Proceedings of IMECE'04
2004 ASME International Mechanical Engineering Congress and RD&D Expo
Anaheim, California, USA, November 13-19, 2004

IMECE2004-62224

FINDING AN OPTIMAL SET OF CUTTER RADII FOR 2D POCKET MACHINING

Irena Nadjakova
Computer Sciene
University of California
Berkeley, California 94720
Email: ienadjak@eecs.berkeley.edu

Sara McMains
Mechanical Engineering
University of California
Berkeley, Califronia 94720
Email: mcmains@me.berkeley.edu

ABSTRACT

We describe an approach to finding an optimal (within a requested approximation ratio) set of cutter radii for machining a given 2-dimensional pocket. We do not assume that there is a predetermined set of cutter radii to choose from or a predetermined number of cutters to use.

Given an initial set of cutters to choose from, we derive an upper bound on the approximation ratio of what is achievable choosing from this set compared to the ideal set. We then use this bound to subdivide the intervals between the given radii until the requested approximation ratio is achieved.

We also look at the machinable area as a function of the tool radius. We show that this area is continuous everywhere, except at a certain set of radii determinable by constructing the Voronoi diagram of the pocket. This lets us avoid subdividing the intervals around the points of discontinuity, improving both running time and the size of the output tool set.

INTRODUCTION

As defined in [1],

> Milling is the mechanical process of removing material from a piece of stock through the use of a rapidly spinning circular milling tool in order to form some desired geometric shape... Among the most common milling problems is simple 2-dimensional pocket milling: cut a given 2-dimensional region – the pocket – down to some constant depth using a given set of milling tools.

This 2-dimensional problem typically arises as a subproblem of 2.5D milling: the part that needs to be cut is horizontally split into constant depth layers and the pocket on each layer is a subset of the pocket on the layer above. Thus, we can generate a plan for cutting out the pocket of one layer independently of the plan for the pocket on any other layer, reducing both the dimensionality and, hopefully, the complexity of our problem.

The problem of cutting out a 2D pocket can be split into two subproblems as well. One is, given a tool and an area to cut, generate the most efficient path for the tool to cut out the area. In this paper, we are assuming that we are given an algorithm to generate this tool-path for us. Another problem – the one we are addressing – is choosing a set of tool radii to use.

If we are just using one tool, the radius has to be small enough to fit in every "corner" of the pocket we want to cut out without gouging the given pocket boundary. Unfortunately, such a tool may be too small to efficiently clear out large spaces within the pocket interior. A solution to this is to first use one or more larger tools to quickly cut out large areas of the pocket and then smaller tools to cut out the remaining material from the areas where the larger tool couldn't fit. Now the question is which larger tools should we use? Even if our choice is restricted to one large and one small tool, picking an efficient set may be hard: should we choose a big and very fast tool after which we have to do a lot of cleanup in the corners, or a smaller and slower tool after which we just have to go around the boundary once, if at all?

PROBLEM STATEMENT

The problem we address is, given a 2-dimensional pocket or set of pockets in the same layer, find an "optimal" set of cutter diameters we can use to machine them. An optimal set is a set for which the total "cost" (the machining time) of machining is minimized. The total cost is the sum of the costs of using each individual tool in the sequence. Each individual cost is assumed to be given to us by the tool path generation subroutine. We do *not* assume that we are given a fixed set of cutters to pick our tools from; we generate the set of radii from scratch to better optimize.

The pocket is defined by its boundary B, which we assume has been preprocessed to meet certain restrictions. The boundary is defined by one or more closed, non-intersecting sequences of line segments and circular arcs, its "edges." (For a pocket with a general curved boundary, preprocessing must approximate those curves with circular arcs.) The edges define one or more outer pocket boundaries and zero or more non-machinable islands within the outer boundaries. The angle between each pair of consecutive edges has to be *reflex*, i.e. greater than or equal to 180^o as seen from the inside of the pocket. This is necessary to ensure a machinable pocket, since if a boundary contained a non-reflex angle, no matter how small of a radius we picked for our flat end-mill, there would always be some area in the corner that the tool could not reach. (Again, in practice this restriction is removed by preprocessing, using the position tolerance for the part definition as a fillet radius to fillet non-reflex angles.)

PREVIOUS WORK

Early work in the area of cutting tool selection tried to find the single best tool to mill a particular feature, with customized decomposition strategies [2,3]. Bala and Chang introduced a tool selection algorithm that chooses two different tool sizes to mill a pocket more efficiently than would be possible with a single tool [4]. Instead of trying to build an optimal set of tools from scratch, most researchers who look at multiple tool solutions assume that a fixed set of tools is given, and then try to pick a best possible subset from those tools for machining a particular pocket. Veeramani and Gau developed an algorithm that could be used with an unlimited number of tool sizes, working from a database of potential tools (four in the examples in their paper), using a dynamic programming approach to find the optimal sequence, and a Voronoi mountain approach to calculate the area machined by each tool [5].

