Approximation Algorithms for Lawn Mowing and Milling *

Esther M. Arkin† Sándor P. Fekete‡ Joseph S. B. Mitchell§

Abstract

We study the problem of finding shortest tours/paths for "lawn mowing" and "milling" problems: Given a region in the plane, and given the shape of a "cutter" (typically, a circle or a square), find a shortest tour/path for the cutter such that every point within the region is covered by the cutter at some position along the tour/path. In the milling version of the problem, the cutter is constrained to stay within the region. The milling problem arises naturally in the area of automatic tool path generation for NC pocket machining. The lawn mowing problem arises in optical inspection, spray painting, and optimal search planning.

Both problems are NP-hard in general. We give efficient constant-factor approximation algorithms for both problems. In particular, we give a $(3+\epsilon)$-approximation algorithm for the lawn mowing problem and a 2.5-approximation algorithm for the milling problem. Furthermore, we give a simple $\frac{6}{5}$-approximation algorithm for the TSP problem in simple grid graphs, which leads to an $\frac{11}{5}$-approximation algorithm for milling simple rectilinear polygons.

Key Words: lawn mowing, milling, NC machining, watchman routes, shortest paths, traveling salesman (TSP), approximation algorithms, computational geometry

*A preliminary version of this paper was entitled "The Lawnmower Problem" and appears in the *Proc. 5th Canad. Conf. Comput. Geom.*, pages 461–466, Waterloo, Canada, 1993.

†estie@ams.sunysb.edu; Department of Applied Mathematics and Statistics, SUNY Stony Brook, NY 11794–3600, USA. Partially supported by NSF Grants CCR-9204585 and CCR-9504192. Parts of this research were conducted while the author was a participant in the Special Semester in Computational Geometry at Tel Aviv University, 1995.

‡sandor@zpr.mi.uni-koeln.de. Center for Parallel Computing, Universität zu Köln, 50923 Köln, Germany. Parts of this work were done during a stay at SUNY Stony Brook, supported by NSF Grants ECSE-8857642 and CCR-9204585.

§jsbm@ams.sunysb.edu; Department of Applied Mathematics and Statistics, SUNY Stony Brook, NY 11794–3600, USA. Partially supported by Hughes Research Laboratories, Boeing Computer Services, and NSF Grants CCR-9204585 and CCR-9504192. Parts of this research were conducted while the author was a participant in the Special Semester in Computational Geometry at Tel Aviv University, 1995.

1 Introduction

Consider the following problem: For a given region covered by grass, find a short path along which to move a lawn mower, such that all the grass is cut. This *lawn mowing problem* arises in several practical applications. Motivations from manufacturing include:

- (Process Planning) Plan the motion of the nozzle of a spray painting device in order to coat the entire surface of an object.
- (Quality Control) Plan the movement of a sensor (camera, detector) in order to check an entire part for imperfections.

Motivations also arise in the planning of geographic surveys, search-and-rescue operations, etc.

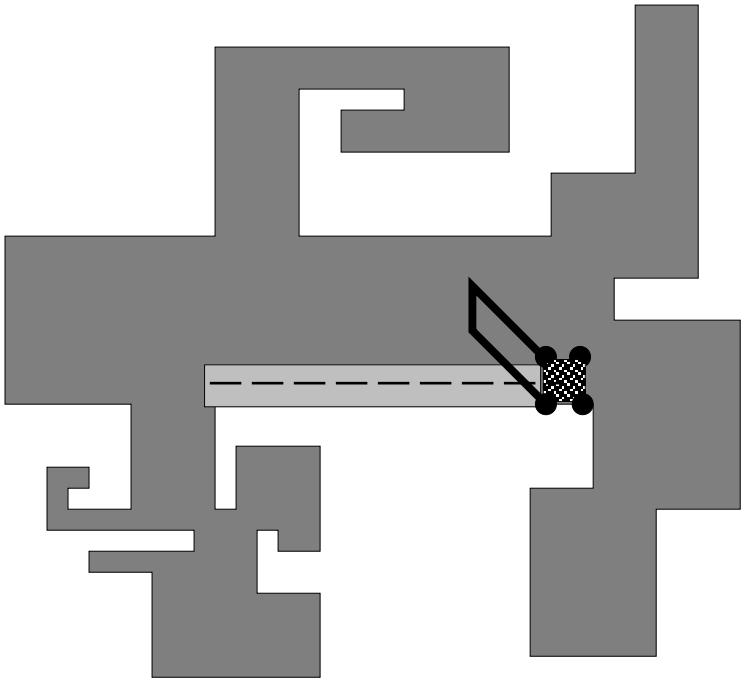

Figure 1: The lawn mowing problem

A closely related problem is that of automatically generating tool paths for NC (Numerically Controlled) pocket machining: Given a workpiece, a cutter head, and the shape of a "pocket" to be milled in the workpiece, determine a route for the cutter such that the cutter removes exactly the material that lies within the pocket. The difference between this *milling problem* and the lawn mowing problem is that in the milling problem we do not allow the cutter to exit the region (pocket) that it must cover, while in the lawn mowing problem it is permitted for the cutter to "mow" over non-grass regions (e.g., one may push the lawn mower over the sidewalk while cutting the grass).

Related Work. The lawn mowing problem is closely related to the geometric Traveling Salesman Problem (TSP) with "mobile clients": Find a shortest tour for a salesman who must visit a given set of clients, each of which is willing to travel up to distance d in order to meet the salesman. It is easy to see that this problem can be modeled as that of "mowing" a given (discrete) set of points (at the locations of the clients), using a "mower" of radius d. This special case of the lawn mowing problem, with a finite (discrete) set of points to be mowed, has been studied by Arkin and Hassin [2], who obtained constant-factor approximation methods

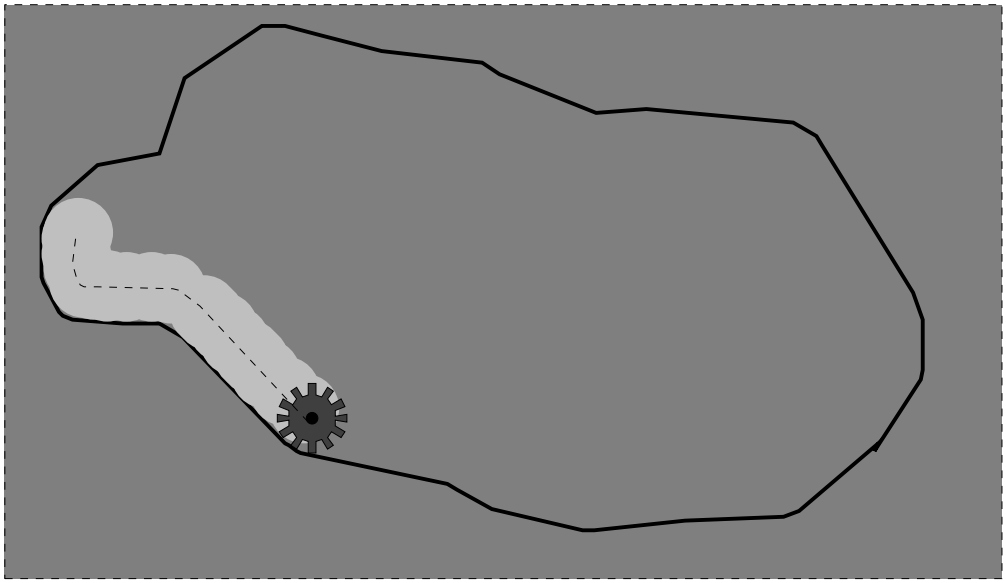

Figure 2: The milling problem

for this and other variants of the "TSP with Neighborhoods" problem (see also [15]). (These problems are clearly NP-hard, from the fact that the Euclidean TSP is NP-hard.)

The lawn mowing problem is also closely related to the "watchman route problem" with limited visibility (or "d-sweeper problem"), which has been studied by Ntafos [19]: How does one "sweep" the floor of a given (polygonal) room, using a circular broom of radius d, so that the total travel of the broom is minimized? By studying the problem of approximating TSP tours on *simple* grid graphs, Ntafos gives an approximation algorithm (with approximation factor $\frac{4}{3}$) for the d-sweeper problem in a simple polygon, *provided* that d is sufficiently "small" in comparison with the dimensions of the polygon.

In the CAD community, there is a tremendous amount of literature on the subject of automatic tool path generation. We refer the reader to the book of Held [11] for a survey. Held was the first author to start a mathematical examination of the algorithmic questions that arise in pocket machining. He was primarily concerned with generating feasible tool paths for milling, and gave efficient algorithms, based on the methods of computational geometry, for implementing some standard heuristics that are used in practice. Aspects of complexity in the optimization problem are only mentioned very briefly in the book; the question of polynomiality or NP-hardness of the milling problem is stated as an open problem, and the question of obtaining approximation algorithms is not discussed.

Recently, Arkin et al. [3] have examined the problem of minimizing the number of retractions for the "zig-zag" pocket machining problem, subject to the constraint that one is *not* allowed to "re-mill"; they show the problem to be NP-complete and obtain constant factor approximation algorithms. (The milling problem addressed here can be thought of as a dual: minimize the amount of re-milling, subject to no retractions of the cutter.)

In [1], we showed that the milling and lawn mower problems are NP-hard, in general. We also provided existence proofs of constant-factor approximation algorithms for these problems. Iwano et al. [13] independently obtained an approximation algorithm for a version of the lawn mower problem; they provide, for any fixed $\epsilon > 0$, a $(9 + \epsilon)$-approximation for the case of rectilinear grass regions.

Main Results This paper represents a continuation and extension of our earlier work [1], with several substantial new results:

- For the lawnmowing problem, we give improved approximation algorithms that not only cut the previous best factor by up to a factor of 2, but also substantially improve the running time. Further, our

method greatly simplifies the proof of [13].

- For the milling problem, we give a 2.5-approximation algorithm, with worst-case time $O(n \log n)$, for a (possibly multiply-connected) pocket whose boundary complexity is n.
- We give new results for the TSP on grid graphs, and applications of these results to related instances of the milling problem:
  - For the case of *simple* grid graphs having no cut vertices (i.e., the removal of a node does not disconnect the graph), we show how to construct efficiently a tour whose length is at most $(6/5)N$, where N is the number of grid points (nodes). This improves upon the best previous bound of $(4/3)N$ [19].
  - We apply this result to obtain a 6/5-approximation algorithm for the milling problem in the case of a unit square cutter, with rectilinear motion, in an integer-coordinate rectilinear simple polygon. The result also extends to arbitrary rectilinear simple n-gons, yielding an 11/5-approximation that runs in linear ($O(n)$) time.
  - For the case of arbitrary connected grid graphs, having no local cut vertices (i.e., the removal of a node does not reduce the genus), we obtain the first nontrivial upper bounds, showing that the length of an optimal tour is at most $1.325N$. (The trivial bound is $2N - 2$, from doubling a spanning tree.) This result also applies to yield improved approximation factors for milling rectilinear polygons with holes.

