Utrecht University

Computing Science

Master Thesis

Project Proposal

Author:
Mark Leenen (6294464)
`m.leenen@students.uu.nl`

Supervisors:
prof. dr. M.J. van Kreveld
Utrecht University

dr. A.I. van Goethem
Eindhoven University of Technology

First Examiner:
prof. dr. M.J. van Kreveld
Utrecht University

Second Examiner:
dr. F. Staals
Utrecht University

May 7, 2020

Utrecht University

Contents

1 Literary Study

1.1 Introduction

1.1.1 Polygons

Cormen et al. [21] describe a *polygon* as a piecewise-linear closed curves in the plane. That is, it is a curve that ends on itself which consists of a sequence of straight line segments, which are called *edges* or sides. The set of points in the plane enclosed by the polygon is known as its interior, while the set of points on the polygon is its boundary. The remaining points, which surround the polygon, collectively make up its exterior. Two points x and y on the boundary or in the interior of the polygon are mutually *visible* if the line segment connecting x and y does not intersect the exterior.

A point joining two consecutive edges is called a *vertex*. If the interior angle between consecutive edges is strictly greater than π radians, the vertex is considered to be a *reflex* vertex, or concave vertex. Otherwise, it is known as a *convex* vertex. A chord of a polygon is a line segment through its interior joining two (non-adjacent) vertices [67].

1.1.2 Polygon classes

A polygon is *simple* if it does not intersect itself. Polygons are often assumed to be simple, unless explicitly stated otherwise. Self-intersecting polygons, on the other hand, are not common in computational geometry and are excluded from the scope of this paper. A convex polygon is a polygon whose vertices are all convex. Equivalently, any two points p, q on the boundary or in the interior of the polygon, must be mutually visible. While a simple polygon is topologically equivalent to a disk, a polygon with holes may be obtained by removing a non-overlapping set of strictly interior, simple subpolygons from the polygon [73]. Some authors allow for degenerate holes such as points or lines. The complexity of polygons and algorithms involving polygons is often measured using the number of vertices of the polygon, typically denoted n. Occasionally the number of reflex vertices or the number of holes in the polygon are taken into account as well, usually denoted N and H, respectively.

1.1.3 Applications

Polygons serve a wide variety of applications, both practical and theoretical. They are used to represent a wide variety of shapes and figures in computer graphics, computer vision, pattern recognition, robotics, and other computational fields. Furthermore, polygons are key components in geographic information systems (GIS) [14]. These tools represent areas in the plane and their boundaries as polygons, which serve for their visualization and to perform computations and queries on them. Polygons may be used to represent planar subdivisions, for instance in the context of thematic cartography [10].

1.1.4 More polygon classes

Some applications call for more restricted polygons. Orthogonal polygons, also referred to as rectilinear polygons, are polygons whose edges are parallel to the (orthogonal) coordinate axes. Therefore, in the Cartesian coordinate system the edges are either horizontal or vertical. Problems in computational geometry involving polygons frequently allow for more efficient algorithms when restricted to orthogonal polygons [73, 75]. Star-shaped polygons are polygons that contain a point from which the entire polygon boundary is visible. The kernel of a polygon is the set of all of the points from which the entire polygon boundary is visible. Note that any convex polygon is a star-shaped polygon where the kernel consists of the entire polygon itself. Star-shaped polygons play a key role in the art gallery problem originally proposed by Chvátal [18], in which one is to find the minimum number of points in a polygon from which each point of the polygon can be seen. Another similar applications are that of graphics, scene analysis, and robotics [75].

1.1.5 Polygon decomposition

Sometimes it may be desirable to consider the components of the polygon rather than the polygon in its entirety. For instance, convex polygons offer nice properties that allow more efficient algorithms for certain computational problems than for polygons in general. This is but one of the motivations for the decomposition of polygons. Many computational geometry algorithms decompose their input polygon into elementary pieces.

Within polygon decomposition various dimensions and parameters can be identified [73]. First of all, there are the elementary components, or primitive components, that the polygon is to be decomposed into. This is typically a type of geometric object such as rectangles or triangles, or a restricted type of polygon such as convex polygons. Then the original input polygon itself may be restricted to a particular domain. For instance, it could be given that the input polygon is orthogonal. Furthermore, whether the polygon is simple or contains holes is a relevant distinction, as it can make the difference between problems that can be solved efficiently and intractable problems. Moreover, some problem statements may explicitly allow or disallow Steiner points, whereas others do so implicitly. For a decomposition to be 'minimal', two measures are commonly used. Either a decomposition is minimal in the number of components in the decomposition, or it is minimal in the total length of its chords, also known as 'ink' used.

However, the most important dimension is that of the type of the decomposition. The class of polygon decomposition problems contains the subclasses polygon covering and polygon partitioning. Generally speaking, in polygon covering a polygon is to be decomposed into a set of given elementary components such that its union is equal to the input polygon. That is, in this case the decomposition permits overlap between the elementary components of the set. Polygon partitioning on the other hand, requires a polygon to be decomposed into a *non-overlapping* set of given elementary components such that its union is equal to the input polygon. Any valid solution to the polygon partitioning problem is therefore a valid solution to the polygon covering problem, and the value of any minimal solution to the former problem can be used as an upper bound for the latter problem. Figure 1 illustrates the decomposition of a rectilinear polygon into the minimum number of components. Consider a rectilinear polygon that is the result of the union of $k \times k$ congruent orthogonal bars, as shown in Figure 1(a). The minimum covering contains $2k$ rectangles whereas the minimum partitioning contains $k(k+1)$ rectangles. Note however, that this covering is not a minimum with respect to ink usage. In fact, Figure 1(b) requires twice the amount of ink that Figure 1(c) does.

Polygon Covering Almost all variations of the covering problem are intractable [73]. Bar-Yehuda and Ben-Chanoch [6] presented a polynomial time, output sensitive solution to one of the few tractable problems of the class of polygon covering problems, minimum square covering, minimizing the number of squares.

Polygon covering lends itself to image processing, where it is used to create a compact description of pictures [5, 78]. Quadtrees were often used to store black and white picture information. However,

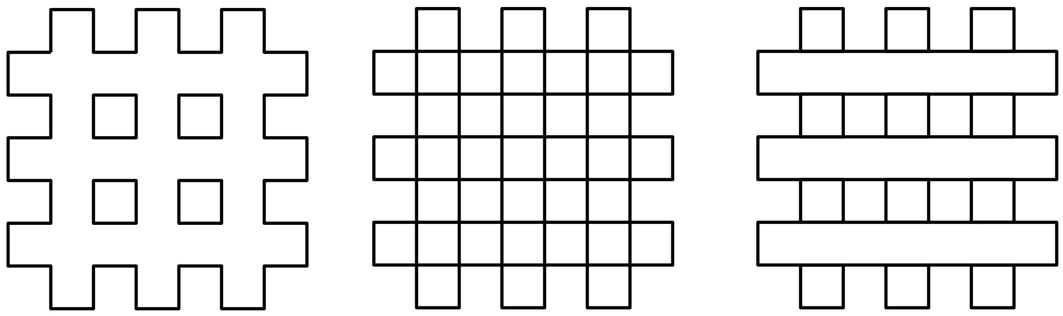

(a) Rectilinear polygon (b) Minimum covering (c) Minimum partitioning

Figure 1: Polygon decomposition of rectilinear polygon in minimum number of components

they took a considerable amount of storage and were very sensitive to the placement of the origin. Polygon covering allowed for storing black squares compactly.

Moreover, Aupperle et al. [5] noted that a solution to the polygon covering problem could be used to create the medial axis of the polygon. Using the digital medial axis transform (MAT) would also allow for picture compression [86].

Another application of polygon covering is in pattern recognition, where the decomposition could be used as feature extraction. It could be used, for example, to recognize Chinese handwriting [3, 29]. The benefit of this approach is that it is translation and rotation invariant. It should be noted however, that modern research makes widespread use of neural networks [84] for this type of feature extraction. It remains unclear whether a hybrid would yield better results.

Polygon Partitioning Agarwal et al. [2] propose an approach using polygon partitioning to decompose a polygon into convex subpolygons, allowing for efficient construction of Minkowski sums. This is but one of many examples in which decomposing polygons into smaller part that allow for employing more efficient algorithms. Combining the results of the smaller parts may yield (an approximation of) a solution for the original polygon [60].

Polygon partitioning is extensively used in VLSI design, in which geometric patterns are to be flashed from a photomask onto a photosensitive material [48]. The photomask typically consists of a piece of glass that can be considered as an orthogonal polygon. A partitioning into rectangles allows the pattern generator to expose such rectangles. A minimum number of rectangles is desirable, as the mask generation time depends on the number of rectangles.

Additive arc manufacturing is another application of polygon partitioning. As per Ding [26], the 3D geometry of a component is transformed into 2D slices. Then the 2D slices are partitioned into convex or monotone polygons, which makes the implementation of path generation considerably easier.

Enclosing Problem A variant on the covering problem is the enclosing problem, in which one is to find the smallest geometrical object of a given type that encloses all the points in a given a set $\mathcal{S}$ of n points. A variety of geometrical objects have been considered, including the minimum area triangle [72], smallest ellipsoid [85], the minimum bounding box [83], and the minimum area square [9].

