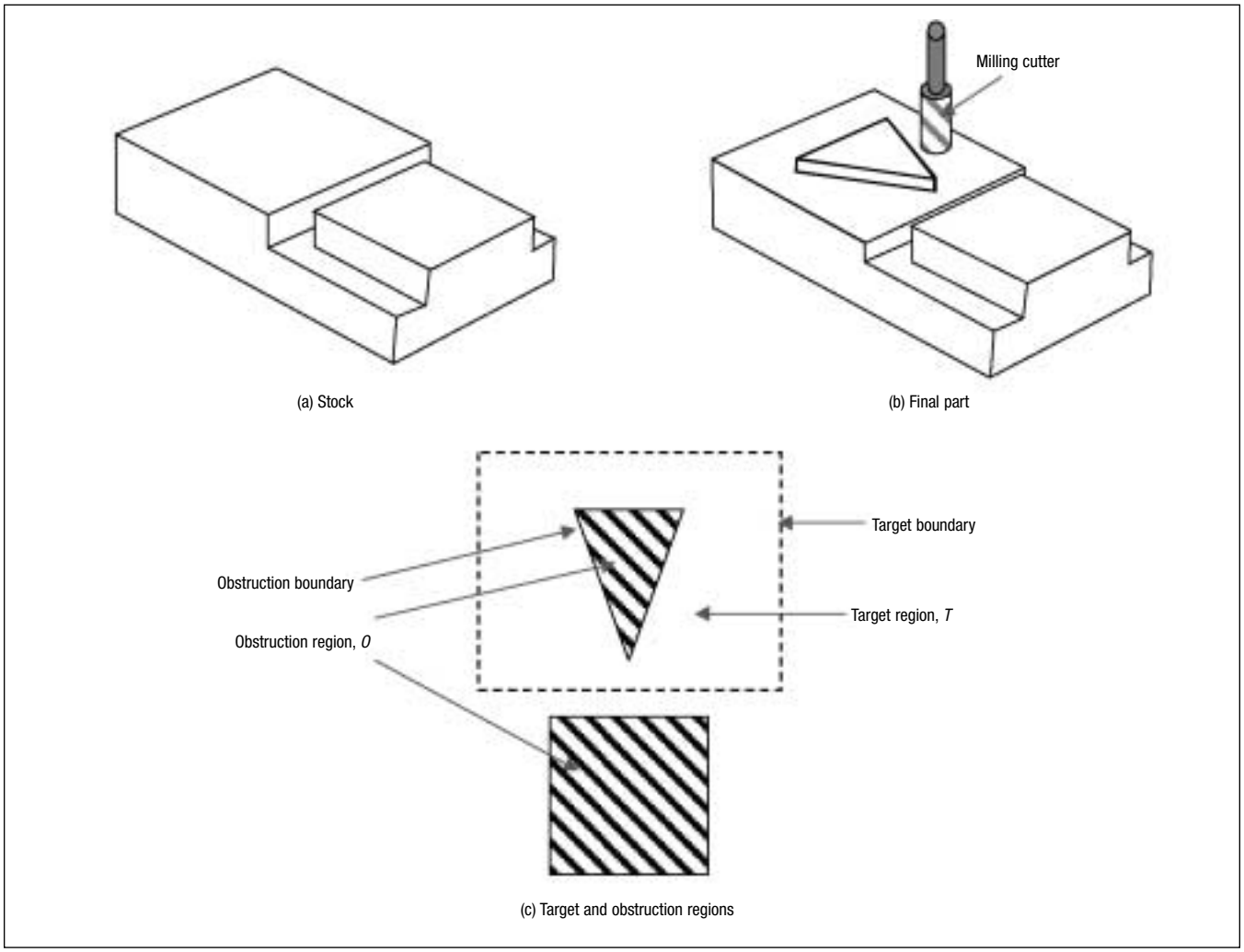

Figure 1
Examples of Stock, Final Part, Target Region, and Obstruction Region

tion region must both be regular sets but may each consist of a number of non-adjacent subregions:

$$T = T_1 \cup ... \cup T_j$$

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

It is assumed that the boundary of each subregion consists only of line segments and segments of circles.

As shown in *Figure 1c*, 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 of the obstruction boundary are called *obstruction edges*. An edge of the target boundary is called a *closed edge* if it coexists with an obstruction edge; otherwise, it is called an *open edge*. (Note that 2-D closed pockets do not have any open edges.) Dashed lines represent open edges, solid lines represent closed edges, and diagonal stripes represent the obstruction region.

Figure 2 shows examples of open and closed edges. Each closed edge, *e*, separates the *material* (that is, 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*:

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

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

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

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*) $\cap^*$ $O = \emptyset$.

where $\cap^*$ is regularized intersection. Similarly, $-^*$ denotes regularized difference and $\cup^*$ denotes regularized union.

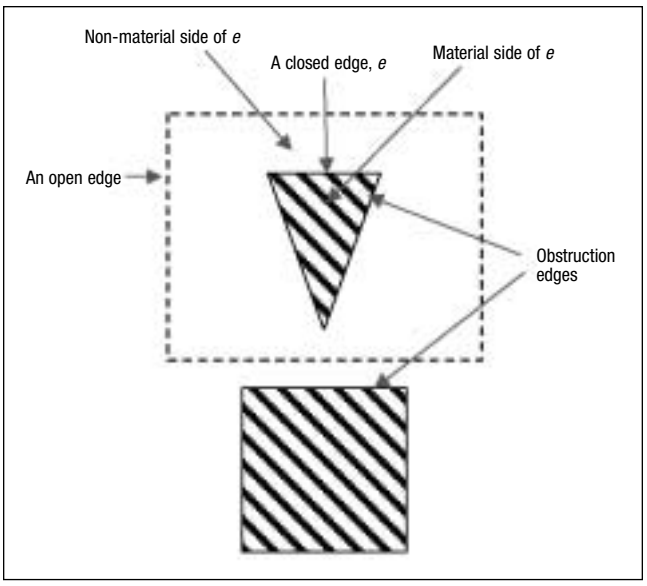

Figure 2
Examples of Open, Closed, and Obstruction Edges

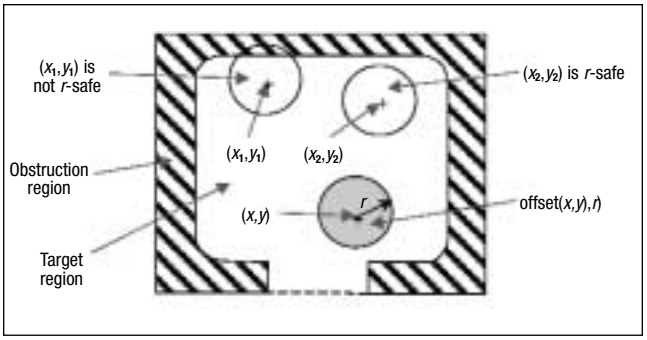

Figure 3
An *r*-offset Region, and Locations that are *r*-safe and Not *r*-safe

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*) = Ø. Safe(*S,r*) is defined 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.