Balasubramaniam et al. observe that milling can be modeled using a network flow formulation. They reduce the computational complexity from an intractable NP-hard problem with 2^n possible permutations of n tools, making simplifying assumptions that allow them to sort and cluster tools based on diameter and material removal rate to obtain a near-optimal solution, using Dijkstra's algorithm to find the shortest path through the network in $O(n^2)$ time [6]. They also extended the concept of machinable area to include the constraints not only of 2D accessibility based on the tool diameter, but also the 3D constraints on tool movement due to tool length, tool holder, and spindle geometry. They show examples on a databases of up to twelve potential tools. Yao et al. take a similar approach, using a network flow graph and Dijkstra's algorithm [7, 8]. They look for the optimal set of cutters for a whole set of parts, to take into account the cost of tool loading time when the tool changer does not hold all the tools needed for the different parts. They show results obtained with databases of ten to twelve tools. D'Souza et al. also build a network flow graph and use Dijkstra's algorithm, calculating actual tool paths for more accurate cost estimates [9–11]. They show examples using a database of six to seven tools.

These algorithms are appropriate for tool selection in a job shop environment focused on quick turnaround for high-mix, low-volume parts, such as are typically found at national labs or universities, where the choice of tools is typically limited to a small number currently on hand. The algorithms' asymptotic computational complexity, however, limits their scalability to databases containing the thousands of different tools used in industry. (Balasubramaniam et al. [6] report that a one hour aerospace job may require several hundred tools for a single part!) Thus in our current work, instead of starting from a small fixed database of tool sizes, we start from the geometry of a given pocket in order to determine what tool sizes it makes sense to consider for cutting that particular geometry. This eliminates testing for tool sizes that are unlikely to further improve processing time while simultaneously expanding the potential choices beyond the tools currently on hand or entered into the database. This approach is most suitable for process planning for large production runs, where tools will be ordered for a specific production line in such large quantities over the life of the product (due to tool breakage and wear) that a machine tool manufacturer would be open to a special order for a tool of a particular size even if it did not appear in their current catalog.

We are using both the set of assumptions and an extended version of the algorithm described in [10] for our problem, so let us first recap those here.

There are three main simplifying assumptions that we make in our algorithm. The first is that the cost for cutting out an area with a larger tool is always smaller than the cost to cut out the same area with a smaller tool. This virtually always holds in practice, since larger tools generally have higher material removal rates (unless the tool lengths are drastically different, making the smaller tool so much stiffer that it can be run with a feedrate sufficiently higher to make up for its smaller size, thus resulting in an ultimately higher material removal rate). The second two assumptions relate to ignoring 3D accessibility constraints. We assume that the area machinable by a larger tool is a subset of the area machinable by a smaller tool. This will be true if both tools have the same length or the pocket we are cutting out is not deeper than the shorter of the tools. Our final assumption is that there is nothing to gain from using the same tool twice in a sequence – we would only add to the cost by having to swap out a tool and then swap it back in (this is equivalent to assuming no accessibility problems because we could only gain from re-using the same tool if its accessible area had grown).

From the last assumption it follows that we can just order and use all our tools once (at most) in a sequence from, say, largest to smallest. Now since a larger tool is more efficient, we cut out as much area as possible with any current tool before proceeding with the next one. Then the area left over after using a tool is independent of any of the tools we used before the last one.

Based on the above assumptions, D'Souza et al. model the problem as a weighted directed graph: the nodes are the available tool radii and the directed edges go from a tool to every tool of a smaller radius. The weight on an edge is just the cost to machine the left-over area from the larger tool (the start of the edge) with the smaller tool (the end of the edge). Then the problem reduces to finding the shortest path from the first node (one associated with an 'infinite' radius – no area has been cut out) to the last node (the one associated with the smallest tool in the sequence – all the area has been cut out). This can be done in $O(n^2)$ time using Dijkstra, for example [12].

ALGORITHM

We reduce our problem to the problem above by generating beforehand a set of radii that we can build the graph from and run Dijsktra on. This can also be split into two subproblems: one is finding the range that the usable tool radius can be in and the other is discretizing this range into a finite set of radii to create the graph nodes from. After running Dijkstra on the generated graph, the produced set of cutters should be within a certain approximation ratio of the real optimum.

Finding the Range of Radii to Search

The cost for cutting out an area with a larger tool is smaller than the cost to cut out the same area with a smaller tool. From this follows that the range of radii of cutting tools we want to consider is bounded below by the radius r_{min} of the largest tool that can cut out the entire given pocket. It is bounded above by the radius r_{max} of the largest circle that fits in the pocket without intersecting its boundaries.