In the related p-center problems, a point set $\mathcal{S}$ of n points is to be covered using p geometric shapes, such that the maximum area is minimized. One application of such problems is deciding on the placement of p distribution centers such that the union of their service areas is to cover all points in $\mathcal{S}$ [44]. Rectilinear p-center problems consider ℓ_1 or ℓ_∞ distance measures, resulting in squares (albeit in different orientations). Note that any problem instance that is to be covered using axis-aligned rectangles of a given aspect ratio may be transformed into a problem instance that is to be covered using axis-aligned squares by scaling the point set along one coordinate axis of the input with the aspect ratio.

1.1.6 Packing

Various definitions of the packing problem exist. Typically one is given a container, in this case a two-dimensional shape, and a set of objects that are to be packed into the container. The goal of the problem is usually to either pack as many of the set of objects into one container without overlap, or to determine the minimum number of copies of the container to be able to pack the entire set of objects into containers. For this paper, the scope is limited to the former, which is also known as the nesting problem.

The packing problem applied to polygons is similar to the problems in the class of polygon decomposition, in that the resulting solution can be considered as a crude approximation or simplified description of the input polygon. Polygon covering and polygon partitioning require the union of the solution, a set of shapes, to be exactly equal to the input polygon. The polygon enclosing problem as described in section 1.2.6 only requires the solution to contain the input polygon. The packing problem, on the other hand, requires the solution to be contained in the input polygon instead. Polygon packing is a considerably difficult problem, however, as most if not all non-trivial varieties of it are NP-complete.

1.2 Covering

1.2.1 Introduction

In the polygon covering problem, given some simple polygon P defined by n points in the two-dimensional plane, one is to find a smallest set of geometric objects of a given type (e.g. rectangles) such that their union is equal to P. Polygon covering is a subclass of polygon decomposition. Various geometric objects are often considered, such as squares, rectangles, star-shaped polygons, convex polygons, and triangles.

Most problems within the class of polygon covering have been shown to be NP-hard. A summary of the results can be found in Table 1. For all of the problems listed, Steiner points are permitted and the minimization measure is the number of components of the decomposition.

1.2.2 Minimum square cover

In the minimum square cover problem, a polygon is to be covered using as few squares as possible.

Initially, the image processing literature [86] had assumed the minimum square cover to be NP-hard following NP-completeness results for the minimum rectangle cover problem, which is discussed in section 1.2.3. However, Aupperle et al. [5] managed to disprove this assumption and showed that the minimum cover can be found quickly after all. Their approach considers the minimum square cover problem restricted to orthogonal polygons whose vertices are all at integral coordinates. Within these coordinates, a unit square is called a block. They reduce the polygon to a chordal graph. Using a known algorithm for finding a minimum clique cover of a chordal graph by Gavril [34], they find a minimum square cover in $O(B^{2.5})$, where B is the number of blocks in the polygon. Aupperle [4] then improved the complexity to $O(B^{1.5})$.

Table 1: Results on the covering of (orthogonal) polygons into various types of components.

Holes	Component	Polygon	Result	Author(s)
Yes	Squares	Orthogonal	$O(B^{1.5})$	Aupperle [4]
Yes	Squares	Orthogonal	$O(n + k)$	Bar-Yehuda & Ben-Chanoch [6]
No	Squares	Orthogonal	NP-complete	Aupperle et al. [5]
No	Squares	Orthogonal	MAXSNP-hard	Berman & DasGupta [11]
Yes	Rectangles	Orthogonal	NP-complete	Masek [68]
Yes	Rectangles*	Orthogonal	$O(n + k)$	Bar-Yehuda & Ben-Chanoch [6]
No	Rectangles	Orthogonal†	$O(n^2)$	Franzblau & Kleitman [31]
No	Rectangles	Orthogonal	NP-hard	Culberson & Reckhow [22]
No	Convex	Polygons	NP-hard	Culberson & Reckhow [22]
Yes	Convex	Polygons	NP-hard	O'Rourke & Supowit [71]
Yes	Star-shaped	Polygons	NP-hard	O'Rourke & Supowit [71]
Yes	Spiral	Polygons	NP-hard	O'Rourke & Supowit [71]

* Axis-aligned rectangles of fixed aspect ratio;
† x-monotone polygons

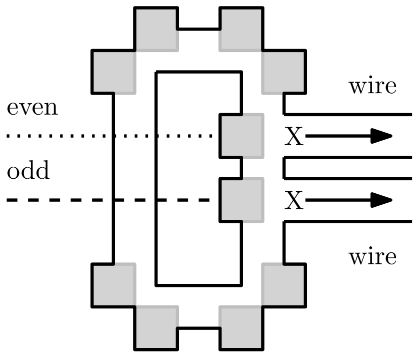
The problem was later solved by Bar-Yehuda and Ben-Chanoch [6] for hole-free, rectilinear polygons and axis-aligned squares with an amortized $O(n + k)$ time algorithm, where k is the output size. Their approach considers all maximal squares within the input polygon. In order for a square to be maximal, it must be bounded on two of its opposite sides by a segment of the input polygon. Each maximal square must be either a continuator or a separator with respect to the input polygon. Continuators are maximal squares that, when intersected with the outline of the polygon, result in a single, continuous partial outline. Separators are maximal squares that, when removed from the input polygon, disconnect it. These topological properties allow for identifying essential squares which must be in the minimum cover and eraseable regions which are already covered and may therefore be disregarded. This leads to a simple local optimization approach which completes in $O(n + k)$ iterations, where k is the output size. Combined with a data structure for visibility queries, the result is an $O(n + k)$ amortized time algorithm.

Aupperle et al. [5] further showed that the minimum square cover problem is NP-hard for rectilinear polygons that contain holes. Their proof consists of a reduction from Planar 3-SAT [58], which makes use of three constructs, typically referred to as 'gadgets', which are placed on a grid:

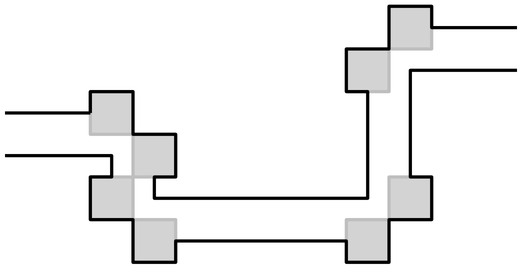
1. *Variable Loop* (see Figure 2(a))
   This construction is placed in the grid, such that even and odd grid lines represent positive or negative boolean values respectively. The *Variable Loop* is placed on this grid as a loop shaped, rectilinear polygon. It can be grown to arbitrary height to accommodate for the required number of instances of the variable. Due to the spacing between the indents along the side, in any minimum cover either all even or all odd connections to wires must have a square protruding into their respective wires, analogous to a TRUE value, whereas the other connections must all have a covering flush to the loop's side, representing a FALSE value. This results in what corresponds to the consistency of a variable's value among its positive and negative instances.
2. *Wire* (see Figure 2(b))
   The fundamental design of this gadget consists of a horizontal, straight and open-ended strip of even length and width equal to two. As a result, any minimum covering of the strip

must either be flush on both ends, or have squares protruding from both ends, allowing for the propagation of a boolean value from its source to its destination. The fundamental design can be augmented by allowing the wires to bend, although to maintain aforementioned propagation property, the source and destination of the wire are to be centered on horizontal grid lines of equal parity (both even or both odd).

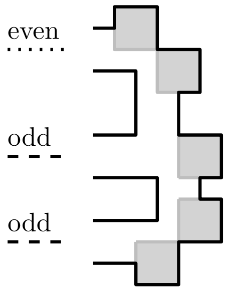
3. *Junction* (see Figure 2(c))
   The clauses of Planar 3-SAT are modeled as Junctions. By itself, it can only test the value of the logical AND of three input variables. The logical OR, which is required to represent the clauses of the Planar 3-SAT, can be acquired by applying DeMorgan's law: Each literal in the problem instance is negated and each OR is replaced by an AND. The Junction construction comes in two varieties, but both have their left boundary aligned on even vertical grid lines. One for the case that all literals are either negated (aligned on odd horizontal grid lines) or non-negated (aligned on even horizontal grid liens), and another for clauses in which there are two negated variables and one non-negated, or vice versa. By the commutative property of the logical OR, these two variants suffice to represent any clause of three variables. A Juncion can be covered with 12 squares if and only if all three input wires carry a TRUE value, otherwise 13 squares are required to do so. As for using the Junction to emulate the logical OR, rather than finding an assignment of truth values such that all clauses are satisfied, i.e. Junctions are covered with 12 squares each, the task is now to find an assignment such that each clause is not satisfied, in which the corresponding Junctions are covered with 13 squares each.



(a) Variable loop



(b) Wire



(c) Junction

Figure 2: Gadgets for Planar 3-SAT hardness reduction to minimum square cover. Gray squares represent squares that must always exist in a minimum square cover. Even and odd grid lines are represented as dotted and dashed lines, respectively.

Using these gadgets, an instance of Planer 3-SAT can be transformed to an instance of covering a rectilinear polygon with squares in polynomial time. A minimum threshold value can be computed from the boolean formula and the design of the gadgets which corresponds to whether the formula is satisfiable based on the number of squares used in the minimal cover. In this particular construction, let C be the number of squares in a minimum cover, and V, W, J, the variable generators, the number of squares required to cover the wires, and the junctions of the polygon, respectively. If $C = V + W + 13J$, then the formula is satisfiable. Else, if $C < V + W + 13J$, then the formula is not satisfiable.

