CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Partitioning polygons into disjoint axis-parallel strips

Master's thesis in Computer Science and Engineering

August Matseke
Chiara Cesarini

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Partitioning polygons into disjoint axis-parallel strips

August Matseke
Chiara Cesarini

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Partitioning polygons into disjoint axis-parallel strips
August Matseke Chiara Cesarini

Supervisor: Peter Damaschke, Department of Computer Science and Engineering
Examiner: Peter Damaschke, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Partitioning polygons into disjoint axis-parallel strips
August Matseke, Chiara Cesarini
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

A relevant problem for manufacturing is that of finding a minimum-turn path that covers a surface. One way of doing this is by converting the surface into a grid-adhering polygon and partitioning it into single-width rectangles, or "strips". The focus of this thesis is on finding such a partition that contains the minimum number of strips. For this an approximation algorithm is developed which finds a strip partition in polynomial time, with an approximation ratio that depends upon properties of the problem instance. The results also include analysis of the problem in the form of multiple lower bounds on the number of strips in optimal partitions, as well as proving certain properties of optimal solutions.

The latter is done by introducing the concept of "switchlines": lines that separate strip regions of different orientations. Depending on the shape of said lines, it is possible to make claims on the optimality of a solution. The analysis also lays a foundation for the development of future algorithms to further improve the effectiveness and accuracy in solving this problem.

Keywords: Computational geometry; Geometric optimization; Approximation algorithms; NP-completeness; milling.

Acknowledgements

We would like to thank our supervisor and examiner Peter Damaschke for all the support and useful input along the way.

Gothenburg, June 2, 2026,

Chiara Cesarini, August Matseke

Contents

List of Figures

List of Tables

Glossary

approximation ratio An upper bound on the ratio between an algorithm's solution and the optimal solution. 5, 23

cell A square with the length of one unit within a polygon. Every polygon can be represented as a set of cells. 1

concave corner A corner in a polygon that is 270° with respect to the inside of the polygon. 3

convex corner A corner in a polygon that is 90° with respect to the inside of the polygon. 3

grid graph A connected set of nodes inside an orthogonal polygon. 3

integral point A set of coordinates in an $n-$dimensional space where each individual coordinate is an integer. For example $(3, 5)$ is a valid integral pair of coordinates in two dimensions, while $(3.5, 5.8)$ is not. 3

orthogonal polygon A cycle-based graph of axis-parallel sides and integer-coordinate corners that form either convex or concave right angles. 3

partition A division of a set into smaller sets such that the union of the sets cover the entirety of the original set, and the sets have no elements in common. 5

segment A straight part of the border of a polygon. 3

sequence A set of adjacent segments. 3

strip A set of connected cells in the same row or column. 1

1

Introduction

It is often the case, in both industrial and naturally occurring scenarios, that a surface has to be efficiently traversed, such as the painting of a car door, polishing of parts used in manufacturing or even the mowing of grass. More efficient traversals can often equate to faster treatment, or fewer resources being required. Efficient paths can for example be created by minimizing turn costs, which is the additional time the tool takes to change direction. An example of this is the case of NC-machining, where the cost of a tool traveling in a straight line is negligible, but the cost of turning is significant. Minimizing the number of turns therefore has direct practical value, and motivates a deeper algorithmic study of how regions can be partitioned into straight segments. Similar considerations can also be observed in autonomous vehicle routing, robotic vacuum cleaners, agricultural field coverage, and even circuit board manufacturing [1].

One natural approach to minimizing turns is to decompose the region into a minimum number of non-intersecting straight strips, since any path must change direction when transitioning between strips. The minimum number of strips thus provides a lower bound on the number of turns required, and in practice tends to yield more efficient tours. The intuition behind this is that a path which traverses a long strip follows a straight line with no turns, while a fragmented path forces constant reorientation. Keeping reorientation to a minimum is therefore a reasonable goal, which translates into higher efficiency in the real world. In this paper, the focus is on this decomposition step of partitioning an orthogonal integral polygon into a minimum number of non-intersecting strips, of which a few methods are presented. The subsequent construction of a minimum-turn traversal path from such a partition is left outside the scope of this research.

Variants of this problem have been discussed before; a few of them are elaborated in this paper and used as inspiration for algorithm development. The most crucial of these is a version where the same single-width strips are *allowed* to intersect, which has been solved by Arkin et al. [1]. In their paper, it is proven that a minimum strip cover on a polygonal region with n edges and N pixels can be computed in $O(N^{2.376})$ or $O(n^{2.5} \log N)$ time, resulting in polynomial time-bound. It turns out that a relaxation allowing intersection quite drastically simplifies solutions, as there is no need to take the relative positions of the strips into account as much as when it has to be guaranteed that they will not share a cell. Regardless, these solutions can be used to construct lower bound solutions to our problem as will be explored

later.

Suk et al. [2] instead show how a polygonal region can be divided into non-overlapping regions. The distinction from our setting is that their regions may have different widths, whereas strips always have width one. The vertices determining their regions are not required to lie on integer coordinates. By dropping the integrality requirement, their framework can leverage continuous geometric techniques that are unavailable in our context, making a straightforward adaptation of their approach non-trivial.

A similar problem closer to the one in this paper is that of finding the minimum number of rectangles, for which Eppstein [3] has devised an algorithm. This can be done in polynomial time, and despite rectangles possibly being wider than strips, they can still be used to determine strip placement and facilitate analysis in approximation algorithms.

1.1 Aim and Problem Definition

The aim of this project is to develop an approximation algorithm which can partition any given orthogonal polygon into a near-minimum set of straight single-width strips. This means that the strips must cover the entire polygon, with the constraint that the strips are not permitted to intersect (i.e. no part of the polygon is allowed to belong to multiple strips). It would be desirable for the algorithm to have a constant approximation ratio, but it is not a requirement. For this project the intersection limit has been treated as absolute to limit the scope, but algorithms that instead try to minimize intersections may be interesting to investigate in future projects. Practical scenarios often motivate this kind of strict non-overlap, since physical tools occupying the same cell simultaneously can lead to collisions, redundant effort, or interference.

It is not determined here whether the problem is NP-complete or not, but because of there being indications of that being the case, an approximation algorithm as opposed to an exact algorithm is presented here. Furthermore, the approximation ratio and time complexity of the developed algorithm are also analyzed, with the hope that they can serve as a foundation for future exact or improved approximate solutions. This is particularly relevant as interest in automated coverage planning continues to grow across both academic and industrial domains.

2

Preliminaries and Related Literature

2.1 Orthogonal Polygons

In this paper, an orthogonal polygon is defined as a polygon whose corners strictly have integer coordinates. In other words, all sides are parallel to either the horizontal or vertical axis, and all corners exist at integral points in the plane. We describe this structure using a *polygon graph*, $P = (V, E)$. In this graph, the vertex set V represents the corners of the polygon, and the edge set E represents the sides connecting them.

The graph P must be a union of closed cycles; these cycles form the outer contour of the polygon as well as any internal holes (subgraphs fully enclosed by the contour). An example of a polygon graph is shown in Figure 2.1.

Additionally, the corners in a polygon can be either *concave* or *convex corners*. If the 90° angle points inwards to the polygon, it is a concave corner, otherwise it is a convex corner.

Within the context of this thesis, a segment refers to a straight part of the border, and a sequence is a set of adjacent segments.

2.2 Grid Graphs

A grid graph can be defined in a number of ways [4]. For the purposes of the definitions and proofs in this thesis, all points are defined to have integral coordinates shifted by 0.5 in both axes. In some contexts, a grid-graph is a rectangle with dimensions $m \times n$ such that it contains $m \cdot n$ vertices. However, as this paper aims to find strip covers of orthogonal polygons, the first definition of offset integral coordinates is used. As the eventual purpose of the strip cover is to find a continuous tour, there is an added constraint that the graph needs to be connected.

Orthogonal polygons can be represented with grid graphs by creating a point at the center point of each cell enclosed within the polygon. As these points are exactly halfway between integral coordinates on both axes, their coordinates will be of the form $(k + 0.5, s + 0.5)$ for $k, s \in \mathbb{Z}$. Within the graph, a vertex is then created for

every such point, and for every pair of points with a distance of one, an edge is drawn between their corresponding vertices, as described above.

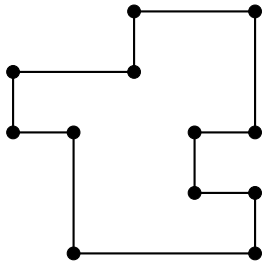

Figure 2.1: Graph P for a polygon. The edges and vertices align with the sides and corners of the polygon.