  Note that recent results of Grigni et al. [10] provide a polynomial-time approximation scheme for TSP in grid graphs, allowing one to approximate the optimal tour within a factor $(1 + \epsilon)$ for any $\epsilon > 0$, in $N^{O(1/\epsilon)}$ time. For comparison, our results establish bounds not just on the ratio of the length A of the approximation to the length of optimal, L^*, but on the ratio of A to N. We are able to show that our bound of 6/5 is best possible for this ratio. Furthermore, our approximation algorithm is relatively simple and much more efficient ($O(N)$).

2 Preliminaries

We are given a planar region, R, that describes the grass to be mowed or the pocket to be machined. In general, R may consist of several connected components, each having "holes". We assume here that each component of R is a (multiply connected) polygon; our results extend to more general regions whose boundaries are described by a discrete set of simple curved arcs (straight segments, circular arcs, etc). We let n denote the total number of vertices of R, and δR denote the boundary of R.

We are also given a *cutter*, χ. Throughout this paper, we assume that χ is either a circle or an axis-aligned square. Without loss of generality, we scale our problem instance so that the χ is a unit circle (radius 1) or a unit square (side length 1). The *reference point* for the cutter χ is its centerpoint. We let $\chi(p)$ denote the *placement* of χ at the point $p \in \mathbb{R}^2$ (i.e., the unit circle/square with centerpoint at p).

A *lawn mower path/tour* π is a path/tour such that every point of the region R is covered by some placement of χ along π; i.e., $R \subseteq \cup_{p \in \pi} \chi(p)$. A *milling path/tour* π is a path/tour such that every point of R is covered by some placement of χ along π, *and* no placement of χ along π ever hits a point *outside* of R; i.e., $R = \cup_{p \in \pi} \chi(p)$.

We consider two cases of allowed motions (translations) of the cutter: *rectilinear* (axis-parallel) and *unrestricted* (arbitrary translation). We measure the length of a path/tour of the cutter as its Euclidean (L_2) length. In the case of rectilinear motion, measuring the Euclidean length amounts to the same thing as measuring the L_1 length of the path/tour. (By the isometry that exists between the L_1 and L_∞ metrics, we are able to handle the L_∞ case as well.)

The points $(x, y) \in \mathbb{N}^2$ having integer coordinates define the *grid points* in the plane. The *integer grid graph* refers to the (infinite) graph whose nodes are the grid points and whose edges join two points at distance 1; thus, in the integer grid graph, each grid point has degree exactly 4. A *pixel* is a unit square whose center is determined by integer coordinates.

It is easy to see that, for any region R, there always exists a lawn mower path/tour; however, it may be that there exists no milling path/tour for a (connected) region R, as the cutter may not be able to fit into the "corners" of R or the pass through the "bottlenecks" of R.

We note that even if the input size is combinatorially very small (e.g., a rectangular region, R, with $n = 4$), a lawn mower tour may require a combinatorially very large description if it is described as a polygonal walk. See Figure 3.

In fact, the number of bends in the output tour may be exponential as a function of the magnitudes of the input coordinates of R. We can address this issue in at least a couple different ways:

1. We can consider the input complexity to depend on the *magnitude* (e.g., bit complexity) of the numbers describing the coordinates of R, rather than on the *combinatorial* size, n, of the input. This leads to algorithms whose complexity is *pseudopolynomial* in the input size.

2. We can consider tours that consist of a polynomial number of pieces each having a regular structure, allowing a succinct representation even if the number of bends in the tour is quite high. For example, in his book, Held [11] concentrates on two natural strategies that are used for milling in practice – "contour-parallel" milling and "axis-parallel" ("zig-zag") milling. (See Figure 4.) It seems reasonable to assume that partial tours following one of these two strategies can be encoded efficiently. Of course this does not resolve the problem of getting closed tours of short length that cover the complete area to be milled. We will show in Section 5 how the above strategies can be used to find a closed tour of bounded length.

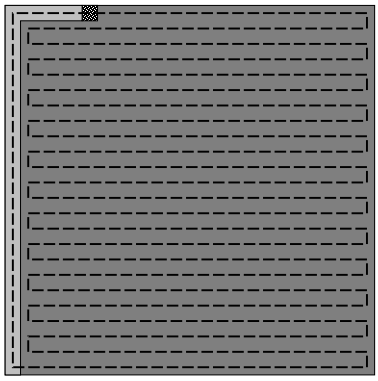

Figure 3: A lawn mower tour may require a large number of bends, even for a very simple input region shape, R.

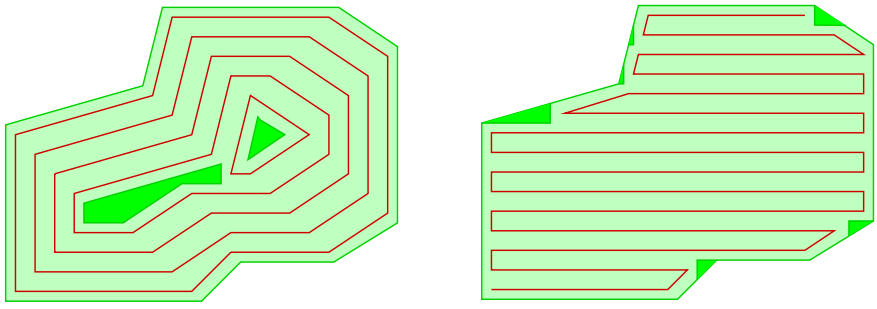

Figure 4: Left: Contour-parallel milling. Right: Axis-parallel milling.

3 NP-Hardness Proofs

Theorem 1 *The lawn mowing problem for a connected polygonal region is NP-hard for the case of an aligned unit square cutter χ.*

Proof. Our proof makes use of the reduction from the (NP-hard) problem Hamiltonian Circuit in Planar Bipartite Graphs with Maximum Degree 3 to the problem Hamiltonian Circuit in Grid Graphs, as used by Johnson and Papadimitriou [14] in their proof of hardness of Hamiltonian Circuit in Grid Graphs. (See also Itai, Papadimitriou and Swarcfiter [12].)

First, a planar bipartite graph G with n vertices (each of maximum degree 3) is represented by a grid graph $\overline{G}$ having $m = O(n)$ vertices, such that $\overline{G}$ has a Hamiltonian circuit if and only if G has a Hamiltonian circuit. Next, we define the (polygonal) region R to be the union of all placements of χ (a unit square) at the (grid) vertices of $\overline{G}$. Figure 6 shows an example of this construction that corresponds to the bipartite graph shown in Figure 5.

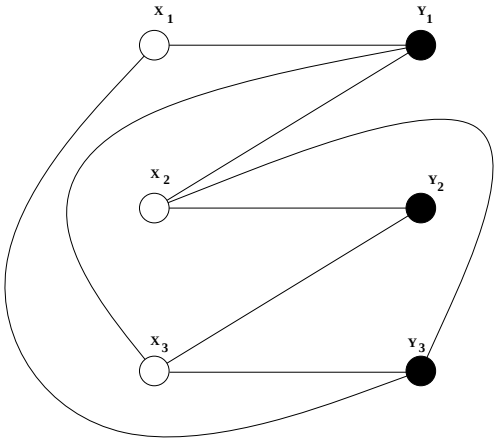

Figure 5: A planar bipartite graph G, with maximum vertex degree 3.

It is easy to see that the existence of a tour of length m on the grid vertices of $\overline{G}$ implies the existence of a lawn mower tour of length m. On the other hand, a lawn mower tour of length m can mow all of R (which has area m) only if no point in the region is mowed more than once. This means that the tour partitions the region R into nonoverlapping strips (rectangles) of width 1; clearly, these strips must have integer length. Since traveling a strip corresponds to traveling the associated grid vertices, this implies that a lawn mower tour of length m induces a tour of length at most m in the grid graph. (See Figure 6.) □

Corollary 1 *The lawn mowing problem is NP-hard even for simple polygonal regions R.*

Proof. We can modify the region R used in the proof of Theorem 1 to make it a *simple* polygonal region R', by making very narrow "slits" that interconnect the holes of R. See Figure 7.

As in the proof of Theorem 1, a Hamiltonian tour (of length m) on the grid graph $\overline{G}$ yields a lawn mower tour of length m for R, and hence for R'. Conversely, if the slits in R' have been made sufficiently narrow (e.g., less than width δ/n, for a small constant δ), then an optimal lawn mower tour, of length L^*, for R' can be slightly perturbed into a lawn mower tour of length $L^* + \epsilon$ for R, where $\epsilon < 1$. Then, if $L^* + \epsilon < m + 1$, we can conclude that $\overline{G}$ has a Hamiltonian cycle. □

Observing that the optimal lawn mower tours for the construction given in the proof of Theorem 1 are also feasible milling tours, we obtain

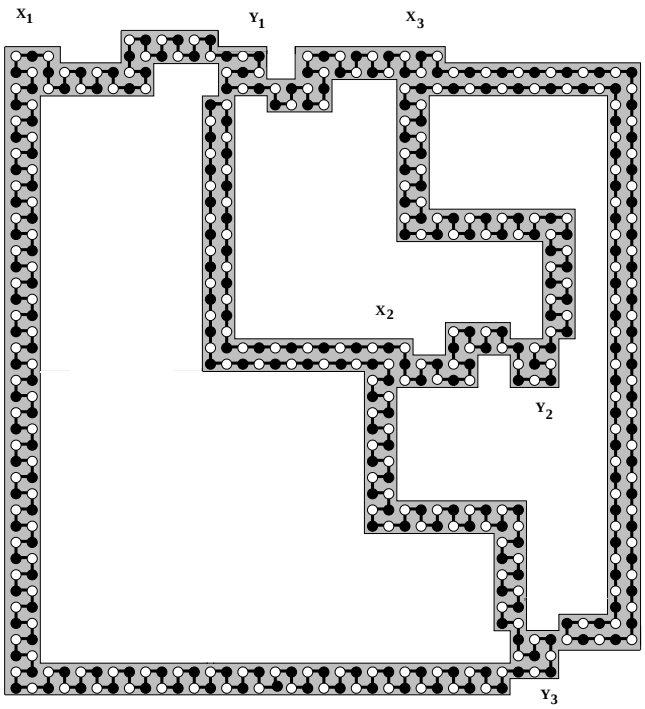

Figure 6: The construction used in proving the NP-hardness of the lawn mowing problem.