In fact, Berman and DasGupta [11] later proceeded to show that the problem with polygons that contain holes is MAXSNP-hard, presenting the first MAXSNP-hardness proof for a geometric problem. Their results ruled out the possibility of a polynomial time approximation scheme.

1.2.3 Minimum rectangle cover

Covering a hole-free, orthogonal polygon using as few rectangles as possible is known as the minimum rectangle cover. Masek [68] showed that covering a rectilinear polygon, which may contain holes, with rectangles is NP-complete.

For polygons that may contain holes, Bar-Yehuda and Ben-Chanoch [6] extend their results on the minimum square cover to covering with axis-aligned rectangles of a given aspect ratio, by scaling the input set in one coordinate axis by the specified aspect ratio. Doing so results in an problem instance that can be solved using the algorithm for the minimum square cover, which therefore runs in $O(n + k)$ time as well.

Franzblau and Kleitman [31] provided an algorithm for covering an orthogonal polygon R of n vertices without holes minimally using rectangles, on the condition that the polygon is x-monotone. That is, if the polygon is intersected with any vertical line, the result is either empty or a single, continuous line segment. Their solution is based on a reduction to an equivalent, 1-dimensional problem. First, the polygon is divided up in *horizontal slices*, a connected set of squares in the same row of R, with both ends on the boundary, i.e. it is horizontally maximal. Observe that a horizontal slice uniquely identifies a maximal rectangle in R. However, two horizontal slices that determine the same maximal rectangle are called equivalent. A top to bottom sweepline algorithm can obtain the set of inequivalent horizontal slices. These sets are then projected onto the x-axis, resulting in a set S of intervals. A set G generates S if it consists of the elementary intervals of S, i.e. every interval in S is equal to the union of one or more intervals in G. A generating set G for S can be constructed into a set of vertically maximal rectangles in R which cover R. Moreover, the authors prove that a minimum generating set MGS of S can be used to construct a covering that uses the least number of rectangles. In order to compute MGS, generating set G is reduced iteratively until each interval has at least one subinterval that is not covered by other intervals in G. The entire procedure can be completed in $O(n^2)$ time.

Culberson and Reckhow [22] proved the NP-hardness of the following covering problems, even for polygons without holes:

1. Covering an arbitrary polygon with the least number of convex polygons
2. Covering the boundary of an arbitrary polygon with the least number of convex polygons
3. Covering an orthogonal polygon with the least number of rectangles
4. Covering the boundary of an orthogonal polygon with the least number of rectangles

The NP-hardness proofs rely on reduction from SATISFIABILITY using three gadgets:

1. The *Beam machine* structure forces one of two slim convex polygons to be present in a minimum covering, corresponding to the assignment of its matching variable.
2. *Variable generators* ensure a consistent assignment for each variable.
3. *Clause checkers* model the clauses of boolean formulas and can only be covered minimally if their corresponding clause can be satisfied.

1.2.4 Other minimum covers

The problems of covering with a minimum number of convex polygons, star-shaped polygons, or spiral polygons was later shown to be NP-hard by O'Rourke and Supowit [71].

Batchelor [7] investigates a procedural approach using convex *sum/difference* decompositions. Their decomposition consists of a tree structure, the concavity tree. The root consists of the convex hull of the polygon, representing a sum operation. Its children consist of the convex hulls of concavities in the polygon, representing a difference operation. The algorithm proceeds in alternating fashion, each level representing either a sum or a difference operation, until there are no more concavities. While introducing the concept of removing polygonal areas makes the algorithm more powerful, the presented algorithm also exhibits instability. That is, a small change in the input shape results in a major change of the resulting concavity tree.

1.2.5 Approximation algorithms

A large portion of the covering problems, while provably intractable, still have plenty of practical applications and thus call for efficient solutions that find approximations of their respective optimal solutions. An overview of approximation algorithms for the polygon covering problem can be found below.

It is trivial to observe that simple polygons with acute angles cannot be covered by squares. For general polygons without acute angles, it was unclear whether the problem could be solved, even in exponential time, until Levcopoulos and Gudmundsson [56] gave an $O(n)$ time approximation algorithm with an approximation factor of $O(\alpha(n))$, where α represents the inverse Ackermann function.

Hertel and Mehlhorn [36] describe a simple and fast algorithm to find a 4-approximation for a minimum cover using convex subpolygons, given a triangulation for the polygon. For each reflex vertex, add at most two edges such that the subangles are all convex. What remains is a partitioning with at most 4 times the minimum number of convex subpolygons.

Eidenbenz and Widmayer [28] propose a polynomial-time approximation algorithm for finding the minimum convex covering for polygons with or without holes. The authors first consider an optimal solution to a discretized version of the problem where vertices may only lie on a quasi-grid. After showing that the optimum solution to the restricted variant of the problem contains at most three times as many convex polygons as the unrestricted variant, a dynamic programming approach is applied iteratively to obtain a solution for the unrestricted problem that has an $O(\log n)$ approximation factor. Moreover, the problem is shown to be APX-hard.
The problem of covering an orthogonal polygon that contains holes with rectangles was shown to be NP-complete by Masek [68]. Since it is a special case of the general set covering problem, it admits a polynomial time approximation algorithm with approximation factor $O(\log n)$ using a greedy scheme by Johnson [41]. However, Kumar and Ramesh [54] improved on this bound and presented an $O(\sqrt{\log n})$ factor approximation algorithm for the problem.

1.2.6 Rectilinear p-center problems

Rectilinear 1-center problem Given a convex polygon P of n vertices, Bereg et al. [9] consider the rotation of its enclosing square and identify a linear number of key events that may result in a minimum size of the enclosing square. Each event can be processed in constant time, which would yield an $O(n)$ time algorithm. Non-convex polygons can also be covered in $O(n)$ time by transforming the non-convex polygon into a convex hull using Melkman's linear time convex hull algorithm for simple polygons [69]. The approach can also be applied to point sets by computing their convex hull in $O(n \log n)$.

Rectilinear p-center problems For the case where it does not matter whether the resulting shapes overlap, Drezner [27] established an optimal solution for the rectilinear 2-center problem with an $O(n)$ time algorithm, whereas Hoffmann [37] presented an $O(n)$ time algorithm for the rectilinear 3-center problem. Using the notion of p-piercing, Sharir and Welzl [80] show that the rectilinear 4-center can be solved in worst-case optimal $O(n \log n)$ and the rectilinear 5-center problem can be solved in $O(n \log^5 n)$ time. The latter gives rise to an algorithm that solves the rectilinear p-center problem for $p \geq 5$ in $O(n^{p-4} \log^5 n)$ time. For large values of p [37], the best known asymptotic running time for solving the rectilinear p-center problem is $O(n^{8\sqrt{p}+10} \log n)$ by Agarwal and Procopiuc [1].

Discrete rectilinear 2-center problem Katz et al. [44] consider the problem of covering the points in set $\mathcal{S}$ by two "constrained" axis-parallel squares whose center must coincide with points from $\mathcal{S}$, while minimizing the area of the largest square. This problem is known as the discrete rectilinear 2-center problem. In fact, the authors generalize the problem by introducing point set $\mathcal{C}$, which consists of m points. In the generalized problem statement, the centers of the axis-parallel squares must coincide with points from $\mathcal{C}$, rather than with points from $\mathcal{S}$. In their approach, the decision variant of this generalized problem is first solved: Given a set $\mathcal{S}$ of n input points and area $\mathcal{A}$, find two discrete axis-parallel squares of area $\mathcal{A}$ each that cover $\mathcal{S}$. In order to solve the decision problem, they introduce the notion of $(p, \mathcal{C})$-*coverable*, which holds if $\mathcal{S}$ is contained in the union of p rectangles of area $\mathcal{A}$, each constrained to a points in $\mathcal{C}$. The original decision problem could then be rephrased as to determining whether $\mathcal{S}$ is $(2, \mathcal{S})$-coverable. In order to determine whether $\mathcal{S}$ is $(2, \mathcal{C})$-coverable, $\mathcal{S}$ is partitioned into four quadrants by two orthogonal lines. Without loss of generality, let $\mathcal{S}_2$ be the subset of points of $\mathcal{S}$ which reside in the top-right quadrant, and let $\mathcal{S}_1 = \mathcal{S} \setminus \mathcal{S}_2$. Note that there are n^2 distinct configurations for the two orthogonal lines. An $n \times n \times 2$ matrix is computed such that each entry represents, for a particular configuration of lines, whether either $\mathcal{S}_1$ or $\mathcal{S}_2$ is $(1, \mathcal{C})$-coverable. The coverability can be computed efficiently by dynamically maintaining an orthogonal range tree. Employing an approach similar to searching in monotone matrices by Sharir [79], finding an entry for which both sets are $(1, \mathcal{C})$-coverable results in an $O(n \log n)$ time algorithm. The solutions to the decision problem can be transformed to a solution to the optimization problem by applying a sorted matrix technique by Frederickson and Johnson [32].