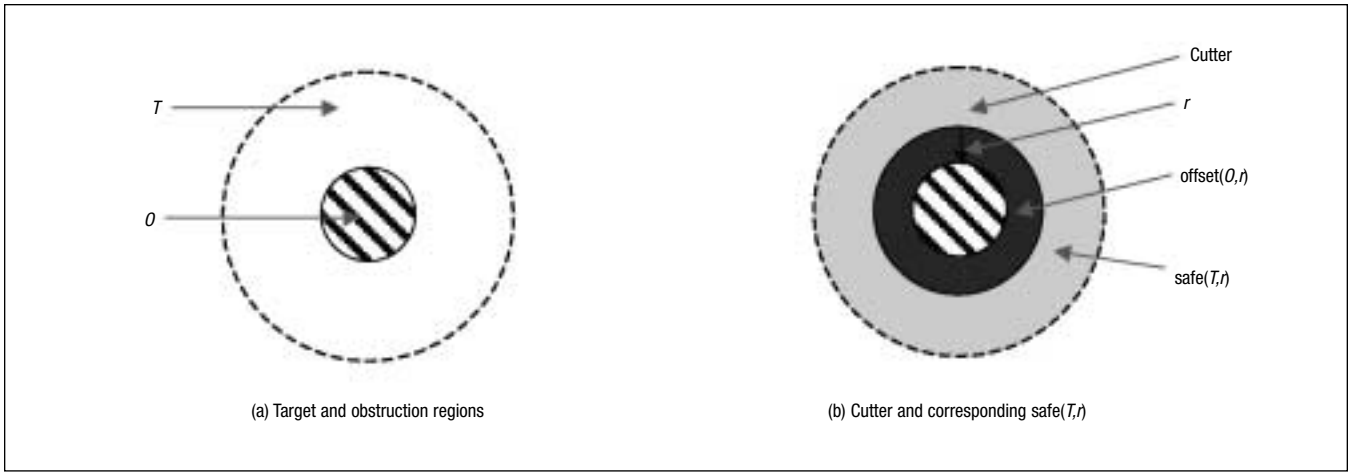

Figure 4
Example of safe(*T,r*)

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, it is first noted 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 "offset" is used for "r-offset," "cuttable" is used 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 5a*. This is equivalent to saying that a cutting tool C of radius r is feasible if T's Voronoi diagram[6] is r-safe. In *Figure 5a*, C_1 is an example of the largest feasible cutter based on this definition.

Alternative 2: cutter feasibility based on a continuous tool path. Rather than forcing the cutter to go through every bottleneck, the cutter can be allowed to go around some of them instead (*Figure 5b*). 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 5b*, C_2 is an example of the largest feasible cutter based on this definition.

Alternative 3: cutter feasibility based on cuttability. If the machining process can be interrupted briefly, each bottleneck can be jumped over by lifting the cutter up and putting it down again on the other side of the bottleneck (*Figure 5c*). Thus, it might be said 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 5c*, 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.

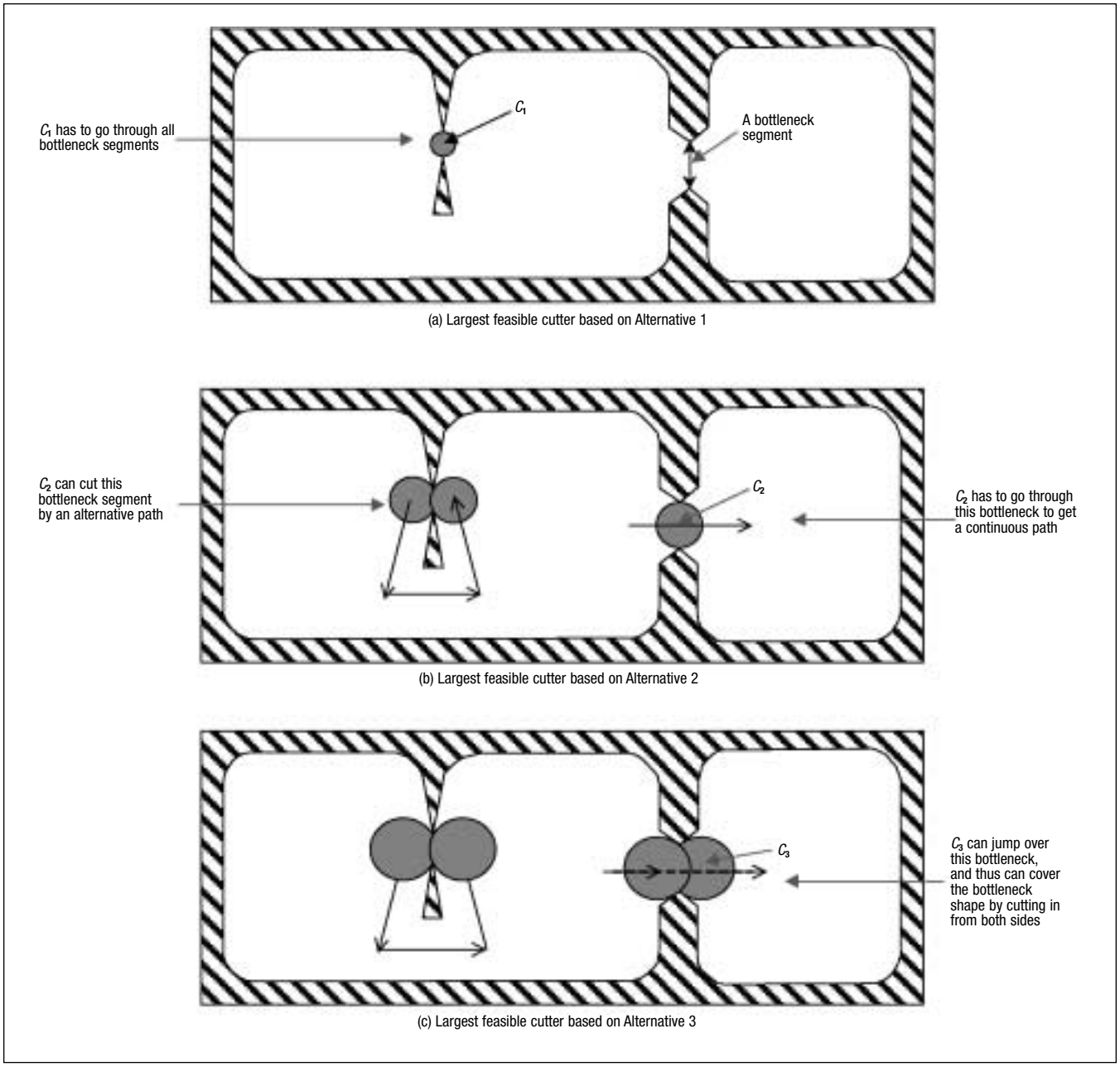

Figure 5
Different Largest Feasible Cutters Resulting from Different Definitions of Cutter Feasibility

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 the largest cutter, it is the definition of cutter feasibility that is preferable in many machining problems. Thus, the definition is what is used in this paper.