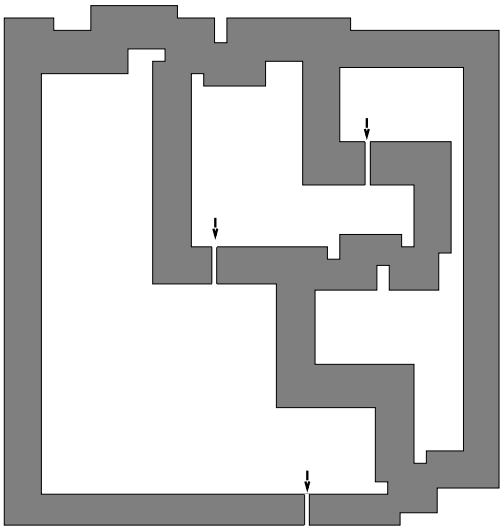

Figure 7: NP-hardness of the lawn mowing problem for simple polygonal regions.

Corollary 2 *The milling problem is NP-hard for the case of an aligned unit square cutter χ and a multiply-connected polygonal region R (with holes).*

We do not know how to extend the NP-hardness result for the milling problem to show hardness of the case with *simple* polygonal regions R (i.e., with no holes). In fact, it is an outstanding open question if the Hamiltonian circuit problem can be solved in polynomial time in a "simple grid graph" (without holes); a polynomial-time solution to this problem would imply a polynomial-time solution to the milling problem in integer rectilinear simple polygons (for the case of an aligned unit square cutter χ).

4 Approximation Methods for the Lawnmowing Problem

In [1], two different approximation methods for the lawn mowing problem were given, depending on the connectedness of the region of grass. In the following, we show that one method is sufficient to achieve a better approximation factor for both cases.

We begin by assuming that the cutter χ is a unit square, aligned with the axes; at the end of this section, we will consider the case of a circular cutter. We consider both the case of rectilinear motion and of unrestricted motion. Let T^* denote an optimal lawn mower tour, of length ℓ^*.

Let S be the set of (integer grid) centerpoints for the pixels, $\mathcal{P}$, that intersect the given polygonal region R. Let $N = |S|$. We can identify S in time $O(N + n \log n)$ from a given description of R as a polygonal region with n vertices.

Our approximation algorithm is remarkably simple: *Construct an approximate TSP tour on the set of points S.*

The length of a tour is measured in terms of L_1 length (for rectilinear motion case) or L_2 length (for unrestricted motion case). Let α_{TSP} denote the approximation factor for the TSP for points in the plane. One can achieve $\alpha_{TSP} = 2$ by simply doubling a minimum spanning tree, or one can achieve $\alpha_{TSP} = 1.5$ by applying the Christofides heuristic (or achieve $\alpha_{TSP} = 1.5 + \epsilon$, using an efficient approximation thereof [22]). Furthermore, we can use the approximation schemes by Arora [5] or Mitchell [16, 17] to get $\alpha_{TSP} = 1 + \epsilon$ in polynomial time (with an exponent of $O(\frac{1}{\epsilon})$).

The following lemma establishes the feasibility of our approximation tour; its proof is immediate.

Lemma 1 *Any tour of the point set S is a feasible lawn mower tour for a (unit square) cutter χ.*

The goal of the next lemma is to establish the sufficiency of searching for tours on the integer grid graph (whose vertices contain the centerpoints S). We refer to the unit square faces in the embedding of this (planar) grid graph as the "dual pixels" (with the word "dual" being suggestive of the plana dual of the integer grid graph). The corners of the dual pixels lie at half-integer coordinates.

Lemma 2 *For a unit square cutter, restricted to axis-parallel motions, there exists a tour, T_G, of length ℓ_G, that lies on the integer grid graph, such that T_G intersects the same set of (closed) dual pixels as does T^*, and such that $\ell_G \leq \ell^*$.*

Proof. If T^* lies fully within a block of four dual pixels that share a common corner, then we can simply define T_G to be the (degenerate) tour consisting of the single point at the shared corner.

Now assume that T^* does not lie fully within a block of four dual pixels sharing a common corner. We claim that, without loss of generality, we can assume that all bend points of the (rectilinear) tour T^* lie on the edges of the integer grid graph (i.e., on the boundaries of dual pixels). To see this, assume to the contrary that there is a bend point p in the tour T^*, with p interior to some dual pixel, P. Let $a \in \delta P$ (resp., $b \in \delta P$) be the last point of P encountered by traversing T^* forward (resp., backward). (Such points exist, since T^* does not lie fully within P.) Now, we simply replace the subpath of T^* that goes from a to b with an "L-shaped" path whose single bend point lies on the boundary of P. (In case a and b have a common x-coordinate or y-coordinate, the path linking a and b may in fact be a single vertical or horizontal segment, having *no* bend points.) The new subpath is no longer than the original subpath; further, the modified tour, T', intersects the same set of dual pixels as the original tour, T^*.

Now, all of the bend points of the modified tour T' lie on the edges of the integer grid graph. Our goal now is to transform T' into a tour, T_G, that lies entirely on the integer grid graph (i.e., that has all of its bend points at grid points), while T_G intersects the same set of dual pixels as does T'.

There are two possible kinds of bend points in T': 90-degree bends and 180-degree bends (*turn-abouts*). If all of the bend points of T' already lie at integer grid points, we are done ($T_G = T'$). Thus, consider an arbitrary bend point, b, on T', such that b does not lie at an integer grid point. Without loss of generality, assume that b lies on a vertical grid line. We now describe an adjustment procedure to convert T' into T_G. There are two cases:

1. If b is a 90-degree bend point, then b lies at the intersection of two segments of T' — a horizontal segment, call it ab, having the other endpoint a also lying on a vertical grid line different from that containing b, and a vertical segment, call it bc, having the other endpoint c lying on the same vertical grid line as b. There are two subcases, depending on whether the vertical segments incident at a and b are (a) on opposite sides of the horizontal line through ab, or (b) on the same side of ab; refer to Figure 8.

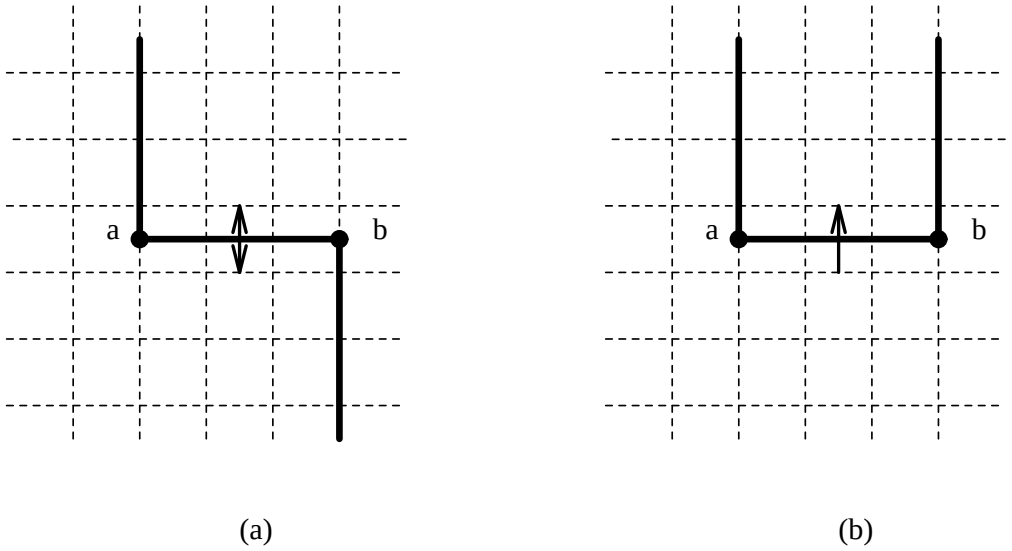

Figure 8: Two cases for adjusting a rectilinear tour to obtain a tour, T_G, on the integer grid.

(a) In case (a), we can translate ab vertically upwards or downwards, without changing the length of T', until one of the endpoints of ab hits (i) a grid point, or (ii) another bend point of T'. This translation does not change the set of dual pixels intersected by the tour. In subcase (i), both a and b hit grid points, so we have succeeded in moving b (as well as a) to a grid point. In subcase (ii), we have decreased the total number of bend points, and, in particular, we have decreased the number of bend points that do not lie at grid points.

(b) In case (b), we can translate ab vertically in the direction that *decreases* the length of T', again until one of the endpoints of ab hits (i) a grid point, or (ii) another bend point of T', exactly as in case (a).

2. If b is a turn-about, then there are two subcases:

(a) If the two edges of T' incident at b lie on the grid line containing b, then both edges can be shortened, either until b coincides with a grid point or until b coincides with another bend point.

(b) If the two edges of T' incident at b are orthogonal to the grid line containing b, then then these two edges (call them ab and cb) can be translated until a or c coincides with a grid point (so that b also coincides with a grid point), or until a or c coincide with another bend point.

When this adjustment procedure terminates, we have all bend points lying at grid points, and the resulting tour is the desired T_G. □

Consider a tour, T_G, of length $\ell_G \leq \ell^*$, obtained from T^*, as in the above lemma. Let S_G denote the set of integer grid points (pixel centerpoints) visited by T_G. While T_G intersects the same set of dual pixels as does T^*, it is not necessarily a lawn mowing tour, since the "rounding" onto the integer grid may have made it infeasible. (If, by chance, we have $S_G \supseteq S$, then T_G is in fact a lawn mowing tour.) The purpose of the next lemma is to show that we can convert T_G into a lawn mowing tour (in particular, a tour visiting all points of S), at a cost of increasing its length by at most a factor of 3.

First, let Π be the path obtained from T_G by cutting it open, by removing one grid edge, of length 1. (We assume that T_G has at least two grid edges; otherwise, the problem is trivial.) Let F denote the (open) region obtained by offsetting Π, convolving it with an (open) L_∞ disk of radius 1 (i.e., F is obtained by sweeping an axis-aligned square of side length 2, with its center following path Π).

