DECOMPOSING AND PACKING POLYGONS

Dania El-Khechen

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy (Computer Science)

Concordia University

Montréal, Québec, Canada

April 2009

Your file Votre référence
ISBN: 978-0-494-63446-2
Our file Notre référence
ISBN: 978-0-494-63446-2

Abstract

Decomposing and packing polygons

Dania El-Khechen, Ph.D.

Concordia University, 2009

In this thesis, we study three different problems in the field of computational geometry: the partitioning of a simple polygon into two congruent components, the partitioning of squares and rectangles into equal area components while minimizing the perimeter of the cuts, and the packing of the maximum number of squares in an orthogonal polygon.

To solve the first problem, we present three polynomial time algorithms which given a simple polygon P partitions it, if possible, into two congruent and possibly nonsimple components P_1 and P_2: an $O(n^2 \log n)$ time algorithm for properly congruent components and an $O(n^3)$ time algorithm for mirror congruent components.

In our analysis of the second problem, we experimentally find new bounds on the optimal partitions of squares and rectangles into equal area components. The visualization of the best determined solutions allows us to conjecture some characteristics of a class of optimal solutions.

Finally, for the third problem, we present three linear time algorithms for packing the maximum number of unit squares in three subclasses of orthogonal polygons: the staircase polygons, the pyramids and Manhattan skyline polygons. We also study a special case of the problem where the given orthogonal polygon has vertices with integer coordinates and the squares to pack are (2×2) squares. We model the latter problem with a binary integer program and we develop a system that produces and visualizes optimal solutions. The observation of such solutions aided us in proving some characteristics of a class of optimal solutions.

Acknowledgements

This work would not have been possible without my supervisors Thomas Fevens and John Iacono. I thank them for their constant encouragement, motivation and advices. I also thank them for giving me the great opportunity to travel and attend many conferences and workshops

I thank all my co-authors. It has been a pleasure to work with every one of them. In particular, I thank Godfried Toussaint for giving me the opportunity to attend his wonderful annual workshop on Computational Geometry in Barbados: an occasion to work on challenging problems, meet great researchers and enjoy the beach during Montréal's cold winter. I thank all the Mcgill lunch group with whom I enjoyed lunch from time to time. I thank all the researchers with whom I worked on the interdisciplinary project with the faculty of fine arts, professors: Cheryl Dudek, Thomas Fevens, Sudhir Mudur, Lydia Sharman and Fred Szabo. I also thank Ramgopal Rajagopalan and Eric Hortop with whom it was a lot of fun to drink coffee and discuss symmetry groups.

I thank Günter Rote for his unpublished manuscript which inspired the material in Chapter 4. I thank Ken Brakke for his software Surface Evolver that we used in Chapter 5. I also thank Tobias Achterberg for his solver SCIP that aided us for Chapter 6 results.

I thank the graduate program advisor Halina Monkiewicz and the office assistant Hirut Adugna for always answering my numerous questions with a smile. I thank the teaching assistants coordinator Pauline Dubois who gave me the priceless opportunity to teach.

I thank Vašek Chvátal for lending me so many (excellent) books and movies, making me discover many (good) restaurants in Montréal, introducing me to so many amazing people and transmitting a great enthusiasm for Mathematics and a great joy of life. I also thank him for his entertaining classes and his (crystal clear) way of transmitting information.

I thank all my friends for their constant support. Fatmé el-Moukaddem for discussing our research problems, reading my nagging over MSN and for or never-ending-after-defence plans, Simon Kouyoumdjian for helping me recover most of the material in Chapter 5 after my hard disk crashed and for sharing many precious moments, and Alessandro Zanarini, with

his incredible sweetness, for many useful Mathematical discussions. I thank my long distance friends: Nisrine Jaafar for not only helping me get through the first year in Montréal but for turning it into a wonderful one, Malak Jalloul for her unlimited phone calls plan to Canada, Abir Baz and Rouba Choueiry for continuously yeling "yalla, khalssina!", Alaa Abi-Haidar for our great exchanges, Narjess Fathalla, Mayssan Maarouf, Ali Mourad, Houssam Nassif and Rima Sleiman for many many reasons. I thank my Montréal friends: Khaled AbdelHay for our long studying sessions, Ruddy Avalos for his super parties, Tamara Diaz (with her unique laugh) and Duhamel Xolot for the great overnight discussions we had, François Grandchamp for his Québécois lessons, John Alexander Lopez for the many things he taught me, Mahitab Seddik and Rania Khattab for always listening. I also thank Chloé Guillaume, Layla Hussain, Bassem Hussami, Marie-André L'espérance, Daria Madjidian and George Peristerakis for their continuous attention. I thank my two dear and constantly-traveling friends Lama Kabbanji and Hicham Safieddine for being there even when they are not. Finally, without my friend JJ, the thesis journey would have been less fun and much harder.

I thank all my dance and literature teachers who made my life richer. In particular, I thank my first and current bellydancing teachers Sheila Ribeiro and Any Massicotte for being an inspiration. I also thank my sweet "bellysisters" Wendy Corner and Ruth Gover.

I thank my family. The Atwi family: my uncles Wajih, Said, Bassam, Bassel and Houssam (who accompanied me here the first month) and my aunt Samia (who supported me in my first years in Montréal). I thank all my cousins! In particular, I thank Douaa, Mayssa, Mohamad, Mostafa and Ahmad for being the siblings I never had. Their mother Safaa Serhan is a precious gift to all of us. I thank also my caring uncle Ali El-Khechen.

Je remercie Nikolaj van Omme pour tout ce qu'il m'a appris sur la programation mathémathiques et pour les discussions enrichissantes. Je le remercie d'être si patient et attentioné. Je le remercie d'avoir une passion contagieuse pour toutes les choses de la vie. Je le remercie aussi de m'avoir présenté son père Albert Carton, un homme extraordinaire. Merci mon chanteur préféré.

I dedicate this thesis to Hind Atwi. A brilliant woman. A great militant. A silent inspiration. C'est grâce à elle si je suis devenue qui je suis. Merci Mama.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Consider the following questions:

- **Cowhide cutting.** Cowhide is used in car industry to make car seat cushions, car flooring and other car parts. A ready-to-cut hide can have local damage. How can the undamaged parts of the hide be cut to minimize the wastage [262]?
- **Flexible circuit layout.** Suppose you are a VLSI layout designer. You want to place p functionally identical circuits on a rectangular chip of area A. Thin rectangles are not desirable since they lead to long wire length. How do you design your board [180]?
- **Collision detection.** You have a set of objects that are represented with geometric models. Collision detection aims at detecting a geometric contact between these objects. How are collisions detected efficiently?
- **Terrain covering.** Suppose you have a set of robots that need to explore a terrain. Each part of the terrain should be visited by one robot. The relative capabilities of the robots are determined based on the area of the terrain they can cover. How do you divide your terrain among the robots [153]?
- **Art gallery guarding.** You are the owner of an art gallery with valuable paintings. You would like to place guards (cameras) so that each point in your gallery is visible to

some guard (camera) and you have budget constraints. How do you find the minimum number of guards (cameras) needed [83]?

Consider also the following (less serious) questions:

- **Ice cream search.** Suppose you are walking around Montreal and you feel an urgent need for an ice cream. There are many ice cream parlours around the city but you would like to go to the nearest one. How do you know which one is the nearest?
- **Animals and fence.** Imagine you have p animals and you have a piece of land that you would like to divide equally among the animals without wasting too much fence material. How can you achieve such a partition?
- **Cake cutting.** Imagine you have a cake (of arbitrary shape) and two kids. No matter how you try, the kids are convinced that unless you give them the exactly-same-shape pieces, the division will not be fair. How do you partition the cake in two similar shapes? Can you always do it?
- **Board filling.** You have a kid to distract. You give her a board and a (very) large number of identical square pieces and you ask her to fill the board with the maximum number of squares. How fast can the kid do it? Are you asking her a too difficult question?

The nine questions above are the sort of problems posed to (or by) a computational geometer. What is *computational geometry*? Computational geometry is a discipline of the field of algorithms (design and analysis) and data structures which involves studying problems of a geometric nature by analysing their computational complexity and developing algorithms and data structures to solve them. The foundations of what is called "Computational Geometry" nowadays were laid in the late 1970s by M.I. Shamos in his Ph.D. thesis [261,263].

Let us look again at the nine questions posed and formulate them in computational geometric terms:

- **Cowhide cutting.** The cowhide can be approximated as a two-dimensional planar figure with straight line edges, call it polygon P. The damage can be considered to

be "islands" inside the polygon boundary which we call a set of H holes. The cutting equipment are objects of some shapes of which only the boundary is of interest to us, call them a set S_{CE} of k polygons $S = \{CE_1, CE_2, \ldots, CE_k\}$. The problem is: what is the maximum number of CE_i that can be packed in $P - H$ where $1 \leq i \leq k$ and i can be duplicated?

- **Flexible circuit layout.** The circuit board can be represented as a rectangle R that needs to be decomposed into a set of p rectangles $S = \{r_1, r_2, \ldots, r_p\}$ such that $R = \bigcup_{r \in S_r} r$ and that the maximum rectangle perimeter is minimized or the sum of the perimeters is minimized [180].

- **Collision detection.** Polygons where all interior angles between edges are less than 180° are called convex. Given two polygons P and Q for which we want to detect collision, how can it be detect efficiently given that the intersection of two convex polygons is faster to compute than the intersection of arbitrary ones?

- **Terrain covering.** The terrain can be modelled by its two-dimensional projection, a polygon P [153]. Let $Area(P)$ denote the area of P. The proportions of the area that each robot should be assigned are represented by a set of values c_i, $i = 1, 2, \ldots, p$ with $0 < c_i < 1$ and $\sum_{i=1}^{p} c_i = 1$. The problem is: given P and p, partition it into p nonoverlapping regions $P_1, \ldots, P_p$ such that $Area(P_i) = c_i Area(P)$ [153].

- **Art gallery guarding.** Consider the floor plan of the art gallery to be a polygon P. The problem can be formulated as follows: decompose P into the minimum number of star-shaped components. The number of guards is equal to the minimum number of components assuming that the guards can see 360° [83].

- **Ice Cream Search.** Consider the city of Montreal to be modelled as a polygon P with a set of q points $M = \{m_1, m_2, \ldots, m_q\}$ inside P representing the ice cream parlour locations. Partition P into q regions such that the region of ice cream parlour m_i consists of all points that are closer to this parlour than they are to any other in the city (all points p_j in P such that $d(p_j, m_i) < d(p_j, m_k)$ for all $m_k \in M$, $k \neq i$).

- **Animals and fence.** Consider the piece of land to be a polygon P. The problem is: partition P into p equal area components such that the perimeter of the cuts is minimized.

- **Cake cutting.** Two polygons are congruent if they are equivalent up to an isometry. Consider the cake to be a polygon P. The problem is: can P be partitioned into two congruent components?

- **Board filling.** The board can be modelled with a polygon P. The problem is: what is the complexity of computing the maximum number of identical squares that can be packed (without overlap) into P?

The problems posed above belong to two areas of computational geometry: geometric object decomposition and geometric object packing. Geometric object decomposition involves decomposing a general geometric object into simpler components. The decomposition can be the goal of the algorithm but is often an intermediate or a preprocessing step. Fast existing algorithms are applied to the simpler components and the partial solutions are then combined to obtain a general one. The collision detection problem is an example where intersection is detected between convex components before reporting a collision. Also, Hert and Lumelsky assume a decomposition of the polygon into convex components prior to solving the terrain-covering problem [153]. There are two major kinds of decomposition: partition and covering. Covering allows components to overlap while partition requires them to be disjoint. Informally, packing can be defined as placing a given set (or subset) of objects in some containers. The goal is either to pack everything (all the given objects) in the best container or pack the best subset possible of objects. The cowhide cutting and the board filling problems are both of the latter sort. Both the decomposition and the packing problems have been extensively studied in the literature and yet many variants of the problems remain open. The existence of a huge literature on these types of problems can be informally explained by the fact that are many ways in which we can decompose (pack) an object and there many types of objects to decompose (pack into). The study of decomposition and packing problems is the subject of this thesis. Our focus is on problems in

the plane; our objects to decompose, our components, our objects to pack and our containers are all two-dimensional polygons.

In this thesis, we make the following contributions:

1. We design three polynomial time algorithms for partitioning a simple polygon P with n vertices into two congruent and possibly nonsimple components P_1 and P_2: an $O(n^2 \log n)$ time algorithm for properly congruent components (equivalent up to translation and rotation) and an $O(n^3)$ time algorithm for mirror congruent components (equivalent up to reflection and glide reflection). The previous algorithms, which solve the problem, output simple components P_1 and P_2. However, the reported running time is erroneous and is not even polynomial for the case of mirror congruent components; our proposed algorithm is the first to provide a polynomial time algorithm for this latter case.

2. We experimentally find new bounds on the optimal solutions for partitioning squares and rectangles into k equal area components while minimizing the perimeter of the cuts. Allowing straight line and sections of circular arcs cuts, we present the best determined solutions for partitioning the unit square and the 1×2 rectangle into k components where $3 \leq k \leq 10$ and $3 \leq k \leq 6$ respectively. We conjecture some characteristics of a class of optimal solutions. Most of the previous results use only straight line cuts and the bounds for circular cuts are only known for $k = 3$ and for $k = 4$.

3. We present three linear time algorithms for packing the maximum number of unit squares in three subclasses of orthogonal polygons: the staircase polygons, the pyramids and Manhattan skyline polygons. We also study a special case of the problem where the given orthogonal polygon has vertices with integer coordinates and the squares to pack are (2×2) squares. We model the latter problem with a binary integer program and we develop a system that produces and visualizes optimal solutions. We prove some characteristics of a class of optimal solutions. Our results support the long standing conjecture that the problem of packing the maximum number of squares in a

general orthogonal polygon is polynomial.

In the next chapter, we define the terms and the notation that are needed for the rest of this thesis. The terms and the notation that are chapter-specific are defined In the corresponding chapter.In Chapter 3, we review the work done in the area of polygon decomposition and polygon packing. At the beginning of every chapter, we review in detail the previous work related to that chapter. In Chapter 4, we present two polynomial time algorithms for partitioning a polygon, if possible, into two properly and mirror congruent components. In Chapter 5, we present experimental work and conjectures on partitioning squares and rectangles into equal area components while minimizing the perimeter of the cuts. In Chapter 6, we present three polynomial time algorithms for packing the maximum number of unit squares in three subclasses of orthogonal polygons and we prove some characteristics of the optimal solution for packing (2×2) squares in general grid orthogonal polygons. Each chapter is concluded with a summary of the contributions and future work.

Chapter 2

Background information and notation

2.1 Polygon definitions

A *polygon* P is defined as a closed plane figure bounded by straight line segments. In this thesis, the boundary of a polygon P will be referred to by δP. A polygon P is said to be *simply connected* or *simple* if it is not self-intersecting and it has no holes. Figure 1 shows two polygons: (a) shows a simple polygon and (b) shows a polygon with holes.

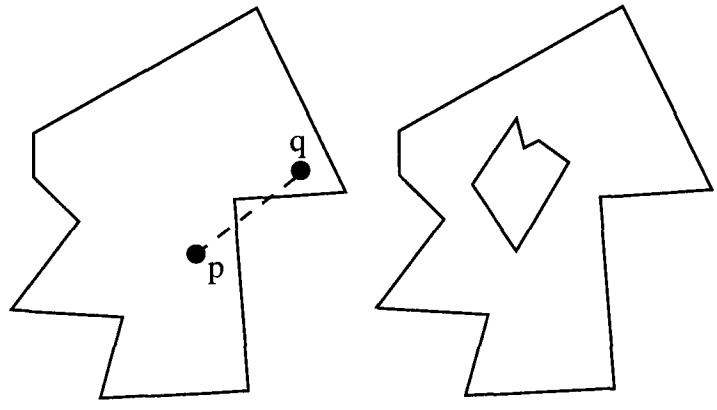

Figure 1: (a) A simple polygon, (b) a polygon with a hole.