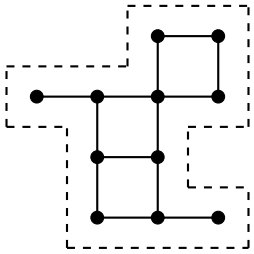

Figure 2.2: The polygon represented as a grid graph instead. Each cell is represented by a vertex and adjacent cells have an edge between their corresponding vertices.

2.3 Minimum-Turn Covering Tours

A covering tour for a given graph $G = (V, E)$ is defined as a sequence of edges in E such that each vertex in V is visited at least once, and that the first and last edge have at least one endpoint in common. A minimum-turn covering tour is a covering tour that has a minimum number of turns.

Further, in some variants there is the added restriction that each vertex needs to be visited exactly once instead. This can be done by, for example, creating extra shortcut edges to circumvent already visited vertices.

2.4 Formal Problem Description

Given an orthogonal polygon, it can be converted to a corresponding grid graph $G = (V, E)$ in the following way:

Define a *cell* to be a unit square whose corners have integral coordinates in the plane. Then, for every such cell that is enclosed within the figure, create a vertex in the grid graph. For every pair of cells that are immediate neighbors on the horizontal or vertical axis, we can consider them connected and create an edge between their corresponding vertices.

For this grid-graph, we can then use the following definitions:

A *strip* s is defined as a set of vertices $\{v_1, v_2, \ldots, v_k\} \subseteq V, s \neq \emptyset$ that are all on the same axis and connected.

There are two kinds of strips, *horizontal* and *vertical*. In a horizontal strip, all vertices share the same vertical coordinate, and vice versa.

Let $\mathcal{S}$ denote the set of all possible strips in V.

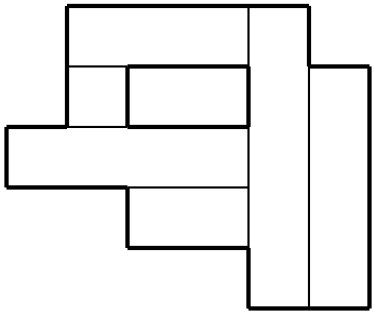

Figure 2.3: An example of a partition of a grid graph into strips.

A *partition* of the grid is a subset of strips $\mathcal{P} \subseteq \mathcal{S}$ that adheres to the following properties:

1. **Completeness:** Every vertex is covered, $\bigcup_{S \in \mathcal{P}} S = V$.
2. **Non-intersecting:** No two strips intersect, $S_i \cap S_j = \emptyset$ for $S_i, S_j \in \mathcal{P}, i \neq j$.

The primary objective of this research is to find an optimal partition $\mathcal{P}^*$ that minimizes the total number of strips:

$$\mathcal{P}^* = \arg\min_{\mathcal{P}} |\mathcal{P}|$$

An example of such a partition is shown in Figure 2.3.

2.5 Approximation Algorithms

When developing an algorithm for an NP-complete problem, it can be desirable to aim for an approximation of a solution instead of an exact solution, as the latter needs more than polynomial time (assuming P $\neq$ NP) to be computed unlike the former. The algorithm can use any methods to give a valid solution to the problem, but for the algorithm to be useful in practical scenarios there needs to be some guarantee as to how far the solution will be from the optimal solution. Let S^A be the number of strips in the solution created by the algorithm and S^* the number of strips in the optimal solution for a given problem instance.

Such a guarantee would therefore need to find a way to bound $\frac{S^A}{S^*}$ for all possible problem instances.

One way of doing this is by finding an upper bound u on S^A, and a lower bound l on S^*. Then the bound is given by $\frac{u}{l}$ since $S^A \leq u$ and $S^* \geq l$ so that $\frac{S^A}{S^*} \leq \frac{u_A}{l}$. This bound can also be called an approximation ratio. The upper bound u_A can be found for example by finding the worst case scenario of the algorithm by utilising its weaknesses, and l can usually be found by determining how properties of the problem limit the effectiveness of even an optimal solution.

2.6 Relevant Literature

There is vast literature covering ways to efficiently travel across a grid graph, but significantly fewer on the subject of partitioning said graphs. This section is an overview of the literature in both the former and the latter categories.

2.6.1 Minimum Strip Covers and Tours with Overlaps

A problem very similar to the one in this thesis is that of partitioning an orthogonal polygon into strips, but allowing them to overlap. This was investigated by Arkin et al. in 2005 [1]. They present multiple variants of graphs and polygons with various restrictions that serve as input to the problem, including the orthogonal integral polygon used for the problem in this thesis. For all these variants, they conclude the minimum turn covering tour problem to be NP-complete. Furthermore, they discuss both an exact polynomial-time algorithm for the strip covering problem and approximation algorithms for the minimum turn covering tour problem.

The problem variants presented are the following:

1. *Discrete Milling:* The problem input is a graph with a maximum degree d. At each vertex, a given set of pairs of the incident can be *collinear*, which means that the turn cost between them for a tour that traverses the two edges directly after each other is 0. The cost for traversing the same edge twice in succession, which is the same as doing a U-turn, is 2. The turn cost between all other possible pairs of edges not included in the above is 1. The problem aim is finding a tour that covers all vertices with a minimum turn cost.

2. *Nonintegral Orthogonal Milling:* In the orthogonal case, $d = 4$ and each segment of the tour has to be axis-parallel. Two edges that are incident to the same vertex and are parallel to the same axis are considered collinear. Two edges that instead meet and are aligned to different axes have a degree of 90° between them and a turn cost of 1.

3. *Integral Orthogonal Milling:* The integral orthogonal milling differs from the non-integral case, as all sides of the polygon align to an integral grid. As described in section 2.2, the polygon can also be represented as a grid graph.

4. *Thin Discrete Milling:* This is similar to *Discrete Milling*, except instead a graph were each vertex needs to be visited at least once, each edge needs to be visited at least once, and at every vertex there is a turn cost as described above. This represents a network of channels connected by junctions.

5. *Thin Orthogonal Milling:* Like the *Thin Discrete Milling* problem described above, except the figure can be represented as an integral orthogonal polygon where the *channels* have a width of one and the junctions are cells in the polygon. In other words, it is the same as the *Integral Orthogonal* case with the exceptions that no areas of 2×2 or larger are permitted to be contained within the polygon.

NP-completeness of all mentioned variants follows from proving the NP-completeness of Thin Orthogonal Milling, from which Discrete Milling, Orthogonal Milling, and Integral Orthogonal Milling can be reduced.

Proving the NP-completeness of Thin Orthogonal Milling is done by using the result that it is NP-hard to decide whether a grid-graph has a Hamiltonian cycle, and proving that any grid-graph can be represented as a Unit Segment Intersection graph [1], which makes the same problem in the latter type of graph also NP-hard. Then, it is shown that this problem can be reduced to the Minimum-turn Covering Tour problem.

Their finding most relevant to this thesis is the algorithm that finds a *minimum cover* of a grid-based region P, such that the strips are allowed to overlap. In their model, a *strip* is defined as a maximal straight segment whose Minkowski sum with the tool is contained within P. To solve this problem, Arkin et al. introduces the concept of a *rook*. In general, when a rook is placed on a cell, no other rooks can be placed in any other cell reachable via a collinear path within the region. In this case, the cells where no other rooks can be placed are the ones accessible via a straight, axis-parallel path within the region; the cells that would be reachable in one move by a chess rook if the polygon was a chess board.

Further, a maximum rook placement is the largest possible set of rooks where no two rooks can attack one another. This maximum placement is then related to the minimum strip cover in the following way: Each rook in a maximum placement requires at least one unique strip (either horizontal or vertical) to cover it. Because no two rooks share a line of sight, one strip cannot cover two rooks. Thus, the size of a maximum rook placement is a fundamental lower bound on the number of strips. Arkin et al. models this as a matching problem in a bipartite graph $G = (V_1, V_2, E)$, where V_1 represents all potential vertical strips and V_2 represents all horizontal strips. An edge exists if a vertical and horizontal strip intersect at a common pixel. In this bipartite version, the size of the maximum matching (rook placement) is equal to the size of the minimum vertex cover (strip cover).

Finally, the following approximation algorithms are presented for the different problem variants described above:

1. *Discrete Milling:*

   The approximation algorithm presented is based of the concept of a *star cover*. The construction works by greedily placing rooks in cells until none more can be placed without attacking another. Each rook then generates strips in every possible direction, which creates the appearance of a star. The number of strips created per rook will at most be ρ, where ρ is the maximum number of collinear pairs of edges connected to a vertex. In this case this number is 2: the vertical and the horizontal directions. Therefore this produces a ρ-approximation of a minimum strip cover. Finally tours are created for each strip, and all tours merge. The algorithm takes $O(n)$ time where n is the number of vertices in the graph.