Lemma 3 *Each centerpoint of S is either visited by the path Π or is within L_∞ distance 1 from some centerpoint that is; i.e., $S \subseteq \Pi \cup \delta F$.*

Proof. Let $p \in S$ be a centerpoint of a grass pixel, and let P be any one of the four dual pixels having corner p. Let π denote that portion of T^* that lies within dual pixel, P. (Note that π need not be connected.) The total region mowed by π must lie within the union of the four grass (non-dual) pixels whose centerpoints are the four corners of the dual pixel P. Since T_G (and, hence, Π) visits the same set of dual pixels as does T^*, we know that at least one corner of P lies in S_G. Thus, all four corners of P lie in the set $\Pi \cup \delta F$, which includes S_G, together with all centerpoints within L_∞ distance 1 from some point of S_G. □

Let $S'_G = S \cap (\Pi \cup \delta F)$ be the subset of S that lie on T_G, or within L_∞ distance 1 of some point of T_G. The following lemma shows that there exists a relatively short tour that covers S'_G, and, hence, by Lemma 3, that covers S, thereby making it a lawn mowing tour.

Lemma 4 *There exists a tour that covers S'_G (and hence S) with length at most $3\ell_G + 6$ (and hence at most $3\ell^* + 6$).*

Proof. For a path of length ℓ and a disk of circumference c, it is known (see [8, 13]) that there exists a tour of length at most $2\ell + c$ that covers the boundary of the Minkowski sum of the path and the disk. Thus, since the L_∞ disk of radius 1 has circumference 8, we know that there exists a tour, T', covering δF of length at most $2\ell_\Pi + 8 = 2\ell_G - 2 + 8 = 2\ell_G + 6$.; But the tour T' must cross T_G, at a point within distance 1 of the edge of T_G that was removed to obtain Π. Thus, we can concatenate the tour T' with the tour T_G to obtain a tour T'' of length at most $3\ell_G + 6$ that covers all of Π and δF, and hence all of S'_G and S. □

Thus, we can conclude that there exists a rectilinear tour of length at most $3\ell^* + 6$ that covers S, and hence is a lawn mowing tour, implying the following theorem:

Theorem 2 *Finding a rectilinear TSP approximation (with factor α_{TSP}) on the set of centerpoints S yields a lawn mowing tour of length at most $\alpha_{TSP}(3\ell^* + 6)$, where ℓ^* is the length of an optimal lawn mowing tour.*

Running time. The complexity of our approximation algorithm is simply the time, $f(N)$, required to approximate the TSP on the N nodes S. By using one of the approximation schemes of Arora [5] or Mitchell [16, 17], we can achieve $3\alpha_{TSP} = 3(1 + \epsilon)$ in a running time of $O(N^{O(\frac{1}{\epsilon})})$. If we use the Christofides heuristic, we obtain an overall approximation factor of $3\alpha_{TSP} = 4.5$, at a running time of $f(N) = O(N^{2.5} \log^4 N)$ (the bottleneck being the computation of the minimum-weight matching [21]). Alternatively, we can apply the approximate matching result of Vaidya [22], within the Christofides heuristic, to achieve a factor of $4.5 + \epsilon$ at a time complexity of $f(N) = O(N^{1.5} \log^{2.5} N)$. Finally, by simply doubling the minimum spanning tree, we obtain a factor of 6, with a simple algorithm of complexity $f(N) = O(N \log N)$.

We can also obtain time bounds for computing an approximate length of an optimal tour, as well as an implicit representation of an approximating tour, in time that depends on n (the combinatorial size of the input) rather than on N (which depends on the numerical size of the input data). For this discussion, we restrict attention to the case of a rectilinear region R, having all of its edges parallel to the coordinate axes. Our strategy will be to compute a minimum spanning tree of S, and then double it (so that $\alpha_{TSP} = 2$).

First, we decompose the region R into rectangles; this takes time $O(n)$, if R is simply connected ([6]), or time $O(n \log n)$, if R is multiply connected (e.g., by standard plane sweep). Next, we compute the L_1 Voronoi diagram of the set of rectangles, treating them as the sources, and using the L_1 (rectilinear) metric. This gives a planar subdivision, of size $O(n)$, of the complement of R, which allows us to compute the nearest neighbor information for each rectangle. Thus, for each rectangle τ, we know which other rectangles are Voronoi neighbors (as well as which ones share edges with τ).

For rectangle τ, we let S'_τ denote the set of pixel centerpoints for the set of pixels intersected by τ. With this definition, notice that a centerpoint can belong to more than one set S'_τ. In order to obtain a partition of the centerpoints S, we define sets $S_\tau \subseteq S'_\tau$ by assigning centerpoints uniquely to rectangles. For pixels that intersect more than one rectangle, we determine which set S_τ should "own" the centerpoint, based on a convention, such as the following: Sweep a vertical line across the pixel, stopping at the first instant that one or more rectangles is intersected (this instant may, in fact, be at the left boundary of the pixel); we assign the pixel to the rectangle that is topmost in the intersection with the sweepline. An implicit representation of the sets S_τ can be determined using the Voronoi neighbor information for rectangles. Further, each set S_τ has an $O(1)$-size implicit description of its minimum spanning tree; e.g., we can simply link up the points S_τ, row by row, and then link the rows together.

We now want to compute a minimum spanning tree (MST) of all of S, by linking together the spanning trees of the sets S_τ. (Note that, by standard properties of minimum spanning trees, we know that the (unit-length) edges in the spanning trees of the sets S_τ are in an MST of S.)

To link together the resulting connected components, we apply a Kruskal-like greedy algorithm, starting with a forest consisting of the spanning trees of the sets S_τ, and, at each stage, linking a pair of components with an edge whose L_1 length is minimum among edges linking centerpoints from different components. Such edges can be identified using Voronoi neighbor information for the rectangles. (Note too that if two rectangles are touching, then their trees can be linked easily, with a unit-length edge, which is clearly of minimum length.) These edges are used in the Kruskal algorithm to link up the components to obtain the overall minimum spanning tree of S. (The fact that we can restrict attention to Voronoi neighbors follows from standard properties of minimum spanning trees.)

Finally, we remark that the method sketched above should be applicable to more general regions than just rectilinear ones; e.g., if the region R has edges from some small fixed set of rational slopes, then a trapezoidal decomposition can be used, with an implicit representation of the pixel centerpoints associated with each. Then, by approximating arbitrary regions with such polygonal regions, it should be possible to obtain similar $O(n \log n)$ time bounds for approximating any region R.

Removing the additive term. The above approximation bound includes both a multiplicative and an additive error term. We now give a second approximation method that can be applied in cases in which the length of an optimal lawn mower tour is small. Then, by selecting the shorter of the two tours obtained by our two methods, we can give an error bound based on a purely multiplicative approximation factor.

Our second approach begins by finding the axis-aligned bounding box (rectangle) of R; assume this rectangle is of size a-by-b, with $a \leq b$.

If $a \leq 1$, mowing this rectangle trivially yields an optimal solution.

If $1 < a \leq 2$, then we can again obtain an optimal tour easily, by noting that the mower must travel at least distance $2(a + b - 2)$ in order to touch all four edges of the bounding rectangle with the boundary of the lawn mower. Thus, by simply mowing the rectangle with a rectangular tour of length $2(a + b - 2)$, we obtain an optimal tour.

If $a > 2$, then, as can be seen from Figure 9, we can mow the rectangle by concatenating a series of concentric rectangular subtours. For all except possibly for the innermost subtour, these concatenations can be performed without extra cost by swapping edges as shown in Figure 9. Joining the innermost subtour may cost an extra two unit edges if $b - 1 - 2\lfloor \frac{a}{2} \rfloor < 1$. Therefore, we can mow the rectangle by traveling a total length of no more than

$$2 + \sum_{i=0}^{\lfloor \frac{a}{2} \rfloor} 2(b - 1 - 2i) + 2((a - 1 - 2i)$$

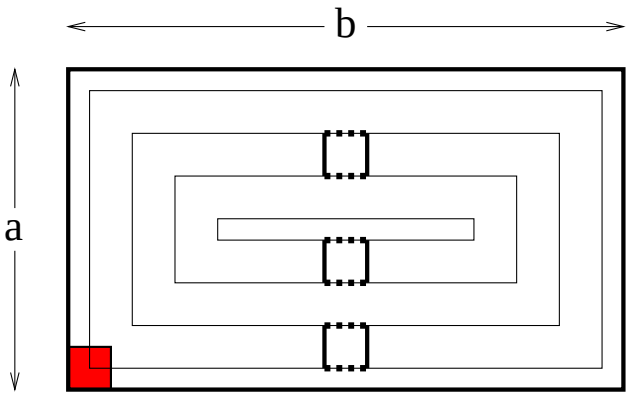

Figure 9: Mowing a rectangle

$$= 2 + 2(a+b-2)\left(\left\lfloor\frac{a}{2}\right\rfloor + 1\right) - 4\left(\left\lfloor\frac{a}{2}\right\rfloor\right)\left(\left\lfloor\frac{a}{2}\right\rfloor + 1\right) =: \ell_{rect}.$$

Now

$$\frac{\ell_{rect}}{\ell^*} \leq \frac{\ell_{rect}}{2(a+b-2)} = \left(\left\lfloor\frac{a}{2}\right\rfloor + 1\right)\left(1 - \frac{2\left\lfloor\frac{a}{2}\right\rfloor}{a+b-2}\right) + \frac{1}{a+b-2} \leq \left(\left\lfloor\frac{a}{2}\right\rfloor + 1\right).$$

This guarantees an approximation factor better than $3\alpha_{TSP}$, as long as we only consider $\left(\left\lfloor\frac{a}{2}\right\rfloor + 1\right) \leq 3\alpha_{TSP}$. Therefore we can assume $a \geq 6$, hence $\ell^* \geq 20$. This means that we can drop the additive term in our approximation estimates by adding no more than $\theta = \frac{3}{10}$ to the multiplicative estimates. (This estimate for θ can be improved; we omit a tighter analysis in the interest of brevity.)

Unrestricted motion of a square cutter. If we consider a unit square cutter that is allowed to translate arbitrarily (while not rotating), rather than only in a rectilinear fashion, then we can apply our same approach as before, with a slight modification to the perturbation scheme used in the proof of Lemma 2. In particular, we argue that we can perturb an optimal tour T^* onto a *grid with diagonals* determined by centerpoints of pixels. In the grid with diagonals, we join two centerpoints with a horizontal or vertical edge if they are separated by distance 1, and by a diagonal (45-degree) edge if they are separated by distance $\sqrt{2}$. A simple calculation shows that any straight segment, of length ℓ, joining a pair of centerpoints can be replaced by a path of length at most $\beta \cdot \ell$, in the grid with diagonals, where $\beta = \frac{2}{\sqrt{2+\sqrt{2}}} \approx 1.08$. The results go through as before, but with an additional factor β in the approximation ratio.