A *polygonal chain* or a *polyline* that is a subset of δP is specified by a startpoint a and an endpoint b (not necessarily vertices). A polyline is always considered to be directed clockwise around the boundary of a simple polygon P. We denote it by $P[a \,..\, b]$.

Two points are *visible* if the line segment joining them lies entirely inside P. P is said to be *convex* if every pair of points in P are visible from each other. Note that in a nonconvex polygon, there exists at least two pairs of points in P that are not visible from each other, see points p and q in Figure 1 (a). A vertex of a polygon is said to be convex if its internal angle is less than 180° and is said to be *reflex* otherwise. A convex polygon is one that has only convex vertices (see Figure 2). In what follows, n will be used to refer to the number of vertices of a polygon and N to the number of reflex vertices of a polygon ($N < n$) unless we specify otherwise. P is *regular* if it is equiangular and all its sides are of equal length, see Figure 3 for examples.

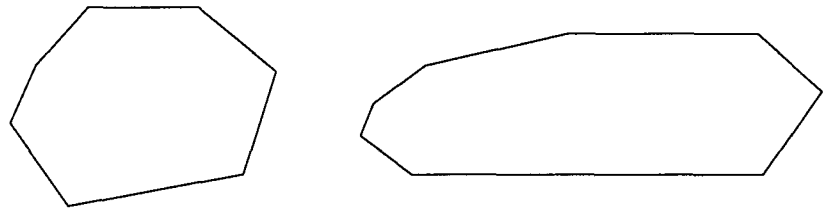

Figure 2: Two convex polygons.

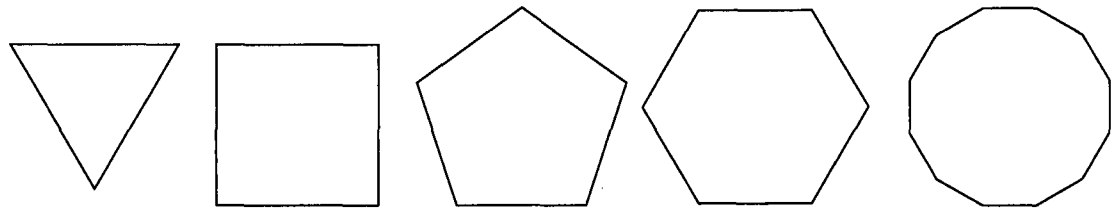

Figure 3: Regular 3-gon (equilateral triangle), 4-gon (square), 5-gon (pentagon), 6-gon (hexagon), 12-gon (dodecagon).

P is *star-shaped* if there exists at least one point $x \in P$ from which the entire polygon is visible. Figure 4 shows two star-shaped polygons. P is *spiral* if it has exactly one *concave* subchain (a chain with only reflex vertices). Figure 5 shows an example of a spiral polygon.

A polygonal chain C is said to be *monotone* with respect to a line l if the projections of the vertices of C on l occur in the same order as in C. P is *l-monotone* if there exists a

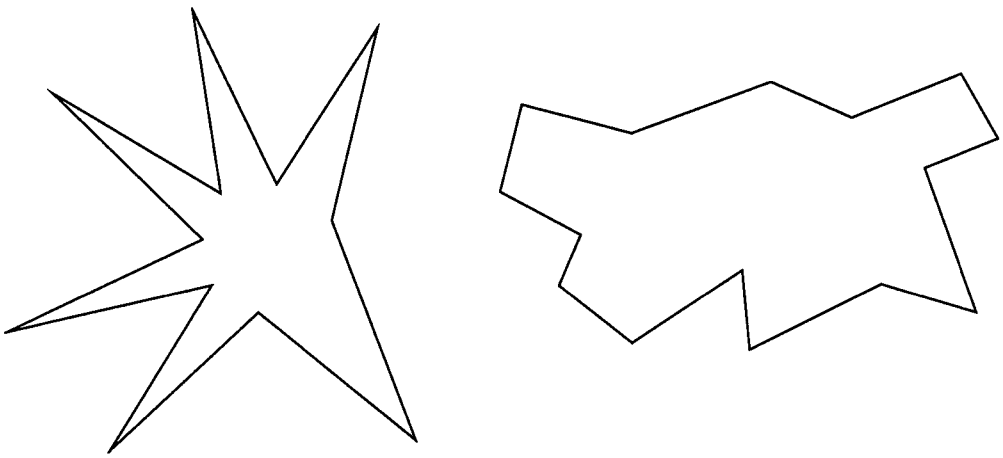

Figure 4: Two star-shaped polygons.

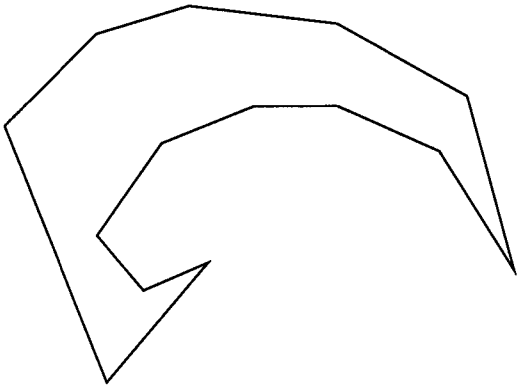

Figure 5: A spiral polygon.

line l such that δP can be partitioned into two monotone polygonal chains with respect to l. Figure 6 shows an example of a y-monotone polygon and the projection of polygonal chains $P[a\,..\,b]$ and $P[b\,..\,a]$ on two lines parallel to the y-axis.

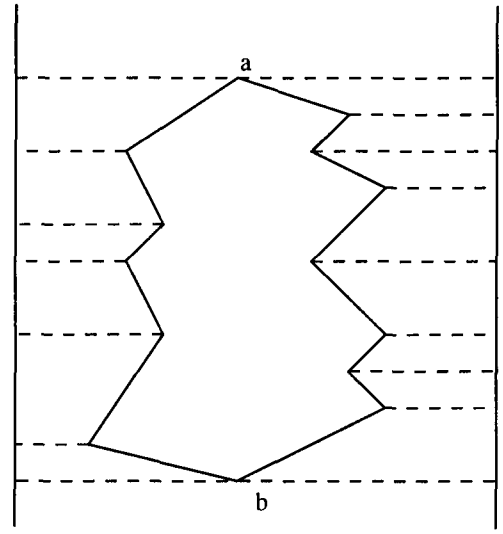

Figure 6: A monotone polygon.

P is *orthogonal* if all its edges are either horizontal or vertical. Figure 7 shows an example of an arbitrary orthogonal polygon. An orthogonal polygon P is said to be *horizontally (vertically)* convex if any horizontal (vertical) segment joining two of its vertices lies inside the polygon. A segment is called a *chord* in an orthogonal polygon P if it is interior to P and if it joins a pair of points $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ in P such that either $x_1 = x_2$ or $y_1 = y_2$. In an orthogonal polygon P, internal angles are either 90° or 270° and a *dent* is defined as an edge in which both endpoints have internal angles of 270°. The orientation of a dent is defined in terms of compass direction. If the polygon is aligned such that the north corresponds with positive y-axis and the dent is parallel to the x-axis then it is called a north dent and is referred to by N-dent. Figure 8 shows a horizontally convex polygon with a west dent.

There are several classes of orthogonal polygons defined in the literature according to the orientation of the dents. A class k orthogonal polygon contains dents of k different orientations. The class 0 is the class of orthogonally convex polygons of which an example is shown in Figure 9. A vertically or horizontally convex polygon is of class $2a$. Class $2b$ refers

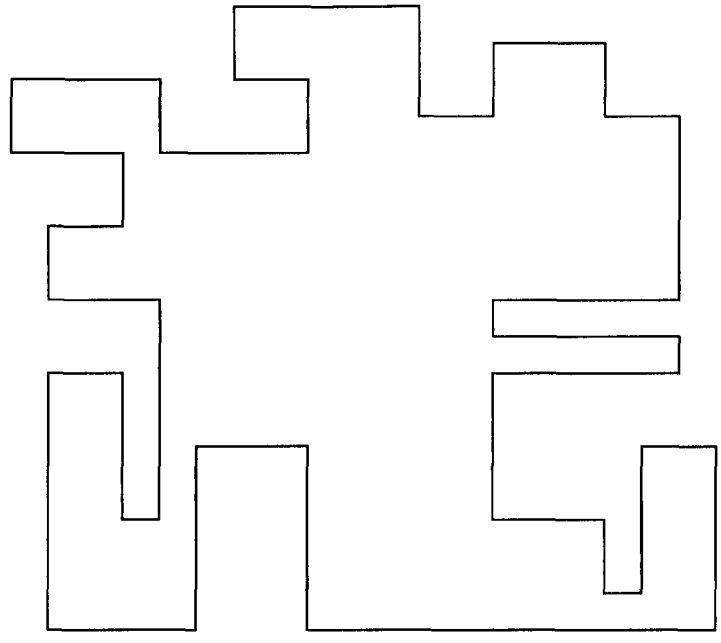

Figure 7: An orthogonal polygon.

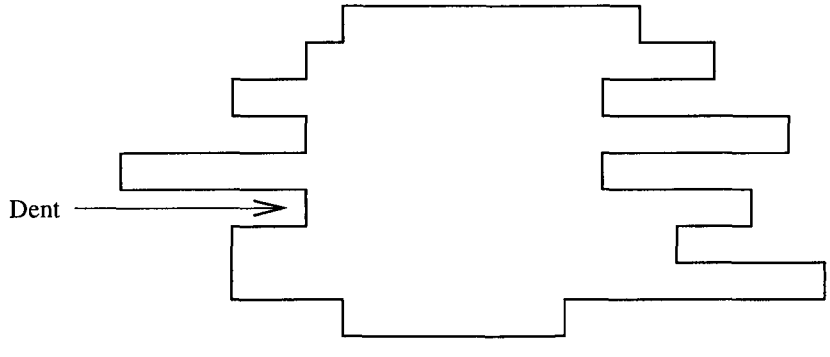

Figure 8: A horizontally convex orthogonal polygon.

to polygons that have two dent orientations orthogonal to each other. Class 4 refers to general orthogonal polygons.

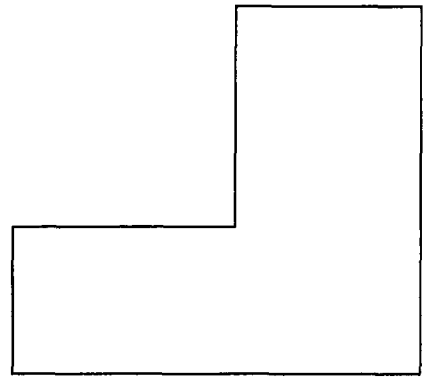

Figure 9: An orthogonally convex polygon.

For orthogonal polygons, other notions of visibility are defined. Two points of a polygon P are said to be *r-visible* if there exists a rectangle (inside P) that contains the two points. Two points of a polygon P are said to *s-visible* if there exists an orthogonally convex polygon (inside P) that contains both points. Hence, it is natural to define *r(s)-star-shaped* polygons: an $r(s)$-star-shaped polygon is an orthogonal polygon P such that there exists at least one point for which all other points in P are $r(s)$-visible. Two examples of an r-star-shaped polygon and an s-star-shaped polygon are shown in Figure 10. Two points p_1 and p_2 indirectly see each other in an orthogonal polygon P if there exists a third p_3 in P such that p_3 is s-visible to both p_1 and p_2.

The *kernel* of a polygon P is defined as the set of all points from which each point in P is visible. The *diameter* of a polygon is the diameter of the smallest enclosing circle and the *width* of a polygon is the diameter of largest inscribed circle. The *aspect ratio* of a polygon is defined as the ratio of its diameter to its width. Figure 11 shows a hexagon with its inscribed circle and circumscribed circle.

A *diagonal* of a polygon is a line segment that joins two vertices and that is interior-disjoint from δP.

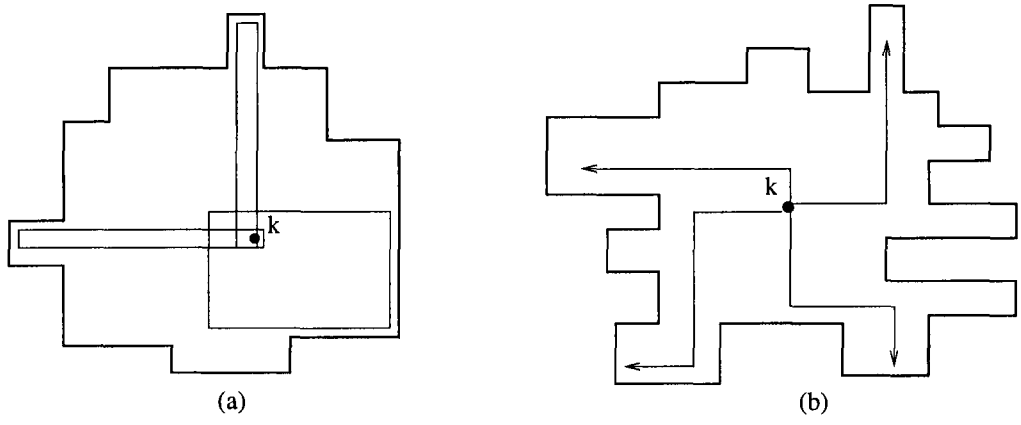

Figure 10: (a) An r-star-shaped polygon with an example kernel point, (b) an s-star-shaped polygon with an example kernel point.

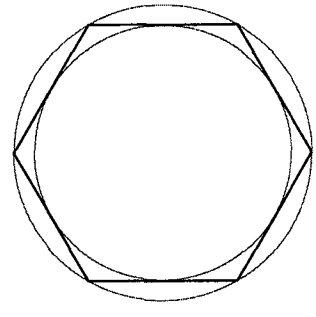

Figure 11: The smallest enclosing circle and largest inscribed circle of a regular polygon.

2.2 Graph definitions

Many problems in polygon decomposition and packing can be reduced to their dual in graph theory and hence are solved using a graph theoretic approach. Some graph theory definitions are needed in this context. A *graph* G is an ordered pair denoted by $G = (V, E)$ where V is a set of *vertices* and E is a set of *edges*. An edge is a 2-element subset of V. We say an edge is *incident* to the two vertices that define it and that are called the endpoints of the edge. The endpoints are said to be *adjacent* vertices in the graph. The degree of a vertex in a graph is the number of edges incident to that vertex, denoted by $deg(v)$ for a vertex v. The maximum degree of a graph G is the maximum degree of its vertices and is denoted by $\Delta(G)$.

A graph $H = (V', E')$ is a *subgraph* of a graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. The subgraph of G whose vertices are V' and whose edges are the edges of G that have both endpoints in V' is called an *induced subgraph* of G. A *walk* in a graph G is a finite non-null sequence of alternating vertices and edges which starts and ends at a vertex. If the edges and the vertices of a walk are distinct except for its first and last vertex which are the same, it is called a *cycle*.

An *independent* set of a graph G is a subset of its vertices such that no two vertices in the set are adjacent in G. A maximum independent set of a graph G is a largest such set in G and its size is denoted by $\alpha(G)$. The graph in Figure 12 has $\alpha(G) = 5$.