To find these radii (and, later on, discretize the search region) we construct the Voronoi diagram of the given boundary B. This can be done in $O(n \log n)$ time, where n is the complexity of B. We'll consider the Voronoi diagram just within the boundary contour.

Definitions Before we explore the properties of the Voronoi diagram, we need to define a few entities, to be used in the proofs to follow.

(1) Let V be the set of the Voronoi vertices and E the set of Voronoi edges *not touching* the boundary B. (In other words, we are considering just the edges from the medial axis of the given region.)

(2) Let E_P be the set of Voronoi edges separating the cells of two points and let E_E be all other Voronoi edges, $E_E = E \setminus E_P$. (A point of the boundary is a point between two adjacent boundary edges.)

(3) Let $d(p)$ be the radius of the maximum empty circle centered at p, for any point p of the interior of the boundary B. In other words, $d(p)$ is the distance from p to the boundary.

(4) Let $d_e = \min\{d(p) | p \in e\}$ and $D_e = \max\{d(p) | p \in e\}$, for each Voronoi edge e. In other words, d_e is the smallest distance from a point on the Voronoi edge e to the boundary and D_e is the largest. For each edge, those distances can be found in $O(1)$ time as follows:

   (a) If the edge is a straight line, it separates the Voronoi cells of two lines, two points, or two circular arcs of equal radius. If it separates two lines (Fig. 1a), the extremums are just at the endpoints of the edge. If it separates two points or arcs facing away from each other (Fig. 1b), the minimum is half the distance between the points (arcs) or at one of the endpoints; the maximum is at one of the endpoints. If it separates two arcs facing towards each other (Fig. 1c), the minimum is at one of the endpoints, the maximum – either at an endpoint, or along the line through the centers of the arcs.

   (b) If the edge is a hyperbola or an ellipse, it separates two arcs or an arc and a point (Fig. 1d-f). The extremums will be either at a point on the edge where the line passing through the two foci intersects the hyperbola/ellipse or at one of the endpoints of the edge, if the intersection point is outside the edge.

   (c) If the edge is a parabola, it separates a line and a point or a line and an arc (Fig. 1g-h). The extremums will be at half the distance from the point (or arc) to the line.

   This covers all the cases for a Voronoi edge. For a complete description, see [13].

(5) For each Voronoi edge $e \in E_P$, let p be the endpoint that is closer to the boundary compared to the other endpoint and define $d'_e = d(p)$.

(6) Let $S = \{d(v) \mid \text{where } v \text{ is a Voronoi vertex }\}$. S can be found in linear time by traversing all the Voronoi vertices.

(7) Let $T = \{d_e,\ D_e \mid e \in \text{Voronoi edges}, e \text{ doesn't touch } B\}$. From (4), we can find T in linear time as well.

(8) Let $R(r)$ be the region that can be covered by a tool of radius r within the given pocket. We take $R(r)$ to be closed.

(9) Let $B(r)$ be the boundary of the region that can be covered by a tool of radius r. ($B(r) \subset R(r)$.)

(10) Let $R = R(r_{min})$, in other words, R is the entire given pocket.

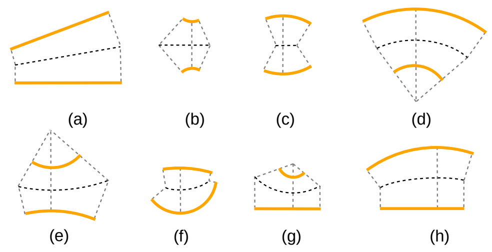

Figure 1. Voronoi edge types

Area vs. Radius We are going to use the area a tool can cut out as an initial approximation for the cost of using that tool (see next subsection for details). We therefore want to explore the relationship between the area and the tool radius. It turns out that we can use the radii in the finite sets S and T to simplify the description.

Claim 1: Let $V = \{r_1, ..., r_m\}$ be the elements of $S \cup T$ in increasing order. Recall that $R(r)$ is the region that can be covered by a tool of radius r. $R(r)$ is monotonically decreasing (not necessarily strictly) on the interval $[r_1, r_m]$, where the comparison is containment.

Proof: Take some r and r' with $r < r'$. Let p be a given point in $R(r')$. Then there is an empty disk d' of radius r' that contains p. Just take a disk d of radius r contained in d' that also contains p – since it is in d' it will also be empty, so $p \in d \subset R(r)$. Hence $R(r') \subset R(r)$. □

The above claim describes the covered region for the entire interval of radii in V. We'd like to see what happens on each of the intervals $[r_i, r_{i+1}]$:

Claim 2: $A(r) = Area(R(r))$ is continuous on every open interval (r_i, r_{i+1}).