Problem Formulation: With the above definition in mind, the *cutter selection problem* is defined 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, it is said that the cutter selection problem is *unsolvable*. In all other cases, it is *solvable*.

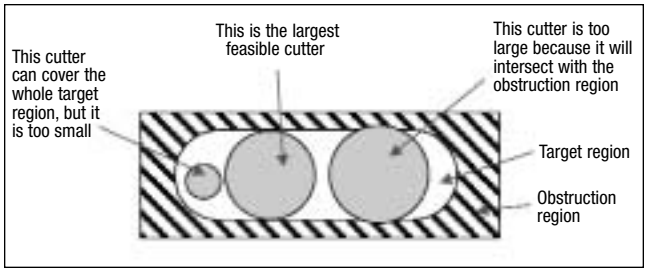

Figure 6
Simple Example of Maximal Cutting Tool

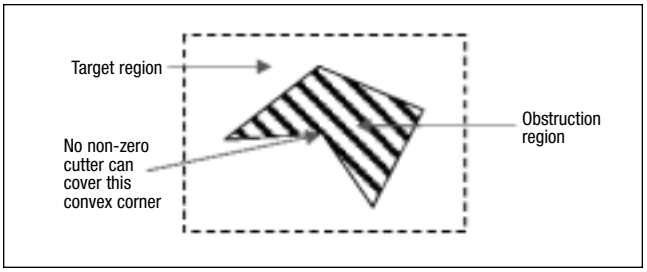

Figure 7
Example of Unsolvable Cutter Selection Problem

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 8c* 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 8c*. From this definition, the following lemma follows immediately:

Lemma 3.

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

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:

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

Below are 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°.
- 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. Because the above examples suggest 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. It will first be proved 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, that is, 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 $R = \text{offset}(q,r)$. Every point of $\text{offset}(q,r)$ is in $E(r)$, so $\text{offset}(q,r) \cap^* O = \emptyset$, whence *q* is *r*-safe. Thus *p* is in the *r*-offset region of an *r*-safe point, so *p* is *r*-cuttable.

Now, it will be proved that if $E(r) \cap^* O \neq \emptyset$ then some point in $T -^* K(r)$ is not *r*-cuttable. Suppose $E(r) \cap^*$

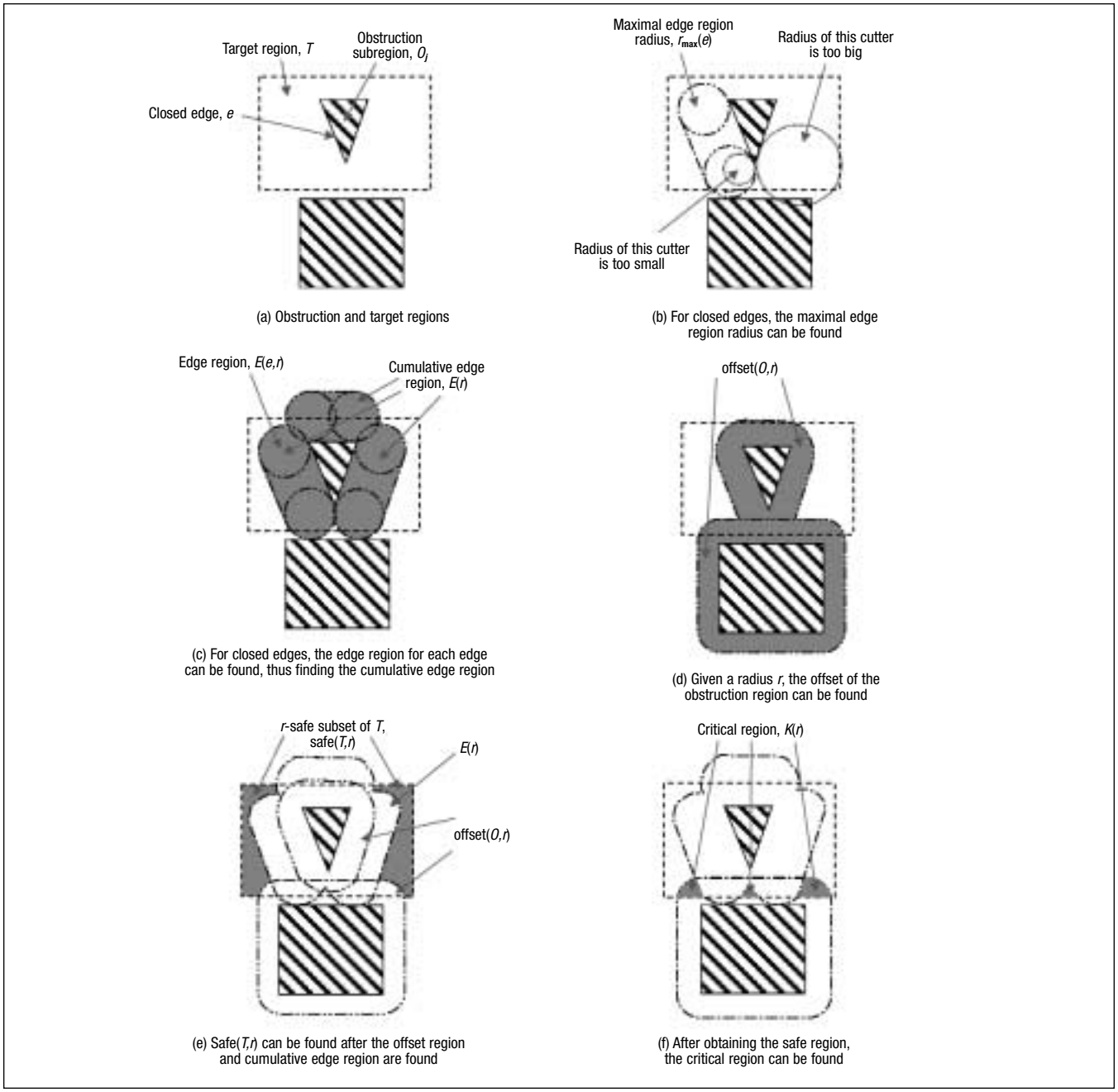

Figure 8
Examples of Edge Region, Safe Region, and Critical Region

$O \neq \emptyset$. 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 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.