2. *Nonintegral Orthogonal Milling:*

   For a region with n edges and N pixels, there is a 4.5-approximation for a minimum turn cycle cover, and a 6.25-approximation for a minimum-turn tour which runs in $O(n^{2.5} \log N + n^3)$ time. This is done by first creating a 2.5−approximate cycle cover, or 3.75−approximate tour, which might still leave some parts of the boundary uncovered. To cover this missed boundary, at worst 2 more turns will be needed. For a cycle cover, an addition of the remaining boundaries will be necessary, which will at worst be $\frac{1}{2}$OPT. This results in the claimed $(2.5 + 2) = 4.5$-approximation cycle cover and $(3.75 + 2.5) = 6.25$-approximation for minimum-turn tours.

3. *Integral Orthogonal Milling:*

   For this variant two algorithms are presented: one that is faster, and one that is slower but results in a better approximation ratio.

   The faster algorithm subdivides the region using vertical and horizontal chords through its n vertices, creating a grid of at most n^2 cells C_{ij}, each a rectangle of size $x_i \times y_j$. For each cell, $r_{ij} = \min\{x_i, y_j\}$ rooks are placed along its diagonal and combined into a single *fat* rook. Placing a fat rook uses up r_{ij} units of both the vertical strip X_i and horizontal strip Y_j, reducing their remaining widths by r_{ij}. Since r_{ij} equals the smaller of the two widths, at least one strip is fully consumed and removed at each step, so the process terminates in at most $2n-1$ steps, giving a greedy rook cover in $O(n \log n)$ time. Each fat rook then generates a set of cycles covering all pixels it *attacks*, which are merged together and finally combined into one tour, giving an approximation ratio of 12.

   The slower algorithm instead computes a tour with an approximation ratio of 3.75 in $O(n^{2.5} \log N)$ time. First, the algorithm finds a minimum strip cover which gives a lower bound on an optimal solution. It then finds the cheapest method to connect the strip endpoints together, which costs at most half of the optimal value since any valid tour must connect them somehow. Combining the strips and matching edges forms a cycle cover costing at most 2.5 times the optimal value. Finally, all cycles are merged into a single tour, and since each merge costs at most 2 turns and each cycle has at least 4, this inflates the cost even more, giving a final approximation ratio of 3.75.

4. *Thin Discrete Milling:*

   Similar to Integral orthogonal milling, two algorithms are presented. The fast algorithm ($O(n \log n)$) time, 6-approximation for tours) doubles each strip into a cycle with 4 turns via two U-turns, forming a cycle cover within a factor of 4 of optimal, and then merges cycles at a cost of at most 2 turns each to produce a tour within a factor of 6.

   The slower algorithm ($O(n^3)$ time, 3.5-approximation for tours) instead computes a minimum-weight matching on strip endpoints, yielding a cycle cover costing at most $1.5 \cdot \text{OPT}$, and then merges the resulting cycles into a tour at

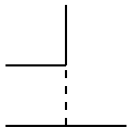

Figure 2.4: A "bad vertex". Notice how it gives rise to three rectangle corners, one at the vertex itself and two where it meets a good diagonal or the side of the figure.

a cost of at most $2 \cdot \mathrm{OPT}$ additional turns.

5. *Thin Orthogonal Milling:*

   Three cases are considered.

   For the basic case, an optimal boundary cycle cover is computed in $O(n^3)$ time, which automatically covers all pixels by the structure of thin polygons, and merging gives a 1.5-approximation tour.

   For the special Eulerian case (all pixels are even degree), the boundary cycle cover is already optimal with no unmatched vertices, and a tighter lower bound of $\mathrm{OPT} \geq t + c$ (where t is the cycle cover cost and c the number of cycles) is combined with $t \geq 4c$ yields a $\frac{6}{5}$-approximation in $O(n \log n)$ time.

   For the general thin case, odd-degree pixels require some edges to be retraced; the algorithm finds an optimal cycle cover, merges any cycles sharing collinear edges for free, and uses a minimum matching on remaining odd-degree pixels to connect components, paying at most 2 turns per remaining merge. A careful analysis of the number of cycles, path turns, and matching costs proves this achieves a $\frac{4}{3}$-approximation in $O(n^3)$ time.

2.6.2 Minimizing the Number of Rectangles

Another problem similar to the one stated here is that of partitioning an orthogonal polygon into a minimum number of rectangles, to which Eppstein [3] states a solution in a survey of graph-theoretic solutions to computational geometry problems. Since integral orthogonal polygons are a special case of orthogonal polygons the results still apply to the former, the focus of this thesis. The proof makes use of some definitions: *good diagonals* and *bad vertices*. A *good diagonal* is an axis-parallel line that connects two concave vertices. For the solution, the largest set G of good diagonals needs to be selected such that none of them intersect. A *bad vertex* is a concave vertex such that it is not an endpoint of a good diagonal. From these parameters, the number of necessary rectangle corners in the figure is computed, and in turn the minimum number of rectangles by dividing this number of corners by four.

Every vertex of the polygon creates one rectangle corner at minimum. Additionally, a line needs to be extended from every bad vertex, as a rectangle cannot contain concave corners. Where this line meets either the edge of the figure or a good diagonal, two additional corners are created, see Figure 2.4. The number of necessary rectangle corners is therefore the number of vertices in the figure added to twice the

number of bad vertices. The number of bad vertices b can be written in the following way:

$$b = \frac{n}{2} + 2h - 2 - g$$

which together with n makes the number of rectangle corners $2n + 4h - 4 - 4g$, and finally the minimum number of rectangles $\frac{n}{2} + h - 1 - g$.

2.6.3 Optimization Problems Related to Zigzag Pocket Machining

Arkin, Held and Smith [5] investigated algorithms for minimizing the number of tool retractions in pocket-machining paths. A *tool retraction* is simply any time a tool has to be removed from the surface in order to relocate it. This is often costly in industrial use and should be minimized when possible.

In the setting of this thesis, a pocket can be viewed as a non-integral, non-orthogonal polygon. This means that its vertices do not need to have integer coordinates, and its edges do not need to be axis-aligned.

A tool path can fill such a polygon in two main ways. *Contour-parallel* machining where the tool follows the boundary inward in a spiral-like fashion, and *direction-parallel* machining where the tool sweeps along a set of parallel lines. The authors focus on the *zigzag* variant of the latter, in which successive passes alternate direction, sweeping back and forth across the pocket.

The problem is reframed in graph-theoretic terms. Each pocket becomes a *machining graph* with three kinds of edges. *Compulsory edges* (c-edges) represent the parts which must be traversed exactly once. *Non-compulsory edges* (nc-edges) represent travel along the pocket boundary and may be traversed at most once, but can also be avoided completely. And finally *Retraction edges* represent lifting and restarting the tool. This is what they aim to minimize. The goal is therefore to traverse the graph so that every c-edge is covered once, no nc-edge is reused, and as few retractions as possible are made.

There are then two different cases of these pockets that must be solved in different ways:

For a pocket without holes, the machining graph is broken into smaller pieces, each of which can be traversed with the minimum possible number of retractions in linear time. Stitching the k small traversals back together adds at most $2(k-1)$ further retractions, resulting in at most $5 \cdot \mathrm{OPT}$ retractions, where OPT is the (unknown) minimum achievable.

Pockets with holes are solved by utilizing the solution to the hole-free case. At the worst-case, each hole can cause cost at most $6h$ extra retractions, so a pocket with h holes can be machined with at most $5 \cdot \mathrm{OPT} + 6h$ retractions overall.

Computing a truly optimal path is, however, intractable in general: the authors prove that the problem is NP-hard for pockets with holes via a reduction from Planar 3-SAT. Whether it remains NP-hard for pockets without holes is left open.

Finally, these bounds are worst-case, and shown to be better in practice. Experiments on 15 sample pockets without holes suggested a typical approximation factor closer to $1.5 \cdot \mathrm{OPT}$.

3

Problem Analysis

Any optimal strip partition must follow certain geometric constraints as well as adhere to certain lower bounds. By identifying behaviors that must exist in an optimal solution, we can conversely define traits that can *not* exist in an optimal one, limiting the space of possible solutions. Additionally, reasoning about lower bounds for optimal solutions provides a tool with which to analyze algorithms, as shown in Section 4.

3.1 Switchlines

The primary tool for analyzing possible solutions is the *switchline*. A switchline is defined as the boundary where regions of horizontal and vertical strips meet. By analyzing the shape of a given switchline, it is possible to make claims regarding a figure's optimality. The process of creating switchlines can be done in linear time, by visiting every cell in the image and comparing neighboring alignments. Figure 3.1 demonstrates a straight switchline separating two orientations.