Discrete rectilinear 2-center problem with constraining set In previous research, Bespamyatnikh and Segal [12] considered covering $\mathcal{S}$ by two axis-parallel boxes that are not constrained, i.e. their centers need not coincide with any point from an additional point set $\mathcal{C}$. Moreover, rather than using the area of the boxes directly, the algorithm optimizes monotone measures of the rectangles that can be evaluated in constant time, such as the perimeter of the box, the diagonal of the box, or the area of the box. First the notion of *determinators* is introduced, which are the extreme points of each of the d axes resulting in $2d$ determinators. For the two-dimensional case, they show that it suffices to traverse the points in sorted x-order, using two pointers as a sliding window for configurations that yield valid values of the monotone measures. Therefore, for a presorted list of points, the algorithm can solve the problem in linear time.

For higher dimensions, the authors consider all $\binom{2d}{d}$ configurations where for each axis, one determinator is fixed. Without loss of generality, starting from an initial solution of two dimensions, the authors consider all tuples of $d-2$ points in $\mathcal{S}$, resulting in n^{d-2} tuples. A linear time procedure is discussed that solves the problem with one specific tuple, resulting in a total running time of $O(n \log n + n^{d-1})$.

1.2.7 Enclosing problems

Bereg et al. [9] solve the enclosing problem for up to four unconstrained squares in a wide variety of settings for point sets of n points. These problems are similar to the rectilinear p-center problems, in that a set of points is to be covered using squares of minimum size. However, in addition to axis-aligned squares, some of these enclosing problems may allow for arbitrary rotation of the squares.

In addition to specifying the number of squares and their orientation, the parameters to the enclosing problems include whether the squares may overlap. Either they may, or they may not in which case they must be disjoint. Additionally, they present algorithms which are not constrained as such. The results are summarized in Table 2.

Table 2: Minimizing the size of the largest square. AP = "axis-parallel", AO = "arbitrary orientation", D = "disjoint", D = "non-disjoint" (overlapping), and DC = "don't care" (the squares involved are not constrained to be D or ND).

Number of squares	D/ND/DC	AP/AO	Previous result	Bereg et al. [9] result
1	-	AO	$O(n^2 \log n)$ [23]	$O(n \log n)$
1	-	AO	-	$O(n \log n)$
2	D	AP	$O(n \log n)$ [40]	$O(n)$
2	ND	AP	-	$O(n)$
2	DC	AP	$O(n)$ [80]	$O(n)$
2	D	AP-AO	$O(n^2 \log n)$ [50]	-
2	DC	AP-AO	-	$O(n^3 \log n)$
2	DC	AO	-	$O(n^4 \log n)$
3	D	AP	-	$O(n \log n)$
4	D	AP	-	$O(n^2 \log^2 n)$

For 2 disjoint axis-parallel squares, Bereg et al. partition the set of points in two almost equally sized subsets using a vertical line. That is, the size of the sets may differ by at most one. For both subsets, determine the smallest enclosing square and consider the smallest of the two. Observe that it is defined by its extreme points and that if those are covered by one square, all other points in this subset are covered too. Those non-extreme points can therefore be removed from consideration. This process is repeated with the remaining points until the smallest enclosing squares are of equal size, in which case the algorithm is finished, or when the total size of the set of points falls below a preselected constant, in which case the remainder is solved by brute force. This results in an $O(n)$ time algorithm. Note that, unless the constant is chosen to be too small, the vertical line will not separate points that were jointly considered as extremes before.

For 2 overlapping axis-parallel squares, Bereg et al. initially consider two rectangles, each covering two adjacent extremes of the points. Without loss of generality, the first rectangle has sides on the leftmost and topmost points, while the other rectangle has sides on the rightmost and bottom-most points. The rectangles are then expanded towards the center, while maintaining equal size, until they touch each other. If all points are covered, the rectangles are shrunk to minimal size. Otherwise, determining how far both squares are to expand requires a constant time computation for each of the remaining points. It should be noted that from one point it may suffice to only grow one of the squares. This results in an $O(n)$ time algorithm.

For the problem of covering two squares of which one is axis-parallel and the other is arbitrarily oriented, where overlapping or disjointness is irrelevant, Bereg et al. provide an $O(n^3 \log n)$ time algorithm. The algorithm considers all $O(n^2)$ pairs of x coordinates and y coordinates as bottom-left of the axis-parallel square. For each pair, the corresponding square S is expanded, encountering $O(n)$ events where a new point touches the boundary of the square, defining a new axis-parallel square S. The minimum (arbitrarily oriented) enclosing square S' of the points that are not contained in S can be computed and maintained using a modified Graham scan in $O(n)$ time. Clearly, as the size of S increases, the size of S' is monotonically non-increasing. Therefore, a binary search on the size of S can be used to minimize the maximum of the sizes of S and S'.

This solution can be extended for the case of two squares which are both independently arbitrarily oriented, where overlapping or disjointness is irrelevant. Observe that for a pair of points, their projection on the x-axis may reverse by rotating the x-axis. Accordingly, a single point has $n - 1$ critical angles, while for all points collectively, there are $O(n^2)$ critical angles.

Consider the bottom left point of S, which consists of the x and y coordinate of two points in S (which may be the same point), respectively. Rotating the x-axis yields $O(n)$ critical angles, as described before. The previous algorithm can then be applied on the rotated set of points, such that what is now axis-parallel square S originally corresponds to a square which is rotated by the critical angle. This imposes an additional linear factor on the original algorithm, resulting in a $O(n^4 \log n)$ running time.

Note that between two subsequent critical angles, it may occur that the width of the minimum enclosing *rectangle* becomes larger than its height or vice versa, which constitutes another critical angle. Overall there will however still be $O(n^2)$ critical angles.

A minimal covering with three axis-parallel, disjoint squares may take on exactly one of six different patterns. Either the three squares are aligned in a line, horizontally or vertically, or there is a line that can separate one square from the other two from the left, right, top or bottom. In order to find a minimal covering for a given point set, for each pattern the points are partitioned into two subsets using the corresponding dividing line and solve the problem optimally for one square and two squares, respectively. The optimal placement for each line can be determined by a binary search, incurring an additional $O(\log n)$ factor, which leads to an $O(n \log n)$ algorithm.

In order to proceed to four disjoint axis-parallel squares, the notion of *guillotine k-partition* is introduced. A k-partition of a rectangle R is a partitioning of R into k subrectangles. A *guillotine* cut is a cut from top to bottom or from left to right. The authors describe classify 6 distinct 4-partitionings, as illustrated in Figure 3. Using one guillotine cut, each can be decomposed into two subproblems of at most three rectangles. The underlying point set in each subproblem can then be covered using one of the previous algorithms. There are $O(\log n)$ choices for the guillotine cut, and each subproblem can be solved in $O(n \log n)$ time, resulting in an $O(n \log^2 n)$ algorithm.

However, there remains one configuration of point sets that must be taken into account. Four rectangles may permit a hole, as shown in Figure 4, setting it apart from the 4-partitions, topologically speaking. In order to solve this case, let R be the axis-aligned bounding box of the point set. For each pair of points, the vertical positions of H_u and H_l are fixed on their vertical coordinates, respectively. A nested binary search can then be used to determine the horizontal positions of V_r and V_l such that the area of the largest subrectangle is minimized. The mirror image of this case is solved analogously. This procedure results in an $O(n^2 \log^2 n)$ time algorithm.

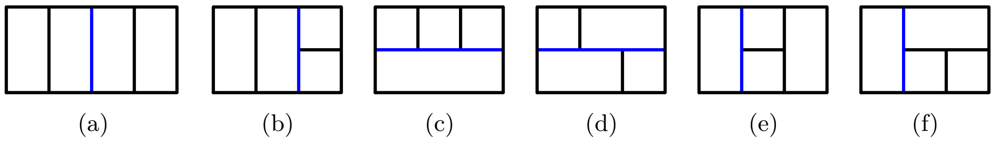


Figure 3: Distinct 4-partitions. The blue line denotes the first guillotine cut.

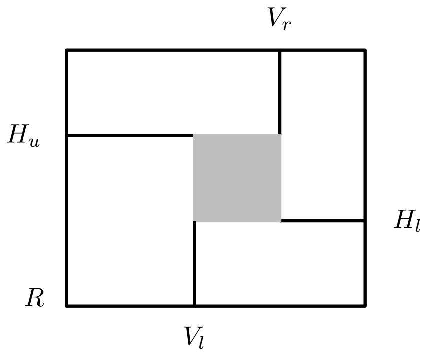


Figure 4: Non-guillotine 5-partition of rectangle R. The gray subrectangle may not contain any points.

Enclosing polygons In polygon covering, given a polygon P (or alternatively, a set of polygons) and a number $p \in \mathbb{N}$, a set of p geometric objects of a given type and minimal size is to be found such that any point in (any polygon in) P is contained in at least one of the p shapes. Hoffmann [37] considers polygon covering of a set of non-intersecting polygonal regions with axis-parallel congruent squares. For $p = 2$, he first generalizes the problem to line segment covering and subsequently reduces it to point set covering, which was shown to be solvable in $O(n)$ time. For $p \leq 3$, Hoffmann reduces the problem to covering the boundary edges of the polygon, which can similarly be solved in $O(n)$ time.

1.3 Partitioning

1.3.1 Introduction

Given a simple polygon P defined by n points in the two-dimensional plane, the polygon partitioning problem is to find a smallest set of disjoint simpler subpolygons, such that their union is equal to P. The problem is very similar to the polygon covering problem, except in that the partitioning problem requires the decomposition to have no overlap at all. These two problems jointly make up the class of polygon decomposition problems.