Circular cutters. We can further extend the method to the case of circular cutters. Assume that χ is a unit-radius disk that is allowed to move arbitrarily in the plane. In this case, we use another form of "pixel" — instead of tiling the plane with unit squares, we tile the plane with regular hexagons, each of diameter 2. The centerpoints of these hexagons lie on a regular lattice. We join two centerpoints by an edge if they lie at distance $\sqrt{3}$; this results in a planar graph whose faces are equilateral triangles of side length $\sqrt{3}$. See Figure 10. Any straight segment of length ℓ between two centerpoints can be approximated by a path in the graph of length at most $\gamma \cdot \ell$, where $\gamma = \frac{2\sqrt{3}}{3} \approx 1.15$.

As in the case of a mowing square, we can remove the additive term from the approximation bound by considering the smallest disk of radius $\rho \geq 1$ that contains R. Mowing that disk can be done with a tour of no longer than

$$\sum_{i=0}^{\frac{\lfloor\rho-1\rfloor}{2}} 2\pi(\rho - 1 - 2i) + 2\lceil\rho - 2\rceil \approx \frac{\rho^2\pi}{2} + 2(\rho - 2),$$

while an optimum tour must have a length of at least $4(\rho - 1)$.

In summary, we have

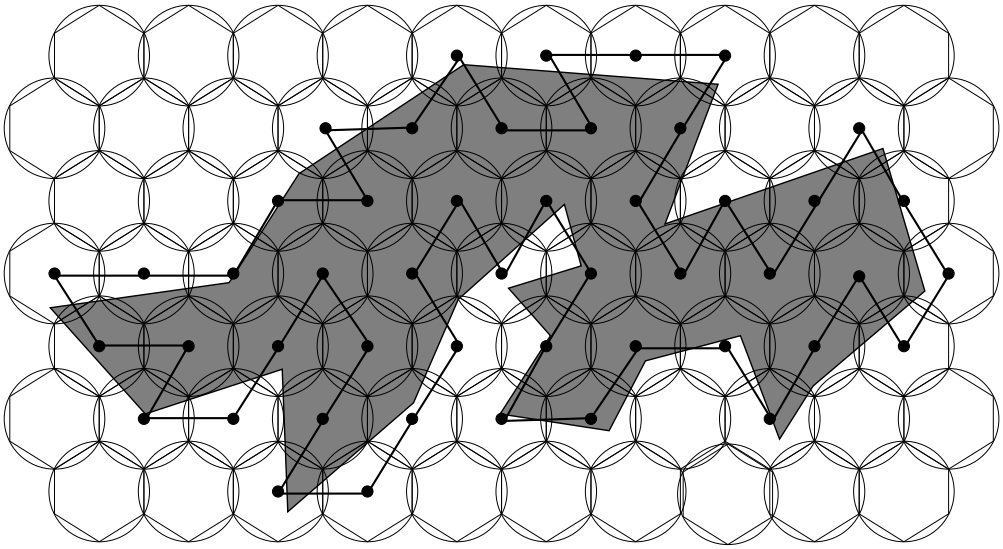

Figure 10: Approximation for the case of a circular cutter χ.

Theorem 3 *The lawn mowing problem has a constant-factor approximation algorithm that runs in polynomial time (dependent on the TSP heuristic employed). For the case of an aligned unit square cutter, the approximation factor is $3\alpha_{TSP}$ for rectilinear motion, and is $3\beta\alpha_{TSP}$ for arbitrary translational motion. For the case of a unit circular cutter, with arbitrary motion, the approximation factor is $3\gamma\alpha_{TSP}$. Here, $\beta = \frac{2}{\sqrt{2+\sqrt{2}}} \approx 1.08$, and $\gamma = \frac{2\sqrt{3}}{3} \approx 1.15$.*

5 Approximation Methods for the Milling Problem

We consider now the problem of pocket machining, in which we must compute a milling tour with the cutting tool required to stay within the region (pocket) R that is to be milled.

Theorem 4 *In time $O(n \log n)$, one can decide whether a (multiply) connected region with n sides (straight or circular arc) can be milled by a unit disk or unit square, and, within the same time bound, one can construct a tour of length at most $2\frac{1}{2}$ times the length of an optimal milling tour.*

Proof. One can check for millability in $O(n \log n)$ time using a medial axis to compute offsets based on the cutter χ. (Offset the boundary inward, then offset this outward, and compare to the original region.)

So assume that R can be milled. Let $B \subset R$ denote the inward offset region of all points within R that are feasible placements for the center point of the milling cutter. B is connected. The length $L_{\delta B}$ of the boundary δB of B is a lower bound for length L_{OPT} of an optimal milling tour. We write $R_{\delta B}$ for the region milled by moving along δB. Note that δB may consist of several pieces if R is not simple. In the following, we will refer to these pieces as δB_i. If $R_{int} := R \setminus R_{\delta B}$ is nonempty, we can cover it by a set of s horizontal strips S_i of vertical width 1 and disjoint interior, as shown in Figure 11: The vertical coordinates of the center lines of any two strips differ by a multiple of 2. (In the following, we will refer to the center line of a strip as a "strip line".) Each strip lies completely inside of R, so there may be several strips that have the same vertical coordinate. By a simple area argument, we need at least the length $L_{str} = \sum_{i=1}^{s} L_{S_i}$ to mill R, so we conclude that $L_{str} \leq L_{OPT}$.

By the choice of the strips, every δB_i contains an even number of endpoints of strip lines. This means that these endpoints partition every δB_i into two sets of pairwise disjoint subportions. We will refer to these two sets as the two "matchings" $M_1(\delta B_i)$ and $M_2(\delta B_i)$ of δB_i. For every δB_i, let $M^*(\delta B_i)$ be the shorter matching. Clearly, the combined length of all $M^*(\delta B_i)$ is at most $\frac{L_{\delta B}}{2} \leq \frac{L_{OPT}}{2}$.

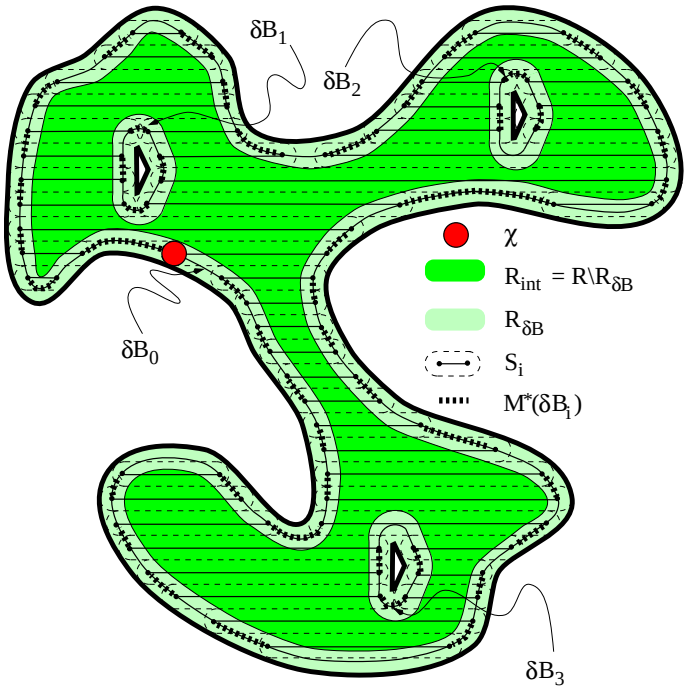

Figure 11: Approximating a milling tour

Now consider the graph formed by the endpoints of strip lines, plus the points where a strip line touches a δB_i. These vertices are canonically connected by the center lines of the strips, the δB_i and the $M^*(\delta B_i)$. Clearly, this graph is connected and every vertex has degree 4. This means that it has a Eulerian tour, which forms a feasible milling tour of length no longer than $\frac{5}{2}L_{OPT}$. □

6 TSP on Simple Grid Graphs

Consider the special case of the milling problem with a unit square cutter, under rectilinear motion, and a region R given by a simple rectilinear polygon having vertices with integer coordinates. It is easy to see that the milling problem in this case is equivalent to the problem of finding an optimal TSP tour on the *simple grid graph*, G, induced by the centerpoints of the pixels that comprise R. Specifically, G is the graph whose node set is the set of centerpoints of pixels within R, and two nodes are joined by an edge if and only if the corresponding centerpoints are at distance 1 from each other. (It is "simple" in the sense of having no "holes"; i.e., all grid points interior to any simple cycle in G must also be nodes of G.) Here, since G may not be Hamiltonian (e.g., G may consist of a single horizontal row of centerpoints), by a "tour" we mean a closed walk that visits every node at least once (possibly revisiting some nodes).

A node v is a *cut vertex* if its removal disconnects G. If G has a cut vertex v, then we can consider separately the approximation problem in each of the components obtained by removing v, and then splice the tours back together at the vertex v to obtain a tour in the entire graph G. Thus, we concentrate on the case in which G has no cut vertices.

The problem was considered by Ntafos [19], who showed (constructively) that, provided there are no cut vertices, there exists a tour of length at most $(4/3)N$, where N is the number of grid points in the graph. This immediately gives a 4/3-approximation algorithm (since N is a trivial lower bound on the length of any tour), thereby improving, in this special case, on the Christofides bound of 3/2. In this section, we improve the result of Ntafos by giving a method to obtain a tour of length at most $\frac{6}{5}$ times the length of an optimal

TSP tour, and this result is tight for the case of no cut vertices. Note that it is still open whether an exact solution for TSP instances on simple grid graphs can be found in polynomial time. Grigni et al. [10] have given a polynomial approximation scheme for arbitrary grid graphs, as did Arora [5] and Mitchell [16, 17] for the general Euclidean case; however, the algorithms from those approximation schemes are not very practical for small approximation factors, whereas our method achieves optimal running time of $O(n)$.

Theorem 5 *Let G be a simple grid graph, having N nodes at the centerpoints, V, of pixels within a simple rectilinear polygon, R, having n (integer-coordinate) sides. Assume that G has no cut vertices. Then, in time $O(n)$, one can find a representation of a tour, T, that visits all N nodes of G, of length at most $\frac{6N-4}{5}$.*

Proof. Let $V_B \subseteq V$ denote the set of centerpoints of pixels on the boundary of R (i.e., one or more sides of the pixel are shared with the boundary of R). Let $V_I \subset V$ denote the set of centerpoints of *non*-boundary ("internal") pixels. Thus, the node set for the grid graph G is partitioned into boundary nodes (V_B) and internal nodes (V_I).

As defined, the points V_B and V_I lie at half-integer coordinates in the plane (since they are centers of pixels). However, for simplicity of exposition, we can translate the points $V_B \cup V_I$ by $(\frac{1}{2}, \frac{1}{2})$, and work from now on with the assumption that G is a subgraph of the integer grid graph.