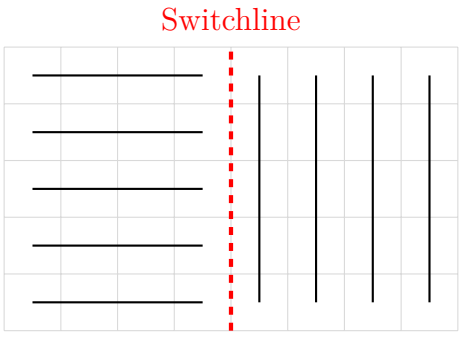

Figure 3.1: A vertical switchline acting as a boundary between regions of horizontal and vertical strips.

The problem of strip decomposition is initially framed as the minimization of the total number S of disjoint strips required to partition a polygon P. However, we can rephrase this objective by shifting our focus from the interior of the strips to their *terminations*. Every strip in an orthogonal decomposition is a $1 \times L$ rectangle, possessing exactly two ends of unit width. These ends occur in only two locations: when a strip meets the contour of the figure, and at a switchline.

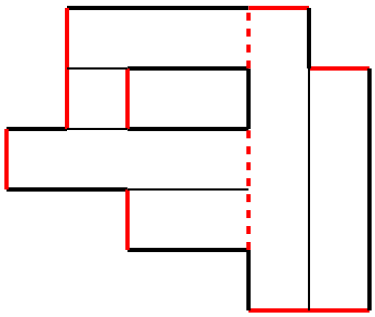

Figure 3.2: A partition where strip terminations are marked in red. Switchlines are dashed lines and the borders are solid lines.

Therefore the total number of strip terminations is equal to the total length of the switchlines and the total length of the portions of the border that serve as strip terminations, or border terminations as they also can be referred to. Since the number of strips is equal to half the number of strip terminations, the number of strips is also equal to half the sum mentioned previously. Minimizing the number of strips is therefore equivalent to minimizing these two quantities: the switchline lengths and the sum of the border terminations. Figure 3.2 shows a partition where the strip terminations are marked as red lines.

3.1.1 Equivalence and Implications

The rephrased goal of lengths of switchlines and strip terminations on the border does not change the optimal solution, but it changes the search for an optimal solution. Instead of trying to fill the volume, an optimal algorithm now creates the minimum possible total length of all switchlines. This perspective is particularly useful for irregular orthogonal shapes because it highlights the cost of a switchline. A long switchline is expensive because it cuts through many potential strips, forcing them to terminate and increasing the total count $|S|$. Conversely, a switchline that connects two nearby concave vertices is cheap because its length is minimal, effectively neutralizing two strip-generating points with the smallest possible impact on the total length. This is explored further in the following sections.

3.2 Restrictions on Optimal Solutions

As established, the total number of strips $|S|$ is directly proportional to the total length of all strip ends. Therefore, an optimal partition must minimize the combined length of switchlines and strip terminations. Using this minimization goal, we can identify specific switchline configurations that cannot be present in an optimal solution.

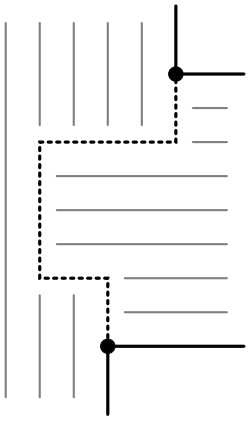

Figure 3.3: A non-monotone switchline.

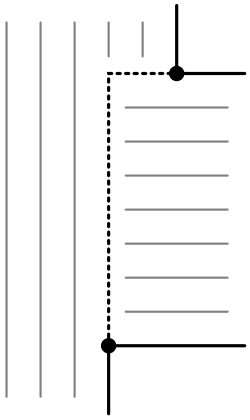

Figure 3.4: A monotone switchline with the same endpoints as the switchline in Figure 3.3.

3.2.1 Monotonicity

As long as the endpoints of all switchlines are constant, minimizing the length of the switchlines will minimize the number of strips. The shortest possible path between two switchline endpoints that adheres to the grid is the Manhattan distance between them, which is equal to the difference in vertical position added to the difference in horizontal position. This distance is achieved iff the switchline is any monotone sequence between the two endpoints, which we define here as a sequence such that no axis-parallel line can cross the switchline twice. Therefore, in an optimal solution, we can assume that all switchlines are monotone. It follows that as long as the switchline adheres to the above condition of monotonicity, it can take on any form without affecting the number of strips in the solution.

3.2.2 Switchline Simplification

As long as all switchlines are monotone and their endpoints unchanged, they can be modified without affecting the number of strips. This means that all switchlines can be simplified, or *folded*, which in turn simplifies the search for an optimal solution. Figure 3.5 shows a switchline with many small segments that in Figure 3.6 has been simplified to a switchline with just two segments. While this does not affect the number of strips, it narrows the possible solution space when searching for optimal switchlines: instead of looking for all possible paths of Manhattan length we have limited it to only two. This means that all switchlines can be regarded as having two components: one horizontal and one vertical. However, in some cases, one of these components overlaps with the border, and in that case the switchline will appear to be straight instead of L-shaped.

The switchline in Figure 3.4 is an example of a switchline that could be simplified in this way to appear straight. We will refer to those switchlines as *straight switchlines* and those which cannot be simplified in this way as *L-shaped switchlines*, despite that straight switchlines can be folded to have an L-shape.

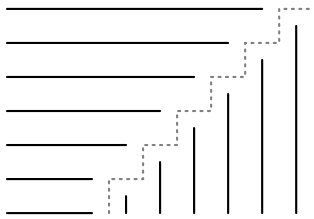

Figure 3.5: A switchline separating two regions in a staircase pattern.

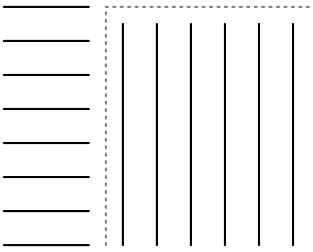

Figure 3.6: The staircase switchline in Figure 3.5 folded into one large step.

Note that the switchline cannot go outside the bounds of the figure.

3.2.3 Parallel Switchlines

Theorem 1. *Let ℓ_1 and ℓ_2 be two parallel switchlines, each of length L. We can then observe region R of width W (the perpendicular distance between them) as the space between the top of ℓ_1 and bottom of ℓ_2. If $W < L$, the partition can be strictly improved by reorienting all strips within R.*

Proof. Let $\mathcal{O}_{\parallel}$ denote the strip orientation parallel to the switchlines ℓ_1 and ℓ_2, and let $\mathcal{O}_{\perp}$ denote the orientation perpendicular to them. Figure 3.7 displays an example of this where $\mathcal{O}_{\parallel}$ is vertical and $\mathcal{O}_{\perp}$ horizontal. By the definition of a switchline, the orientation must change across the boundary. Therefore, region R placed between the switchlines, and the exterior regions must have different orientations.

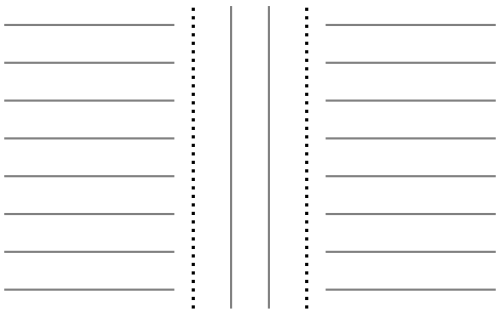

Figure 3.7: Two parallel switchlines (dashed lines), creating three distinctly oriented regions.

Define the width of R as W and the height L. We can denote the external regions as $\mathcal{O}_{ext}$ and the interior region R follows $\mathcal{O}_{int}$, where $\{\mathcal{O}_{ext}, \mathcal{O}_{int}\} = \{\mathcal{O}_{\parallel}, \mathcal{O}_{\perp}\}$.

The number of strips can be rephrased in terms of the length of the switchlines as well as the number of border segments used as strip terminations. The two switchlines contribute $2L$ to the first term. The contribution to the second term depends on the orientation of the strips in R: If $\mathcal{O}_{int} = \mathcal{O}_{\perp}$, the strips in R terminate on the switchlines. These ends are already accounted for by the switchline length, contributing 0 to the sum of border terminations. If $\mathcal{O}_{int} = \mathcal{O}_{\parallel}$, all strips in R terminate on the polygon boundary, contributing $2W$ to the sum of border terminations.

Let C_{ext} be the constant contribution of the exterior strips to the boundary ends. The total count is:

$$2|S| = \begin{cases} 2L + C_{ext} & \text{if } \mathcal{O}_{int} = \mathcal{O}_{\perp} \\ 2L + 2W + C_{ext} & \text{if } \mathcal{O}_{int} = \mathcal{O}_{\parallel} \end{cases}$$

We reorient the strips in R to match the exterior orientation ($\mathcal{O}_{int} \to \mathcal{O}_{ext}$). This eliminates the switchlines ℓ_1 and ℓ_2, as there is no longer a transition between orientations.