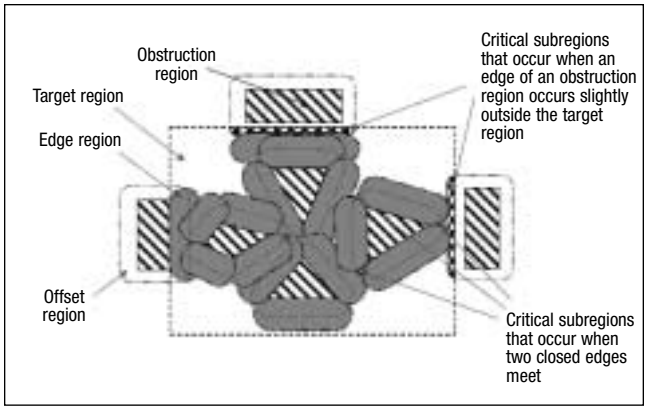

Figure 9
Some Examples of Critical Regions

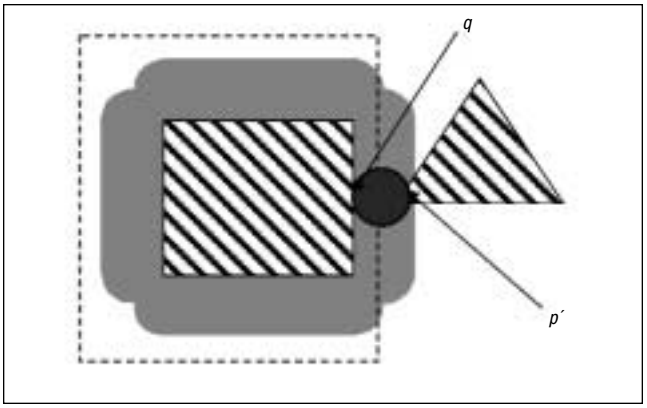

Figure 10
In this example, if $E(r) \cap^* O \neq \emptyset$, then some points in $T -^* K(r)$ are not r-cuttable

4. Algorithm for Finding the Maximal Cutter

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

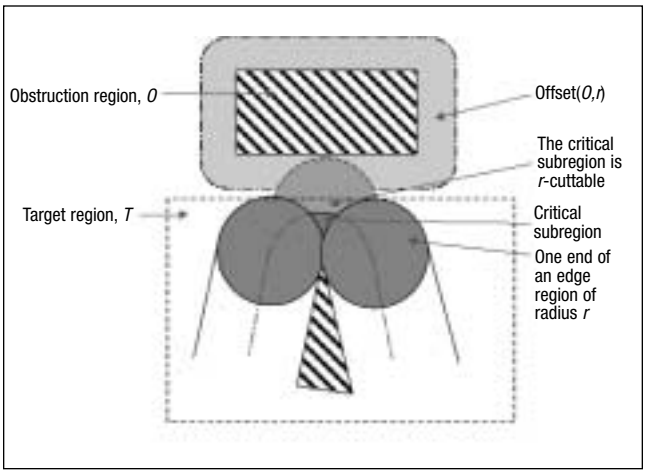

Figure 11
Example of a Critical Subregion That Can Be Covered Without Reducing the Tool Radius

8. Else return $r = r_E$;

5. Finding the Maximal Swept Cutter for a Closed Edge

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 this arc is split into two arc segments such that each arc segment's angle is less than or equal to 180°. For each closed edge a, the following is done:

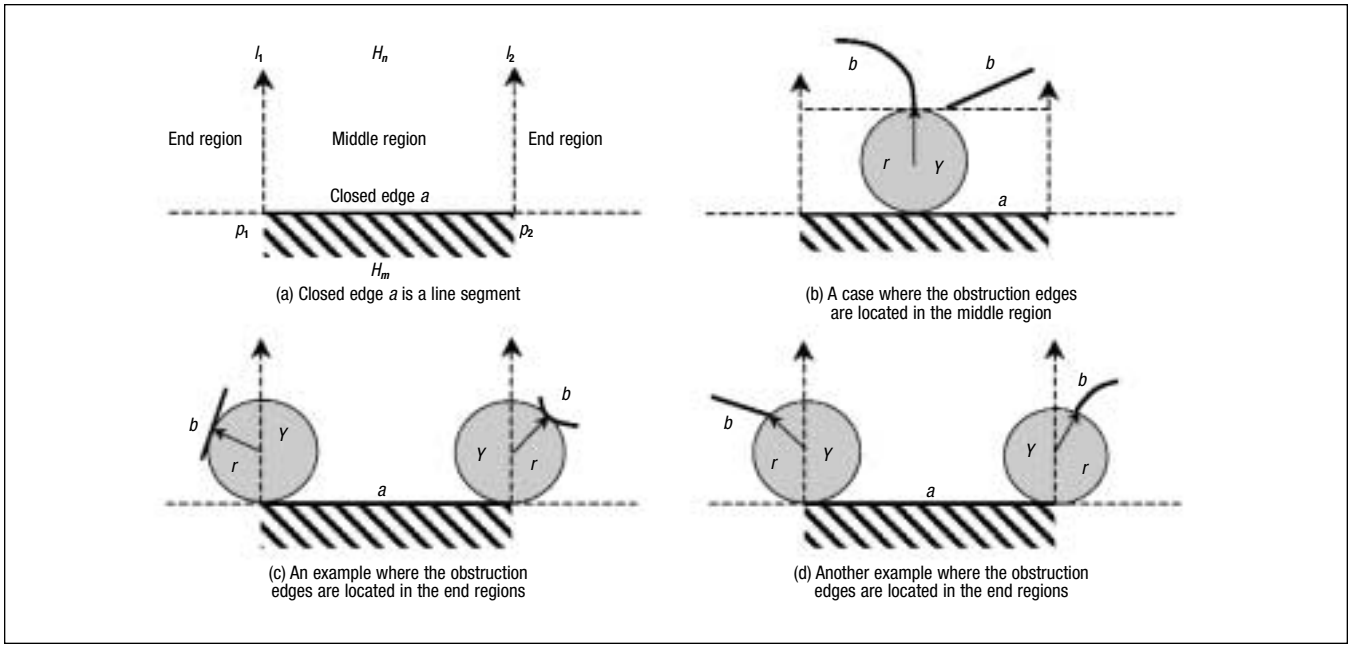

Figure 12
Finding Maximal Cutter Radius for a Linear Closed Edge in Presence of Different Types of Obstruction Edges