The fact that G has no cut vertices implies that there exists a *simple* cycle, C, through only boundary nodes V_B, visiting each boundary node exactly once; we call C the *contour* of G. Figure 12 shows an example of a contour C (shown as a bold outline) through the nodes V_B; the internal nodes V_I are indicated by (hollow) circles.

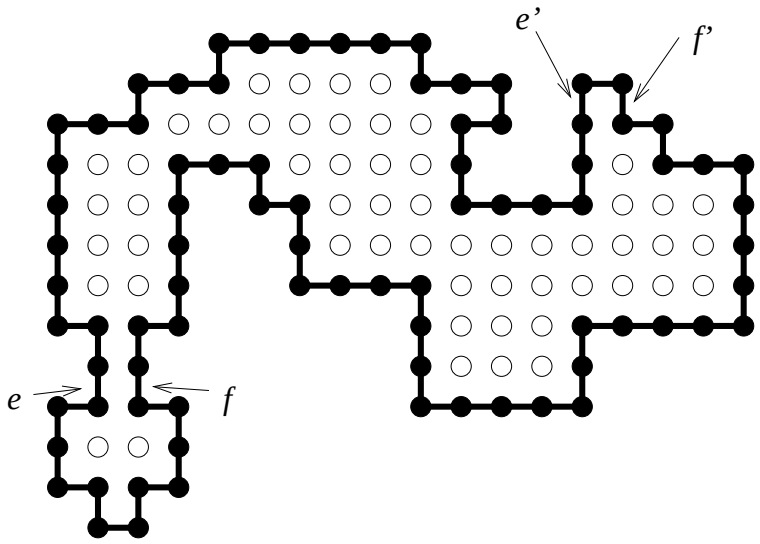

Figure 12: A contour C (shown bold) surrounding the internal nodes V_I of a simple grid graph that has no cut vertex.

We say that two edges, e and f, of C form a *bottleneck* if

1. e and f are parallel, separated by distance 1, and
2. there does not exist an edge g of C such that e, g, and f form a subpath of C of length 3.

(An alternative definition is to say that e and f form a bottleneck if they are opposite sides of a unit square, such that neither of the other two sides of the square is an edge of C.) For example, in Figure 12, the pair (e, f) of edges forms a bottleneck, while the pair (e', f') does not.

The following lemma shows that it will suffice to consider the case in which C has no bottlenecks.

Lemma 5 *Consider a simple grid graph G with N vertices that does not have any cut vertices. Suppose we can find a tour of length at most $\frac{6N-4}{5}$ for any simple grid graph* without bottlenecks*, such that any degree 2 vertex is connected to both of its neighbors. Then we can find a tour of length at most $\frac{6N-4}{5}$ for* any *simple grid graph without cut vertices, such that any degree 2 vertex is connected to both of its neighbors.*

Proof. We proceed by induction over the number of bottlenecks. Let $e = (v_1, v_2)$ and $f = (v_3, v_4)$ form a bottleneck, separating the pieces G_1 and G_2, consisting of N_1 and N_2 vertices, resp., so $N = N_1 + N_2 + 4$. (See Figure 13.) Then we can decompose G into two pieces, H_1 induced by G_1, v_1, v_2, v_3, v_4, and H_2 induced by G_2, v_1, v_2, v_3, v_4. In H_1, v_2 and v_4 are vertices of degree 2, as are v_1 and v_3 in H_2. Therefore we have a tour of length at most $\frac{6N_1+20}{5}$ of H_1 that consists of a path p_1 from v_1 to v_3 and the edges (v_2, v_4), e, f; similarly, we have a tour of H_2 which is no longer than $\frac{6N_2+20}{5}$ and consists of a path p_2 from v_2 to v_4 and the edges (v_1, v_3), e, f. Therefore, the union of p_1, f, p_2, e forms a tour of G in which all degree 2 vertices are connected to their neighbors. Its length is at most $\frac{(6N_1+20)+(6N_2+20)+5+5}{5} = \frac{6(N_1+N_2+4)-4}{5} = \frac{6N-4}{5}$. □

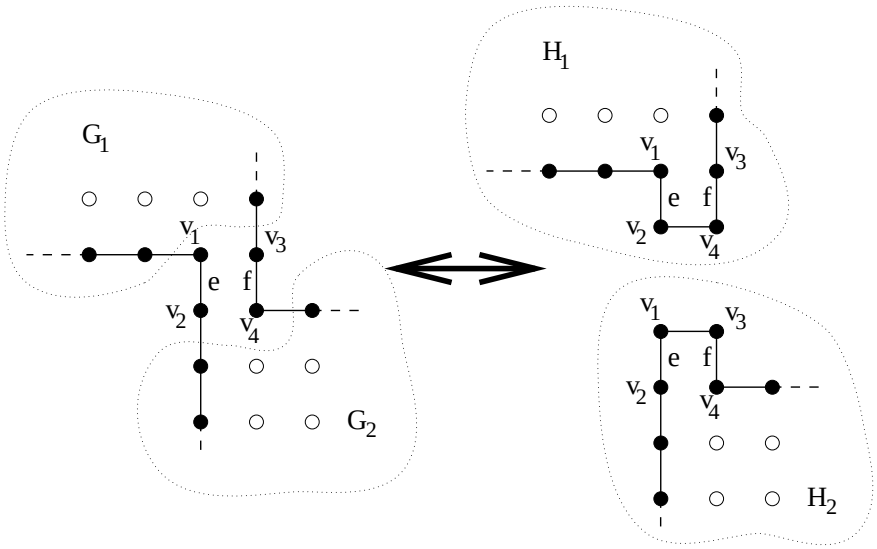

Figure 13: How to deal with bottlenecks

So, for the remainder of the proof of Theorem 5, let us assume that C has no bottlenecks.

To construct the tour T, we begin with the contour C through the boundary nodes V_B. We will then argue that we can modify C, through a series of detours, into a tour that also visits every internal node, in such a way that whenever the detour causes us to "waste" an edge (by going to a node that has been visited already), we can "charge" this wasted edge off to some group of at least 5 nodes. No node will ever belong to more than one group, and no group will be charged more than once for a wasted edge; thus, in all, there will be at most $\frac{6}{5}N$ edges in the final tour T.

We can assume, without loss of generality, that the internal nodes V_I have (integer) y-coordinates $1, 2, \ldots, k$. In order to define the "groups" of nodes that will be charged with wasted edges when we make modifications to C, we begin by partitioning V_I into *doublerows* obtained by pairing up the rows, $y = 1$ with $y = 2$, $y = 3$ with $y = 4$, etc.

Consider the internal nodes that lie in a doublerow defined by $y = j$ and $y = j + 1$; i.e., consider those points of V_I whose y-coordinate is either j or $j+1$, for an odd positive integer j. These internal nodes induce a grid graph, whose edges are defined by pairs of nodes at distance 1 from each other. For each (integer) i, this graph has 0, 1, or 2 (internal) nodes that lie in the column determined by the vertical grid line $x = i$; we refer to these columns as being an *empty* column, a *singleton* column, or a *doubleton* column. Note that in a singleton column, there must be exactly one internal node and one boundary node.

We describe a set of possible modifications ("detours") that we will apply to the contour C. Let C' denote the modified contour at any particular stage of the modifications; we maintain the invariant that C' is a closed walk within the grid graph G that visits every boundary node (V_B) at least once. Initially, $C' = C$; once C' visits all nodes ($V_B \cup V_I$), we will be done, and we will take T to be the tour C'. Our modifications are "monotone" in that each modification will result in C' visiting a superset of the nodes that it previously visited. We let $V_I' \subseteq V_I$ denote the set of internal nodes not yet visited by C', and we let G_I^j denote the grid graph induced by those nodes of V_I' that lie in the doublerow determined by $y = j$ and $y = j + 1$.

We say that a modification to the contour C' is *free* if it results in no wasted edges; i.e., the net increase in the number of edges of C' exactly equals the number of (internal) nodes that are newly visited by the modified contour.

We consider two types of free modifications to C'. These are done within a doublerow (say, defined by $y = j$ and $y = j + 1$). Both modifications are based on finding an edge of C' that is parallel to (and at distance 1 from) an edge of G_I^j, and then modifying the edge of C' to detour through the two (internal) nodes that define the edge of G_I^j.

I. If a doubleton column (at, say $x = i$) is adjacent to an empty column (at, say $x = i - 1$) that has a (vertical) edge $(u, v) = ((i - 1, j), (i - 1, j + 1))$ of C', then a *Type I detour* replaces edge (u, v) with the path that goes from u to (i, j) to $(i, j + 1)$ to v. Such a detour can be repeated, "pushing" the vertical edge of C' rightwards, until we run out of doubleton columns. See Figure 14.

II. If two singleton columns (at, say $x = i$ and $x = i + 1$) have their internal nodes in the same row (at, say $y = j$)[1] and C' includes the (horizontal) edge $(u, v) = ((i, j + 1), (i + 1, j + 1))$ joining the other two nodes in these columns, then a *Type II detour* replaces edge (u, v) with the path that goes from u to (i, j) to $(i + 1, j)$ to v. Such a detour can be repeated if there is a path of horizontal edges of C' opposite a row of internal nodes, resulting in a "zig-zag" detour that visits all but one of the internal nodes in the row. See Figure 14.

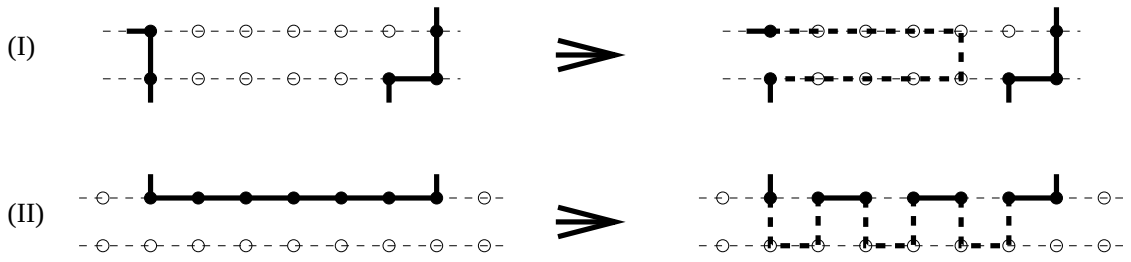

Figure 14: *Top: Type I detours allow C' to visit the internal nodes in a sequence of consecutive doubleton columns. Bottom: Type II detours applied to a horizontal row of edges of C' allow C' to visit all but one of the corresponding internal nodes.*

We continue to perform the above free modifications until no longer possible. In the end, if we have incorporated all internal nodes into the contour C', we are done (we use the tour $T = C'$, and this tour is necessarily optimal, since its length is N). Otherwise, we are left with a set of internal nodes, $V_I' \subseteq V_I$ such that each connected component, H, of G_I^j, corresponding to the doublerow determined by $y = j$ and $y = j + 1$, must satisfy the following properties:

1. No doubleton column of H is adjacent to a vertical edge of C';
2. There are no three consecutive singleton columns of H having the internal nodes all in the same row; and

[1]Hence, there is an edge, $((i, j), (i + 1, j))$, of G_I^j joining these two nodes.