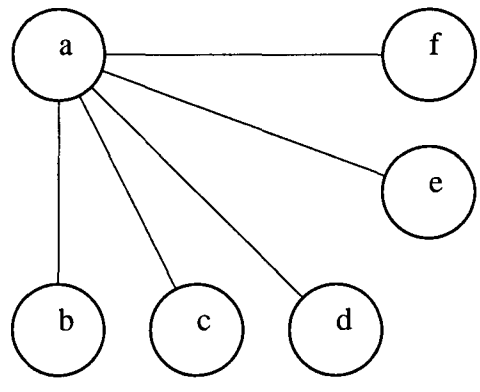

Figure 12: A graph G with $\alpha(G) = 5$.

A *clique* of a graph is a subset of vertices such that each pair in the subset is connected by an edge. A maximum clique of a graph G is a largest such set in G and is denoted by $\omega(G)$. The graph in Figure 13 has $\omega(G) = 3$.

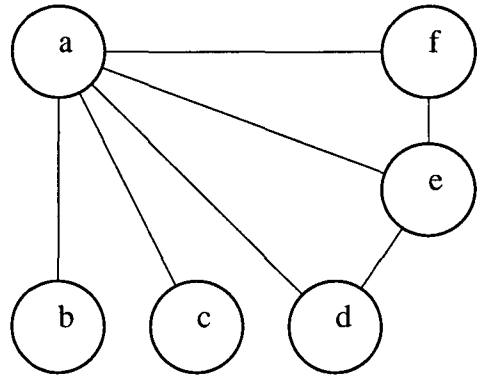

Figure 13: A graph G with $\omega(G) = 3$.

A *minimum clique cover* of a graph G is a minimum set of subsets of vertices $\{V_1, V_2, \ldots, V_k\}$ such that for every $1 \leq i \leq k$, V_i is a clique and such that for every edge $(u, v) \in E$ there is some V_i that contains both u and v. A *chord* is an edge joining two vertices that are not adjacent in a cycle. A *chordal graph* is a graph possessing no chordless cycle. The *chromatic number* of a graph is the smallest number of colours needed to color the vertices of a graph such that no two adjacent vertices share the same color; it is denoted by $\gamma(G)$. A *perfect* graph G is such that for every induced subgraph of G, the size of the largest clique equals the chromatic number ($\omega(G) = \gamma(G)$).

Given a set $S = s_1, s_2, \ldots, s_n$ of geometric objects in $\mathbb{R}^d$, the *intersection graph* of S, $G_s = (V, E)$ is defined as follows: each vertex $v_i \in V$ corresponds to the objects o_i and $e_{ij} \in E$ if $o_i \cap o_j \neq \emptyset$.

2.3 Complexity classes and algorithmic techniques

Problems are said to belong to *complexity classes* according to a measure of their "hardness". There are four main complexity classes that we refer to in this document: **P**, **NP**, **NP**-complete and **NP**-hard. Before defining the complexity classes, let us look at the notion of

reduction. Problem Y is said to be *polynomially reducible* to problem X if X is no more than a polynomial factor harder than Y. In other words, if X can be solved in polynomial time, then Y can be solved in polynomial time. An algorithm is said to be *efficient* if it has a polynomial running time, i.e. given an input of size n, the worst case running time is $O(n^k)$ for some constant k. The class **P** consists of all problems for which there exists an efficient algorithm. The class **NP** consists of all problems for which answers can be checked by a polynomial time algorithm, i.e. if we were given a "certificate" of a solution then we could verify the correctness of the certificate in polynomial time in the size of the input to the problem. The class **NP**-complete consists of all problems that are in **NP** such that all other **NP** problems are reducible to them (or no other **NP** problem is more than a polynomial factor harder). Informally, a problem is **NP**-complete if answers can be verified efficiently, and an efficient algorithm to solve this problem can be used to solve all other **NP** problems efficiently. The class **NP**-hard is informally the complexity class of problems that are harder than **NP**-complete problems: all **NP** problems are reducible to them.

A problem is a *decision* problem if the solution to the problem is a yes or no answer and it is an *optimization* problem if it requires a function to be optimized (minimized or maximized) as part of the solution. An example of a decision problem would be: is polygon P partitionable into convex components such that the perimeter of the cuts is less than k? An optimization version of the same question: partition P into convex components such that the perimeter of the cuts is minimized. Following Motwani [235], we define an optimization problem Π as being characterized by three components:

- **Instances** D: a set of input instances.
- **Solutions** $S(I)$: the set of all feasible solutions for an instance $I \in D$ and $S_D = \bigcup S(I)$ for all $I \in D$.
- **Value** f: a function which assigns a value to each solution, $f : S_D \rightarrow \mathbb{R}$.

A maximization problem Π is: given $I \in D$, find a solution $\sigma^I_{opt} \in S(I)$ such that

$$\forall \sigma \in S(I), f(\sigma^I_{opt}) \geq f(\sigma)$$

A minimization problem is defined similarly. Throughout the document we refer to $f(\sigma^I_{opt})$ by OPT.

A large number of the known optimization problems are **NP**-hard. Complexity theory states that it is impossible to find efficient algorithms for such problems unless the class **P** is the same as the class **NP** (i.e. $\mathbf{P} = \mathbf{NP}$). Relaxing the requirement to obtain an optimal solution for all instances of a problem results in an *approximation algorithm* which returns a near-optimal solution. Approximation algorithms are said to have an approximation guarantee (or performance guarantee) R_A if for any input of size n, the cost $A(I)$ of the solution produced by algorithm A on instance I is within R_A of the cost $OPT(I)$ of an optimal solution of the same instance I. An approximation is said to be a k-approximation if its approximation guarantee is k. Approximation algorithm are classified according to their approximation guarantee:

- **Absolute (or additive) approximation guarantee.** A polynomial time approximation algorithm A for an optimization problem Π is said to have an absolute approximation guarantee of k if for every instance I of Π we have $|A(I) - OPT| \leq k$.

- **Relative (or multiplicative) approximation guarantee.** The relative approximation guarantee is defined as the $R_A(I) = \max\left(\frac{A(I)}{OPT(I)}, \frac{OPT(I)}{A(I)}\right)$ (depending on whether the optimization problem is a minimization or a maximization problem). If for every $\epsilon > 0$ there is a polynomial approximation algorithm for Π with a relative approximation guarantee of $1+\epsilon$ then problem Π is said to have a *Polynomial Time Approximation Scheme (PTAS)*. The running time of an algorithm of the approximation scheme is expressed in terms of the size of the input and in terms of ϵ. If the running time is polynomial in both the input size and $1/\epsilon$ the problem is said to have an *Fully Polynomial Approximation Scheme (FPTAS)*. An algorithm such that R_A is constant, is a *constant guarantee* approximation algorithm. An approximation guarantee k is said to be *asymptotic* if there exists an $n_0 > 0$ such that an algorithm A achieves a k-approximation for all instances of the problem having $OPT \geq n_0$. Similarly, a problem has *APTAS* (asymptotic PTAS) and *AFPTAS* (asymptotic FPTAS) if

for each $\epsilon > 0$ there exists $n_0 > 0$ and a polynomial time algorithm such that the approximation guarantee is $1 + \epsilon$ for all instances having $OPT \geq n_0$.

Linear programming is a branch of applied mathematics concerned with linear programming problems. If $c_1, c_2, \ldots, c_l$ are real numbers and $x_1, x_2, \ldots, x_l$ are real variables a *linear function* is defined by

$$f(x_1, x_2, \ldots, x_l) = c_1 x_1 + c_2 x_2 + \ldots + c_l x_l = \sum_{j=1}^{l} c_j x_j$$

. If f is a linear function and if b is a real number, then the equation $f(x_1, x_2, \ldots, x_l) = b$ is called a *linear equation* and the inequalities $f(x_1, x_2, \ldots, x_l) \leq b$ and $f(x_1, x_2, \ldots, x_l) \geq b$ are called *linear inequalities.* Linear equations and linear inequalities are both referred to as *linear constraints.* A linear programming problem is the problem of maximizing (or minimizing) a linear function subject to a finite number of linear constraints [84]. When the variables are required to be integers, the problem is called an *integer programming* problem. The special cases of integer programming problems where the variables are binary are called *binary integer programming* problems. The following linear program is said to be in the *standard form*:

$$\max \sum_{j=1}^{l} c_j x_j$$

$$\sum_{j=1}^{l} c_{ij} x_j \leq b_i \qquad (i = 1, 2, \ldots m) \qquad (1)$$

$$x_j \geq 0 \qquad (j = 1, 2, \ldots, l) \qquad (2)$$

Dynamic programming is an algorithmic technique to solve optimization problems by caching subproblems solutions rather than recomputing them in order to improve the running time of a given algorithm.

Chapter 3

State of the art

An instance of a *decomposition problem* consists of

- A object O and
- A set $OP = \{op_1, op_2, \ldots, op_j\}$ of object properties,

A *solution* to the decomposition problem consists of

- A collection $C = \{o_1, o_2, o_3, \ldots, o_i\}$ that obey the given properties such that $P = \bigcup_{o \in C} o$. We call the elements of the collection C the *components* of the decomposition.

As such, the collection C is called a *covering* of object O and it is called a *partition* if the shapes in C are interior disjoint. In this thesis, we are mainly interested in the object O and the elements of C being polygons. Hence, a decomposition of a polygon P is called a *partition* if the components $\{P_1, P_2, \ldots, P_i\}$ are not allowed to share a common interior point otherwise it is called a *cover*. An important property in the set of OP is to specify whether additional vertices are allowed or disallowed. These are called *Steiner points.*

An instance of a *packing problem* consists of

- A set of objects $S = \{s_1, \ldots\}$,
- A set of containers $B = \{b_1, \ldots\}$, and
- A set of transformations $T = \{t_1, \ldots\}$.

A *solution* to the packing problem consists of

- The packed set of objects $S^* \subseteq S$, $S^* = \{s_1, s_2, \ldots\}$
- A sequence of transformations which show how the packed objects are positioned: $T^* =< t_1, t_2, \ldots >$ such that each $t_i \in T$.
- A container the objects are packed in: $b^* \in B$
- The assertion that transformed objects $t_i(b_i)$ are disjoint subsets of b^*. This is what makes it a packing.

Both types of problems (decomposition or packing) have decision as well as optimization versions. In what follows, we review the partitioning, covering and packing literature. In the sections about decomposition, we first review the various decomposition of general polygons and then those of orthogonal polygons. Computational geometry has given particular attention to orthogonal polygons since they are encountered frequently in practice. For each type of polygon, we organize the sections around the shape of the components.

3.1 Partitioning

Although there are other types of planar objects that have been decomposed (e.g. splinegons [99], a splinegon being a polygon where edges have been replaced by well-behaved curves), we focus in this section on partitioning general polygons (arbitrary nonself-intersecting polygons with or without holes). We encounter a variety of objective functions for the optimization versions of decomposition problems but the two main ones are: minimizing the perimeter of the cuts (also called *minimum ink partition*) and minimizing the number of components created.

3.1.1 General polygons

Triangles

A decomposition of a polygon into triangles by a maximal set of nonintersecting diagonals is called a *triangulation*: it is known that every simple polygon admits a triangulation. Due to the huge amount of work done on polygon triangulation, triangulation has evolved to become a field of its own. Again, we follow Keil's survey on polygon decomposition [172] and omit this particular decomposition from this chapter. For comprehensive surveys on triangulation and related problems, we refer the reader to [43, 44]. We mention that the triangulation of simple polygons can be computed in linear time [76] and that the problems of decomposition into triangles, trapezoids and convex, star-shaped, monotone and spiral polygons are linearly equivalent [120].

Convex components

- **Minimum number of components for a polygon with holes.** The **NP**-hardness of the convex partition of polygons with holes is proved in [197] for both of the cases where Steiner points are allowed and disallowed. Lingas and Soltan [200] define F to be a given family of directions in the plane. They show that the problem of partitioning a planar polygon P with holes into a minimum number of convex polygons by cuts in the directions of F is **NP**-hard if $|F| \geq 3$ and that it admits a polynomial-time algorithm if $|F| \leq 2$. Martini and Soltan study a special case of the problem combinatorially [218].

- **No minimum-guarantee partition of polygons without holes.** Several algorithms polynomial in n and N that do no allow Steiner points are devised [116, 258]. No optimization is done. Tanase and Veltkamp propose to partition simple polygons into unions of convex regions using straight skeletons, as a preprocessing step for shape matching in image processing [273]. A more recent work partitions a convex polygons into n convex parts each being based on a single side of P and containing a specified share of P [17].

- **Minimum number of components for polygons with neither holes nor Steiner points.** Several constant factor approximation algorithms to approximate the number of components are developed disallowing Steiner points [74, 154]. Green gives two algorithms for the optimal decomposition of polygons into convex parts under the minimum number of components criterion [141]. The running time of the first algorithm is $O(n \log n)$ and the solution produced is at most four times the optimal partition. The second algorithm is an exact algorithm that gives an optimal partition and runs in $O(N^2 n^2)$ time. Independently, Keil solves the same problem in $O(N^2 n \log n)$ time [169]. More recently, Keil and Snoeyink show that a partition of a simple polygon into a minimum number of convex regions without Steiner points can be computed in $O(n + N^2 \min(N^2, n))$ [174] .

- **Minimum number of components for polygons without holes with Steiner points.** Chazelle and Dobkin give an $O(n + N^3)$ time algorithm to decompose a polygon into the union of a minimal number of convex polygons while allowing Steiner points [77]. Since any decomposition must consider removing all the reflex vertices of P, the algorithm is based on the introduction of X_k patterns that remove k reflex vertices without the introduction of new ones and while minimizing the number of components. Figure 14 shows a decomposition with X_2 and X_3 patterns. The equivalent problem is **NP**-hard for polygon with holes.

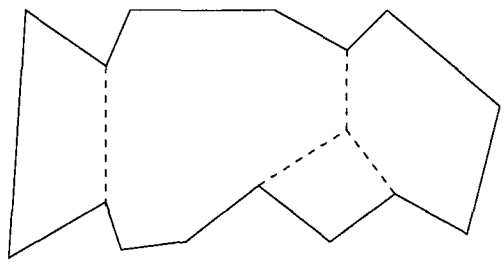

Figure 14: A decomposition with X_2 and X_3 patterns.

- **Minimum ink for a polygon with holes without Steiner points.** In his doctoral thesis [168], Keil shows the **NP**-completeness for polygons with holes.

- **Minimum ink for a polygon with neither holes nor Steiner points.** Keil

develops an $O(N^2n^2 \log n)$ time exact algorithm based on dynamic programming [169].

- **Minimum ink for a polygon with holes and Steiner points.** Lingas et al. prove **NP**-hardness and they devise, for polygons with holes, an $O(n \log n)$ time algorithm that produces a convex partition of size $O((b + m) \log N)$ where b is the perimeter of the polygon and the holes and m is the minimum perimeter of its convex partition. No optimal algorithms for the problem are known when Steiner points are allowed [199].

- **Minimum ink for a polygon without holes with Steiner points.** Levcopoulos and Lingas give an approximation algorithm for the convex partitioning with the minimum ink requirement. Their algorithm allows Steiner points and yields a solution of size $O(b \log N)$ where b is the perimeter of the polygon [191] .

Spiral components

- **Minimum number of components for polygons with holes without Steiner points.** Keil proves that the problem is **NP**-complete [168].

- **No minimum-guarantee partition of polygons with neither holes nor Steiner points.** Feng and Palvidis develop a polynomial time algorithm for partitioning a polygon without holes into spiral components without any minimum guarantee for the number of components [116].