Proof: Let $r \in (r_i, r_{i+1})$. Consider the boundary $B(r)$ of $R(r)$. More specifically, consider the parts of that boundary $B'(r)$ where $B(r)$ isn't on top of B: $B'(r) = B(r) \setminus B$. Any continuous one of those parts is an arc of a circle with radius r and was induced by getting our disk with radius r 'stuck', i.e. touching B at some two or more points, while trying to sweep the area within B. Then the center c of the disk that got stuck lies on a Voronoi edge, by definition of the Voronoi diagram. Moreover, c is not a Voronoi vertex – since otherwise r would be in S, but no $r \in S$ can be in the open interval (r_i, r_{i+1}) since these are adjacent radii in the ordered set $\{r_1, ..., r_m\} \in S \cup T$. Then, since the distance to the nearest site varies continuously over a Voronoi edge and r is not an extremum (r is not in T), there is some small δ' s.t. when we use circles of radii $r' = r - \delta'$ or $r'' = r + \delta'$ instead of r, their "getting stuck" centers will be "close" to c on c's Voronoi edge and so the boundary $B(r)$ will locally change only within some small given ε' (Figure 2). Then we have $|A(r) - A(r \pm \delta)| < \varepsilon$ (taking δ be the minimum over δ'-s and ε' to be sufficiently small to sum up to ε over all the continuous regions of $B'(r)$). So $A(r)$ is continuous on (r_i, r_{i+1}). □

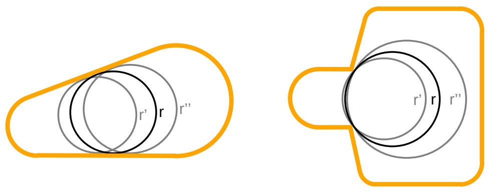

Figure 2. A couple of examples for changes in the boundary (hence in the area) when we use tools of radii $r' = r - \delta'$ and $r'' = r + delta'$ instead of r.

Claim 3: To refine the previous claim, on any given interval (r_i, r_{i+1}), $A(r)$ is either strictly decreasing on the entire interval or constant on the entire interval.

Proof: We'll look at three cases for a given interval (r_i, r_{i+1}). Either (1) this interval is contained in an interval defined by some Voronoi edge $e \in E_E$, (2) it is contained in an interval defined by some Voronoi edge $e \in E_P$, or (3) no Voronoi edge exists for which either of the first two cases hold. We consider each case in turn:

Case 1: There exists a Voronoi edge $e \in E_E$ such that interval $(r_i, r_{i+1}) \subset (d_e, D_e)$. We want to show that $A(r)$ is strictly decreasing on (r_i, r_{i+1}).

Take any $r, r' \in (r_i, r_{i+1})$ with $r < r'$. Let c' be a point on the interior of e with $d(c') = r'$. Since by the definition of E_E, at least one of the two Voronoi cells e separates has a straight line or circular arc for a boundary edge, take p' to be the point on that boundary edge a distance r' from c' (Figure 3). (In other words, cutting the pocket with a tool of radius r' would get the tool 'stuck' with its center at c' and it touching the boundary of the pocket at p'.) Now p' is a point on the boundary B separating the region of the boundary covered by the tool ($B \cap B(r')$) and the region not covered by the tool ($B \setminus B(r')$): looking at the figure, the points in the vicinity of p' and to the 'right' of p' are all a distance larger than r' from e along the perpendicular to the boundary edge and so can be covered by an even larger tool; the points in the vicinity of p' and to the 'left' of p' are all a distance smaller than r' from e along the perpendicular to the boundary edge and so cannot be covered by a tool of radius r'.

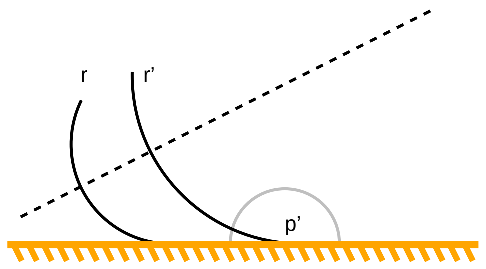

Figure 3. Case 1 (hatched areas are exterior to pocket)

There is a small vicinity of p' on the boundary, all points in which are a distance 'close' to r' from e along the perpendicular to the boundary edge. Thus, the vicinity can be picked so that all those distances are greater than r (since $r < r'$), i.e. the vicinity will be covered by a tool of radius r. Then there exists a disk C with center at p' with sufficiently small radius such that $C \cap R$ is entirely contained in $R(r)$. $C \cap R$ is only partially contained in $R(r')$: take a point p'' on the boundary to the 'left' of p' within C (i.e. $p'' \in C \cap B \setminus B(r')$). Since p'' is not in $R(r')$ and the complement of $R(r')$ is an open set, we can take a small disk C'' centered at p'' that is disjoint from $R(r')$. Then $C \cap C' \cap R$ is a region with non-empty area not contained in $R(r')$ but contained in $R(r)$. So $A(r) > A(r')$ and therefore $A(r)$ is strictly decreasing on the interval.