In the new partition the switchline contribution becomes 0. If the new uniform orientation is $\mathcal{O}_{\parallel}$, the W strips now terminate on the polygon boundary, contributing $2W$. If the new uniform orientation is $\mathcal{O}_{\perp}$, the strips now pass through the space where the switchlines once were and terminate on the outer polygon boundary. The change in the end-count is effectively $-2L$ relative to the switchline cost.

In either configuration, the new count $2|S|'$ satisfies:

$$2|S|' = \begin{cases} 2W + C_{ext} & \text{if reoriented to } \mathcal{O}_{\parallel} \\ C_{ext} & \text{if reoriented to } \mathcal{O}_{\perp} \end{cases}$$

Comparing the counts, the net change $\Delta(2|S|)$ is:

- $(2W + C_{ext}) - (2L + C_{ext}) = 2(W - L)$ in the first case.
- $C_{ext} - (2L + 2W + C_{ext}) = -2L - 2W$ in the second case.

In both cases, since $W < L$, the change is strictly negative. Specifically, for the configuration where R was the only region with orientation $\mathcal{O}_{\perp}$, the improvement is exactly $W - L$ strips.

□

3.2.4 Possible Switchline Positions and Maximum Number of Switchlines

Something that can facilitate finding and reasoning about solutions is limiting what such solutions could look like. In this section, we find a way to limit the maximum number switchlines an optimal solution can contain, and what their form is with the following theorems. This can in turn be used in algorithm analysis in the following ways.

Lemma 1. *Every valid solution to a given figure can be modified such that all straight switchlines have at least one endpoint in a concave corner of the figure, without increasing the number of strips.*

Proof. For any given figure, assume that there is a straight switchline such that none of its endpoints coincide with a concave corner.

The switchline can then be moved along the borders without increasing the number of strips if it is moved in the direction such that the strips perpendicular to the switchline are lengthened. An example of this is shown in Figure 3.8. This holds until a concave or convex corner is encountered at one of the endpoints of the switchline, since expanding the perpendicular strips further might break up strips that are parallel to and span longer than the switchline, as is the case in Figure 3.8. The corner closest to the switchline might be a part of a shape that splits the funnel into two which is shown in Figure 3.9. In that case the switchline splits into two switchlines, each one with a concave corner, which completes the proof for this scenario.

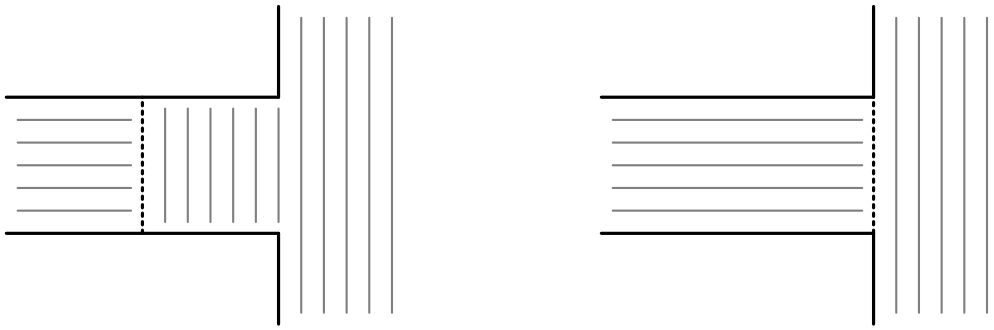

Figure 3.8: An example of moving a switchline in a way which decreases the number of strips. If it were to be moved past the end of the funnel, the vertical strips would get broken up, creating new strips

Figure 3.9: A switchline that can be moved to the right without increasing the number of strips. For the purpose of the proofs in this section it is enough that each straight switchline can be modified to have at least one endpoint on a concave corner, and therefore there is no point in moving the two newly created switchlines further to the right.

When that is not the case, one of the straight borders must eventually meet a convex or concave corner. If this corner is concave, like in Figure 3.8, nothing more needs to be done.

Otherwise, if it encounters a convex corner, there are three scenarios of which only two are feasible, shown in Figure 3.10. The first is that the border continues forwards until it meets the other endpoint of the switchline, which means that the shape comes to a dead end and the switchline disappears as it completely overlaps with the border. The second scenario is that the border turns outwards after going forwards in which case a concave corner is formed which serves as an endpoint. The final scenario would be that the border turns inwards which is not possible since it

would eventually result in a convex or concave corner which the switchline would have encountered first in that case, and if it goes past the starting position of the switchline, the switchline would not have been possible to begin with.

Figure 3.10: The two possible scenarios where one of the straight segments ends in a convex corner. In the first scenario, the switchline will overlap completely with the border and therefore disappear.

Therefore we have shown that in a valid solution every straight switchline can be moved so that at least one of the endpoints coincide with a concave corner in the figure without increasing the number of strips, completing the proof. □

Lemma 2. *Switchlines that consist of a visible L-shape can be modified to have both their endpoints at concave corners of the figure without increasing the number of strips.*

Proof. Assume we have an L-shaped switchline that has either endpoint on a concave corner, and the other on a straight segment of the border, or both on straight segments. Only straight switchlines can have one of their endpoints on a convex corner, as when the switchline is simplified into two segments, one of them will be hidden by the border thus making the switchline appear straight.

The endpoint(s) that are on a straight segment can then be moved in a straight line along the borders such that the Manhattan distance between the endpoints decreases without increasing the number of strips. This is only possible in one direction for each endpoint. In the other the Manhattan distance will increase instead. In the example in Figure 3.11, this direction is up for endpoint A and left for endpoint B. The endpoint being moved in parallel to the direction of the "exterior" strips of the L will lengthen the exterior strips, and remove one strip on the interior per unit length it moves. The other will instead lengthen a subset of the exterior strips to span the length of the side of the L.

As before, each endpoint can continue to be moved until a convex or concave corner is encountered somewhere on that endpoint's half of the switchline, and once again, if the corner is concave, that endpoint can stop there. For the switchline to be L-shaped, this corner has to be encountered before the switchline is on the same axis-parallel line as the other endpoint, since it would otherwise get straightened out and therefore not be a true L-shaped switchline.

A concave corner here directly satisfies the lemma. If the corner is convex, it eventually has to come to a concave corner somewhere within the rectangle that is formed between the original positions of the endpoints. Otherwise the original L-shape would have been obstructed and therefore not possible. In that case, this concave corner becomes the new endpoint of the switchline, and in the case where this would

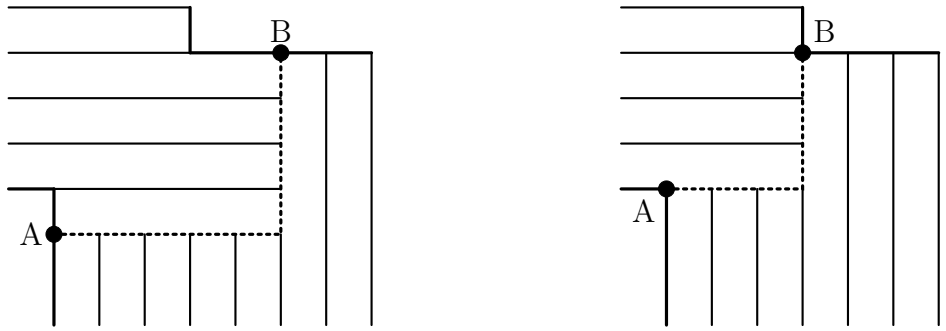

Figure 3.11: An L-shaped switchline, moved so that both endpoints are on concave corners of the figure. Note that the total number of strips has decreased after the move.

create a gap between the part of the border the new endpoint is on and the straight segment of the border the switchline previously attached to, a new straight switchline is created directly in line with the old switchline.

Another way of viewing this, is the switchline splitting in two, the reason being that no new strips will be created by preserving the strips on the other sides of the switchlines and not breaking them up. An example is shown in Figure 3.12.