- **Minimum number of components for polygons with neither holes nor Steiner points.** In his doctoral thesis, Keil solves this problem by developing a dynamic programming $O(n^3 \log n)$ time algorithm based on his dynamic programming formulation of the convex partition [168].

- **Minimum ink for polygons with neither holes nor Steiner points.** Similarly, Keil solves the minimum ink problem using an $O(n^4 \log n)$ time algorithm [169].

Star-shaped components

- **Minimum number of components for polygons with holes without Steiner points.** Keil proves the **NP**-completeness of the problem [168].

- **No minimum-guarantee partition for polygons with neither holes nor Steiner points.** Toussaint and Avis provide an $O(n \log n)$ time algorithm to partition into at most $\lceil \frac{n}{3} \rceil$ star-shaped components [18]. As cited in Keil's survey [172], Aggarwal and Chazelle [6] improve this result. Their algorithm partitions a polygon into at most $\lceil \frac{n}{3} \rceil$ components in $O(n)$ time.

- **Minimum number of components for polygon with neither holes nor Steiner points.** Keil presents an $O(n^5 N^2 \log n)$ time dynamic programming algorithm [169].

- **Minimum number of components for polygon without holes with Steiner points.** Shapira and Rappoport solve a restricted version of the problem where the kernel of each component contains a vertex of the polygon, a partition that does not always exist. They, therefore, allow Steiner points [264].

- **Minimum ink for polygons with neither holes nor Steiner points.** Keil provides an $O(n^4 \log n)$ time algorithm [169].

Monotone components

- **Minimum number of components for polygon with holes without Steiner points.** Keil proves the **NP**-completeness of the problem [168].

- **No minimum-guarantee partition for polygons without holes.** As a by-product of their triangulation algorithm, Garey et al. provide an $O(n \log n)$ time algorithm for partitioning a polygon in monotone components [128].

- **Minimum number of components and minimum ink for polygons with neither holes nor Steiner points.** Keil uses the same approach to solve both problems with an $O(Nn^4)$ time algorithm [168].

- **Minimum number of uniformly monotone components for polygons without holes.** If all the components in a partition are monotone with respect to the same line then they are said to be *uniformly monotone*. Liu and Ntafos provide two algorithms for this partition problem, one that disallows Steiner points and runs in $O(nN^3+N^2n\log n+N^5)$ time and one that allows them and run in $O(N^3n\log n+N^5)$ time [208].

Quadrilaterals

Quadrilateralization (or quadrangulation) is the term given to partitioning a polygon into quadrilaterals (quadrangles). Quadrilateralization has received considerable attention in the literature. Several applications for this problem are mentioned in Toussaint's survey [277]. A special case of quadrilateralization is partitioning into trapezoids often referred to as trapezoidation. Trapezoidation is in many cases a by-product of an intermediate step in triangulation algorithms.

- **Quadrilateralization for polygons with holes.** Lubiw shows that the problem of deciding whether a polygon with holes admits a quadrilateralization is **NP**-complete [214].

- **Convex quadrilateralization for polygons with or without holes.** Using Steiner points, Everett et al. shows that polygons without (with h) holes can always be quadrilateralized into $\frac{5(n-2)}{3}$ ($\frac{8(n+2h-2)}{3}$) convex quadrilaterals [113].

- **Minimum convex quadrilateralization for convex polygons.** Müller-Hannemann and Weihe present a linear time algorithm for partitioning convex polygons into the minimum number of strictly convex quadrilaterals. Steiner points are not allowed on the boundary of the polygon [238].

- **Minimum ink quadrilateralization for polygons without holes.** Conn and O'Rourke devise an $O(n^3\log n)$ time algorithm [87].

- **Fat convex quadrilaterals.** Van Kreveld gives an $O(n\log^2 n)$ time algorithm to

partition a fat simple polygon P with n vertices into $O(n)$ fat convex quadrilaterals. Van Kreveld defines the fatness of a polygon P by the wideness of every quadrilateral formed by every four points in P [278].

- **No minimum-guarantee trapezoidation.** As a by-product of a triangulation algorithm, Chazelle gives a linear time algorithm for trapezoidation with no minimum guarantee on the number of components [76]. Seidel presents an algorithm with $O(n \log^* n)$ expected running time [260].

- **Minimum number of trapezoids.** Asano et al. present an $O(n^2)$ time algorithm for partitioning a polygon without holes into the minimum number of trapezoids. Also they present an $O(n \log n)$ constant factor approximation algorithm in the case where the polygon has holes [13].

Other components

- **Minimum ink T-gon partition.** Levcopoulos et al. give an algorithm that given an input polygon P and an integer T, partitions a polygon into T-gons while minimizing the perimeter of the cuts. The cuts are restricted to diagonals and thus, no Steiner points are allowed. The algorithm runs in $O(n^3T^2)$ time [193].

- **Minimum number of α-fat components for polygons without holes.** Damian-Iordache and Pemmaraju call a polygon *fat* if it has a small aspect ratio and is called α-fat if this ratio does not exceed a certain α. An α-small polygon is a polygon whose diameter does not exceed α. Damian-Iordache and Pemmaraju present polynomial time algorithms to partition simple polygons into a minimum number of α-fat and α-small components while disallowing Steiner points. The algorithms are based on a dynamic programming framework and special algorithms are given for convex polygons [93, 94].

- **Most circular partition of regular n-gons: convex and nonconvex components.** Damian and O'Rourke define the *circularity* of a polygon in terms of the closeness of its aspect ratio to 1 and they discuss partitioning regular n-gons into

convex circular components [92]. For the equilateral triangle, it is shown that the most optimal partition can only be achieved with an infinite number of components. For $n > 5$, the optimal partition is the one piece partition. The question remains open for the square. The idea is pursued for nonconvex circular partitioning in an unpublished manuscript [91].

- **Minimum ink NEWS partition.** Motivated by geographic information retrieval, van Kreveld and Reinbacher presented polynomial time algorithms for partitioning a polygon into four parts that can be considered as the North, East, West and South (NEWS). There are several criteria to satisfy for such a partition (nonoverlapping simply connected adjacent regions, equal-area partition, boundary with simple shape) [279].

- **Minimum ink and nonoptimal area partitioning.** Given a polygon P and a number p, an *area partition* of P is a set of components $\{P_1, \ldots, P_p\}$ of P each of a specified area, such that union of the interior of the parts equals the interior of P. If, in addition, each polygon P_i $(1 \leq i \leq p)$ has a particular point site S_i in it the partition is called an *anchored* area partition on $S_1, \ldots, S_n$. Given a set of areas A=$a_1, \ldots, a_p$, Bast and Hert propose a $O(pn)$ algorithm to partition a simple polygon into p components with the given areas [30]. The algorithm first partitions P into q convex components and then "sweeps" over the components to merge or divide them when necessary. The running time is actually $O(pq + n)$, where q is the number of convex components but q is bounded by n since Steiner points are disallowed. The minimum ink partitioning for this problem is shown to be **NP**-hard. Anchored area partitioning is solved with a polynomial time divide-and-conquer sweep-line algorithm for any simply or nonsimply connected polygon [153].

- **Equal area partition of convex bodies.** Guardia and Hurtado study the equipartition (equal area) of convex bodies using chords. The authors show that there is no solution for the set of nonsymmetrically convex n-gons for $m > n$ chords and that when the problem has a solution the straight lines determined by the chords are area bisectors of the polygon: i.e. each cuts the polygon into two equal areas [143].

- **Minimum ink two equal partition.** Even partitioning a polygon into two (possibly disconnected) equal area components under the minimum ink criterion for the cuts is shown to be **NP**-complete. Two algorithms are presented: a PTAS that partitions a polygon into approximately equal parts and an exact algorithm for partitioning convex polygons with $O(n^2)$ running time [181].

- **Perfect partitioning.** A partitioning of a polygon P is said to be *perfect* if the components have the same perimeter as well as the same area. A partitioning into k perfect components is denoted by k-partitioning (see Figure 15 for a 3-partitioning of a square). Akiyama et al. study perfect partitioning of convex sets in the plane and show that for any k, any convex set admits a perfect k-partitioning. The authors also explore radial partitioning in which all cut lines are required to meet into a single point. They prove that every convex set admits a radial perfect 3-partitioning [9,10].

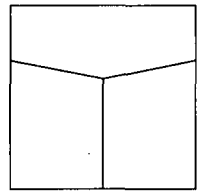

Figure 15: A perfect 3-partitioning of a square.

- **Two congruent components partition.** Eriksson considers the partition of a polygon into two congruent components (see Figure 16 for such an example of such partition). The running time for his algorithm is claimed to be $O(n^3)$ but he neglects the need for a data structure to check possible boundary intersections. The author also claims without proof that the algorithm generalizes to polygons with holes [111]. In a draft criticizing the paper, Rote gives a counterexample to this claim [252]. This problem is the subject of Chapter 4.

- **Pseudo-convex partition.** A novel partition is proposed by Aichholzer et al. [8]: the pseudo-convex partition is one where both convex polygons and pseudo-triangles are allowed. A pseudo-triangle is a planar polygon that has exactly three convex vertices, all other vertices are concave. They establish an existence proof of a pseudo-convex

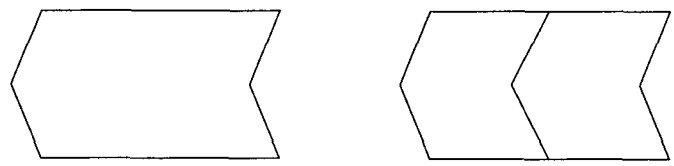

Figure 16: (a) A polygon, (b) its decomposition into two congruent components.

decomposition for every simple polygon. Gerdjikov and Wolf present an algorithm that determines a minimum pseudo-convex decomposition of a simple polygon in $O(n^3)$ time [129].

3.1.2 Orthogonal polygons

Rectangles and squares

The use of Steiner points is inherent in partitioning orthogonal polygons into rectangles.

- **Minimum number of rectangles for polygons with holes.** The first algorithms for partitioning polygons with holes run in $O(n^{\frac{5}{2}})$ [117,207,243] and in $O(n^{\frac{3}{2}} \log n)$ [159, 205,206]. The results were extended to point holes in [270]. Gudmundsson et al. show an $\Omega(n \log n)$ lower bound on the time complexity of the problem (for optimal or approximative partition) [144].

- **Minimum number of rectangles for polygons without holes.** The first two algorithms were constant factor approximation algorithms with $O(n^2)$ [138] and $O(n \log n)$ [241] respective running times as cited in Keil's survey [172]. Liou et al. present an $O(n \log \log n)$ time algorithm to find the optimal partition into rectangles for polygons without holes by taking a graph theoretic approach. They also present a linear time algorithm for convex and horizontally (vertically) convex orthogonal polygons [203]. More recently, Bajuelos et al. show tight lower and upper bounds on the number of rectangles in a partition [21].

- **Minimum number of fat rectangles.** O'Rourke et al. study the partition of orthogonal polygons into fat rectangles. The goal is to maximize the shortest side γ of

the rectangles in a partition and among all partitions with the same γ, the authors seek the one with the fewest number of rectangles [244, 246]. O'Rourke and Tewari study the structure of a partition for several types of cuts: vertex cuts that have at least one end at a vertex, anchored cuts that have endpoints on the boundary of the polygon and unrestricted cuts that are floating inside the boundary of the polygon [246]. Figure 17 shows a partition of an orthogonal polygon into rectangles with vertex and anchored cuts. O'Rourke et al. focus on vertex cuts and gives an $O(n^5)$ time dynamic programming algorithm. The authors prove that their algorithm generalizes for anchored cuts and give more specialized algorithms for monotone, pyramids and staircase polygons with better running times [244].

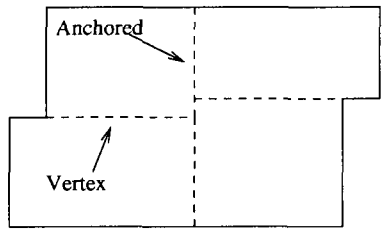

Figure 17: An example of a partition into rectangles with vertex and anchored cuts.

- **Minimum ink partition into rectangles for polygons with holes.** Lingas et al. prove the problem to be **NP**-complete [199]. Approximation algorithms with constant factor approximations were presented by Lingas [198] and Levcopoulos [187] as cited in Keil's survey [172]. Levcopoulos presents an $O(n \log n)$ time constant factor approximation improving on previous results [186].

- **Minimum ink partition into rectangles for polygons without holes.** Lingas et al. give an $O(n^4)$ to partition a rectilinear polygon into rectangles under the minimum ink criterion. Knowing that the optimal cuts lie on the grid induced by the polygon boundary, the search for optimal solution is restricted to the $O(n^2)$ candidate points determined by the grid intersections for which there are $O(n^2)$ matching points to complete the rectangles [199].

- **Minimum ink partition of rectangles into rectangles.** If the partitioned shape

is a rectangle with point holes, several constant factor approximation algorithms were presented [135, 136, 136]. Gonzalez et al. devise an approximation algorithm that allows only guillotine cuts, i.e cuts that are obtained by starting recursively cutting the rectangle into two rectangles by a line orthogonal to one of the axes [134].

- **Minimum ink partition of rectangles, squares and circles into equal area components.** Bose et al. study the optimal —in terms of the perimeter of the cuts— partition of squares, rectangles, circles and prisms into k equal area components. They show that the diameter cuts the circle into two equal areas optimally. The optimal solution to cut a circle into k equal area components such that the chords forming the cuts do not intersect and the solution to the same problem such that the cuts intersect into one Steiner point are also presented [51, 52, 54]. The minimum ink partition of rectangles and squares is reviewed in detail in Chapter 5.

- **Area partitioning of rectangles under different optimization criteria.** Given a rectangle with area α and a set of n positive real numbers $A = \{a_1, a_2, \ldots, a_n\}$ with $\sum_{a_i \in A} a_i = \alpha$, the problem consists of partitioning R into n rectangles r_i with area a_i $(i = 1, 2, \ldots, n)$ while minimizing an objective function. The decision version of the problem is known to be **NP**-complete [32]. Three main objective functions are discussed in the literature: minimizing the sum of the perimeters of rectangles r_i, minimizing the aspect ratio of the rectangles r_i and minimizing the maximum perimeter of the rectangles r_i. For the latter objective function, Kong et al. solve the problem in polynomial time when all the areas a_i are equal [179, 180]. Constant factor approximation algorithms for the general problems and all three objective functions can be found in [33, 240].

Other components

- **Minimum number of L-shaped components and other optimization criteria.** Lopez and Mehta present two algorithms for partitioning rectilinear polygons into L-shapes and rectangles by using horizontal cuts only. Both algorithms run in $O(n +$

$h \log h$) time where h is the number of N-dents (the authors call them H-pairs) [211].

- **Minimum number of star-shaped components for monotone simple polygons.** Allowing Steiner points, Liu and Ntafos present a linear time algorithm for partitioning monotone orthogonal simple polygons into star-shaped components as well as an $O(n \log n)$ constant factor approximation algorithm for general orthogonal polygons [209].

- **Minimum number of $\leq k$ and fixed number of vertices components.** As cited in Keil's survey [172], Gunther gives a polynomial time algorithm for partitioning an orthogonal polygon into orthogonal polygons with k or fewer vertices [146] and Gyori presents a partition into components with fixed number of vertices [147] .

- **Minimum number of quadrilaterals.** Toussaint and Sack show that a star-shaped orthogonal polygon can always be decomposed into convex quadrilaterals [256] and Kleitman et al. generalize the result for arbitrary simple orthogonal polygons [167]. Later, Sack and Toussaint give an $O(n \log n)$ time algorithm [255, 257] and Lubiw proves $\Omega(n \log n)$ as a lower bound for the complexity of the problem [214].