Case 2: There exists a Voronoi edge $e \in E_P$ such that $(r_i, r_{i+1}) \subset (d'_e, D_e)$. (Recall that d'_e is the distance from the boundary to the closer endpoint of the edge e.) Again, we want to show that $A(r)$ is strictly decreasing on (r_i, r_{i+1}).

Let the endpoints of the Voronoi edge e be v_1 and v_2, with v_1 being the vertex for which $d(v_1) = d'_e$ and v_2 being the vertex for which $d(v_2) = D_e$ (Figure 4). Of the two points from the boundary between which e is a Voronoi edge, let p be the one which required v_1, i.e. the one for which $v_1 p$ separates a Voronoi cell of p and of the boundary edge l adjacent to p (Figure 4). Let the other boundary point be p'.

Take any $r, r' \in (r_i, r_{i+1})$ with $r < r'$. Since $d'_e \le r_i < r < r' \le D_e$, $r > d'_e = |v_1 p|$ and $r < D_e = |v_2 p|$. Thus the points in the vicinity of p on l are not contained in $R(r)$ and the points in the vicinity of p on l' are contained in $R(r)$ – both by the similar reasoning as in Case 1. Take a small circle C with center at p (such that $C \cap l$ is in the above vicinity, i.e. $C \cap l$ are not in $R(r)$ while $C \cap l'$ are in $R(r)$). $R(r)$ contains p, which is inside C, and p', which is outside C. Since there is a path between p and p' within $R(r)$ (for example the arc of the disk that touches both p and p' on the 'right' side of pp'), $C \cap R(r)$ is non-empty. Now C contains a point from $B(r)$, $C \cap l'$, so it has to contain at

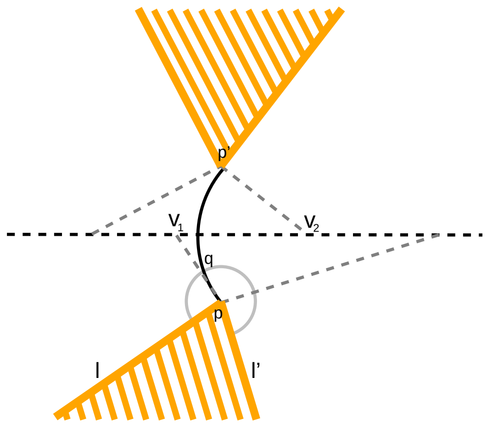

Figure 4. Case 2 (hatched areas are exterior to pocket)

least one another point of $B(r)$ – call it q. q can't be on l since we've established that the points on l in the vicinity of p are not contained in $R(r)$. Therefore q can only be in the interior of the pocket.

Suppose q is contained in $R(r')$. If q were in the interior of $R(r')$, then we could inscribe a circle of radius $r < r'$, containing q in its interior, within the circle of radius r' that contains it. This is a contradiction since $q \in B(r)$, not in $R(r) \setminus B(r)$. So q would have to be on $B(r')$. Since q is also in the interior of the pocket, there is a small disk C, centered at q, such that C is entirely contained in the pocket. But then we can build a disk of radius r, that is entirely contained in the pocket, and also contains q in its interior – see Figure 5. Again, contradiction with $q \notin R(r) \setminus B(r)$.

So q is outside of $R(r')$ and therefore there is some small disk C, centered at q and entirely outside of $R(r')$. Since $q \in B(r)$, the intersection of C and $R(r)$ has a non-empty area, not contained in $R(r')$, and thus $A(r) > A(r')$.

Case 3: Edges from the above two cases don't exist. In other words, if $(r_i, r_{i+1}) \subset (d_e, D_e)$ for some $e \in E$, then $e \in E_P$ and moreover $(r_i, r_{i+1}) \subset (d_e, d'_e)$. We claim that in this interval the area is constant.

Take any $r \in (r_i, r_{i+1})$. Consider a point p on B, separating the region covered by the tool, $B \cap B(r)$ and the region not covered by the tool $B \setminus B(r)$. It has to be a distance r from some Voronoi edge ('stuck' point), so the edge can be only in E_P and the point itself has to be a point between two boundary edges. But the vicinity of p on the boundary is contained in $B(r)$ – same reasoning as in Case 1 – so p can't be a separating point. Therefore no separating points exist, i.e. the entire boundary B is contained in $R(r)$. From that follows that the entire pocket is also contained in $R(r)$ so the area on the interval (r_i, r_{i+1}) is constant. □

Now, using Claim 3, we can show that $r_{min}, r_{max} \in S \cup T$.