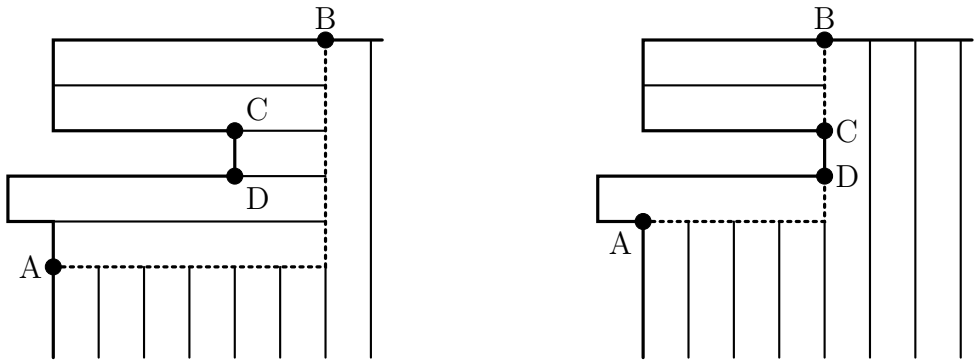

Figure 3.12: An L-shaped switchline that is moved until concave corners C and D are encountered, splitting the switchline in two without increasing the total number of strips.

In summary, we have shown that in all theses cases the L-shaped switchline can be modified such that both its endpoints are on concave corners without increasing the number of strips, and thus proved the original statement. □

Lemma 3. *Every vertex in the figure can at most be the endpoint of one switchline in an optimal solution.*

Proof. Each vertex in the figure is either a concave or a convex corner, and we will begin with the former case:

Assume there are two switchlines Q and R. They share one endpoint in vertex C, and have their other endpoint in A and B respectively. According to the monotonicity

results in Section 3.2.1 it follows that Q and R can be any monotone sequence between C and A or B, respectively without affecting the solution. For the purposes of this proof, fold the switchlines such that they consist of only one "step" each and that they meet at a right angle in C. Then, the strips "outside" the switchlines on one side can be extended to cover the entirety of the rectangle partly formed by the halves of the switchlines meeting at C.

Thus the switchlines have been merged into a new switchline with endpoints A and B, and C is no longer the endpoint of a switchline. If more than two switchlines have endpoints in C, this procedure can be repeated on pairs of adjacent switchlines until none are left. In this context, adjacent means that if a straight line is drawn between C and every other endpoint, the two lines corresponding to the switchlines will be next to each other.

For the latter case: similar to above, assume there is a convex corner on which two switchlines Q and R each have one endpoint, and that A, B, and C otherwise are the same as above. From Lemmas 2 and 4, it can be inferred that these two switchlines need to be straight. Further, from the definition of endpoints Q and R need to be able to be modified into any monotone sequence between C and A or B without modifying the number of strips. This is only possible if the strips on the side of the switchline that continues in the direction opposite of the other endpoint are perpendicular to the switchline.

Figure 3.13 shows an example of this. Define $\mathcal{O}_C$ to be the orientation of the region between Q and R, $\mathcal{O}_A$ the orientation of the other region Q borders, and likewise with $\mathcal{O}_B$ and R. From the definition of switchlines, $\mathcal{O}_A \neq \mathcal{O}_C$, $\mathcal{O}_C \neq \mathcal{O}_B$, and thus $\mathcal{O}_A = \mathcal{O}_B$. Then, either both Q and R will be parallel or perpendicular.

In the first case, if Q denotes the switchline furthest away from C, $\mathcal{O}_A$ denotes the orientation that needs to be perpendicular to Q, and similarly for $\mathcal{O}_C$ and R. However, $\mathcal{O}_A \neq \mathcal{O}_C$, but Q and R have the same orientation. Therefore the conditions necessary for both Q and R being valid switchlines with endpoints in C are not fulfilled. In the other case, Q and R are perpendicular to each other. We know that $\mathcal{O}_A = \mathcal{O}_B$, but since Q and R have different orientations, once again only one of the switchlines can have an endpoint in C.

This proof also applies for when there are more than two switchlines, if Q and R are the two switchlines closest to C, since we only used the orientations directly bordering the switchlines and did not make assumptions about the existence of other switchlines beyond that point.

Finally, we have shown for both concave and convex corners that two or more switchlines can't have the same corner as endpoint, which completes the proof.

□

Lemma 4. *Each straight switchline has one endpoint on a concave corner, and the other either on a concave or convex corner.*

Proof. We know from Lemma 1 that every straight switchline has at least one con-

Figure 3.13: Two different sections of polygons that are identical except for the orientations of the strips. A switchline beginning in D can have E as its other endpoint in the left figure, but not the right. This is because in the left figure the strips opposite of E are perpendicular to the switchline, while the opposite is true for the right. Because of this, a switchline in the left figure can take on all monotone sequences between D and E while leaving the number of strips unchanged by extending and contracting strips, while some monotone sequences in the right figure can create new strips.

cave corner as an endpoint. If the other endpoint is also on a vertex in the polygon graph, the statement is already satisfied. Otherwise, the visible part of the switchline will end somewhere on a straight segment of the border. Therefore we want to show that this switchline can be modified to be any monotone sequence between its original concave corner, and some concave or convex corner without creating or destroying existing strips.

It has to be true that the straight segment where the switchline ends is capped off by two corners which can be either convex or concave. The strips parallel to the border on one side of the switchline can be fully extended to the end of the straight segment without affecting the number of strips; the perpendicular strips are simply shortened as is shown in Figure 3.14. All other monotone sequences will be somewhere between those two extremes, and by modifying the length of the parallel strips, all of those can be attained. Therefore the switchline can take on any monotonous sequence between those two endpoints, which completes the proof.

□

Theorem 2. *The number of switchlines in an optimal solution can be upper bounded by* $\left\lfloor \frac{v}{2} \right\rfloor$ *where* v *is the number of vertices in the polygon graph.*

Proof. From Lemmas 1, 2 and 4, it follows that every switchline in an optimal solution can be assigned to exactly two corners in the figure, either concave or convex. Since each corner can only be the endpoint of exactly one switchline according to Lemma 3, the number of corners in the figure has to be at least twice the number of switchlines. This relation can be reformulated as $T \leq \frac{v}{2}$ where T is the number of switchlines and v the number of vertices in the polygon graph of the figure. The bound can be further refined by noting that the number of switchlines has to be an integer, and that fewer than 2 vertices are not enough for a switchline, so that $T \leq \left\lfloor \frac{v}{2} \right\rfloor$. □

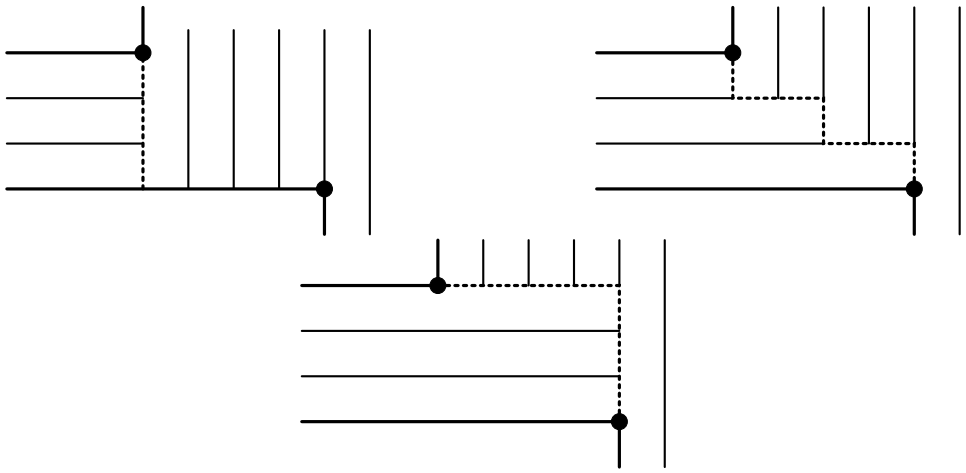

Figure 3.14: Three of the possible monotone sequences as switchlines between the two endpoints (black dots). Notice that the number of strips remains the same in the different versions.

3.3 Lower Bounds on S^*

To compute the approximation ratio for algorithms that solve the given problem, the number of strips placed by the algorithm in the worst case needs to be compared to the minimum possible number of strips needed for any valid solution, including an optimal solution S^*. This section contains several such lower bounds which will be used in the analysis of the algorithm in section 4.

3.3.1 Minimum Number of Rectangles

Each valid strip partition is also a valid rectangle partition according to how the latter is described in [3]: every strip can trivially be converted to a single-width rectangle of the same length as the strip. This means that the optimal strip partition is also a valid rectangle partition, and that the latter has to contain a less than or equal number of rectangles than the former has strips. If that is not the case, the optimal strip partition can simply be converted to a rectangle partition with an equal number of rectangles as there originally were strips. Then, there is a rectangle partition that contains fewer rectangles than the minimum rectangle partition, creating a contradiction. Therefore, for any given figure, the minimum number of possible rectangles is a lower bound of the minimum number of strips in a minimum strip partition.