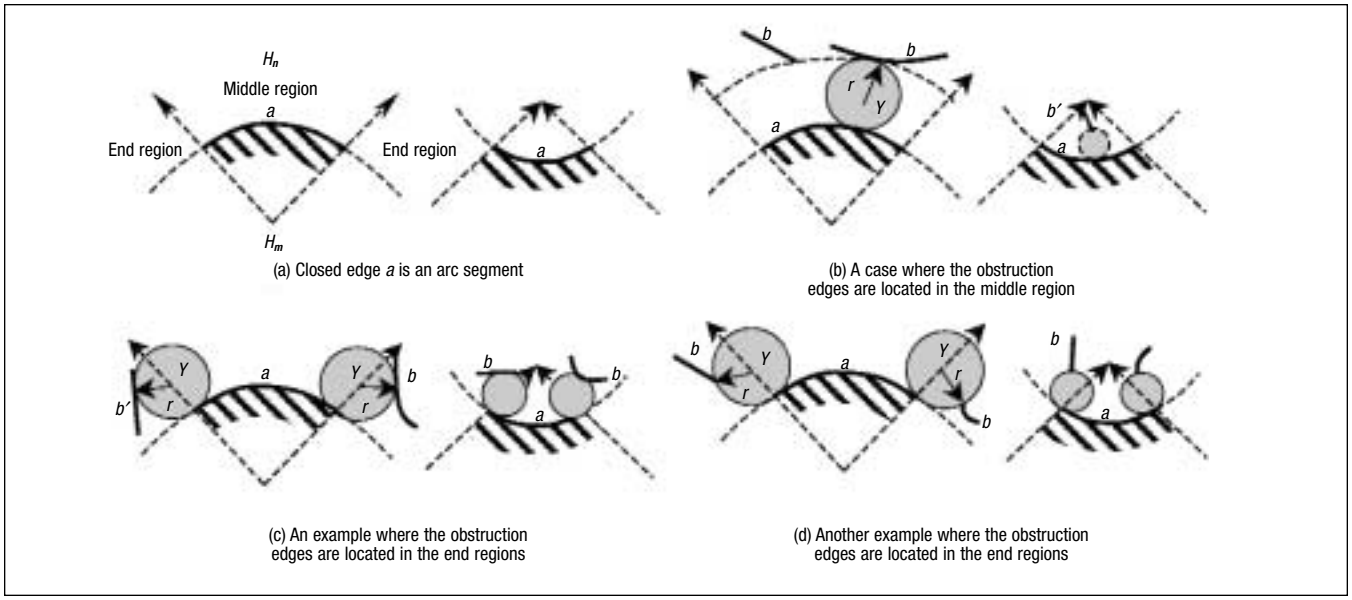

Figure 13
Finding Maximal Cutter Radius for Circular Closed Edge in Presence of Different Types of Obstruction Edges

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-spaces, H_m and H_n. H_m is the half-space that contains the material side of *a*. H_n is half-space that contains the 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*. *Figure 12a* shows examples of these regions when *a* is a line segment, and *Figure 13a* 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 is 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 the 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 *Figures 12b* and *13b*.
3. If *b* is in the end region of *a*, then
   - Let *e* be whichever end point of *a* is closest 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 a circle can be found that is tangent to both *a* and *b* at *p*, then *Y* is that circle. Otherwise, *Y* is found by finding the minimal circle that is tangent to *a* at *p* and passes through one end point of *b*. Examples are shown in *Figures 12c* and *12d* and *Figures 13c* and *13d*.

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 is available on 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 *U* is computed by computing the bounding box of $T \cup^* O$.
2. loop
   - $P = U -\!\!-^*$ 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 *Figures 14c* and *14d*, $r_K = r_E$, and $K \not\subset S$. After one or more iterations, $r_K = r_E - cr_I$, where *c* is a constant, then $K \subseteq S$ as shown in *Figures 14e* and *14f*.

7. Discussion of Correctness of Algorithm

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

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

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 $\max\{r : E(e,r) \cap^* O = \emptyset\}$. Thus from Lemma 3, r_E is the largest number such that $E(r_E) \cap^* O = \emptyset$. Therefore, from Theorem 1, a cutter of radius r_E will be able to cut all of *T*. For any $r > r_E$, $E(r) \cap^* O \neq \emptyset$. 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).

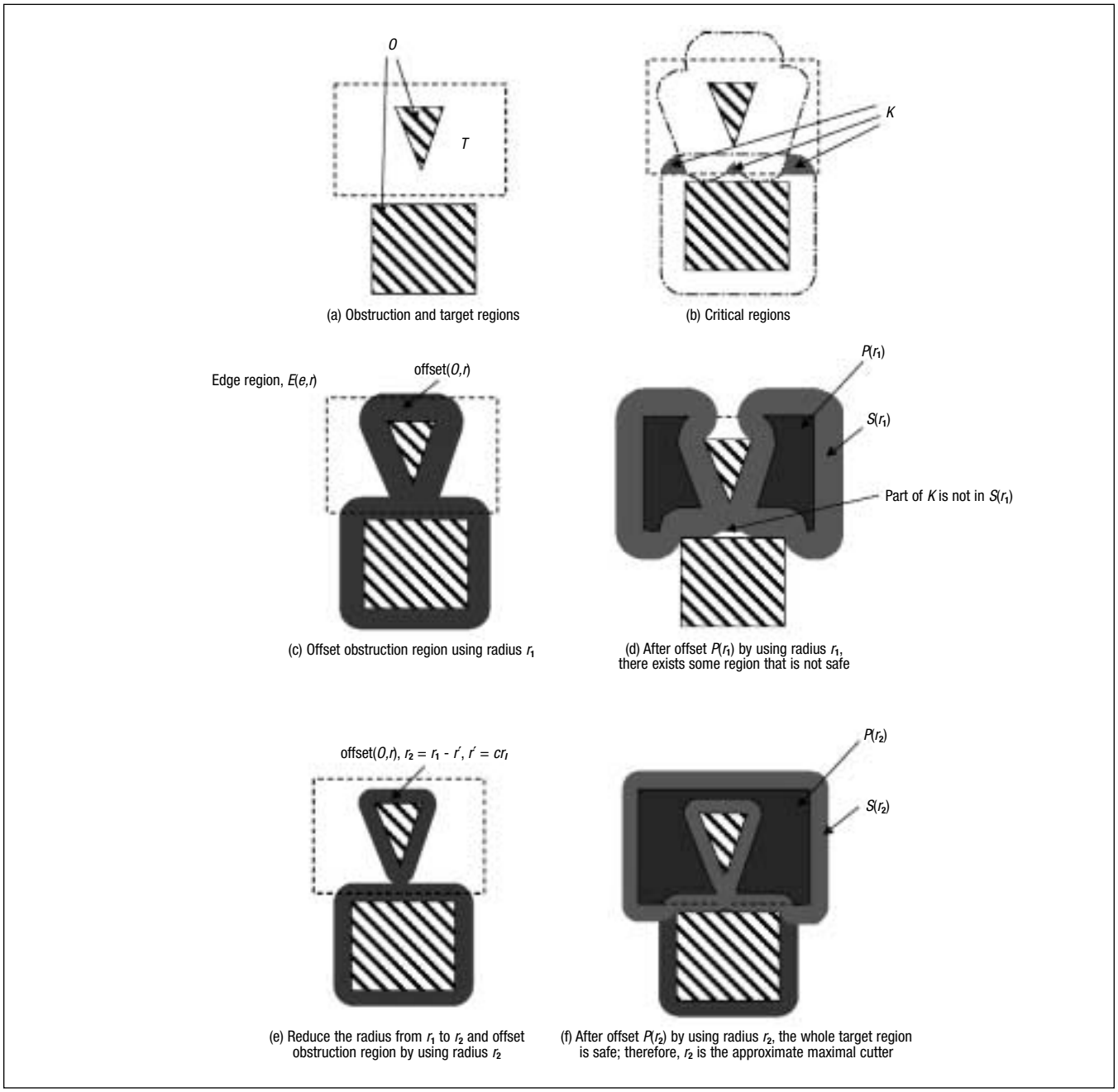

Figure 14
Using Approximation Algorithm to Find Near Maximal Cutter for Critical Region