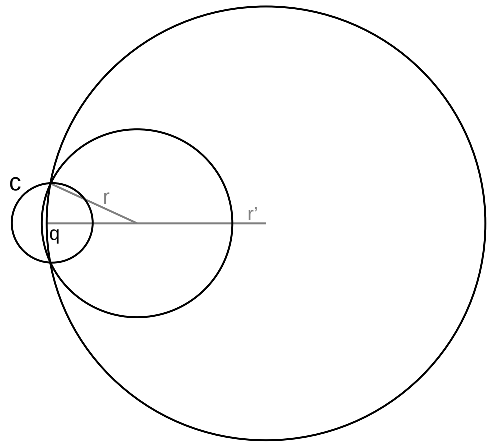

Figure 5. Finding a contained disk of radius r

First we need $r_{min}, r_{max} \in [r_1, r_m]$. For this we just have to show that a tool with radius r_1 can cover the entire region within the boundary B and that r_m is the largest tool that fits within B.

For the former, given a point p in the interior of B, we can find a disk d with center lying on a Voronoi edge not touching B (we defined all the angles of the pocket to be reflex) that contains p and lies in the interior of B. Since r_1 is the minimum distance between B and the Voronoi edges not touching B, r_1 has to be less than or equal to the radius of the chosen disk. Then we can just take a disk of radius r_1 that contains p and lies in d and therefore the interior of B.

For the latter, suppose there is a disk of radius r larger than r_m contained in the interior of B. Then for the Voronoi cell that it intersects, one of the Voronoi edges has to have a point that is distance r or greater from the boundary. This means the extremum of that edge (contained in $T \subset T \cup S$) is greater than or equal to r which is greater than r_m – the maximum of the set. This is a contradiction. Since r_m corresponds to a circle that fits within the boundary, and $r_{max} \leq r_m$, we get that $r_m = r_{max}$.

Finally, to show that r_{min} is also in $S \cup T$, suppose $r_{min} \neq r_i$ for all i. Then for some i, $r_{min} \in (r_i, r_{i+1})$ with $r_i < r_{i+1}$. By Claim 3, $A(r)$ is strictly decreasing on (r_i, r_{i+1}), so we have that for some $r \in (r_i, r_{min})$, $A(r) > A(r_{min}) = A$, where A is the total area. Contradiction.

Finding a Set on Which to Run Dijkstra

We assume that for a given tool, the cost to machine a (portion of a) pocket is proportional to the area. This assumption may seem simplistic, given that today typical tool paths will often vary in the extent to which they overlap themselves, depending on the pocket geometry. But optimal G & M code for cutting a pocket would keep cutting forces constant for maximum efficiency with minimum tool wear; the two typical approaches towards achieving this goal are to modify the tool path to achieve constant cutter engagement [14] or modify the feed rates, increasing feed rates when cutter engagement is reduced, to smooth cutting forces [15]. For some of the clean-up cutting operations performed with smaller tools, however, there may not be enough material left to be removed by that tool in some areas to achieve constant cutter engagement; thus it seems that feed rate adjustment might be a better solution to keeping material removal rate (MRR) constant. As constant MRR concepts become incorporated in commercial tool path planners, our assumption of cost (cutting time) proportional to area will become very close to actual practice.

Before we try finding a set of radii, let's estimate what the efficiency of the shortest path among a *given* set of radii is compared to the ideal sequence. Let's say that the optimal sequence of tool radii is $t_1 > t_2 > \ldots > t_l$ (even if we never manage to find it, we know that this sequence exists for a given pocket). Suppose we also have another sequence of radii, $r_1 > r_2 > \ldots. > r_k$ – an arbitrary set of tools (not to be confused with $r_1, \ldots, r_m \in S \cup T$) from which we can pick the optimal tool sequence for this particular set. We can assume that $t_1 \leq r_1$ and $t_l = r_k$.

Denote with $up(t_i)$ the smallest $r \in \{r_1, \ldots, r_k\}$ s.t. $t_i < r$ and $down(t_i)$ the largest $r \in \{r_1, \ldots, r_k\}$ s.t. $t_i \geq r$. Now, assuming we managed to pick the r_is so that the t_is are in distinct intervals separated by at least two r_is,

the cost incurred by the i^{th} tool in the ideal sequence is the cost to machine the area left after the use of the $(i-1)^{th}$ tool in the ideal sequence with a tool of radius t_i

$\geq$

the cost to machine the area left after the use of tool with radius $down(t_{i-1})$ with a tool of radius t_i (since $down(t_{i-1}) \leq t_{i-1}$ the left-over area from it is smaller)

$\geq$

the cost to machine the area left after the use of tool with radius $down(t_{i-1})$ with a tool of radius $up(t_i)$ (since $up(t_i) \geq t_i$ and tools with larger radius are more efficient)

$=$

$\frac{A_L(down(t_{i-1}))-A_L(up(t_i))}{A_L(up(t_{i-1}))-A_L(up(t_i))}$ times the cost to machine the area $A_L(up(t_{i-1}))$ with a tool of radius $up(t_i)$, where $A_L(r)$ denotes the area left over after the use of a tool of radius r. (This uses the assumption that the cost to machine an area with a given tool is approximately proportional to its area.)