Examples of the applications of polygon partitioning are pattern recognition, (robot) motion planning [55], and more advanced computational geometry problems. Partitioning may result in a compact description of the original figure [31], and may allow for the use of more efficient algorithms that only apply to simpler figures [60].

Polygon partitioning problems may allow for the use of Steiner points. These are additional vertices introduced to facilitate the partitioning [10]. If not allowed, a partitioning may only use the vertices of the input polygon.

1.3.2 Convex partitioning with holes

For polygons with (degenerate) holes, Pagli et al. [66] show that the problem of partitioning the polygon into a minimum number of convex components is NP-hard, if Steiner points are not allowed. Lingas [59] followed up and proved that even when allowing Steiner points, the problem remains NP-hard. Their proof is similar to the hardness proofs for covering, but reduces from a modified planar satisfiability problem (MPLSAT), where each clause with three literals must contain at least one positive and one negative clause. The proof makes use of wire loops which can only be partitioned in exactly two manners, corresponding to the boolean assignment of a variable. Junctions are designed such that it can only be partitioned minimally if each of the connected wire loops is partitioned as specified by the one-to-one correspondence with its clause.

1.3.3 Convex partitioning without holes

For polygons without holes, Keil [46] developed an $O(N^2 n \log n)$ time algorithm that disallows the use of Steiner points. Here, n represents the number of vertices of the polygon and N represents the number of reflex vertices of the polygon. Keil and Snoeyink [49] later improved the approach and achieved an $O(n + N^2 \min\{N^2, n\})$ time algorithm. The improvements include the use of stacks in place of a search structure, and simplification of the input which retains the same minimum partitioning but has a bound of $O(\min(N^2, n))$ sides.

For the case that allows for Steiner points, Chazelle and Dobkin [17] present a polynomial time algorithm for partitioning a polygon into a minimum number of convex components. Their approach makes use of dynamic programming to identify X patterns in the polygon, which represent an interconnection between reflex vertices which, when used as decomposition, remove those reflex angles and introduce no new ones. An X_k pattern is an X pattern that connects k reflex vertices. It follows that decomposition using the most X patterns is minimal. However, determining whether a set of reflex vertices can be interconnected via an X pattern appeared too involved of a process. The introduction of Y patterns resolves this problem. Y patterns can be regarded as X patterns with an additional structural property. Aside from X_4 patterns, any X pattern can be advantageously replaced by a Y pattern. Moreover, Y patterns can be constructed in polynomial time using dynamic programming. This approach leads to an $O(n + N^3)$ time algorithm.

1.3.4 Minimizing ink

Another optimization criterium is the minimization of the total edge length, or ‘ink’. Partitioning a polygon into convex subpolygons such that their total edge length is minimized if Steiner points are allowed was shown to be NP-complete by Lingas et al. [61], regardless of whether or not the polygon may contain holes. Their approach is very much alike the typical approach with gadgets for rectilinear partitionings of rectilinear polygons. The one exception is that the *phase shifter*, a gadget used to line up other gadgets, now needs to incorporate non-rectilinear lines. Greene [35] presents a minimum ink algorithm for partitioning into convex subpolygons based on Keil [46], in $O(n^2 N^2)$ time.

The results are summarized in Table 3.

Table 3: Results on the partitioning of polygons into convex subpolygons. The results either minimize for number of components (C) or edge length (E).

Holes	Steiner points	Optimizing	Result	Author(s)
Yes	Yes	C	NP-hard	Lingas [59]
Yes	No	C	NP-hard	Pagli [66]
No	No	C	$O(N^2 n \log n)$	Keil [46]
No	No	C	$O(n + N^2 \min\{N^2, n\})$	Keil & Snoeyink [49]
No	Yes	C	$O(n + N^3)$	Chazelle & Dobkin [16]
Yes	Yes	E	NP-complete	Lingas et al. [61]
Yes	No	E	NP-hard	Keil [45]
No	Yes	E	NP-complete	Lingas et al. [61]
No	No	E	$O(N^2 n^2)$	Greene [35]

1.3.5 Polygon triangulation

Partitioning a polygon into triangles, widely known as triangulation, plays an important role in computational geometry. Many algorithms for polygons begin by triangulating the polygon. Triangulation can be used to create mesh networks and are often used in representing terrains.

Garey et al. [33] proved an $O(n \log n)$ time algorithm as early as 1978. Ten years later, Tarjan and Van Wyk [82] discovered an $O(n \log \log n)$, which was later simplified by Kirkpatrick et al. Chazelle [15] followed up with an asymptotically optimal running time, showing that any simple polygon can be triangulated in $O(n)$ time.

1.3.6 Rectangular partitioning

When partitioning orthogonal polygons, rectangles become an important component to consider. Partitioning orthogonal polygons into axis aligned rectangles has various applications in image processing and VLSI design. Steiner points in this setting are typically assumed to be allowed. Lipski et al. [66] used a maximum bipartite matching approach on the vertical and horizontal chords to develop an $O(n^{5/2})$ time algorithm for partitioning orthogonal polygons with holes into a minimum number of rectangles. Due to the discovery of a special structure of the bipartite graph, the running time was improved to $O(n^{3/2} \log n)$ by Lipski [64, 65] and Imai and Asano [39]. Later Soltan and Gorpinevich [81] extended the algorithms to achieve the same running time while allowing for holes that degenerate into points. Liou et al. [62] show an $\Omega(n \log n)$ time lower bound for the problem with holes using a reduction from the sorting problem.

As for orthogonal polygons that do not contain holes, Liou et al. [63] developed an algorithm that originally solved the rectangular partitioning problem in $O(n \log \log n)$, based on the algorithm Tarjan and Van Wyk [82] that triangulates simple polygons in $O(n \log \log n)$ time. However, Chazelle [15] published an algorithm that achieves linear time triangulation of simple polygons, which in turn allows for the algorithm of Liou et al. to run in $O(n)$ time. The three dimensional version of the problem is shown to be NP-complete by Dielissen and Kaldewaij [24].

When optimizing for minimum edge length rather than minimum number of rectangles, Lingas et al. [61] present an $O(n^4)$ time algorithm for orthogonal polygons without holes. Additionally, they show that if a polygon were to contain holes, the problem becomes NP-complete following the same proof of partitioning general polygons into convex polygons.

Table 4: Results on the partitioning of orthogonal polygons into rectangles. The results either minimize for number of components (C) or edge length (E).

Holes	Optimizing	Result	Author(s)
Yes	C	$O(n^{3/2})$	Lipski et al. [64, 65, 66]
No	C	$O(n \log \log n)$	Liou et al. [63]
No	C	$O(n)^*$	Liou et al. [63]
Yes	E	NP-complete	Lingas et al. [61]
No	E	$O(n^4)$	Lingas et al. [61]

* Using the linear time polygon triangulation by Chazelle [15].

1.3.7 Partitioning into squares

Supposedly, StackOverflow user 'Realz Slaw' shows that partitioning an orthogonal polygon into squares is NP-hard by reduction from PLANAR-3-SAT [1] . Remarkably, this result is an interesting juxtaposition with the covering problem, where covering with rectangles is NP-hard but covering with squares is possible in polynomial time.

1.3.8 Quadrilateralization

If Steiner points are disallowed, quadrilaterals are used instead, as a generalization of rectangles. Partitioning a polygon into a convex quadrilaterals is referred to as quadrilateralization. Kahn et al. [43] show that it is always possible to find a quadrilateralization for orthogonal polygons, whereas it is not always possible for arbitrary polygons. Based on their proof, Sack [77] developed an $O(n \log n)$ time algorithm to find a partitioning into a minimum number of quadrilaterals. Moreover, they show that quadrilateralization of arbitrary monotone and star-shaped orthogonal polygons can be quadrilateralized in $O(n)$ time. Lubiw [67] takes a new approach to the results of Kahn et al. [43] by showing that rectilinear polygons without holes and rectilinear with holes are CQ hereditary and provides an $O(n \log n)$ time algorithm. A class P of polygonal regions is CQ hereditary if every polygonal region in P which is not itself a convex quadrilateral has a removable quadrilateral which leaves polygonal regions. As for minimum edge length quadrilateralization, Conn and O'Rourke [20] presented an $O(n^3 \log n)$ time algorithm, which was an improvement on the algorithm by Keil and Sack [47] which runs in $O(n^4)$ time.

1.3.9 Approximation

For polygons without holes, disallowing Steiner points, Greene, and Hertel and Mehlhorn [35, 36] provide $O(n \log n)$ time algorithms that find a convex partitioning that contains at most four times the optimal number of components. Allowing Steiner points, Levcopoulos and Lingas [57] present an $O(n \log n)$ time algorithm that results in a solution of size $O(p \log N)$, where p is the length of the perimeter of the polygon. For polygons with holes, they provide another $O(n \log n)$ time algorithm that results in a solution of size $O((b+m) \log N)$, where b is the combined length of the perimeter of the polygon and its holes, and p is the minimum length of its convex partition.