- **Minimum ink quadrilateralization.** Two independent works show that the minimum ink quadrilateralization can be computed in $O(n^4)$ time [173,214]. Toussaint and Sack show that star-shaped or monotone polygons can be quadrilateralized in linear time [256].

3.1.3 $3D$ partition

This thesis does not address partitioning of $3D$ objects. Therefore, our citing of the work on partitioning polyhedra is not comprehensive and is just cited to give a flavour of the work done in the field. It is important to note that whereas every polygon can be triangulated, this is not the case in three-dimensions as shown in a result from 1911 cited in [44]. Work on triangulation/tetrahedralization (when it is possible), convex partition and rectangle partition of polyhedra can be found in the literature [20,27,37,42,44,73,75,78,81,98,126,152,196,254].

3.2 Covering

Covering problems are known to be bound by partition since the minimal cardinality of a cover is at least equal to the cardinality of a partition. In other terms, any partition is a valid cover while a cover may or may not be a valid partition. In covering, Steiner and Steiner-free decomposition are also considered. Figure 18 shows a partition and a cover of the same polygon.

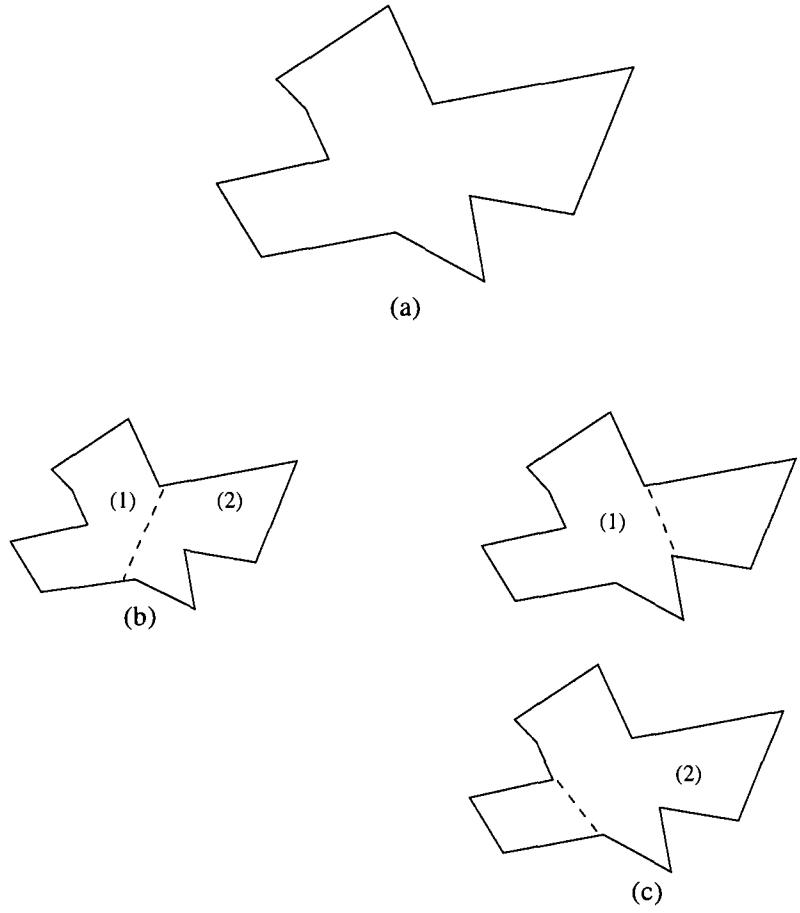

Figure 18: (a) A simple polygon, (b) its partition into two components, (c) its cover with two components.

3.2.1 General polygons

Convex components

- **Minimum number of components for polygons with holes.** For polygons with holes, O'Rourke and Supowit show that the problem is **NP**-hard by with or without Steiner points [245].

- **Minimum number of components for polygons without holes.** Culberson and Reckhow prove that, for polygons without holes, covering the interior or boundary of an arbitrary polygon with convex polygon is **NP**-hard, that covering the vertices of an arbitrary polygon with convex polygons is **NP**-complete and that covering the interior or the boundary of a polygon with rectangles is **NP**-complete [90].

- **Recognizing the polygons without holes that can be divided into a fixed number of convex polygons.** Shermer explores the properties of three classes of closely related polygons. U_2 is a polygon that can be expressed as the union of two convex polygons. P_3 is a polygon such that for any three points in the polygon, at least two of them are visible to each other. A KR polygon is one such that all its reflex vertices belongs to its kernel. A KR polygon where $N \neq 3$ is P_3. A KR polygon with N even is U_2. Shermer's linear time algorithm classifies an input polygon P by outputting: KR (if KR but not P_3), P_3 (if P_3 but not U_2), U_2 or NO (if P belongs to none of the classes) [266]. Keil mentions in his survey [172] Belleville's work that solves the problem of recognizing polygons that can be covered by three convex polygons in linear time [35, 36].

- **Minimum number of O-convex components for polygons without holes.** Let O denotes a set of line orientations. A connected point set is called O-convex if its intersection with any line with orientation in the set O is connected. Two points in a polygon P are said to be O-visible if there is an O-convex path between them that does not intersect δP. A polygon P is O-convex if and only if every pair of points is O-visible. Bremner and Shermer characterize a class of polygons that admits a

polynomial time algorithm for finding an O-convex cover [59].

- **Covering a convex polygon with translates of convex polygons.** Given a set of n convex polygons, determining whether they can be translated to cover a fixed convex polygon is proven to be **NP**-hard by Wang et al. [280].

Star-shaped components

- **Minimum number of components for polygons with holes.** O'Rourke and Supowit show that the problem is **NP**-hard for polygons with holes with or without Steiner points [245].

- **Minimum number of components for polygons without holes.** For polygons without holes, it is mentioned in Keil's survey [172] that Aggarwal proves **NP**-hardness of the problem in his Ph.D. thesis [5]. Approximation algorithms with an $O(\log n)$-approximation factor are devised for restricted versions of the problem [7, 132].

- **Covering with two star-shaped polygons.** Belleville, as cited in Keil's survey [172], solves the problem of recognizing polygons that can be covered by two star-shaped polygons in $O(n^4)$ time [34].

Rectangles and squares

- **Minimum number of components for polygons with or without holes.** It is not known whether the optimal covering can be computed in exponential time. Several experimental and theoretical results have been developed for this problem [150, 185, 188–190, 192]. Gudmundsson and Levcopoulos present an $O(\log n)$ factor, $O(n \log n + OPT)$ time approximation for covering obtuse-angle-only polygons (possibly with holes) with rectangles provided that the vertices of the input polygon are given as polynomially bounded integer coordinates. In a more recent work, the same authors give the first constant factor $O(n^2 + OPT)$ time approximation algorithm for covering with rectangles with bounded aspect ratio. They also present several constant factor approximations algorithms for covering with squares and fat rectangles [189].

Other components

- **Minimum number of spiral components for polygons with holes.** For polygons with holes, O'Rourke and Supowit show that the problem is **NP**-hard with or without Steiner points [245].

- **Covering a fat convex quadrilaterals with fat triangles.** Van Kreveld develops a method to cover a fat convex quadrilateral with $O(1)$ fat triangles [278].

3.2.2 Orthogonal polygons

Rectangles and squares

Covering with rectangles and squares refers to axis-aligned ones.

- **Minimum number of rectangles for polygons with holes.** As mentioned in Keil's survey [172] on polygon decomposition, covering an orthogonal polygon with holes with the minimum number of rectangles is **NP**-complete [219] even in the special case where only the boundary or only the reflex vertices need to be covered [86]. Berman and DasGupta prove that no polynomial time approximation schemes exists unless $\mathbf{P} = \mathbf{NP}$ and present a few constant factor approximation algorithms for some variations of the problem such as covering the boundary and the vertices of a given polygon [39, 40]. Franzblau presents an $O(\log OPT)$-approximation [122]. Bern and Eppstein [45] present a constant factor approximation algorithm for a polygon with holes when the polygon is in general position (no two boundary segments are collinear). The question whether the general problem admits a constant factor approximation remains open [45]. Heinrich-Litan and Lübbecke conjecture that it does and support their conjecture by experimental work (integer programming) [151]. Kumar and Ramesh present an $O(\sqrt{\log n})$-approximation algorithm for covering polygon with holes [182].

- **Minimum number of rectangles for polygons without holes.** Culberson and Reckhow prove that the general problem is **NP**-complete [90]. However, several

polynomial time algorithms exist for special cases: orthogonally convex polygons [66, 204], horizontally convex polygons [123], 2-staircase polygons [58], polygons that do not contain a rectangle that touches the boundary only at two opposite corners of the rectangle [215]. Two constant factor approximation algorithms are devised by Cheng et al. [80] and Franzblau [122].

- **Optimal nonpiercing covering.** Keil presents an algorithm for optimal nonpiercing covering for orthogonal polygons without holes. Nonpiercing covering consists of covering with rectangles where every rectangle R_i and R_j in the cover are such that $R_1 - R_2$ or $R_2 - R_1$ is connected [171].

- **Minimum number of squares for polygons with holes.** Aupperle et al. prove that the problem is **NP**-complete [16].

- **Minimum number of squares for polygons without holes.** The problem is solved with a graph theoretic approach. A graph is associated with a given grid orthogonal polygon P (one with integer coordinates). P is seen as composed of unit squares called *the blocks.* A *square* is a square subset of the blocks of the orthogonal polygon. A *square cover* is a collection of squares whose unions is equal to the polygon. Each block in the polygon is made to correspond to a vertex in the graph and two vertices are adjacent in the graph if and only if the corresponding blocks belong to a square of P. Albertson and O'Keefe show that this graph is perfect for a polygon without holes [11] . Aupperle et al. show that the resulting graph is chordal and reduce the problem to that of finding a minimum clique cover in a chordal graph [16]. As chordal graphs are perfect, the problem is solvable in polynomial time and the running time of the algorithm is $O(B^2\sqrt{B})$ where B is the number of blocks in the polygon. It is mentioned in Keil's survey [172] that Aupperle, in his thesis, adapted this algorithm further to obtain $O(B\sqrt{B})$ running time [15]. Moitra proposed a parallel algorithm linear in B for solving the same problem using also a graph theoretic approach [232]. Bar-Yehuda and Ben-Chanoch's algorithm runs in $O(n + OPT)$ time [26].

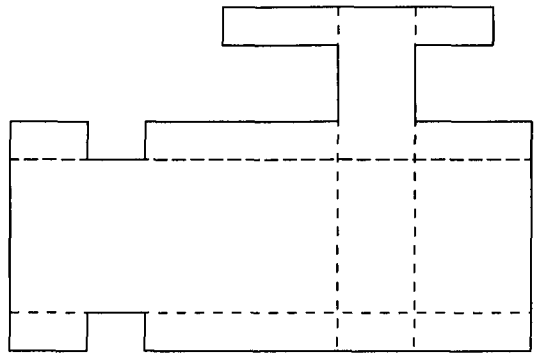

Figure 19: The prolongation of the dents in an orthogonal polygon and its division into regions.

Star-shaped components

- **Minimum number of s-star-shaped components for polygons without holes.** Following a graph theoretic approach, Motwani et al. solve the problem of minimally covering orthogonal polygons with s-star-shaped polygons. The polygon is divided into regions defined by the prolongation of the dents supports until they hit the boundary as shown in Figure 19. A star graph $H = (V, E)$ is defined from this partition. Each region is associated with a graph vertex in H and two vertices are adjacent in H if the two corresponding regions indirectly see each other. It is proven that a minimum clique cover of H corresponds exactly to a minimum cover of P by s-star polygons. The star graph is proven to be a weakly triangulated graph that are again perfect and hence the problem is solved in polynomial time ($O(n^8)$ time) [236].

- **Minimum number of r-star-shaped components for polygons without holes.** Keil follows a geometric approach to solve the problem of optimally covering horizontally convex polygons into r-star-shaped components in $O(n^2)$ time [170]. This result is improved to $O(n)$ time by Gewali et al. [130] and more recently to linear time with only $O(k)$ additional space where k is the size of the optimal solution by Lingas et al. [201]. Keil and Worman, following a graph theoretic approach, settle the open problem for general simple orthogonal polygons by showing that it can be solved in polynomial time ($\tilde{O}(n^{17})$ where $\tilde{O}()$ hides polylogarithmic factors) [281, 282]. More

recently, Lingas et al. present a simple linear time 3-approximation for the same problem [202].

- **Minimum number of components for other star-shapedness definitions.** Optimal number of components algorithms for different star-shapedness definitions can be found in [62,102,216,257] and in [131] as cited by Keil in his survey [172].

Orthogonally convex components

- **Minimum number of orthogonally convex components.** Keil provides an optimal covering in quadratic time of horizontally convex polygons by orthogonally convex components using a geometric approach [170] . Reckhow and Culberson prove a lower bound of $\Omega(n^2)$ for any algorithm that needs to report an explicit representation of the output for an orthogonal polygon. They then provide a linear time counting-based algorithm for finding the minimum number of orthogonally convex polygons to cover a horizontally convex orthogonally polygon. Following a geometric approach, the authors also develop an $O(n^2)$ time algorithm to optimally cover class $2b$ polygons with orthogonally convex components. A complex polynomial time algorithm is presented to handle orthogonal polygons with four dent orientations with the condition that at most a constant number of dent lines intersect any given dent line [250]. For class 2 and 3 polygons, Motwani et al. reduce the problem to a minimum clique cover in the polygon's visibility graph which results in a polynomial time algorithm for this case [237].

3.2.3 $3D$ covering

Not much work has been done on covering polyhedra. Mookherje and Prabhajaran provide an algorithm (without an analysis of its complexity) for approximately covering a convex polyhedron with the minimum number of spheres, a work with an application in the radiation treatment of a tumour [233]. Wang and Yang present an algorithm for the following problem: given two simple polyhedra P^0 and P^1 and a convex polyhedron P^2, determine whether or

not P^0 can be covered by $P^1 \cup P^2$. The running time is polynomial in the size of the three polyhedra [280].

3.3 Packing

In surveying the literature on packing problems, we introduce a taxonomy of these problems:

- **Packing at fixed locations.** Only a finite number of discrete locations for each object to be packed (in the set $S = \{s_1, \ldots\}$, each $s_i \subseteq G$) are considered. Such problems are equivalent to maximum independent set in a graph.
- **Strip packing.** The container is a rectangular strip of a given width and the goal is to pack all the given objects into the strip so as to minimize its height.
- **Packing identical objects.** The objects to pack are identical and the possible locations are not finite.
- **Packing different objects.** The objects in S are different.

For the optimization versions of packing problems, there are two main variants for the objective function:

- Pack everything. If the container is unique, this is a decision problem. If it is not, then you want to pack everything into the best container.
- Pack the "best" subset for varying definitions of best.

In the following subsections of this chapter, n will not denote the number of vertices of a polygon.

3.3.1 Packing at fixed locations

- **Packing in graphs.** The problem of maximum independent set for general graphs with n vertices is hard to approximate even to within a factor of $n^{1-\epsilon}$ [149].

- **Packing objects with fixed locations.** A reduction from independent set on intersection graphs is shown to prove that the problem of finding the largest independent set of objects is **NP**-hard [121, 158].

- **Squares in grid orthogonal polygons.** Hochbaum and Maass give a PTAS for the **NP**-complete problem of packing a maximum number of $(k \times k)$ squares in an orthogonal grid polygon [155]. Variations of this problem are studied in Chapter 6.