Proof. Let r be the number returned by ***FMCR***. Here $r = \min\{ r_E, r_K \}$. Then $E(r) \cap^* O = \emptyset$, so from Theorem 1 it is known 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 the 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

The example shown in the *Figure 15* is used to illustrate the operation of the algorithm. The target

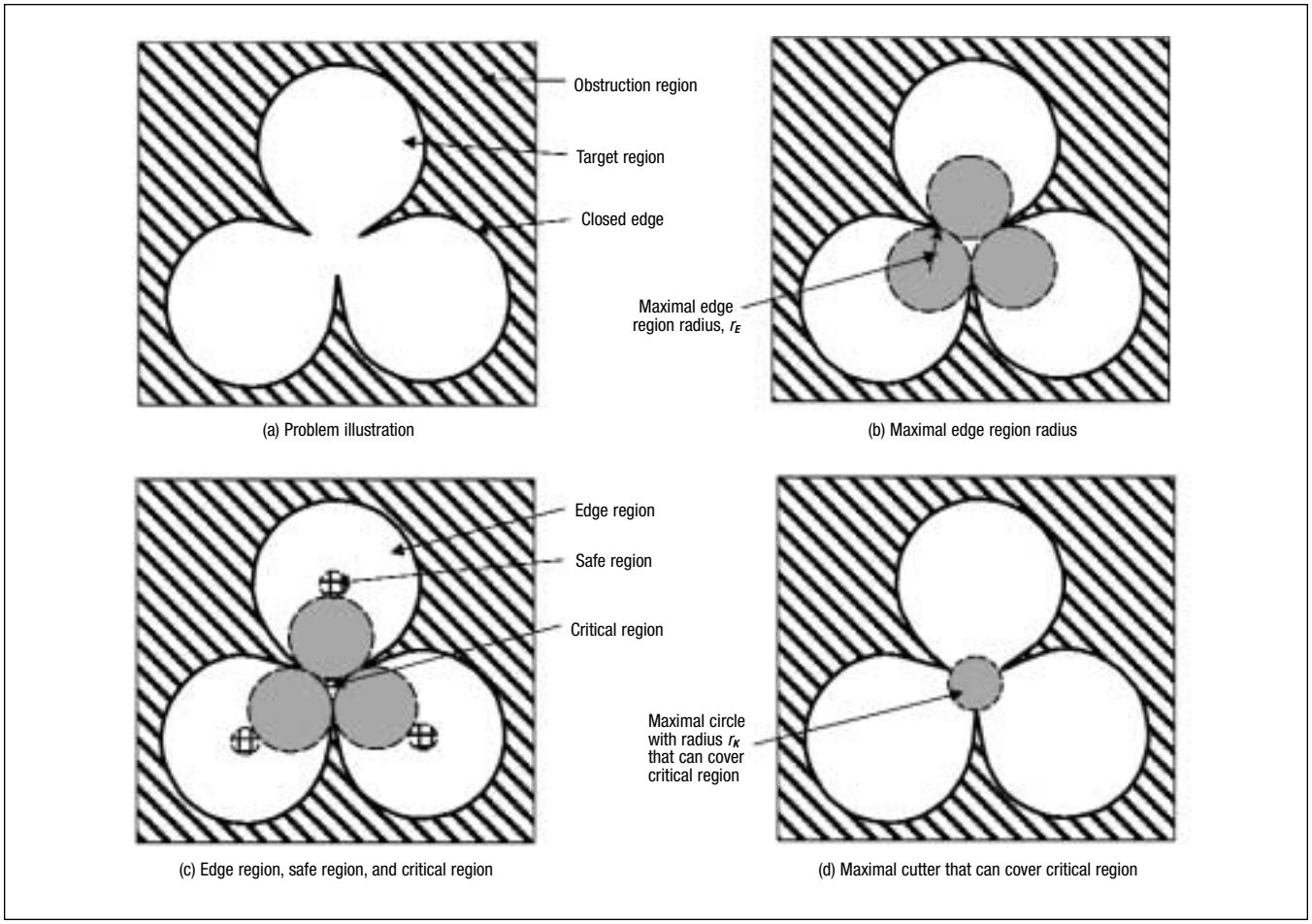

Figure 15
Example of Operation of Algorithm

region and obstruction regions are shown in *Figure 15a*. The details are as follows: The 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.

- First, for every closed edge *a* is found the maximal edge-region radius for *a* (shown in *Figure 15b*).
- The smallest of those radii, r_E, is used to compute the cumulative edge-region $E(r_E)$ and safe region *S* (shown in *Figure 15c*).
- Then the critical region *K* is obtained. For the critical region is found the maximal cutter with radius r_K that covers it and does not interfere with the obstruction region, as shown in *Figure 15d*.
- 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.

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

Figures 16-19 show the results of the maximal cutters selected by the algorithm. The algorithm