3. The only way to have two consecutive singleton columns of H is if H consists of just one (horizontal) edge joining the two nodes of the two singleton columns.

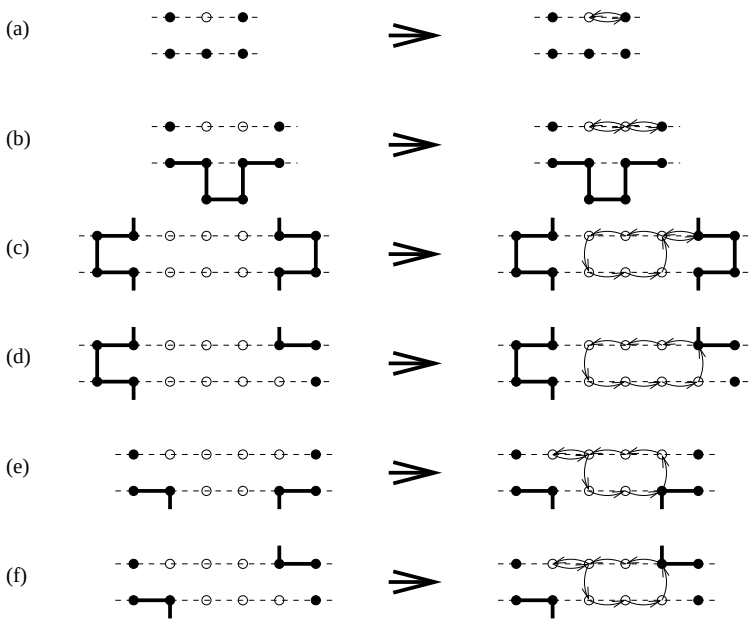

Figure 15: *Six cases for incorporating internal nodes into the modified contour tour. Hollow circles denote internal nodes, V'_I, and solid circles denote nodes of C'. Solid edges are drawn where there* must *be edges of C'.*

This leaves us with six possible cases for a component H, as illustrated in Figure 15. If H has no doubleton columns, then H must either be a single node (case (a)) or a single horizontal edge joining two nodes (case (b)). If H has at least one doubleton column, then we define the *width* of H to be the number of doubleton columns of H. The resulting four cases are distinguished based on the number of singleton columns of H: 0 (case (c)), 1 (case (d)), or 2 (cases (e) and (f)). If there are 2 singleton columns, we distinguish between the case in which the two corresponding internal nodes lie in the same row (case (e)), and the case in which they do not (case (f)).

(a) *H consists of a single node.* In this case, there must be five nodes of C' in the doublerow, in the (singleton) column containing H, and the two neighboring columns, as shown in Figure 15(a).

In this case we waste one edge, but have six nodes to charge. We charge any five of them; since our subdivision into doublerows forms a partition of the interior nodes, none of those five nodes will ever be charged again. The remaining sixth node remains uncharged and may be referred to later, see below.

(b) *H consists of two nodes, joined by a single horizontal edge.* In this case, the two (boundary) nodes that are in the same two singleton columns as H occupies must *not* be joined by an edge of C' (since, otherwise, a Type II detour could be performed). Instead, there must be a vertical edge of C (not just C') incident on each of these two boundary nodes, and, therefore, by our assumption about no bottlenecks, there must also be an edge of C linking the other ends of these two vertical edges (i.e., a horizontal edge at distance 2 from H).

None of the ten nodes that we charge will ever be charged again.

(c) *H consists of only doubleton columns.*

In this case we waste two edges, but have at least 10 nodes to charge. Again, it follows from our partition that none of the ten nodes that we charge will ever be charged again.

(d) *H has one or more doubleton column, and exactly one singleton column.*

In this case we waste one edge, but have at least 10 nodes to charge. None of the ten nodes that we charge will ever be charged again.

(e) *H has one or more doubleton column, and exactly two singleton columns, whose corresponding nodes lie in the same row.*

In this case we waste two edges, but have at least 10 nodes to charge. None of the ten nodes that we charge will ever be charged again.

(f) *H has one or more doubleton column, and exactly two singleton columns, whose corresponding nodes lie in different rows.*

In this case we waste two edges, but have at least 10 nodes to charge. None of the ten nodes that we charge will ever be charged again.

There is another technical detail needed in order to apply Lemma 5: We must argue that we can leave 4 vertices uncharged, after applying all of the above charging scheme, so that we get a bound of $\frac{6N-4}{5}$. Since we are free to choose the parity of our subdivision into doublerows (i.e., have them at coordinates $2i-1$ and $2i$, or at $2i$ and $2i+1$), we can make this choice in a way that leaves a "bottommost" or a "topmost" part of C' (i.e., the set of boundary vertices that have y-coordinate smaller or larger than any interior vertex) out of any doublerow. Clearly, any such piece will contain at least three vertices. If the bottommost part has only three vertices, a doublerow containing it can only have one interior vertex – as in case (a) above. Therefore we can either leave a part with more than three vertices uncharged, or leave three vertices uncharged on one side, plus another vertex as described in case (a) above. In either case, we are done.

Finally, we remark that the running time of the algorithm to produce the claimed approximate tour can be bounded by $O(n)$, the time needed to partition R into horizontal trapezoids. Within each such trapezoid we can perform the free modifications, in block, in $O(1)$ time, after which we are left with only $O(n)$ doublerows to which we apply the above case analysis.

□

It should be noted that the 6/5 bound is asymptotically tight – consider the class of examples shown in Figure 16.

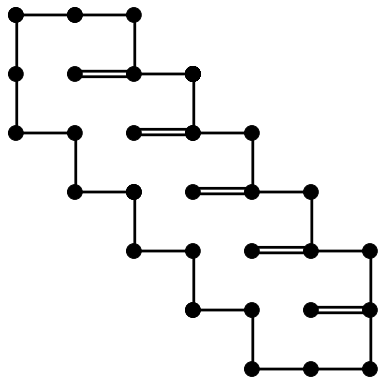

Figure 16: A tight class of examples

A direct corollary of Theorem 5 is a $\frac{6}{5}$-approximation bound for milling rectilinear simple polygons having integer coordinates:

Corollary 3 *The milling problem for a unit square cutter, rectilinear motion, and a simple rectilinear polygon region R having integer coordinates can be approximated within a factor of $\frac{6}{5}$ of optimal, in time $O(N)$, where N is the number of pixels contained within R. If the simple polygonal region has n edges, it is straightforward to show that we can find such a tour in time $O(n)$.*

A further corollary is a $\frac{11}{5}$-approximation bound for milling rectilinear simple polygons having *arbitrary* coordinates:

Corollary 4 *The milling problem for a unit square cutter, rectilinear motion, and a simple rectilinear polygon region R can be approximated within a factor of $\frac{11}{5}$ of optimal, in time $O(n)$.*

Proof. First, one can decide in time $O(n)$ if R has a feasible milling tour, by doing an offset (e. g., using recent linear-time algorithms for computing the medial axis of simple polygons [7]). This tells us if the contour is feasible. Consider the union R_{IN} of all integer pixels. By the above method, we can find a tour of R_{IN} that is within $\frac{6L_{OPT}-4}{5}$ of the length L_{OPT} of an optimum milling tour for R. Combining this with a tour that follows the contour δB of R and has length $L_{\delta B} \leq L_{OPT}$, we get a tour no longer than $\frac{11}{5}L_{OPT} + \frac{6}{5}$. □

7 TSP on Grid Graphs with Holes

We can extend the results of the previous section to grid graphs with holes that satisfy an additional condition on the local structure.

Definition 6 *We say that a vertex v in a grid graph G is a* local cut vertex*, if the removal of v disconnects G or reduces the number of holes.*

We show that all grid graphs without local cut vertices (but possibly with holes) allow a tour of short length; our bound is better than the previously best known estimate of $\frac{4}{3}N$ for simple grid graphs without cut vertices [19].

Theorem 7 *Let G be a connected grid graph, having N nodes at the centerpoints, V, of pixels within a (multiply connected) rectilinear polygon, R, having n (integer-coordinate) sides. Assume that G has no local cut vertices. Then, in time $O(n)$, one can find a representation of a tour, T, that visits all N nodes of G, of length at most $1.325N$.*

Proof. Applying the method in the proof of the previous section, we can link internal grid vertices to a boundary contour, at a cost of only roughly $(1/5)N$ extra edges. But, if there are h holes in the grid graph, then this linking process may result in $h+1$ independent subtours that do not interconnect. We build "bridges" between these subtours in order to link them into a single tour. First, we perform "free" bridges, by identifying pairs of facing edges, from two distinct subtours, that bound a common pixel and can therefore lead to a "swap" that interconnects the respective subtours. We are now left with $h' \leq h+1$ subtours S_i. One of them (say, S_0) contains the outside contour, each of the others is surrounded by S_0 while surrounding at least one hole H_i.

Consider a subtour S_i ($i \neq 0$) with the smallest number of vertices. (See Figure 17.) Let x_a, x_z, y_a, z_z be the smallest and largest x- and y-coordinates of vertices in S_i, and let x_b, x_w, y_b, y_w be the smallest and largest x- and y-coordinates of vertices in H_i. Clearly, $x_a < x_b \leq x_w < x_z$ and $y_a < y_b \leq y_w < y_z$. Furthermore, any coordinate in the interval $[x_b, x_w]$ must contain at least two vertices of S_i and each of the coordinates $x_b - 1$, $x_w + 1$ must contain at least three vertices of S_i. This implies that the rectangle $[x_b - 1, x_w + 1] \times [y_b - 1, y_w + 1]$ contains at least eight vertices of S_i.