As for minimizing total edge length, Plaisted and Hong [74] give a polynomial time algorithm for the partitioning of a polygon into convex parts, guaranteeing a total edge length of at most

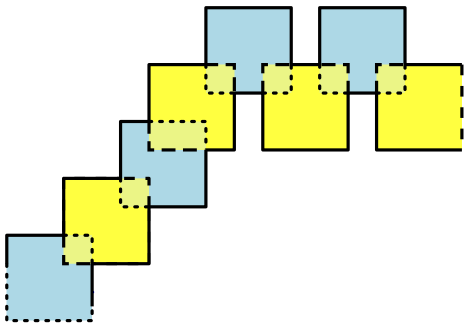
[1] https://cs.stackexchange.com/questions/16661/tiling-an-orthogonal-polygon-with-squares/16801#16801

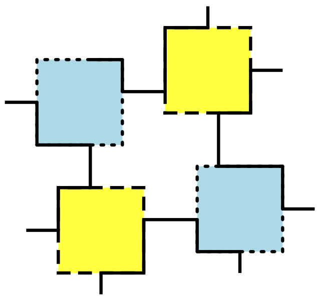
12 times that of the minimum total edge length, leading to an approximation algorithm for the minimum cost triangulation of polygons of an approximation ratio of $O(\log n)$.

1.4 Packing

1.4.1 Packing with squares

Fowler et al. [30] first give an outline for showing the NP-completeness of packing a two-dimensional polygon with squares of fixed size, where all coordinates are integers, by a reduction from maximum independent set problem. They go on to show the NP-completeness of packing squares into a two-dimensional polygon in the general case, denoted as the BOX-PACK problem, by a reduction from 3SAT. Let M, N denote the number of variables and clauses of the 3SAT instance, respectively. Furthermore, the size of the squares that are to be packed is 2×2. The reduction is based on a wire gadget which consists of a chain of connected "slots" for squares to be packed into, as shown in Figure 5(a). That is, for now each slot overlaps exactly two other slots, namely its neighbors. Due to this overlap, there are exactly two maximum packings for such a construction, where either squares are packed in all odd slots or in all even slots. Now let each variable v_i be represented by a closed wire which consists of $2 * K_i$ slots, then a maximum packing must contain K_i squares. The two different maximum packings correspond directly to the value assignment of the corresponding variable. Loops of different variables may cross using a crossover region as shown in Figure 5(b). It is a 3×3 square which is not taken into account when considering the length of each wire. In a maximum packing, each of the N_c crossover regions is always packed with exactly one square. Finally, the clauses of the 3SAT are modeled by bringing the variable loops of each of the involved variables into close proximity in a clause region. The clause region has the property that if at least one of the variables loops is in the a proper state, i.e. the value assignment of the variables satisfies the clause corresponding to the clause region, an extra square can be placed within it. This concludes the construction of a BOX-PACK instance from an 3SAT instance. A conjunctive normal form formula is satisfiable if and only if the corresponding constructed polygon can be packed with $\sum_{i=1}^{M}(K_i)+N_c+N$ squares. Since BOX-PACK is clearly in NP, and the aforementioned construction can be built in polynomial time, BOX-PACK is NP-complete.


(a) Wire gadget. In a maximum packing, either each yellow area (dashed) is packed, or each blue area (dotted) is packed.


(b) Crossover region. In a maximum packing, exactly one yellow (dashed) and one blue region (dotted) must be packed with squares

1.4.2 Rectangle packing

In the rectangle packing problem, a rectangular container is to be packed with rectangles from a given set. Korf et al. [38, 52, 53] show that the rectangle packing problem is NP-complete by a reduction to the bin-packing problem. In the rectangle packing problem, given a rectangular container and a set of rectangles, one is to find a placement of the rectangles inside the container, such that they do not overlap and are contained within the container entirely. It finds applications in the warehouse industry, where real life packing is applied. Other applications are the wood and glass industries, where rectangular components are to be cut from rectangular sheets, and can be applied to decrease wastage in the clothing industry [42]. Bennell and Oliveira [8] explore different heuristic solutions that apply to practical situations. More recently, Junior et al. [42] propose a heuristic for the packing of nonregular shapes into a rectangular strip of constant width, with the goal of minimizing the required length of the strip. Their solution is a hybrid of previous results that makes use of previous results. It combines genetic algorithms, a greedy bottom-left heuristic and the no-fit polygon data structure.

2 Research Questions and Methodology

2.1 Introduction

Schematic Maps Maps are a widely used visual medium for conveying information and placing it within geographic context. Often, exact geographic details are not required to convey the primary information of a map. Thematic maps provide a thematic overlay that is specifically designed to convey information on a particular theme related to a specific geographic area. An abundance of superfluous details may however distract or even obscure from the main purpose of a map [25, 70].

A schematic map often depicts the elements in an abstract and stylized form. The concept of simplification refers to reducing the level of detail of a map in order to support and reinforce the main message of the map and is an essential part of schematization. Schematization can be used to structure and simplify the information in order to lead to a better understanding and a reduced cognitive load for the user. Manual construction of schematic maps is a time-consuming process and along with the need for digital, up-to-date maps, has given rise to an interest in automated schematization.

Design Rules Part of the research in automated schematization focuses on identifying design rules and attempting to formalize them in terms of generalization operators and heuristics. Buchin et al. [13] introduce the notion of an *edge-move* a local operation for polygonal features which is suitable for iterative simplification. Reimer and Meulemans [76] conjecture that an *parallelity* is an important design rule for the construction of chorematic diagrams. *Continuation* of line features can be considered as another desirable design rule [19, 51].

2.2 Paper outline

In this paper, two approaches for the schematization of territorial outlines will be investigated. The territorial outlines are assumed to be simple orthogonal polygons. The results of the experiment on the approaches are eventually compared with respect to a quality measure of the complexity and continuity of linear features. The intent of this paper is to determine whether the suggested approaches are suitable and desirable for the automated schematization of maps.

The quality measure incorporates the notion of complexity and colinearity. Let P be a simple orthogonal polygon of n vertices. Its complexity is measured as the number of edges in the solution, denoted E. The colinearity of P is measured as the number of lines that the edges coincide with, denoted L. Note that L is equal to the sum of number of distinct x and y-coordinates of the vertices of P. Parameter $w \in [0, 1]$ denotes the relative importance of colinearity with respect to complexity. Quality measure Q is defined as follows:

$Q(P) = w \cdot L + (1 - w) \cdot E$

In addition to the quality measure, a (dis)similarity measure (e.g. symmetric difference, Fréchet distance) and a corresponding threshold are introduced that represent the degree of schematization of the solutions. A higher degree of schematization may allow for better solutions with respect to the quality measure, but may diminish resemblance to the original polygon. The (dis)similarity measure, its threshold and w constitute the parameters to each approach.

The first method approximates a polygon with the union of a set of orthogonal rectangles. Intuitively, this approach promotes continuity of line features in that the set difference of any output rectangle and the other output rectangles is expected to contain colinear lines. The (dis)similarity measure and its threshold represent the permitted degree of schematization, or in other words the permitted error with respect to the input polygon.

The second method is based on the notion of an alignment operator, which takes two parallel edges and shifts them along their perpendicular axis until the edges are colinear. This approach is expected to directly improve continuity by making edges colinear.

2.3 Research Questions

1. For each approach, how do variations in the (dis)similarity threshold and w influence the performance of the approach on orthogonal territorial outlines with respect to the quality measure?
2. Which approach yields better results on orthogonal territorial outlines with respect to quality measure?

If time permits, an optional hypothesis may be considered:

A. The proposed colinearity-based approaches result in schematizations that humans consider to have improved visual complexity.

2.4 Research Methodology

At first, the visualization and quality measure are implemented to provide a structure for testing. Then the approaches are to be fully designed and implemented, along with a visual interface for the experiments.

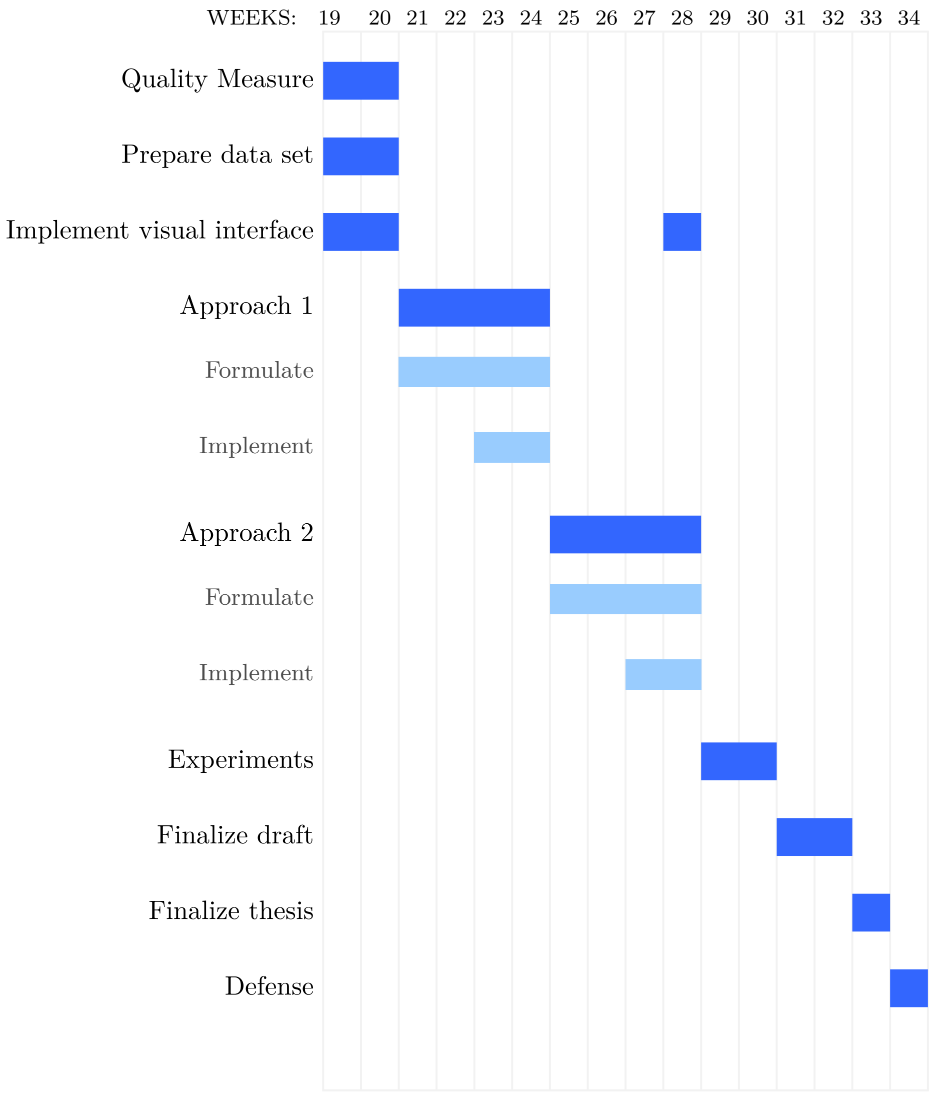
The implementations will be tested against a test suite consisting of a set of territorial outlines. One such test suite was made available, courtesy of the Applied Geometric Algorithms group of Eindhoven University of Technology. The territorial outlines in the test suite must be preprocessed into orthogonal polygons.