3.3.2 Using the Number of Convex Corners

A simple way of lower bounding a solution is to make use of the fact that in a strip coverage of a polygon, each convex corner must correspond to the convex corner of one strip. Since each strip has four convex corners, at best, one strip can therefore cover four convex corners of the polygon. This can be expressed as a lower bound

$$\left\lceil \frac{c}{4} \right\rceil$$

where c is the number of convex corners in the figure. The reason the fraction is rounded up is because the number of strips is integral, and so if a fraction of a strip is necessary, an entire strip has to be used.

3.3.3 Using Width, Height, and Number of Cells

Given the width W, height H, and number of cells n in a figure, we can compute the minimum number of strips needed to cover the figure. The width and height here are the total length of the figure in each axis. For example, for complete rectangles, the minimum number of strips needed is the minimum of the width and the height. This section aims to generalize this to figures that are not complete rectangles using these three parameters. Here it is assumed that $W \geq H$ for simplicity of calculations, and when that is not the case, the figure can simply be rotated 90°.

Theorem 3. *For all solutions S to a given figure with width W and height H such that $W \geq H$, $W, H > 2$ and total number of cells n, $S \geq \left\lceil \frac{n-H}{W-1} \right\rceil + 1$.*

Proof. If $W, H \geq 2$, the minimum number of strips needed to cover the figure is 2. The width and the height, together with the constraint that the figure is connected, means that at least one cell is needed per column and per row. If only one strip is used, either only one column or one row is going to contain cell(s), since the strip has a width of one. Further, if $W, H > 2$ and $n \leq W(H-1)+1$, in the case with the smallest possible n there needs to be at least one horizontal and one vertical strip. The first constraint exists because if $W \leq 2$ or $H \leq 2$, the figure could (in the best case) be covered by two vertical or two horizontal strips, respectively. The second follows from that if the figure is close to being filled, similarly all strips could be either horizontal or vertical. One of these two strips needs to be full length, and the other its full length subtracted by one, so that they together span the full width and height of the figure accounting for the intersection between them. The number of cells required for this base case is therefore $W + H - 1$.

Then, for any cells beyond this number, at least one extra strip is needed. To be exact, up to $W-1$ cells can be added while only adding one extra strip. In general, one strip is needed for every $W-1$ extra cells. With this background we can state the following inequality that must be fulfilled for any correct figure with the parameters W, H, n, and extra strips k besides the two first (and where $W, H > 2$ and $n \leq W(H-1)+1$):

$$W + H - 1 + k(W-1) \geq n.$$

This can be rewritten as such:

$$k \geq \left\lceil \frac{n-H}{W-1} \right\rceil - 1.$$

The reason for rounding up is that k has to be an integer, so when a fraction of a strip is needed an entire strip has to be used. Since $S = k + 2$, this means that $\left\lceil \frac{n-H}{W-1} \right\rceil + 1$. □

3.3.4 Star Rooks

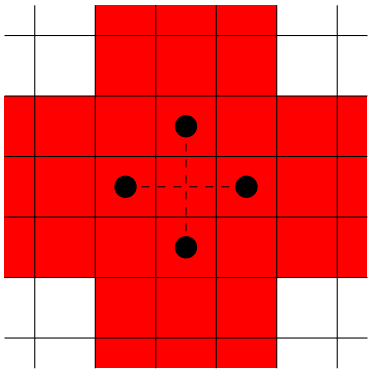

Figure 3.15: An example of a star rook. No other star rooks can be placed in the red cells.

Another way of lower bounding solutions is to use so-called *star rooks*. A star rook is defined as four separate cells, such that two of them are on the same row and the other two are on the same column, and that the cell where the row and column intersect is between these four cells on both axes. Figure 3.15 shows an example of a star rook. Then, at least three strips are needed to cover these four cells. Since all four cells are not on the same line, no fewer than two strips can be used. However, since the lines the cells are on intersect and strips are not allowed to intersect, at least three strips are needed. Furthermore, cells from different stars can not be on the same row or column, because then fewer strips can be used to cover the stars. Consider for example two stars that are immediately next to each other. Then only three strips could be used to cover both stars, which is one and a half strip per star.

Using this, any solution must contain at least $3k$ strips, where k is some number of stars that can be placed. Of course, the more stars that can be placed, the tighter the bound is.

The maximum number of stars can be used by converting the placement problem into a maximum independent set problem. Let each possible star placement be a vertex in a graph $G = (V, E)$, and for every pair of star placements that cannot both be included in a solution, create an edge between their two vertices. Since we want the maximum number of star placements such that they can all exist in the same solution, this is the same as trying to find the maximum number of vertices such that none of them have an edge between them, which in turn is equivalent to a maximum independent set problem.

3.3.5 Variant with Overlaps as a Lower Bound

Finally, the solution stated by Arkin et al. [1] can be utilized to create a lower bound for solutions in this thesis. This follows from every valid solution to the problem where intersections are forbidden, being valid solutions where intersections are allowed. Therefore, the problem solved by Arkin et al. [1] is a more general version of the problem stated in this thesis, and thus any optimal solution to the former problem must be equal to or better than any solution to the latter. Since they find an optimal solution in polynomial time, the number of strips returned by their solution can be used directly as a lower bound.

4

Algorithm

To approximate solutions to the problem stated in Section 2.4 a polynomial-time algorithm has been devised. This section is a presentation of the algorithm itself together with an analysis of its approximation ratio and time complexity.

4.1 Minimum Rectangles Strip Placement Algorithm

Consider the algorithm for finding the amount of minimum rectangles described in [3]. For any given figure, the author states that at least $\frac{v}{2} + h - g + 1$ rectangles are necessary to cover it, where v is the number of vertices in the contour of the figure, h the number of holes, and g the maximum number of pair-wise disjoint *good diagonals*. A good diagonal is an axis-parallel line that connects two concave corners in the figure.

The algorithm has the following steps for any given figure:

1. Find the minimum number of rectangles r in the figure using the algorithm in [3]
2. Divide each rectangle of width w and height h into evenly-sized strips that are all horizontal if $h \leq w$ and all vertical otherwise

Eppstein [3] proves that finding the minimum number of rectangles can be done in $O(v^{2.5})$ time. Further, dividing each of the $O(v)$ rectangles into strips takes $O(vm)$ time where m is the minimum of the overall width and height of the figure, so an upper bound on the number of strips in a rectangle. Therefore the entire algorithm takes $O(v^{2.5} + vm)$ time.

4.1.1 Approximation Ratio

To begin finding the approximation, we know how many cells are in the figure and that the figure can be divided into r rectangles, but not how the cells are distributed among the rectangles. We will need to estimate how many strips are needed to cover each rectangle. We will, however, avoid finding the distribution for the time being, and instead focus on the strips inside each rectangle. A rectangle that consists of n cells and is covered by strips optimally as in the algorithm, between 1 and $\lceil \sqrt{n} \rceil$

strips will be needed, depending on the rectangle's dimensions. Therefore, if we can find the distribution of n among the rectangles in the worst case, we have an upper bound on the number of strips required by the algorithm. To simplify the calculations, we can use $\sqrt{n}+1$ as the upper bound instead of $\lceil\sqrt{n}\rceil$.

Lemma 5. *The highest possible upper bound on the number of strips in the figure, given r, n, is achieved when all rectangles in the figure are of equal size, or as close in size as possible while the solution is still feasible.*

Proof. Assume that we have two rectangles r_1 and r_2 with n_1 and n_2 cells each, such that $n_1 \leq n_2$.

It then follows that

$$u = \sqrt{n_1} + \sqrt{n_2} + 2$$

acts as the upper bound for the sum of their strips u. Then, suppose that we were to "move" some number of cells q from r_2 to r_1. Assume that $q \leq \frac{n_2-n_1}{2}$ for the purposes of this proof. In this scenario it does not matter whether this is feasible or not based on their relative placements as we are only looking for the absolute worst case to create an upper bound. The worst feasible scenario will still be contained within this upper bound. For the same reason we can also disregard the fact that cells are discrete and allow for q to be any real (positive) number. After the move, the new total of the strips will be upper bounded by $u' = \sqrt{n_1+q} + \sqrt{n_2+q} + 2$. These terms can be rewritten as

$$\sqrt{n_1+q} = \sqrt{n_1} + \int_{n_1}^{n_1+q} (\frac{d}{dx}\sqrt{x})dx = \sqrt{n_1} + \int_{n_1}^{n_1+q} \frac{1}{2\sqrt{x}}dx$$

and

$$\sqrt{n_2-q} = \sqrt{n_2} - \int_{n_2-q}^{n_2} (\frac{d}{dx}\sqrt{x})dx = \sqrt{n_2} - \int_{n_2-q}^{n_2} \frac{1}{2\sqrt{x}}dx$$

respectively. Naturally, the integrals could be resolved directly, but by splitting the terms like this, their sum can be compared with u more easily in the following way:

$$\begin{aligned} u' &= \sqrt{n_1} + \int_{n_1}^{n_1+q} \frac{1}{2\sqrt{x}}dx + \sqrt{n_2} - \int_{n_2-q}^{n_2} \frac{1}{2\sqrt{x}}dx + 2 \\ &= u + \frac{1}{2}\left[\int_{n_1}^{n_1+q} \frac{1}{\sqrt{x}}dx - \int_{n_2-q}^{n_2} \frac{1}{\sqrt{x}}dx\right]. \end{aligned}$$