Consider the case ($x_a = x_b - 1$) or ($x_z = x_w + 1$) or ($y_a = y_b - 1$) or ($y_z = y_w + 1$), so w. l. o. g. ($x_a = x_b - 1$). There must be three vertices $v_1 = (x_a, y_1)$, $v_2 = (x_a, y_1 + 1)$, $v_3 = (x_a, y_1 + 2)$, such that v_1 and v_3 are neighbors of v_2 in S_i. Now consider the $u_1 = (x_a - 1, y_1)$, $u_2 = (x_a - 1, y_1 + 1)$, $u_3 = (x_a + 1, y_1 + 2)$. Since S_i is surrounded by S_0, u_2 must be contained in some subtour $S_j \neq S_i$. u_2 cannot be adjacent to either u_1 or u_3 in S_j, since this would allow a free bridge between S_i and S_j. Therefore, S_j must enter and leave u_2 from the vertex $t_2 = (x_a - 2, y_1 + 1)$. Without any additional cost, we can replace the two unit edges between t_2 and u_2 by two unit edges between u_1 and u_2. This allows a free bridge between the (modified) subtour containing u_1 and the subtour S_i, since (u_1, u_2) and (v_1, v_2) are parallel edges bounding a common pixel.

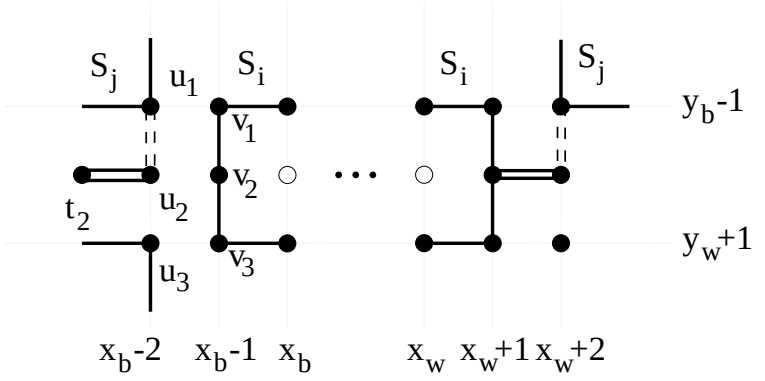

Figure 17: A smallest subtour must contain 16 vertices

Now assume $(x_a < x_b - 1)$, $(x_z > x_w + 1)$, $(y_a < y_b - 1)$, and $(y_z > y_w + 1)$, and consider the set $U = \{x_w + 2\} \times [y_b - 1, y_w + 1] \cap S_i$. (See Figure 17.) As in the previous paragraph, it is straighforward to argue that we can create a free bridge and link S_i to another subtour if $|U| < 2$. Therefore we may assume $|U| \geq 2$. Similarly, the (disjoint) sets $\{x_b - 2\} \times [y_b - 1, y_w + 1] \cap S_i$, $\{y_b - 2\} \times [x_b - 1, x_w + 1] \cap S_i$, and $\{y_w + 2\} \times [x_b - 1, x_w + 1] \cap S_i$ can all be assumed to contain at least two points. With the eight points in the rectangle $[x_b - 1, x_w + 1] \times [y_b - 1, y_b + 1]$, this implies that S_i contains at least 16 vertices; hence, $h' \leq N/16$.

At a cost of at most 2 links per bridge, we can now construct (non-free) bridges to link up the remaining subtours, resulting in a total tour length of not more than $(6/5)N + 2(N/16) = (53/40)N = 1.325N$. □

8 Conclusion

In this paper, we have given a variety of algorithmic results on the most basic forms of "lawn mowing" and "milling" problems. We have shown that many forms of the problems are NP-hard, and we have given constant-factor approximation algorithms for most of these instances. There are several ways in which the problems can be generalized:

1. We may consider a combination of the lawn mowing problem and the milling problem in which the complement of the region R (the "grass") is partitioned into two kinds of regions — a set X of points over which the cutter may pass freely (e.g., the "sidewalk" and the "driveway"), and a set Y of points over which the cutter is not allowed to pass (e.g., the "flowerbeds" and the "house"). The milling problem is that in which $Y = \mathbb{R}^2 \setminus R$, while the lawn mowing problem is that in which $X = \mathbb{R}^2 \setminus R$.

2. When milling a pocket in practice, it is important to try *not* to allow the cutter to pass over a portion of the region R that has already been milled, as such "over-milling" may lead to imperfections in the smoothness of the cut on the bottom of the pocket. (See Held [11] for a more extensive discussion.) Thus, one often wants to plan a cutter path that picks up the cutter ("retracts" it) and moves it to a new position, in order to avoid passing back over a portion that has been milled already. In such plans, however, one wants to minimize the number of retractions. For the special case of "zig-zag" pocket machining, Arkin et al. [3] have recently obtained approximation algorithms for the problem of minimizing the number of retractions. Ideally, one could combine our version of the "milling problem" with more realistic models of how machining is actually done, and study problems that include both the cutter path length *and* the number of retractions in the objective function.

3. We may also want to consider the issue of "over-milling" from a different point of view. In applications to spray painting and material deposition, the goal may be to cover every point at least k_{min} times (to

guarantee good coverage), and at most k_{max} times (to prevent overcoating, which may lead to "runs"), for a fixed speed of moving the tool. (Alternatively, one may wish to control the speed of tool motion or the flow rate of material through the nozzle.) In such problems, the goal may be to minimize wasted (over-sprayed) material and/or maximize the uniformity of coverage.

4. Another important consideration from a practical point of view is the "shape" of a cutter path/tour. If a path/tour has many sharp turns, it may require a slower processing speed, thus spoiling the benefits of having a "short" path/tour. Thus, we may be motivated to study the problems of minimizing "link distance", possibly in conjunction with Euclidean length and "total turn" (the integral of the absolute value of all changes in direction of motion). For "bicriteria" shortest path problems among obstacles in the plane, see Arkin et al. [4] and Mitchell et al. [18], who consider combinations of link distance, total turn, and Euclidean length. Held [11] has considered the issue of bounding the angles at turns in cutter path planning. For a finite set of points in the plane, the existence of "angle-restricted tours" has been studied by Fekete and Woeginger [9].

5. We may consider the case in which the cutter may move at different speeds through different portions of the material (e.g., "grass" of different heights) or at different orientations with respect to the material (e.g., "grain" effects in using a router in wood).

6. We may consider other allowed motions for cutters and other shapes of cutters.

7. We may consider the option to select different cutter sizes. For example, a large cutter may be used to "rough out" the pocket (or a large tractor may be used to mow the open field), while a smaller cutter is used to do fine details near the boundary of R.

8. We may consider the need to stop the cutting periodically, possibly several times during a single job, in order to resharpen the tool or perform some other function (e.g., empty the bag of grass clippings), which possibly involves sending the cutter to a particular location (depot). When mowing a lawn, one often tries to schedule the cutting route so that one is close to the compost pile when the bag that catches clippings becomes full.

Acknowledgement

We thank Martin Held for useful discussions and for specific suggestions that improved this paper.

References

[1] E. M. Arkin, S. P. Fekete, and J. S. .B. Mitchell. The lawnmower problem. *Proceedings of the 5th Canadian Conference on Computational Geometry*, pp. 461–466, Waterloo, Canada, 1993.

[2] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, **55**, 1994, pp. 197–218.

[3] E. M. Arkin, M. Held, and C. L. Smith. Optimization problems related to zigzag pocket machining. *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms 1996*, pp. 419–428.

[4] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Bicriteria shortest path problems in the plane. *Proceedings of the 3rd Canadian Conference on Computational Geometry*, pp. 153–156, 1991.

[5] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. *Proceedings of the 37th Annual IEEE Symposium on the Foundations of Computer Science*, pp. 2–12, 1996.

[6] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Computational Geometry*, **6**, 1991, pp. 485–524.

[7] F. Chin, J. Snoeyink, and C-A. Wang. Finding the medial axis of a simple polygon in linear time. *Proceedings of the 6th Annual International Symposium on Algorithms and Computing (ISAAC 95)*, pp. 382–391, Lecture Notes in Computer Science, Vol. 1004, Springer-Verlag, 1995.

[8] S. P. Fekete and W. R. Pulleyblank. Traveling the boundary of Minkowski sums. ZPR Report 97-254, `ftp://ftp.zpr.uni-koeln.de/pub/paper/zpr97-254.ps.gz`, 1997.

[9] S. P. Fekete and G. J. Woeginger. Angle-restricted tours in the plane. To appear, *Computational Geometry – Theory and Applications*. Available as ZPR Report 96-224, `ftp://ftp.zpr.uni-koeln.de/pub/paper/zpr96-224.ps.gz`, 1996.

[10] M. Grigni, E. Koutsoupias, and C. Papadimitriou. An approximation scheme for planar graph TSP. *Proceedings of the 36th Annual IEEE Symposium on the Foundations of Computer Science*, pp. 640–645, 1995.

[11] M. Held. *On the Computational Geometry of Pocket Machining*. Lecture Notes in Computer Science, Vol. 500, Springer-Verlag, June 1991.

[12] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal of Computing*, **11**, 1982, pp. 676–686.

[13] K. Iwano, P. Raghavan, and H. Tamaki, The traveling cameraman problem, with applications to automatic optical inspection. *Proceedings of the 5th Annual International Symposium on Algorithms and Computing (ISAAC 1994)*, Lecture Notes in Computer Science, Springer-Verlag, 1994.

[14] D. S. Johnson and C. H. Papadimitriou. Computational complexity and the traveling salesman problem. E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, editors, *The Traveling Salesman Problem*, pp. 68–74. John Wiley & Sons, New York, 1985.

[15] C. Mata and J. S. B. Mitchell. Approximation algorithms for geometric tour and network design problems. *Proceedings of the 11th Annual ACM Symposium on Computational Geometry*, pp. 360–369, 1995.

[16] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem. *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, pp. 402–408, 1996.

[17] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: Part II – A simple polynomial-time approximation scheme for geometric k-MST, TSP, and related problems. To appear, *SIAM Journal of Computing*. Available at `http://ams.sunysb.edu/~jsbm/jsbm.html`.

[18] J. S. B. Mitchell, C. Piatko, and E. M. Arkin. Computing a shortest k-link path in a polygon. *Proceedings of the 33rd Annual IEEE Symposium on the Foundations of Computer Science*, pp. 573–582, 1992.

[19] S. Ntafos. Watchman routes under limited visibility. *Computatinal Geometry – Theory and Applications*, **1** (3), 1992, pp. 149–170.

[20] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, New York, NY, 1985.

[21] P. M. Vaidya. Geometry helps in matching. *SIAM Journal of Computing*, **18**, 1989 pp. 1201–1225.

[22] P. M. Vaidya. Approximate minimum weight matching on points in k-dimensional space. *Algorithmica*, **4**, 1989, pp. 569–583.