- **Unit disks.** Given a set of possible locations for unit disks, Hunt et al. present a polynomial time approximation scheme for solving maximum independent set for an intersection graph where th objects are unit disks [157].

- **Fat objects.** Erlebach et al. focus on the case where the objects to pack have varying sizes but are fat. They give a PTAS for both maximum weighted independent set and minimum weight vertex cover [112]. Chan describes an improved PTAS [68].

- **Line segments and convex objects.** Agarwal and Mustafa present two approximation algorithms for the independent set problem on line segments and convex objects [2].

- **Rectangles (map labelling).** Agarwal et al. describe an algorithm with an $O(n \log n)$ running time and $O(\log n)$-approximation factor for finding the maximum subset in a set of n arbitrary axis-parallel rectangles in the plane. When all rectangles are all unit height, they present a 2-approximation in $O(n \log n)$ time and a $(1+\epsilon)$-approximation in time $O(n \log n + n^{2/\epsilon-1})$ time [4]. Khanna et al. [177], being interested in database applications of the maximum subset packing problem, state independently similar results to Agarwal et al. [4]. Berman et al. [41] and Chan [69] improve the previous on both arbitrary and unit height rectangles.

- **Weighted rectangles.** Given a set of fixed weighted rectangles, a $(2+\epsilon)$-approximation algorithm is presented to find the maximum-weight packing [162].

- ***D*-box graphs.** A d-dimensional box (d-box) is a subset of $\mathbb{R}^d$ that is a Cartesian

product of d intervals in $\mathbb{R}$. Chlebík and Chlebíková prove that no PTAS exists for maximum independent set for d-box graphs for any fixed $d \geq 3$ unless $\mathbf{P} = \mathbf{NP}$ [82].

3.3.2 Strip packing

- **Rectangle strip packing with no rotations.** Given a set of different rectangles of width at most 1 and given a strip of unit width, place the rectangles in the strip so as to minimize the height of the strip. Baker et al. show that the bottom-left approximation algorithm has asymptotic performance guarantee of 3 [23]. Tarjan et al. study different level-oriented algorithms and show their performance guarantees to be 2 and 1.7 and 1.5 [85]. Other works present asymptotic performance guarantees of 2.5 [268], $\frac{4}{3}$ [133] and $\frac{5}{4}$ [22]. The best current absolute performance guarantee is 2 [259, 271]. Fernandez and Zissimopoulos present a $(1+\epsilon)$-approximation for the restricted version of the problem where the height and width of rectangles are bounded by an absolute constant [96]. Kenyon and Rémila present an AFPTAS for the same problem with no restriction on the size of the rectangles [175].

- **Rectangle strip packing with $90°$ rotations.** Miyazawa and Wakabayashi present two algorithms allowing orthogonal rotations: one has an asymptotic performance guarantee of 1.613 [230] and the other a performance guarantee of $\frac{3}{2}$ [108]. Jansen and van Stee present an AFPTAS for this problem [161].

- **$3D$ rectangle strip packing.** Several approximation algorithms are developed for the three dimensional version of the problem [25, 160, 194, 195, 229, 231].

- **Variants.** Variations of the strip packing are also considered such as online strip packing (with and without rotations) [19, 89], as well as strip packing with precedence constraints or release times [14].

- **Packing circles in a strip.** The maximal density of circles in a strip of width w is determined for certain values of w [127].

3.3.3 Packing identical objects

- **Heuristics.** Many computer aided optimization algorithms lead to experimental results that do not contain meaningful algorithmic results. As examples we cite: finding the densest packing of equal disks in an equilateral triangle [212], finding the densest packing of equal disks in a square [63, 64, 217, 272] and identifying patterns of packing equal circles in a square and in various shapes [139, 210, 213].

- **Finite problems.** Packing of small numbers of different simple shapes in larger simple shapes is addressed; the packing centre of Friedman is a clearinghouse for many results [124], and many nontrivial results remain without a proof. Proofs are found for specific cases [119, 221, 222], but others are simply the best known packings for very specific cases. For example, in [223] the packing of 16, 17 and 18 congruent circles in an equilateral triangle is presented with the following claim: "The results have been found by the use of simulated annealing and a quasi-Newton optimization technique, supplemented with some human intelligence." For a recent survey and new results on packing small numbers of unit squares in squares, see [125].

- **Packing squares in squares.** Earliest to ask the question of packing squares with unit squares are Erdös and Graham [109]. They prove that allowing rotations keeps the amount of area uncovered down to at most proportional to $\alpha^{\frac{7}{11}}$ which is for large α is better than linear waste produced by just stacking the squares row by row, where α is the side of the square.

- **Packing two identical disks in a polygon.** The *medial axis* of a polygon P is defined as the locus of all centres of circles inside P that touch δP in two or more points. Bose et al. give two algorithms for packing two disks in a convex n-gon. Their first algorithm, which maximizes the radius of two equal disks, runs in $O(n)$ time. The second algorithm runs in $O(n^2)$ time and maximizes the sum of radii of two disks. It is shown in both cases that the centres of the two optimal disks lie on the medial axis of the polygon [53]. The former problem was first discussed in the

context of finding a folding to hide the largest possible disk in a simple polygon and was solved in $O(n^2)$ time. Kim et al. [178] presented an improved $O(n \log n)$ algorithm for convex polygons. They also present a variation of the problem where the folding line is set to pass through vertices of the polygon and they solve it in $O(n^2 \log^2 n)$ time. Bespamyatnikh [46] improves the running time of this latter version to $O(n \log^2 n)$ time. For simple polygons, he gives an $O(n \log^2 n)$ algorithm for the original problem. Bose et al. improve on this latter result by presenting a randomized algorithm that runs in $O(n \log n)$. Finally, Chan improves the result to a linear expected time algorithm [70].

- **Packing disks in an orthogonal polygon.** The algorithm packs a maximum number of unit disks in a rectangular region with obstacles, represented as holes. The approximation factor is $\frac{2}{3}$ [38].

- **Packing squares in a polygon.** Baur and Fekete [31] present an approximation algorithm for the following **NP**-hard problem: Given a polygonal region P—possibly with holes—with n vertices, pack k many $(L \times L)$ squares into P such that L is as big as possible. The authors prove that the problem has no PTAS unless $\mathbf{P} = \mathbf{NP}$ and give a $\frac{2}{3}$ -approximation algorithm. The problem posed here belongs to a larger set of problems called *dispersion problems* or *obnoxious facility location.*

- **Packing rectangles in a rectangle.** The pallet loading problem consists of packing a large containing rectangle with identical axis-parallel copies of a small rectangle. The problem is not known even to be in **NP**. Dowsland claims the problem to be **NP**-hard [100] and Exeler claims it to be in **NP** [114]. Both claims are erroneous [97]. Neliben presents several heuristics to solve the problem [242] and Tarnowsky proves that a special case of it can be solved in polynomial time under some unnatural assumptions as to the nature of an optimal solution [275]. This problem has received much attention in the literature; Ram surveys many solutions and their industrial applications [248].

- **Heuristics for packing identical boxes into a polyhedron.** Heuristics and

experimental results can be found for the problem of packing identical ($4 \times 2 \times 1$) boxes in a "car trunk" without rotations [104] and with rotations [103].

- **Packing in infinite space.** For more information on the packing density of objects in simple geometric shapes or infinite space see, for example, Tóth's survey [276].

3.3.4 Packing different objects

- **Packing similar triangles into a triangle.** A sufficient condition to pack any sequence of triangles similar to a triangle T is discussed in several papers. Let $a(T)$ denote the largest number such that any finite sequence of triangles similar to T with total area not greater than $a(T) \cdot area(T)$ can be packed into T. Richardson proves that $a(T) \geq \frac{1}{2}$ [251]. Soifer conjectures that $a(T) = \frac{1}{2}$ for any triangle [269]. Finally, Januszewski proves the existence of a triangle with $a(T) > \frac{1}{2}$ [163].

- **Packing squares in squares.** It is proven that the problem of deciding whether a set of squares of different sizes can be packed into a larger square is strongly **NP**-complete [184].

- **Packing harmonic squares in a small rectangle.** Given a set of squares of side lengths $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$ find the smallest rectangle in which these squares can be packed. Several results that improve on the upper bound of the size of such a rectangle are presented [24, 67, 164, 220, 247] with some generalizations to cubes [220].

- **Convex hull packing.** Given two convex polygons in the plane with respective complexities m and n and that are free to translate and rotate, a minimum convex packing of the two polygons is the smallest convex region that they can be packed in. The problem is first solved in linear time without allowing rotations [183]. Tang et al. present an $O(n+m)nm$ algorithm [274]. The problem is solved experimentally using simulated annealing for packing more than two polygons [283].

- **Containment.** Packing a variety of different polygons into some minimal shape is also known as the containment problem. Milenkovic et. al. has studied many variants of this

problem (some of which include strip packing) [95, 224–228]. The variants examined do not have many of the simplistic assumptions that many other papers have, they directly tackle both approximate and exact solutions for packing real polygons (even nonconvex ones) into minimal shapes from various classes (e.g. rectangles). The cases of allowing rotations or translations only are addressed. However, their worst-case run times are all exponential.

- **Packing two polygons into a minimal rectangle.** Alt and Hurtado, inspired by the work of Milenkovic et al. solve the minimum area-rectangle packing problem in polynomial time, but only when there are two polygonal objects to pack [12].

Chapter 4

Congruence

4.1 Introduction

In this chapter, we are interested in partitioning a simple polygon into two congruent components. Symmetry detection algorithms solve problems of the same flavour by detecting all kinds of isometries in a polygon, a set of points or a set of line segments and some classes of polyhedra [101]. Two open problems with unknown complexity were posed by Eades in [101]: the minimum symmetric decomposition (MSD) problem and the minimal symmetric partition (MSP) problem. Given a set D in $\mathbb{R}^d$ ($d \in \{2, 3\}$), the goal is to find a set of symmetric (nondisjoint for MSD and disjoint for MSP) subsets $\{D_1, D_2, \ldots, D_k\}$ of D such that the union of the D_i is D and k is minimum. The following problem is a decision version of a variation of MSP where $k = 2$:

Problem 1. *Given a polygon P with n vertices, compute a partition of P into two (properly or mirror) congruent polygons P_1 and P_2, or indicate such a partition does not exist.*

Eriksson claims to solve the aforementioned problem in $O(n^3)$ time [111]. Rote observes that a careful analysis of Eriksson's algorithm yields a $O(n^3 \log n)$ running time for proper congruence and he shows that the combinatorial complexity of an explicit representation of the solution in the case of mirror congruence cannot be bounded as a function of n [252]. Rote also gives a counterexample where the algorithm fails for a polygon with holes. In

this chapter, we present two algorithms to solve the problem for a simple input polygon P: an $O(n^2 \log n)$ algorithm for properly congruent and possibly nonsimple P_1 and P_2 and an $O(n^3)$ algorithm for mirror congruent and possibly nonsimple polygons P_1 and P_2. In other words, our second algorithm is able to produce solutions unbounded by n in a time polynomial in n using an implicit representation of the output. Since we allow nonsimple polygons as outputs, we use a different definition of partition than the one found in the literature to solve Problem 1.

The chapter is organized as follows. In Section 4.2, we define terms and notation needed for the rest of the chapter. In Section 4.3, we translate into pseudo-code the previous algorithm that solves Problem 1 and we present in detail Rote's critiques of the solution. In the remaining sections, we present our solution of Problem 1 along with some conjectures.

4.2 Preliminaries

Our notions of congruence follow those in [111]. Two polygons are said to be *congruent* if one can be transformed into the other by an isometry, i.e. a transformation that preserves distances. Two polygons are *properly congruent* if they are equivalent up to translations and rotations (see Figure 20) and are *mirror congruent* if they are equivalent up to reflection or glide reflection (see Figure 21). Note that a glide reflection is a reflection followed by a translation parallel to the reflection axis. A reflection along an axis g followed by a rotation or a translation is a reflection around an axis g'. Congruence transforms involving either a translation or rotation $T = (p, \theta)$ may be viewed as a rotation about an arbitrary point p, including points at infinity where θ is the angle of rotation. Let $T^{-1} = (p, -\theta)$. Congruence transforms involving glide reflection are denoted by $T = (g, \mathbf{v})$ where g is the axis of reflection and $\mathbf{v}$ is the vector of translation if any. Let $T^{-1} = (g, -\mathbf{v})$.

Let $\underset{P}{\angle} a$ be the interior angle of a point a on a polygon P. Let $\overline{ab}$ be the line segment with endpoints a and b.

A polyline can be viewed as an alternating sequence of lengths and angles, which always begins and ends with a length. The angles of a polyline are the ones measured in the interior

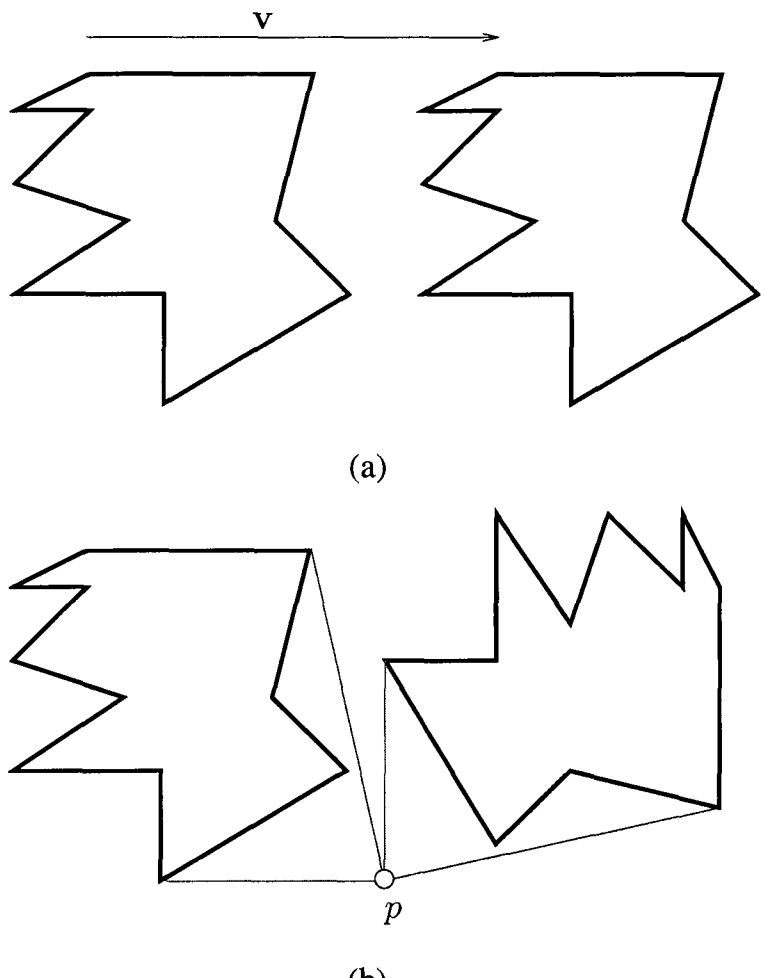

Figure 20: Properly congruent polygons: (a) two translationally congruent polygons with translation vector $\mathbf{v}$, (b) two rotationally congruent polygons with rotation point p.