Different configurations for the parameters will be considered and tested on both approaches. In the case of non-determinism, each configuration is to be tested multiple times. The mean value and standard deviation will be computed, along with visualizations of the experiments.

In case extra time remains, Hypothesis A. may be tested by conducting a user study amongst cartography experts.

Future works may consider polygons that are not orthogonal and independently oriented rectangles and lines, which may include parallelity in the quality measure.

3 Planning


WEEKS:
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
Quality Measure
Prepare data set
Implement visual interface
Approach 1
Formulate
Implement
Approach 2
Formulate
Implement
Experiments
Finalize draft
Finalize thesis
Defense

References

[1] P. K. Agarwal and C. M. Procopiuc. “Exact and Approximation Algorithms for Clustering”. In: Algorithmica 33.2 (June 2002), pp. 201–226.

[2] P. K. Agarwal, E. Flato, and D. Halperin. “Polygon decomposition for efficient construction of minkowski sums”. In: Algorithms - ESA 2000. Ed. by M. S. Paterson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 20–31.

[3] A. Aggarwal, S. Ghosh, and R. Shyamasundar. “Computational complexity of restricted polygon decompositions”. In: Computational morphology. Ed. by G. T. Toussaint. Vol. 6. Machine intelligence and pattern recognition. North-Holland, 1988, pp. 1–11.

[4] L. J. Aupperle. “Covering regions by squares”. In: Master’s Thesis, University of Saskatchewan, Saskatoon, Canada (1987).

[5] L. Aupperle et al. Covering orthogonal polygons with squares. Baltimore, Md.: Johns Hopkins University, Department of Computer Science, 1988.

[6] R. Bar-Yehuda and E. Ben-Hanoch. “A linear-time algorithm for covering simple polygons with similar rectangles”. In: International Journal of Computational Geometry & Applications 06.1 (Mar. 1, 1996), pp. 79–102.

[7] B. G. Batchelor. “Using concavity trees for shape description”. In: IEE Journal on Computers and Digital Techniques 2.4 (Aug. 1, 1979), pp. 157–168.

[8] J. A. Bennell and J. F. Oliveira. “A tutorial in irregular shape packing problems”. In: Journal of the Operational Research Society 60 (sup1 May 2009), S93–S105.

[9] S. Bereg et al. “Optimizing squares covering a set of points”. In: Theoretical Computer Science 729 (June 12, 2018), pp. 68–83.

[10] M. d. Berg et al. Computational Geometry: Algorithms and Applications. 3rd ed. Berlin Heidelberg: Springer-Verlag, 2008.

[11] P. Berman and B. DasGupta. “Approximating the rectilinear polygon cover problems”. In: Proceedings of the 4th Canadian Conference on Computational Geometry. 1992, pp. 229–235.

[12] S. Bespamyatnikh and M. Segal. “Covering a set of points by two axis-parallel boxes”. In: Information Processing Letters 75.3 (Aug. 31, 2000), pp. 95–100.

[13] K. Buchin, W. Meulemans, and B. Speckmann. “A new method for subdivision simplification with applications to urban-area generalization”. In: 19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (ACM GIS). 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2011). Association for Computing Machinery, Inc, 2011, pp. 261–270.

[14] R. A. de By and O. Huisman. Principles of geographic information systems: an introductory textbook. Enschede: The International Institute for Geo-Information Science and Earth Observation (ITC), 2009.

[15] B. Chazelle. “Triangulating a simple polygon in linear time”. In: Discrete & Computational Geometry 6.3 (Sept. 1, 1991), pp. 485–524.

[16] B. Chazelle and D. Dobkin. “Decomposing a polygon into its convex parts”. In: Proceedings of the eleventh annual ACM symposium on Theory of computing. STOC ’79. Atlanta, Georgia, USA: Association for Computing Machinery, Apr. 30, 1979, pp. 38–48.

[17] B. Chazelle and D. P. Dobkin. “Optimal Convex Decompositions”. In: Machine Intelligence and Pattern Recognition. Ed. by G. T. Toussaint. Vol. 2. Computational Geometry. North-Holland, Jan. 1, 1985, pp. 63–133.

[18] V. Chvátal. “A combinatorial theorem in plane geometry”. In: Journal of Combinatorial Theory, Series B 18.1 (Feb. 1, 1975), pp. 39–41.

[19] R. Comer, E. Gould, and A. Furnham. Psychology. Wiley, 2013.

[20] H. E. Conn and J. O’Rourke. “Minimum weight quadrilaterization in O(n^3 log n) time”. In: Proc. 28th allerton conf. Commun. Control comput. Oct. 1990, pp. 788–797.

[21] T. H. Cormen et al. Introduction to algorithms. 3rd. Cambridge, Mass.: MIT press, 2009.

[22] J. Culberson and R. Reckhow. “Covering polygons is hard”. In: [Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science. [Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science. Oct. 1988, pp. 601–611.

[23] S. Das, P. P. Goswami, and S. C. Nandy. “Smallest k-point enclosing rectangle and square of arbitrary orientation”. In: Information Processing Letters 94.6 (June 30, 2005), pp. 259–266.

[24] V. J. Dielissen and A. Kaldewaij. “Rectangular partition is polynomial in two dimensions but NP-complete in three”. In: Information Processing Letters 38.1 (Apr. 12, 1991), pp. 1–6.

[25] T. v. Dijk et al. “Map schematization with circular arcs”. In: 8th International Conference on Geographic Information Science (GIScience). conference; Eighth International Conference on Geographic Information Science. Springer, 2014, pp. 1–17.

[26] D. Ding et al. “A tool-path generation strategy for wire and arc additive manufacturing”. In: The International Journal of Advanced Manufacturing Technology 73.1 (July 2014), pp. 173–183.

[27] Z. Drezner. “On the rectangular p-center problem”. In: Naval Research Logistics (NRL) 34.2 (1987), pp. 229–234.

[28] S. Eidenbenz and P. Widmayer. “An Approximation Algorithm for Minimum Convex Cover with Logarithmic Performance Guarantee”. In: Algorithms — ESA 2001. Ed. by F. M. auf der Heide. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001, pp. 333–344.

[29] H.-Y. Feng and T. Pavlidis. “Decomposition of Polygons into Simpler Components: Feature Generation for Syntactic Pattern Recognition”. In: IEEE Transactions on Computers C-24.6 (June 1975), pp. 636–650.

[30] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. “Optimal packing and covering in the plane are NP-complete”. In: Information Processing Letters 12.3 (June 13, 1981), pp. 133–137.

[31] D. S. Franzblau and D. J. Kleitman. “An algorithm for covering polygons with rectangles”. In: Information and Control 63.3 (Dec. 1, 1984), pp. 164–189.

[32] G. N. Frederickson and D. B. Johnson. “Generalized Selection and Ranking: Sorted Matrices”. In: SIAM Journal on Computing 13.1 (Feb. 1, 1984), pp. 14–30.

[33] M. R. Garey et al. “Triangulating a simple polygon”. In: Information Processing Letters 7.4 (June 1, 1978), pp. 175–179.

[34] F. Gavril. “Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independent Set of a Chordal Graph”. In: SIAM Journal on Computing 1.2 (June 1, 1972), pp. 180–187.

[35] D. H. Greene. “The decomposition of polygons into convex parts”. In: Computational Geometry 1 (1983), pp. 235–259.

[36] S. Hertel and K. Mehlhorn. “Fast triangulation of the plane with respect to simple polygons”. In: Information and Control. International Conference on Foundations of Computation Theory 64.1 (Jan. 1, 1985), pp. 52–76.

[37] M. Hoffmann. “A simple linear algorithm for computing rectilinear 3-centers”. In: Computational Geometry. 11th Canadian Conference on Computational Geometry 31.3 (June 1, 2005), pp. 150–165.