Since $\frac{d}{dx}\frac{1}{\sqrt{x}} = -\frac{1}{4x\sqrt{x}} < 0$ for all $x > 0$, that means that $\frac{1}{\sqrt{x}}$ is strictly decreasing for $x > 0$, and because of the constraint on q, all points in the interval $[n_1, n_1+q]$ are equal to or smaller than all points in the interval $[n_2-q, n_2]$. Therefore,

$$\int_{n_1}^{n_1+q} \frac{1}{\sqrt{x}}dx \geq \int_{n_2-q}^{n_2} \frac{1}{\sqrt{x}}dx$$

since $\frac{1}{\sqrt{x}}$ in all points of $[n_1, n_1+q]$ will be equal to or larger to the same function in all points in $[n_2-q, n_2]$. This means that $u' \geq u$, so that evening out the number of cells between the rectangles increases the upper bound. The maximum upper bound is therefore achieved when all rectangles have an equal number of cells. □

Lower bound	Approx. ratio
r	$\sqrt{\lceil \frac{n}{r} \rceil}$
$\left\lceil \frac{c}{4} \right\rceil$	$\frac{r\sqrt{\lceil \frac{n}{r} \rceil}}{\lceil \frac{c}{4} \rceil}$
$\left\lceil \frac{n-H}{W-1} \right\rceil + 1$	$\frac{r\sqrt{\lceil \frac{n}{r} \rceil}}{\lceil \frac{n-H}{W-1} \rceil + 1}$

Table 4.1: The different approximation ratios for the algorithm resulting from using different lower bounds. Note that $r = \frac{v}{2} + h - g + 1$.

Consequently, the worst case would be when the cells are perfectly evenly distributed. But as cells are discrete, this might not be feasible. Out of all possible cell distributions, one or multiple will result in an upper bound that is the closest to the worst case. However, instead of finding this "perfect" distribution which can be tedious, we can instead settle for the bound achieved by the average case with the cells in each rectangle rounded up, as this provides a looser upper bound. This means that each rectangle has $\lceil \frac{n}{r} \rceil$ cells and therefore at most $r\sqrt{\lceil \frac{n}{r} \rceil}$ strips will be needed to cover the figure using this algorithm.

Since the lower bounds presented in Section 3.3 depend only on properties of the problem instance and not on the algorithm, any of them can be used in this analysis. For the sake of conciseness, all possible approximation ratios using the different bounds are presented in table format in Table 4.1.

5

Discussion

The main findings in this project are the algorithm, the lower bounds and the switchline analysis. This section discusses the value of these findings as well as their limitations, and gives an overview of future work that can be done.

5.1 Algorithm

Beginning with the algorithm, we had hoped to achieve a constant approximation ratio, which unfortunately did not appear possible, at least within the scope of this project. Instead various parametrized approximation ratios were found, depending on the lower bound used. As for the algorithm itself, the found time complexity of $O(v^{2.5} + vm)$ time makes it usable in practice. It should however be observed that an algorithm based on constructing a minimum set of large rectangles may struggle in some cases. It stands to reason that a polygon with a few large solid sections will most likely be easy to solve with a number of strips close to the optimal. In the opposite case, a polygon broken up by many holes and thin regions will result in a higher number of strips.

5.2 Lower Bounds

The simplest lower bound is the one using the number of convex corners (Section 3.3.2) but it will give a very poor lower bound when applied to figures that are large with a simple contour. For example, when applied to a very large rectangular shape, the minimum number of strips will be just one according to this bound, while it may be clear from a glance at the figure that the number of strips needed must be much higher.

Secondly, the lower bound in Section 3.3.3 aims to resolve this weakness by taking in the overall width and height of the figure, together with the number of cells. It does not, however, account for the number of corners in the figure as to not make the bound more complicated. The lower bound works by finding how the cells would be arranged in the best case within the rectangle delimited by the width and height, and so when the cells of a figure actually are configured in that way or close to it, the bound will be the most effective.

Thirdly, the lower bound using the minimum number of rectangles possible in the

polygon is related to the algorithm which uses the same partitioning as a step in the process to divide the figure into strips. With this bound the approximation ratio becomes $\sqrt{\lceil \frac{n}{r} \rceil}$, a simpler expression than the other approximation ratios and so has a better possibility of being understood intuitively.

In addition to this, we have also shown that the solution to the strip partitioning with overlaps also functions as a lower bound. Both this, and the previous bound, can be found in polynomial time. So while they require more computation than the other lower bounds, they can still be found fast on a computer, which is where the algorithm is intended to be used.

Finally, the lower bound using star rooks might be promising if an approximation algorithm is developed to approximate a placement. Until then, this bound remains theoretical and difficult to use in practice since the maximum independent set problem is known to be NP-complete.

5.3 Switchline Analysis

Arguably, the main finding of this project is that of switchlines. Our results can both be used as a way of improving any solution, and to simplify already optimal solutions. With the knowledge that all switchlines in an optimal solution needs to be monotone sequences that can be simplified into L-shapes or be straight, and knowing that their endpoints need to be on corners of the figure, there are much fewer possibilities to choose from when designing a solution. This information is used in Theorem 2 to find an upper limit on the number of switchlines in an optimal solution. Using the possible switchlines together with this upper bound, there are still an exponential number of possible configurations, but a future algorithm could potentially approximate a selection from these limited possibilities and reason about how the result deviates from the optimal solution.

5.4 Limitations and Future Work

There is a middle ground between the work laid out in this paper, and the one presented by Arkin et al. [1]. While minimizing overlap in a milling tour is important for efficiency, it may not be worth the effort. Consider a possible path which does not permit intersection, but in order to achieve said goal, creates a significantly longer path than its intersection-counterpart. It would be worthwhile investigating a compromise between minimizing turns and intersections, to better mirror real-life circumstances.

Additionally, while this research mostly discusses methods to minimize the number of strips, a real tool still needs to create a path connecting these strips into a minimum-turn path. Possible inspiration on how to construct such a path has also been laid out by Arkin et al [1]. Their solutions can of course not be applied directly because of the difference in allowing intersections. As such, an unknown level of modification

would be required.

Furthermore, in real-world implementations of this research (notably NC-machining) the tool has many more alternatives for traveling. Both the polygon borders as well as strip alignments must here align with an orthogonal grid, whereas it is common for NC tools to travel in circular sweeps.

Using switchline analysis as a tool for algorithm design is not something that could be done within the timeframe for this project, and has not been found in other research within the field either. As mentioned earlier, we therefore suggest that there may be space to construct completely new approximation algorithms based on switchlines; for this specific problem outlined, and possibly other variations.

6

Conclusion

In summary, in this project we have analyzed the problem of partitioning an orthogonal polygon into strips and created an approximation algorithm for it. The approximation ratio also provides a guarantee to the quality of the solution so that it can be known in advance if the algorithm is a good choice for a particular situation. In addition to this, we have been able to prove various properties of the problem itself with the use of switchlines, the borders between regions of different orientations. We have also demonstrated a series of lower bounds, which can be used to compute approximation ratios for future research.

Bibliography

[1] E. Arkin, M. Bender, E. Demaine, S. Fekete, J. Mitchell, and S. Sethia, "Optimal covering tours with turn costs," *SIAM Journal on Computing*, vol. 35, pp. 531–566, Jul. 2006. DOI: 10.1137/S0097539703434267.

[2] T. Suk, C. Höschl, and J. Flusser, "Decomposition of binary images-a survey and comparison," *Pattern Recognition*, vol. 45, no. 12, pp. 4279–4291, 2012, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2012.05.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320312002427.

[3] D. Eppstein, "Graph-theoretic solutions to computational geometry problems," in *Graph-Theoretic Concepts in Computer Science*, C. Paul and M. Habib, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–16.

[4] E. W. Weisstein, *Grid graph*, https://mathworld.wolfram.com/GridGraph.html [Accessed: 2026-05-08], n.d.

[5] M. H. E. M. Arkin and C. L. Smith, "Optimization problems related to zigzag pocket machining," *Algorithmica*, vol. 26, pp. 197–236, 2000. DOI: https://doi.org/10.1007/s004539910010.