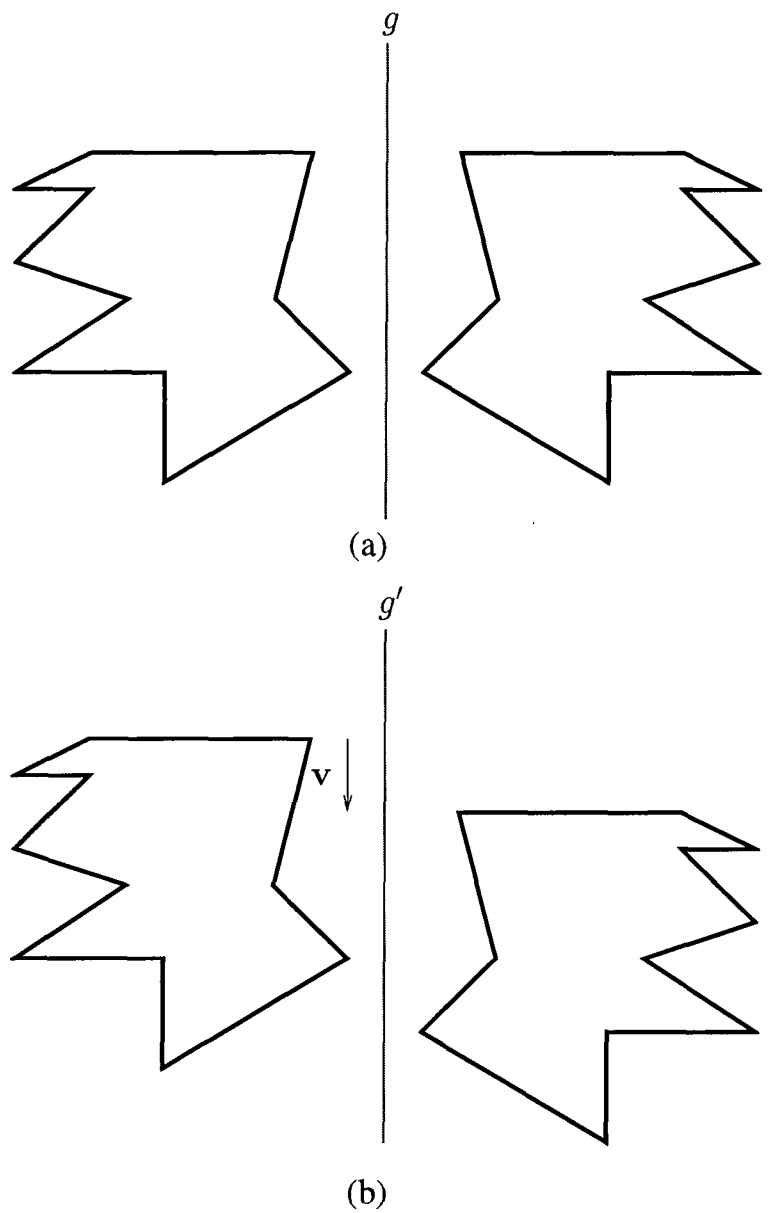

Figure 21: Mirror congruent polygons: (a) two mirror congruent polygons with reflection axis g, (b) two mirror congruent polygons with reflection axis g' and vector $\mathbf{v}$.

of the polygon. Let $P[a\,..\,b]$ be a polyline on the boundary of a polygon P and c be a point on that polyline. $P[a\,..\,b]$ can be written as the concatenation of two polylines and an angle: $P[a\,..\,b] = P[a\,..\,c] + \underset{P}{\angle} c + P[c\,..\,b]$ (where $+$ is the concatenation operator). A polygon P with vertices $\langle p_1, p_2, \ldots, p_n \rangle$ can be viewed as the concatenation of a polyline, an angle, a length and another angle: $P[p_1\,..\,p_n] + \underset{P}{\angle} p_n + |\overline{p_n p_1}| + \underset{P}{\angle} p_1$. A polygon can therefore, be written as the concatenation of several polylines (to which an angle is appended).

Two polylines are congruent if they are represented by the same sequence (see Figure 22). Two polylines are *flip-congruent* if they are represented by the same sequence after replacing all of the angles α_i in one by $2\pi - \alpha_i$ and reversing the order of the sequence (see Figure 23). In other words, two flip-congruent polylines are properly congruent in the pure geometric sense. Two polylines are *mirror congruent* if they are represented by the same sequence after reversing the order of the sequence (see Figure 24).We use $\cong$ to denote proper congruence, $\overset{\text{FLIP}}{\cong}$ to denote flip-congruence and $\overset{\text{MIRROR}}{\cong}$ to denote mirror-congruence. Observe that $P[a\,..\,b] \overset{\text{FLIP}}{\cong} P[b\,..\,a]$.

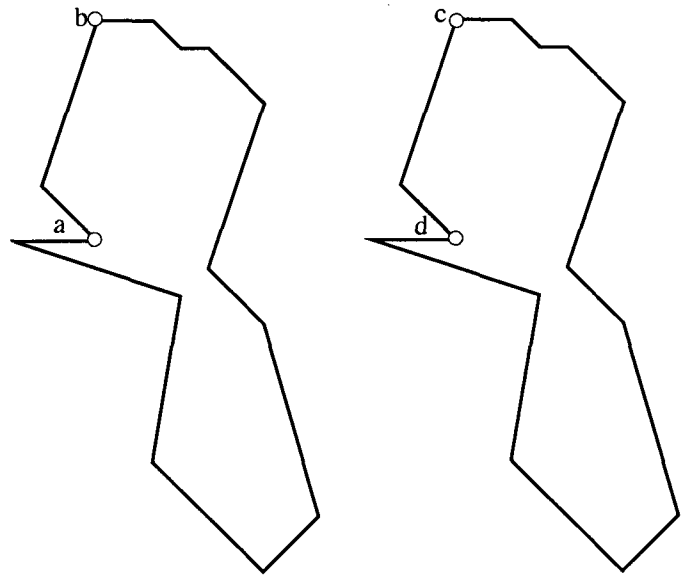

Figure 22: Two translationally congruent polygons P (left) and Q (right) where polylines $P[a\,..\,b]$ and $P[c\,..\,d]$ are congruent.

A simple polygon P is said to be *partitionable* into P_1 and P_2 if there exists two points b and e on δP and a polyline sp connecting them such that $P_1 = P[e\,..\,b] + x + sp + y$

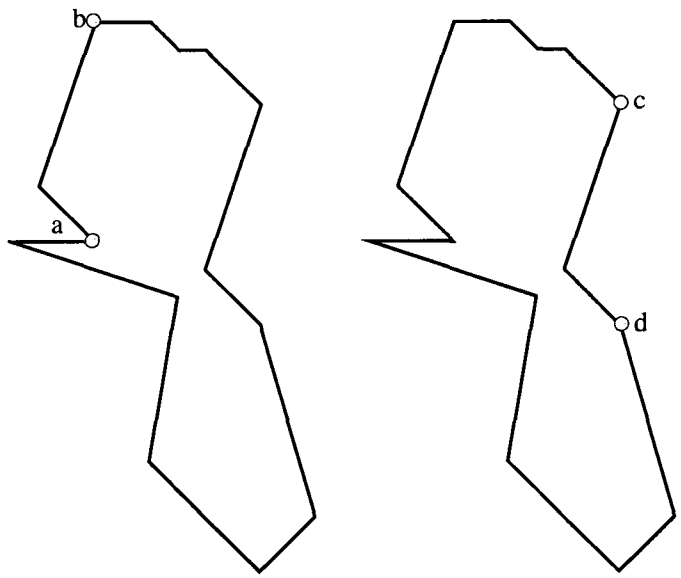

Figure 23: Two translationally congruent polygons P (left) and Q (right) where polylines $P[a\,..\,b]$ and $P[c\,..\,d]$ are flip-congruent.

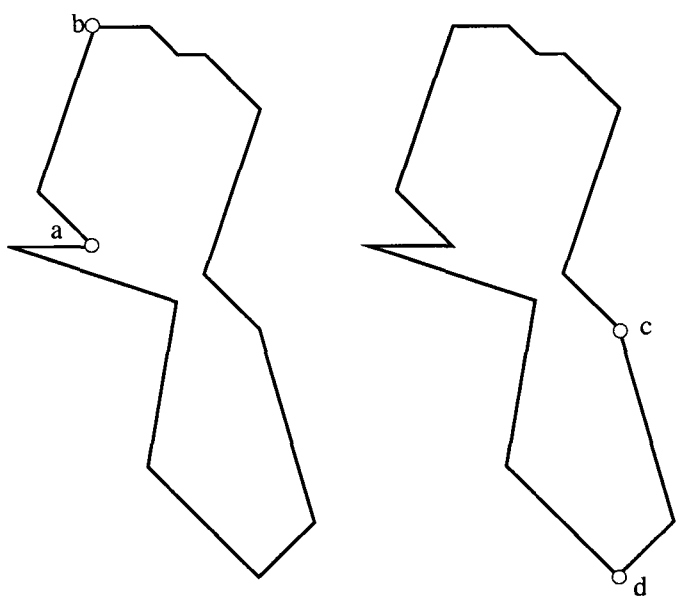

Figure 24: Two translationally congruent polygons P (left) and Q (right) where polylines $P[a\,..\,b]$ and $P[c\,..\,d]$ are mirror congruent.

and $P_2 = P[b\,..\,e] + w + sp + z$ where x, y, w and z are some angles. A partitioning of P, if it exists, is a solution to Problem 1 under this definition of partition and is denoted by $S = (P_1, P_2)$. Components P_1 and P_2 are such that there exists a transformation T_S where $T_S(P_1) = P_2$. We say P_1 and P_2 are congruent if T_S involves rotation and translation such that $T_S(P_1) = P_2$ and mirror congruent otherwise. Using the definition of a polygon as an alternating sequence of lengths and angles (which starts with a length and ends with an angle), components P_1 and P_2 are congruent if and only if their corresponding sequences are cyclic permutations of one another. The polyline sp is called the *split-polyline* and is denoted by Split(S). We are interested in a split polyline that has minimum complexity. When P is symmetric, we call the partition *trivial* and the problem reduces to symmetry detection which has been solved in linear time in [101].

Note if T_S is a reflection it can be determined by one pair of points $(p_i, T_S(p_i))$. If T_S is glide reflection, it can be determined by two pairs of points $(p_i, T_S(p_i))$ and $(p_j, T_S(p_j))$ $(i \neq j)$. We say that two subsets $s_1 \subseteq P_1$ and $s_2 \subseteq P_2$ of congruent polygons P_1 and P_2 are transformationally congruent with respect to congruence transformation T_S if $T_S(s_1) = s_2$. Let $vd(a, b)$ be the vertical distance between the two points a and b. Let $cw_P(a)$ and $ccw_P(a)$ denote respectively the line segments incident to point a clockwise and counterclockwise around δP (P is a simple polygon). A line determined by two points a and b is denoted by (ab). We normalize P to have unit perimeter.

4.3 Eriksson's algorithm

We present Eriksson's algorithm for partitioning a polygon into two congruent components. We have rewritten his algorithm in pseudocode and separate it into two procedures: Algorithm 1 that checks for proper congruence and Algorithm 2 that checks for mirror congruence.

Let us analyse the running time. S' is the set of all vertices and the midpoints of all segments in δP. For every pair of points in S' $(O(n^2))$, both procedures travel around δP $(O(n))$ and check for the intersection of split-polyline with δP $(O(\log n)$ assuming $O(n)$

Algorithm 1 Eriksson's algorithm for partitioning a polygon into two proper congruent components.

Require: A polygon P with vertices $S = \langle p_1, p_2, \ldots, p_n \rangle$
Ensure: Partitions P into proper congruent P_1 and P_2 or indicate that such a partition does not exist

$S' \leftarrow S \cup$ the set of the midpoints of all segments in δP
$Path_1 \leftarrow \emptyset$
$Path_2 \leftarrow \emptyset$
for all pairs (p_i, p_j) of points in S' **do**
  $q \leftarrow p_i$
  $r \leftarrow p_j$
  while $\overline{p_i p_{i+1}} = \overline{p_j p_{j+1}}$ **do** {The two paths are constructed clockwise around δP}
    $Path_1 \leftarrow Path_1 + \overline{p_i p_{i+1}}$
    $Path_2 \leftarrow Path_2 + \overline{p_j p_{j+1}}$
    $p_i \leftarrow p_{i+1}$
    $p_j \leftarrow p_{j+1}$
  end while
  if $Path_1$ and $Path_2$ have reached r and q respectively **then**
    Report that P can be partitionable into $P_1 \leftarrow Path_1$ and $P_2 \leftarrow Path_2$ with the line segment $\overline{rq}$
  else {$Path_2$ adds boundary segments and $Path_1$ add duplicates of the line segments until it intersects the boundary again}
    repeat
      $Path_1 \leftarrow Path_1 + \overline{p_j p_{j+1}}$
      $Path_2 \leftarrow Path_2 + \overline{p_j p_{j+1}}$
      $p_j \leftarrow p_{j+1}$
    until $Path_1$ intersects the boundary again
    Check if the produced cut is valid and report a partition of P into $P_1 \leftarrow Path_1$ and $P_2 \leftarrow Path_2$, if so
  end if
  $p_i \leftarrow q$
  $p_j \leftarrow r$
  while $\overline{p_i p_{i-1}} = \overline{p_j p_{j-1}}$ **do** {The two paths are constructed counterclockwise around δP}
    $Path_1 \leftarrow Path_1 + \overline{p_i p_{i-1}}$
    $Path_2 \leftarrow Path_2 + \overline{p_j p_{j-1}}$
    $p_i \leftarrow p_{i-1}$
    $p_j \leftarrow p_{j-1}$
  end while
  if $Path_1$ and $Path_2$ have reached r and q respectively **then**
    Report that P can be partitionable into $P_1 \leftarrow Path_1$ and $P_2 \leftarrow Path_2$ with the line segment $\overline{rq}$
  else {$Path_2$ adds boundary segments and $Path_1$ add duplicates of the line segments until it intersects the boundary again}
    repeat
      $Path_1 \leftarrow Path_1 + \overline{p_j p_{j-1}}$
      $Path_2 \leftarrow Path_2 + \overline{p_j p_{j-1}}$
      $p_j \leftarrow p_{j-1}$
    until $Path_1$ intersects the boundary again
    Check if the produced cut is valid and report a partition of P into $P_1 \leftarrow Path_1$ and $P_2 \leftarrow Path_2$, if so
  end if
end for
Report that P is not partitionable otherwise.

Algorithm 2 Eriksson's algorithm for partitioning a polygon into two mirror congruent components.

Require: A polygon P with vertices $S = \langle p_1, p_2, \ldots, p_n \rangle$
Ensure: Partitions P into mirror congruent P_1 and P_2 or indicate that such a partition does not exist

$S' \leftarrow S \cup$ the set of midpoints of all segments in δP
$Path_1 \leftarrow \emptyset$
$Path_2 \leftarrow \emptyset$
intersect = false
for all pairs $\langle p_i, p_j \rangle$ of points in S' **do**
  while $\overline{p_i p_{i+1}} = \overline{p_j p_{j-1}}$ and $Path_1 \cap Path_2 = \emptyset$ **do** {The two paths are constructed in opposite direction around δP, say $Path_1$ clockwise and $Path_2$ counterclockwise}
    $Path_1 \leftarrow Path_1 + \overline{p_i p_{i+1}}$
    $Path_2 \leftarrow Path_2 + \overline{p_j p_{j-1}}$
    $p_i \leftarrow p_{i+1}$
    $p_j \leftarrow p_{j-1}$
  end while
  if $\overline{p_i p_{i+1}} \neq \overline{p_j p_{j-1}}$ **then**
    $TempPath_1 \leftarrow Path_1$
    $TempPath_2 \leftarrow Path_2$
    repeat
      $Path_1 \leftarrow Path_1 + \overline{p_j p_{j-1}}$
      $Path_2 \leftarrow Path_2 + \overline{p_j p_{j-1}}$
      $p_j \leftarrow p_{j+1}$
    until $Path_1$ intersects the boundary again
    if $Path_1$ intersects the boundary again and the produced partition is valid **then**
      Report a partition of P where $P_1 \leftarrow Path_1$ and $P_2 \leftarrow Path_2$
    else
      $Path_1 \leftarrow TempPath_1 + P[p_{i+1} \,..\, p_j]$
      $Path_2 \leftarrow TempPath_2$
      repeat
        $Path_1 \leftarrow Path_1 + P[p_{i+1} \,..\, p_j]$
        $Path_2 \leftarrow Path_2 + P[p_{i+1} \,..\, p_j]$
      until $Path_1$ intersects the boundary of P
      Check if the produced partition is valid and report $P_1 \leftarrow Path_1$ and $P_2 \leftarrow Path_2$, if so
    end if
  end if
  Repeat the same construction of $Path_1$ and $Path_2$ exchanging their direction: the former goes counterclockwise and the latter clockwise around δP
end for
Report that P is not partitionable otherwise.