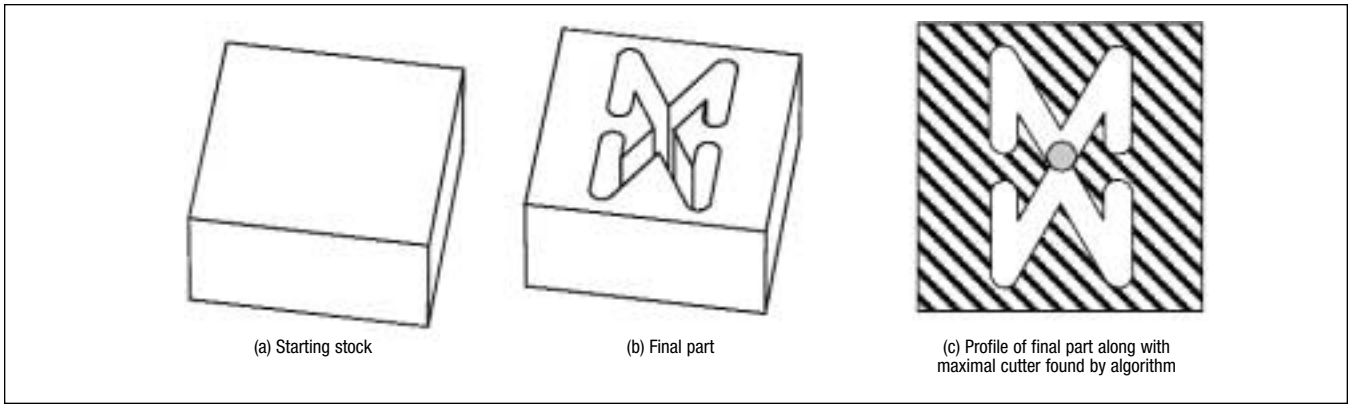

Figure 16
Example 1

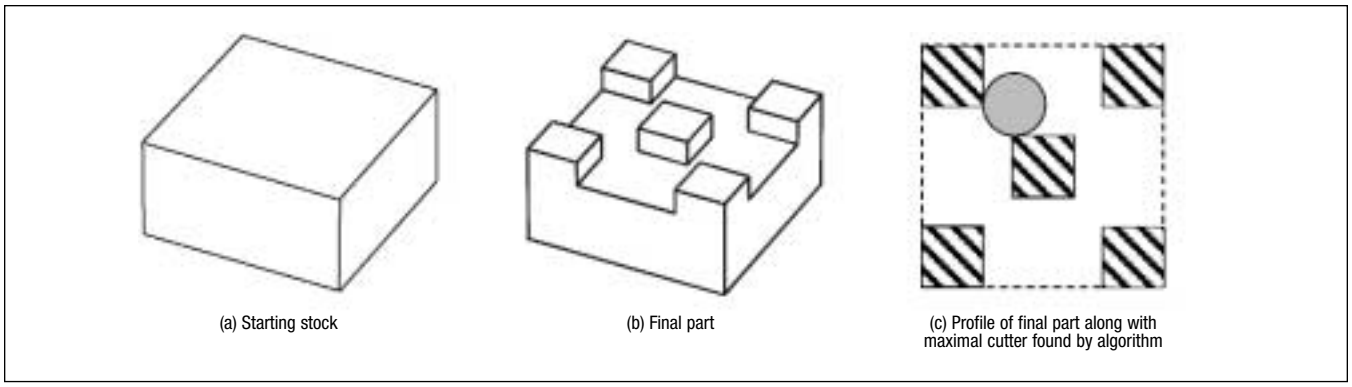

Figure 17
Example 2

solved every one of those examples in less than one second on an Ultra 10 computer.

9. Conclusion and Discussion

This paper presented a geometric algorithm for finding the maximal cutter size for a 2-D milling process. The 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, the 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. The algorithm uses a cutter feasibility definition based on the 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 (for example, either based on covering every bottleneck in the target region or existence of a continuous path between every pair of points in the target region).

The maximal cutter found is based on the geometric constraints. In actual machining, it is necessary to consider several other cutting constraints. Here are some examples:

- There are several cutting parameters that may influence the selection of cutter size. Cutters that conflict with the machining constraints cannot be used. For example, it is known that the material removal rate is proportional to the diameter of the cutter in milling operations. As a result, if a bigger cutter is used, there can be a higher material removal rate, thus saving 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.

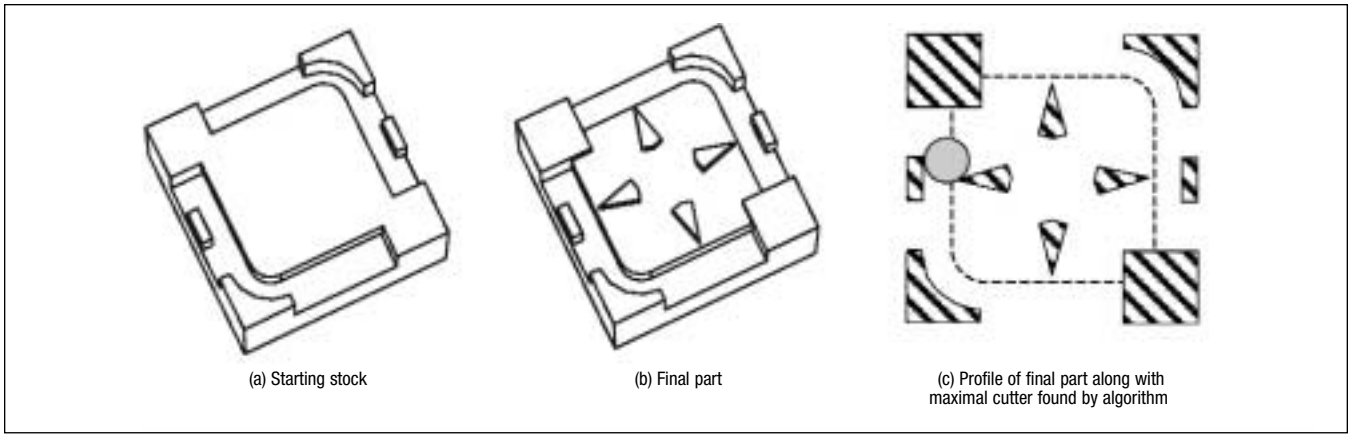

Figure 18
Example 3

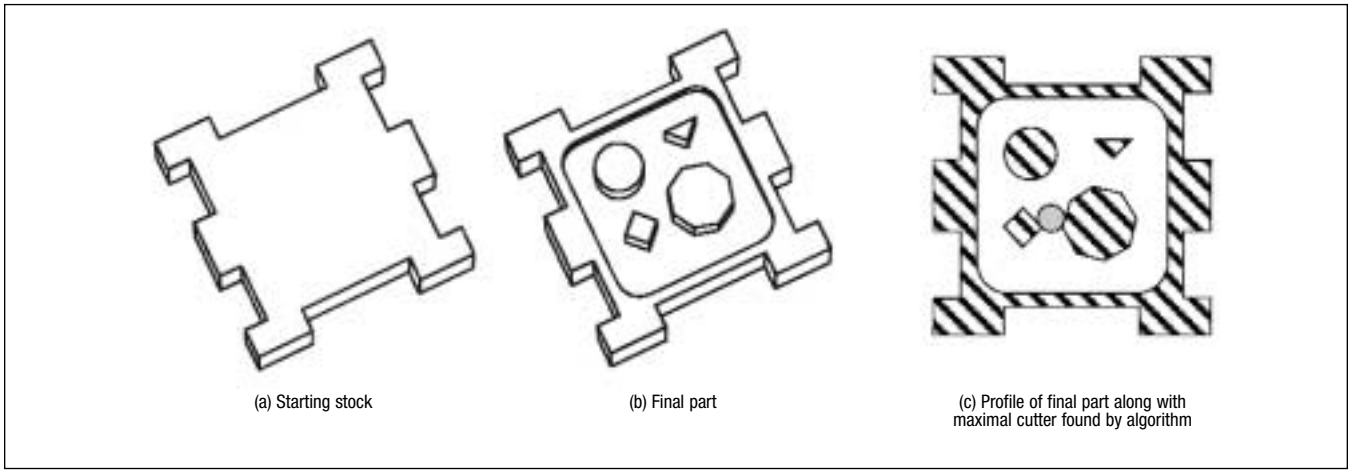

Figure 19
Example 4

- Sometimes the cutter selected by the region-covering idea may not be the best one for manufacturing. For example, *Figure 20* shows an example in which, if the maximal cutter selected by the algorithm is used, it will have to be lifted up and put down several times. In this particular case, the maximal cutter selected may not save total cutting time and may result in a bad manufacturing surface. Besides the geometric constraints, manufacturing knowledge is also needed to help 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.