Now machining the entire pocket with the sequence of tools $up(t_1), \ldots, up(t_l)$ in the given radii is the sum of the costs to machine areas $A_L(up(t_{i-1}))$ with respectively tools of radius $up(t_i)$ and this is, by above, $\leq \max_{i \in \{1,\ldots,l\}} \frac{A_L(up(t_{i-1}))-A_L(up(t_i))}{A_L(down(t_{i-1}))-A_L(up(t_i))}$ times the optimal cost $\leq \max_{i \in \{1,\ldots,m-1\}} \frac{A_L(r_i)-A_L(r_{i+2})}{A_L(r_{i+1})-A_L(r_{i+2})}$ times the optimal cost. The maximum of those fractions is then an upper bound on our current approximation ratio, where we define approximation ratio to be the ratio between the actual and optimal costs.

Thus, to achieve a goal approximation ratio, we just have to recursively look at all pairs of consecutive tools (of some given initial sequence) and insert a radius midway if current approximation ratio is larger than the desired approximation ratio.

Not all initial sequences work equally well if we apply this procedure, however. Recalling the lemma, the function of the area (and therefore the left-over area, as well) is not continuous at the points $r_2, r_3, ..., r_{m-1} \in S \cup T$. This means that we could keep subdividing for example the interval $(r_5 - 0.0001, r_5 + 0.0001)$ indefinitely without improving the approximation ratio at all. Thus, a good initial sequence to try would be $r_1, r_2 - \varepsilon, r_2 + \varepsilon, r_3 - \varepsilon, ..., r_{m-1} + \varepsilon, r_m$. Then, to achieve the goal ratio, we only look at the continuous intervals – we don't subdivide the intervals $(r_i - \varepsilon, r_i + \varepsilon)$.

CURRENT IMPLEMENTATION AND RESULTS

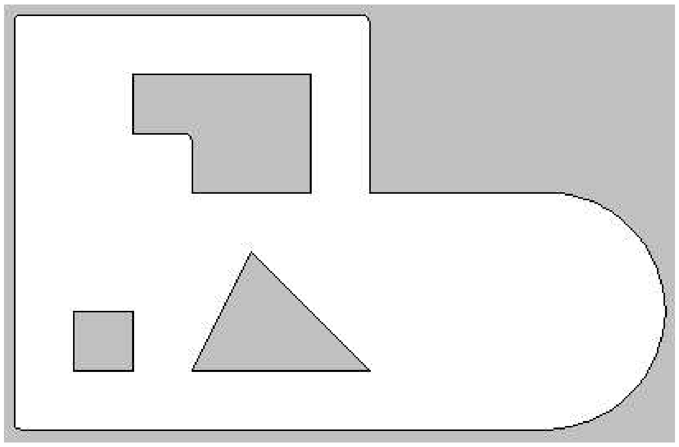

Figure 6. The pocket we ran our test on.

The current implementation doesn't check where the discontinuities are yet, it just subdivides the initial interval $[r_{min}, r_{max}]$ until either the maximum approximation ratio is smaller than the requested ratio or the difference in areas between subdivisions is a small percentage of the total area. It is using ACIS to compute the offset curves and boolean operations needed for finding the regions machinable by a certain tool, as well as the region areas.

We ran the algorithm on a simple example consisting of 23 edges (Figure 6). The radius generation phase came up with 67 radii to check. As expected, at least 60% of these are clustered in the areas where the discontinuities would be. Running the same example with a looser approximation ratio comes up with 19 radii to check and with similar proportions of clustering. In both cases, we can reduce at least the output graph size by comparing the output ratios and trimming the radii that don't improve the ratios by a given percent. In the former case this reduced the number of radii by 30%, in the latter by 10%. Dijkstra afterwards comes up with shortest paths of respectively 5 and 4 tool sizes (the result of the latter run is shown on Figure 7).