[38] E. Huang and R. E. Korf. “Optimal Rectangle Packing: An Absolute Placement Approach”. In: Journal of Artificial Intelligence Research 46 (Jan. 23, 2013), pp. 47–87.

[39] H. Imai and T. Asano. “Efficient Algorithms for Geometric Graph Search Problems”. In: SIAM J. Comput. (1986).

[40] J. Jaromczyk and M. Kowaluk. “Orientation independent covering of point sets in R2 with pairs of rectangles or optimal squares”. In: European workshop on comp. Geometry. 1996, pp. 54–61.

[41] D. S. Johnson. “Approximation algorithms for combinatorial problems”. In: Journal of Computer and System Sciences 9.3 (Dec. 1974), pp. 256–278.

[42] B. A. Júnior et al. “Dealing with Nonregular Shapes Packing”. In: Mathematical Problems in Engineering 2014 (2014), pp. 1–10.

[43] J. Kahn, M. Klawe, and D. Kleitman. “Traditional Galleries Require Fewer Watchmen”. In: SIAM Journal on Algebraic Discrete Methods 4.2 (June 1, 1983), pp. 194–206.

[44] M. J. Katz, K. Kedem, and M. Segal. “Discrete rectilinear 2-center problems”. In: Computational Geometry 15.4 (Apr. 1, 2000), pp. 203–214.

[45] J. M. Keil. “Decomposing polygons into simpler components”. PhD thesis. Toronto: University of Toronto. Department of Computer Science, 1983. 89 pp.

[46] J. M. Keil. “Decomposing a Polygon into Simpler Components”. In: SIAM Journal on Computing 14.4 (Nov. 1, 1985), pp. 799–817.

[47] J. M. Keil and J.-R. Sack. “Minimum Decompositions of Polygonal Objects”. In: Machine Intelligence and Pattern Recognition. Ed. by G. T. Toussaint. Vol. 2. Computational Geometry. North-Holland, Jan. 1, 1985, pp. 197–216.

[48] M. Keil. “Polygon Decomposition”. In: Handbook of Computational Geometry. Elsevier, 2000, pp. 491–518.

[49] M. Keil and J. Snoeyink. “On the Time Bound for Convex Decomposition of Simple Polygons”. In: International Journal of Computational Geometry & Applications 12.3 (June 2002), pp. 181–192.

[50] S.-S. Kim, S. W. Bae, and H.-K. Ahn. “Covering a point set by two disjoint rectangles”. In: International Journal of Computational Geometry &amp; Applications (Nov. 20, 2011).

[51] K. Koffka. Principles Of Gestalt Psychology. Routledge, Oct. 8, 2013. 733 pp.

[52] R. Korf. “Optimal Rectangle Packing: New Results.” In: Proceedings of the 14th International Conference on Automated Planning and Scheduling, ICAPS 2004. Jan. 1, 2004, pp. 142–149.

[53] R. E. Korf. “Optimal Rectangle Packing: Initial Results”. In: ICAPS. 2003.

[54] V. S. A. Kumar and H. Ramesh. “Covering Rectilinear Polygons with Axis-Parallel Rectangles”. In: SIAM Journal on Computing 32.6 (Jan. 2003), pp. 1509–1541.

[55] S. M. LaValle. Planning Algorithms. Cambridge: Cambridge University Press, 2006.

[56] C. Levcopoulos and J. Gudmundsson. “A Linear-Time Approximation Algorithm for Minimum Rectangular Covering”. In: (Oct. 13, 2001).

[57] C. Levcopoulos and A. Lingas. “Bounds on the length of convex partitions of polygons”. In: Foundations of Software Technology and Theoretical Computer Science. Ed. by M. Joseph and R. Shyamasundar. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1984, pp. 279–295.

[58] D. Lichtenstein. “Planar Formulae and Their Uses”. In: SIAM Journal on Computing 11.2 (May 1, 1982), pp. 329–343.

[59] A. Lingas. “The power of non-rectilinear holes”. In: Automata, Languages and Programming. Ed. by M. Nielsen and E. M. Schmidt. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1982, pp. 369–383.

[60] A. Lingas, C. Levcopoulos, and J. Sack. “Algorithms for minimum length partitions of polygons”. In: BIT Numerical Mathematics 27.4 (Dec. 1, 1987), pp. 474–479.

[61] A. Lingas et al. “Minimum edge length partitioning of rectilinear polygons”. In: Proc. 20th Allerton Conf. Commun. Control Comput. 1982, pp. 53–63.

[62] W. T. Liou, J. J. Tan, and R. C. Lee. “Minimum partitioning simple rectilinear polygons in O(n log log n) - time”. In: Proceedings of the fifth annual symposium on Computational geometry. SCG ’89. Saarbruchen, West Germany: Association for Computing Machinery, June 5, 1989, pp. 344–353.

[63] W. Liou, J.-M. Tan, and R. Lee. “Minimum rectangular partition problem for simple rectilinear polygons”. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 9.7 (July 1990), pp. 720–733.

[64] W. Lipski Jr. “Finding a manhattan path and related problems”. In: Networks 13.3 (1983), pp. 399–409.

[65] W. Lipski. “An O(n log n) Manhattan path algorithm”. In: Information Processing Letters 19.2 (Aug. 31, 1984), pp. 99–102.

[66] W. Lipski Jr et al. “On two-dimensional data organization II”. In: Fundamenta Informaticae 2 (1979), pp. 245–260.

[67] A. Lubiw. “Decomposing polygonal regions into convex quadrilaterals”. In: Proceedings of the first annual symposium on Computational geometry. SCG ’85. Baltimore, Maryland, USA: Association for Computing Machinery, June 1, 1985, pp. 97–106.

[68] W. J. Masek. “Some NP-complete set cover problems”. In: Unpublished manuscript, MIT Laboratory for Computer Science (1979).

[69] A. A. Melkman. “On-line construction of the convex hull of a simple polyline”. In: Information Processing Letters 25.1 (Apr. 1987), pp. 11–12.

[70] W. Meulemans. “Similarity measures and algorithms for cartographic schematization”. PhD thesis. 2014.

[71] J. O’Rourke and K. Supowit. “Some NP-hard polygon decomposition problems”. In: IEEE Transactions on Information Theory 29.2 (Sept. 1, 2006), pp. 181–190.

[72] J. O’Rourke et al. “An optimal algorithm for finding minimal enclosing triangles”. In: Journal of Algorithms 7.2 (June 1, 1986), pp. 258–269.

[73] J. O’Rourke, S. Suri, and C. D. Tóth. “Polygons”. In: Handbook of Discrete and Computational Geometry. Nov. 22, 2017.

[74] D. A. Plaisted and J. Hong. “A heuristic triangulation algorithm”. In: Journal of Algorithms 8.3 (Sept. 1, 1987), pp. 405–437.

[75] F. P. Preparata and M. I. Shamos. Computational Geometry. New York, NY: Springer New York, 1985.

[76] A. Reimer and W. Meulemans. “Parallelity in chorematic territorial outlines”. In: Proceedings 14th ICA Workshop on Generalisation and Multiple Representation. 14th ICA/ISPRS Workshop on Generalisation and Multiple Representation. 2011, pp. 1–12.

[77] J. Sack. “An O(n log n) Algorithm for Decomposing Simple Rectilinear Polygons into Convex Quadrilaterals”. In: Proceedings of the 20th allerton conference on communication, control, and computing, monticello. 1982, pp. 64–74.

[78] Scott et al. “TID - a translation invariant data structure for storing images”. In: Communications of the ACM. May, 1986. vol. 29: pp. 418-429 : some ill. includes bibliography 29 (Apr. 1986).

[79] M. Sharir. “A Near-Linear Algorithm for the Planar 2-Center Problem”. In: Discrete & Computational Geometry 18.2 (Sept. 1, 1997), pp. 125–134.

[80] M. Sharir and E. Welzl. “Rectilinear and polygonal p-piercing and p-center problems”. In: Proceedings of the twelfth annual symposium on Computational geometry - SCG ’96. the twelfth annual symposium. Philadelphia, Pennsylvania, United States: ACM Press, 1996, pp. 122–132.

[81] V. Soltan and A. Gorpinevich. “Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles”. In: Discrete & Computational Geometry 9.1 (Jan. 1, 1993), pp. 57–79.

[82] R. E. Tarjan and C. J. Van Wyk. “An O(n log log n)-Time Algorithm for Triangulating a Simple Polygon”. In: SIAM Journal on Computing 17.1 (Feb. 1, 1988), pp. 143–178.

[83] G. Toussaint. “Solving Geometric Problems with the Rotating Calipers”. In: In Proceedings of IEEE MELECON’83 (1983), p. 8.

[84] L. Wan et al. “Regularization of neural networks using dropconnect”. In: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28. ICML’13. Atlanta, GA, USA: JMLR.org, June 16, 2013, pp. III–1058–III–1066.

[85] E. Welzl. “Smallest enclosing disks (balls and ellipsoids)”. In: New Results and New Trends in Computer Science. Ed. by H. Maurer. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1991, pp. 359–370.

[86] A. Y. Wu, S. K. Bhaskar, and A. Rosenfeld. “Computation of geometric properties from the medial axis transform in O(n log n) time”. In: Computer Vision, Graphics, and Image Processing 34.1 (Apr. 1, 1986), pp. 76–92.