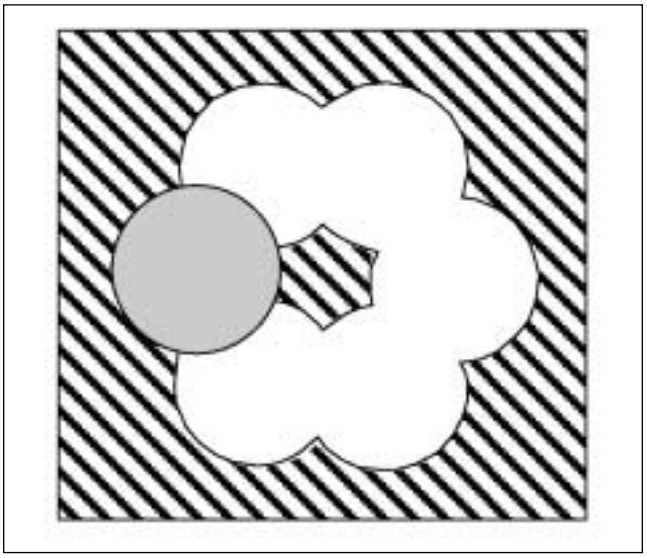

Figure 20
Manufacturing Consideration in Choosing Maximal Cutter

The algorithm is currently being extended to perform cutter selection optimization by considering multiple cutters. Preliminary results are described in Yao, Gupta, and Nau.[1]

Acknowledgment

This research has been supported by NSF grants DMI9896255, DMI9713718, and EIA-9729827. Opinions expressed in this paper are those of the authors and do not necessarily reflect the opinion of the National Science Foundation.

References

1. 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 (College Park, MD: Univ. of Maryland, 2000).*

2. M. Bala and T.C. Chang, "Automatic Cutter Selection and Optimal Cutter-path Generation for Prismatic Parts," *Int'l Journal of Production Research* (v29, n11, 1991), pp2163-2176.

3. D. Veeramani and Y.S. Gau, "Selection of an Optimal Set of Cutting-tool Sizes for 2.5D Pocket Machining," *Computer Aided Design* (v29, n12, 1997), pp869-877.

4. 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* (v31, n5, 1999), pp303-315.

5. B. Mahadevan, L. Putta, and S. Sarma, "A Feature Free Approach to Tool Selection and Path Planning in 3-axis Rough Cutting," *Proc.* of 1st Int'l Conf. on Responsive Mfg., Nottingham, UK, Sept. 1997, pp47-60.

6. S. Arya, S.W. Cheng, and D.M. Mount, "Approximation Algorithm for Multiple-tool Milling," *Proc.* of 14th Annual ACM Symp. on Computational Geometry, 1998, pp297-306.

7. Z. Dong, H. Li, and G.W. Vicker, "Optimal Rough Machining of Sculptured Parts on a CNC Milling Machine," *Trans. of ASME, Journal of Engg. for Industry* (v115, n64, 1993), pp424-431.

8. 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* (v26, n11, 1994), pp787-795.

9. Y.S. Lee and T.C. Chang, "Application of Computational Geometry in Optimization 2.5D and 3D NC Surface Machining," *Computers in Industry* (v26, n1, 1995), pp41-59.

10. Y.S. Lee and T.C. Chang, "Automatic Cutter Selection for 5-axis Sculptured Surface Machining," *Int'l Journal of Production Research* (v34, n4, 1996), pp977-998.

11. T. Lim, J. Corney, and D.E.R. Clark, "Exact Tool Sizing for Feature Accessibility," *Int'l Journal of Advanced Mfg. Technology* (v16, 2000), pp791-802.

12. G. Sun, F. Wang, P. Wright, and C. Sequin, "Operation Decomposition for Freeform Surface Features in Process Planning," *Proc.* of DETC 1999: 1999 ASME Design Engg. Technical Conf., Las Vegas, NV, Sept. 12-15, 1999.

13. C.F. You and C.H. Chu, "An Automatic Path Generation Method of NC Rough Cut Machining from Solid Models," *Computers in Industry* (v26, n1, 1995), pp161-173.

Authors' Biographies

Zhiyang Yao is a PhD student in the mechanical engineering department at the University of Maryland, College Park. He received a BS in 1995 and an MS in 1998 in mechanical engineering from Tsinghua University, P.R. China. His research interests are computer-aided design and manufacturing, concurrent engineering, and geometric reasoning. Currently, he is working on constructing innovative process plans that can significantly reduce time to market and enable cost-effective small-batch manufacturing. He has authored or coauthored 11 articles in journals, conference proceedings, and technical reports. He is a student member of ASME.

Satyandra K. Gupta is an assistant professor at the University of Maryland in the mechanical engineering department and the Institute for Systems Research (ISR). He received a PhD in mechanical engineering from the University of Maryland at College Park in 1994. Prior to joining the University of Maryland, he was a research scientist in the Robotics Institute and an adjunct assistant professor of manufacturing in the Graduate School of Industrial Administration at Carnegie Mellon University. The objective of Dr. Gupta's research is to provide innovative product realization methodologies that can significantly reduce time to market and enable cost-effective small-batch manufacturing. Over the last 10 years, Dr. Gupta has participated in a number of design and manufacturing automation research projects. Representative projects include generative process planning for machining, automated manufacturability analysis, automated redesign, generative process planning for sheet metal bending, assembly planning and simulation, extraction of lumped parameter simulation models for microelectromechanical systems, distributed design and manufacturing for solid freeform fabrication, and automated mold design and fabrication. He has authored or coauthored more than 70 articles in journals, conference proceedings, and book chapters. He has organized several conference sessions on computer-aided design and manufacturing areas. Dr. Gupta has won many honors and awards for his academic excellence and his research contribution to design and manufacturing automation area.

Dana S. Nau is a professor at the University of Maryland in the Department of Computer Science and the Institute for Systems Research (ISR). He also has affiliate appointments with the Institute for Advanced Computer Studies and the Department of Mechanical Engineering. Dr. Nau received a BS in applied mathematics from the University of Missouri at Rolla in 1974. He received an AM (in 1976) and PhD (in 1979) in computer science from Duke University, where he was an NSF graduate fellow and a James B. Duke graduate fellow. He has had summer and/or sabbatical appointments at IBM Research, NIST, the University of Rochester, and General Motors Research Laboratories. Dr. Nau's research interests include AI planning and searching techniques and computer-integrated design and manufacturing. He has co-edited two books and has published more than 200 refereed technical papers. Copies of recent papers and summaries of current research projects are available at *http://www.cs.umd.edu/users/nau*.