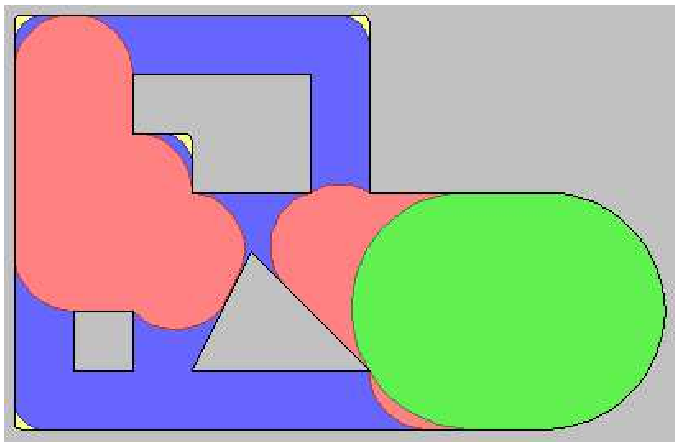

Figure 7. Tool sizes produced by the algorithm: biggest tool can cut out the green area, followed by red, blue, and yellow.

SUMMARY AND FUTURE WORK

We have presented an algorithm that, with no preset restrictions on cutter size, generates a sequence of cutter radii for machining a 2D pocket with a cutting time within a user specified percentage of that for the optimal cutter radii sequence. In future work, we will look at relaxing some of our simplifying assumptions, particularly in regards to 3D accessibility, incorporating tool length, tool holder, and spindle geometry, as well as tool change costs, into the analysis.

ACKNOWLEDGMENTS

We gratefully acknowledge the contributions of the anonymous reviewers who provided counter-examples that revealed flaws in some of our original definitions, and for questions and suggestions that improved the exposition of our algorithm. The researchers were supported in part by UC MICRO grants 02-068 and 03-077 and the Ford Motor Company.

REFERENCES

[1] Arya, S., and Cheng, S., 2001. "Approximation algorithm for multiple-tool milling". International Journal of Computational Geometry and Applications, **11** (3) , pp. 339–372.

[2] Lee, Y.-S., and Chang, T.-C., 1994. “Using virtual boundaries for the planning and machining of protrusion free-form features”. Computers in Industry, **25** (2) December , pp. 173–187.

[3] Lee, Y.-S., and Chang, T.-C., 1995. “Application of computational geometry in optimizing 2.5D and 3D NC surface machining”. Computers in Industry, **26** (1) , pp. 41–59.

[4] Bala, M., and Chang, T., 1991. “Automatic cutter selection and optimal cutter path generation for prismatic parts”. International Journal of Production Research, **29** (11) November , pp. 2163–2176.

[5] Veeramani, D., and Gau, Y.-S., 1997. “Selection of an optimal set of cutting-tool sizes for 2.5 D pocket machining”. Computer-Aided Design, **29** (12) December , pp. 869–877.

[6] Balasubramaniam, M., Joshi, Y., Engels, E., Sarma, S., and Shaikh, Z., 2001. “Tool selection in three-axis rough machining”. International Journal of Production Research, **39** (18) , pp. 4215–4238.

[7] Yao, Z., Gupta, S., and Nau, D., 2001. “A geometric algorithm for selecting optimal set of cutters for multi-part milling”. In Proceedings of the sixth ACM symposium on solid modeling and applications, pp. 130–139.

[8] Yao, Z., Gupta, S., and Nau, D., 2003. “Algorithms for selecting cutters in multi-part milling problems”. Computer-Aided Design, **35** (9) August , pp. 825–839.

[9] Ahn, S., Sundararajan, V., Smith, C., Kannan, B., D’Souza, R., Sun, G., Mohole, A., Wright, P., Kim, J., McMains, S., Smith, J., and Séquin, C., 2001. “CyberCut: An Internet Based CAD/CAM System”. Journal of Computing and Information Science in Engineering, **1** (1) , pp. 52–59.

[10] D’Souza, R., Wright, P., and Séquin, C., 2001. “Automated microplanning for 2.5-D pocket machining”. Journal of Manufacturing Systems, **20** (4) , pp. 288–296.

[11] D’Souza, R., Wright, P., and Séquin, C., 2002. “Handling tool holder collision in optimal tool sequence selection for 2.5-D pocket machining”. Journal of Computing and Information Science in Engineering, **2** (4) , pp. 345–349.

[12] Sedgewick, R., 2002. *Algorithms in C++, Part 5: Graph Algorithms,*. Addison-Wesley.

[13] Kim, D., Hwang, I., and Park, B., 1995. “Representing the Voronoi diagram of a simple polygon using rational quadratic Bezier curves”. Computer-Aided Design, **27** (8) August , pp. 605–614.

[14] Stori, J., 1998. *Simulation based parameter optimization and constant engagement tool path planning*. PhD thesis, University of California, Berkeley, Mechanical Engineering Department.

[15] Bae, S.-H., Ko, K., Kim, B. H., and Choi, B. K., 2003. “Automatic feedrate adjustment for pocket machining”. Computer-Aided Design, **35** (5) April , pp. 495–500.

[16] Veeramani, D., and Gau, Y., 2000. “Cutter-path generation using multiple cutting-tool sizes for 2-1/2D pocket machining”. *IIE Transactions*, **32** , pp. 661–675.