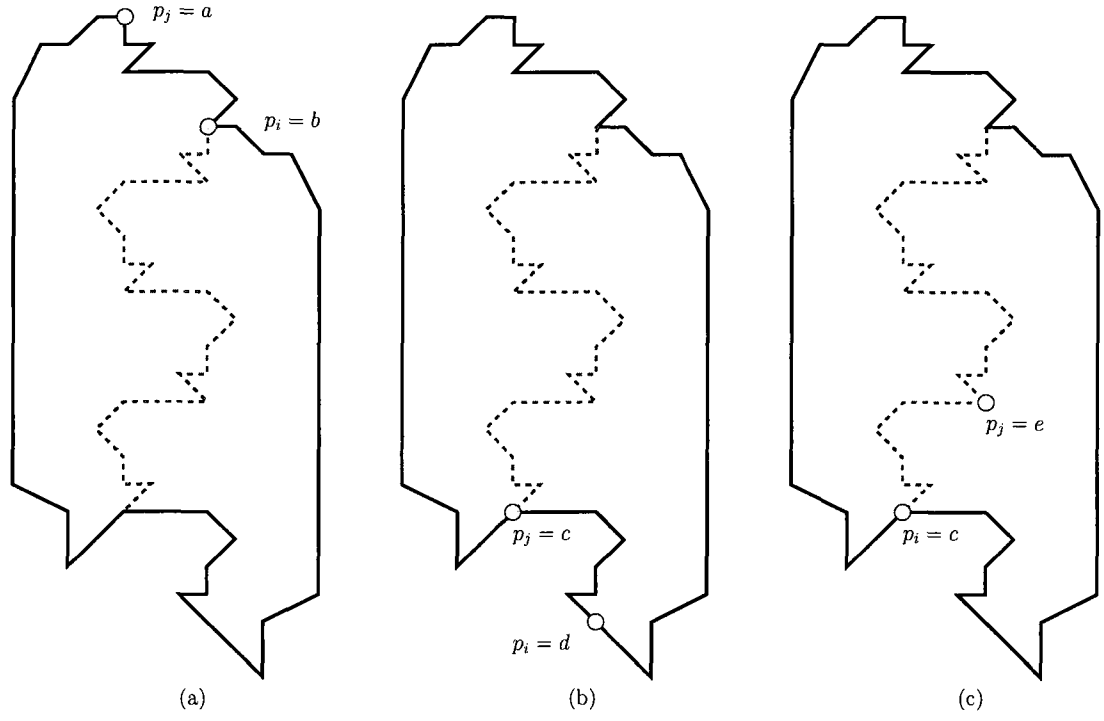

Figure 25: A bad example for Eriksson's algorithm.

preprocessing and space [145]), a total of $O(n^3 \log n)$ running time as opposed to the $O(n^3)$ claimed by Eriksson [252].

In Algorithm 2, every pair of points in S' is considered to trace the congruence of the boundary and to find a potential split polyline to partition P into P_1 and P_2. Consider Figure 25. Suppose that $p_i = a$ and $p_j = b$. (a, b) is a candidate pair that generates a congruent partition of P. Points p_i and p_j "travel" around δP clockwise and counterclockwise respectively until they construct the split polyline and they reach b and a again (when $p_i = d$, $p_j = c$ and when $p_i = c$, $p_j = e$ as shown in Figure 25). As we shall see later in this document, for this case, the split polyline is periodic with period $P[a \,..\, b] + x + T_S(P[a \,..\, b])$ (where x is some angle) and its complexity depends on the vertical distance between two points in δP. Therefore, p_i and p_j might need more than linear time to "travel" around δP and construct the split polyline. This is where Eriksson's analysis of lines 27 to 30 in algorithm 2 fails.

4.4 Preprocessing

Let the length of a polyline $P[a\,..\,b]$ (denoted $d_P(a,b)$) be the sum of the lengths of all the line segments that form this polyline. Given a point $a \in \delta(P)$, we need to locate another $b \in \delta(P)$ such that $d_P(a,b) = x$. Let $d_P^{-1}(a,x)$ be the point b such that $d_P(a,b) = x$. That is, it is the point on δP obtained by walking x units (in the given order of vertices) around δP from a. Note that $d_P^{-1}(a,0.5) = b$ is equivalent to $d_P^{-1}(b,0.5) = a$. We need two preprocessing steps for our algorithms: one to detect the congruence of polylines and the other to, given a point $a \in \delta(P)$, locate another $b \in \delta(P)$ such that $d_P(a,b) = x$.

Congruence of polylines is detected by string matching. Our string representation of polygons and polylines yields Corollary 3.

Theorem 2 ([115]). *Given a string R of length n, an $(n \times n)$ table H of integers in the range $1 \ldots n^2$ can be computed in time $O(n^2)$ such that $H_{i,j} = H_{k,l}$ iff $R_{i,j} = R_{k,l}$ where $R_{i,j}$ is the substring of R from the ith to the jth character.*

Corollary 3. *Given a simple polygon P, with $O(n^2)$ preprocessing and space, queries of the form $P[a\,..\,b] \overset{?}{\cong}_{MIRROR} P[c\,..\,d]$ and $P[a\,..\,b] \overset{?}{\cong}_{FLIP} P[c\,..\,d]$ can be answered in constant time.*

Theorem 4. *Given a polygon $P = \langle p_1, \ldots, p_n \rangle$, with $O(n)$ preprocessing and space, the functions $d_P(a,b)$ and $d_P^{-1}(a,x)$ can be computed in constant time if the points a and b are vertices of the given polygon, and in $O(\log n)$ time if they are not, using standard point location techniques [239].*

Proof. Let $D_P[1\,..\,n]$ be a vector of distances and let $D_P[1] = 0$. For every vertex p_i of P, we store its distance around δP from p_1; $D_P[i] = D_P[i-1] + |\overline{p_i - 1 p_i}|$. Given two vertices a and b of the polygon P with respective indices k and l, $d_P(a,b) = D[l] - D[k]$ if $k \leq l$ and $d_P(a,b) = 1 - (D[l] - D[k]$ otherwise. If a and b are not vertices then we locate in $O(\log n)$ time the segments they belong to on the boundary using standard point location techniques [239] and we calculate $d_P(a,b)$ similarly. Let D_P^{-1} be a hash table of the distances in D_P. Given a point a and a distance x, consider that a is a vertex (if it is not locate it in $O(\log n)$ time using standard techniques) and let b the point which is at distance x from a.

We look $D_P[a] + x$ up in the hash table. If b is a vertex, we find it in D_P^{-1} in constant time. Otherwise, locate b in $O(\log n)$ time by binary searching in D_P. Both the distance vector and the hash table take linear time to construct and linear space to store. □

4.5 Proper congruence

In this section, we assume that if a solution exists then the transformation involves a rotation or a translation and is defined by $T_S = (p, \theta)$. Let b and e denote the endpoints of the split-polyline Split(S), if it exists.

Lemma 5. *Assume that P can be nontrivially partitioned into two properly congruent polygons then there is a solution $S = (P_1, P_2)$ such that either $P_1[b\,..\,e]$ is disjoint from the polyline $T_S\,(P_1[b\,..\,e])$ or $P_1[b\,..\,e]$ must be a single line segment.*

Proof. Let $S = (P_1, P_2)$ where P_1 and P_2 are chosen to minimize the length of Split(S). Assume that $P_1[b\,..\,e]$ overlaps (fully or partially) with $T\,(P_1[b\,..\,e])$, i.e. $T\,(P_1[b\,..\,e])$ overlaps with $P_2[e\,..\,b]$. Let x be a point on $P_1[b\,..\,e]$ and y be a point on $T_S\,(P_1[b\,..\,e])$ such that $y = T_S(x)$ and $x \neq y$. Cut the corners at x and y identically and the transformation is still preserved. A contradiction to the fact that the split-polyline was the shortest possible. □

The section is organized as follows. We first show the necessary conditions for the existence of a solution in Lemma 6, namely that a solution $S = (P_1, P_2)$ can be specified by a sextuple of points on δP satisfying some properties. In Lemma 7, we show how to verify if a given sextuple specifies a solution to Problem 1 or not. In Lemma 8, we show how, given two points of a solution sextuple, we can find the rest of the points in the sextuple. Finally, in Theorem 9, given that (by Lemma 6) at least four points of a solution sextuple are vertices, we present an $O(n^2 \log n)$ algorithm that solves Problem 1 for the case discussed in this section.

For Lemmas 6, 7 and 8, assume that P can be nontrivially partitioned into two congruent components P_1 and P_2 where $S = (P_1, P_2)$ and let $c = T_S(b)$, $d = T_S(e)$, $a = T_S^{-1}(b)$,

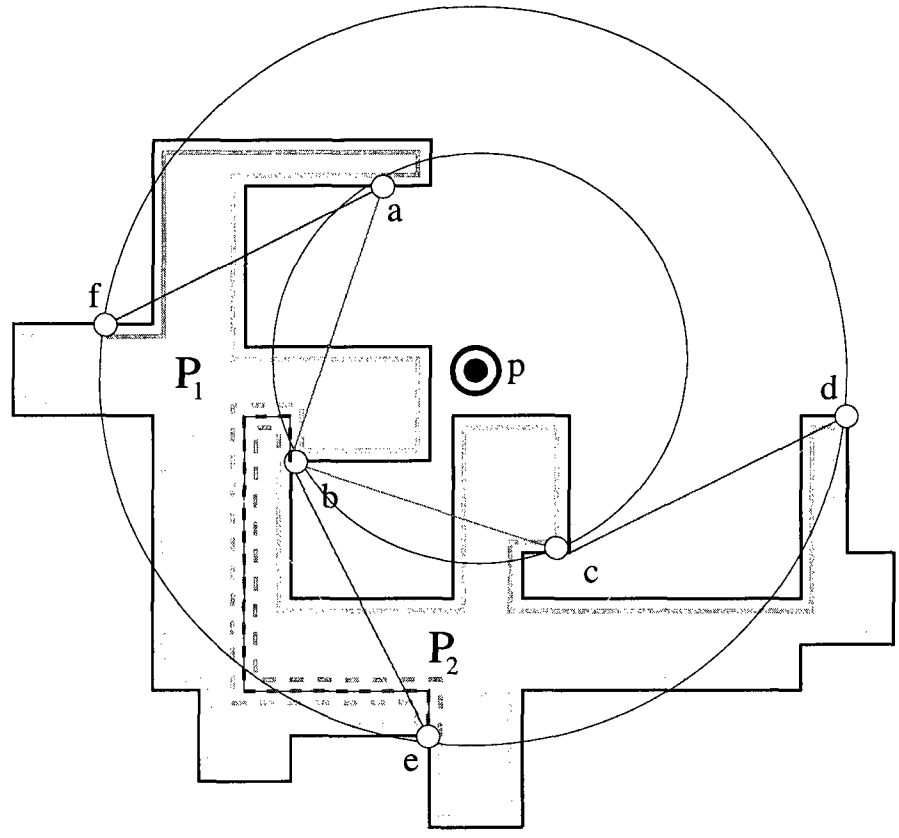

Figure 26: A simple polygon partitioned into two simple rotationally congruent components.

and $f = T_S^{-1}(e)$. Figures 26 and 28 show two polygons respectively partitioned into two rotationally and translationally congruent simple components P_1 and P_2. Figure 28 shows a polygon partitioned into two translationally congruent nonsimple components P_1 and P_2; Figure 29 shows the details of the partition.

Lemma 6. *The following facts hold (see Figures 26, 27 and 28):* (a, b, c, d, e, f) *appear clockwise order on* δP; $P[f \,..\, a] \cong P_2[e \,..\, b]$; $P[c \,..\, d] \cong P_1[b \,..\, e]$; $P[a \,..\, b] \cong P[b \,..\, c]$; $P[d \,..\, e] \cong P[e \,..\, f]$; $P[f \,..\, a] \overset{FLIP}{\cong} P[c \,..\, d]$; $\angle_{P_1} a + \angle_{P_2} c = \angle_{P_1} b + \angle_{P_2} b$; $\angle_{P_1} f + \angle_{P_2} d = \angle_{P_1} e + \angle_{P_2} e$ *and at least two of the points in* $\{a, b, c\}$ *and two of the points in* $\{d, e, f\}$ *are vertices of* P.

Proof. Since, by definition, $P_1[b \,..\, e]$ is a subset of the boundary of P_1 then its flip-congruent polyline $P_2[e \,..\, b]$ is a subset of the boundary of P_2. By Lemma 5, $T_S(P_1[b \,..\, e])$ and

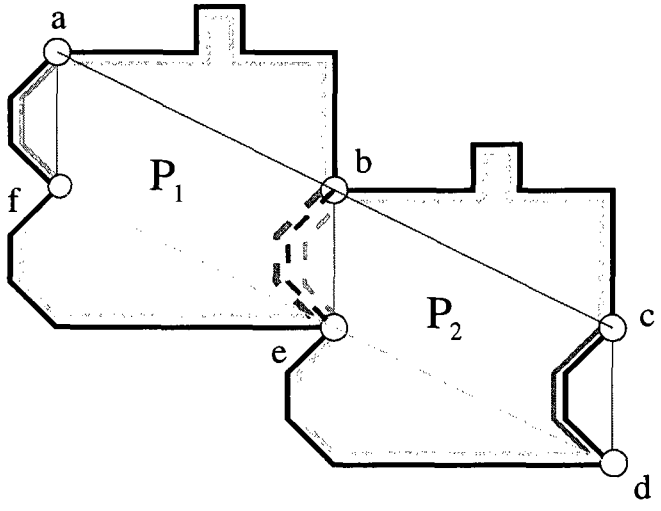

Figure 27: A simple polygon partitioned into two simple translationally congruent components.

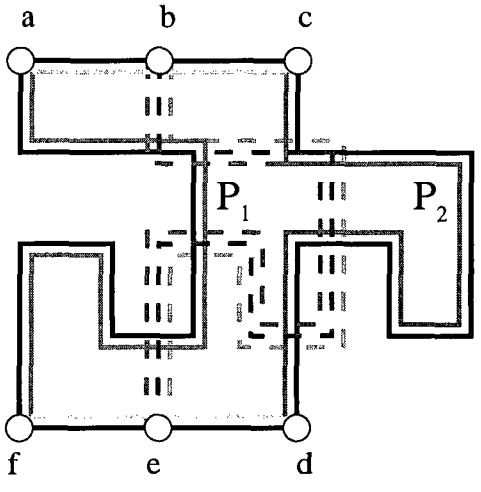

Figure 28: A simple polygon partitioned into nonsimple translationally congruent components.

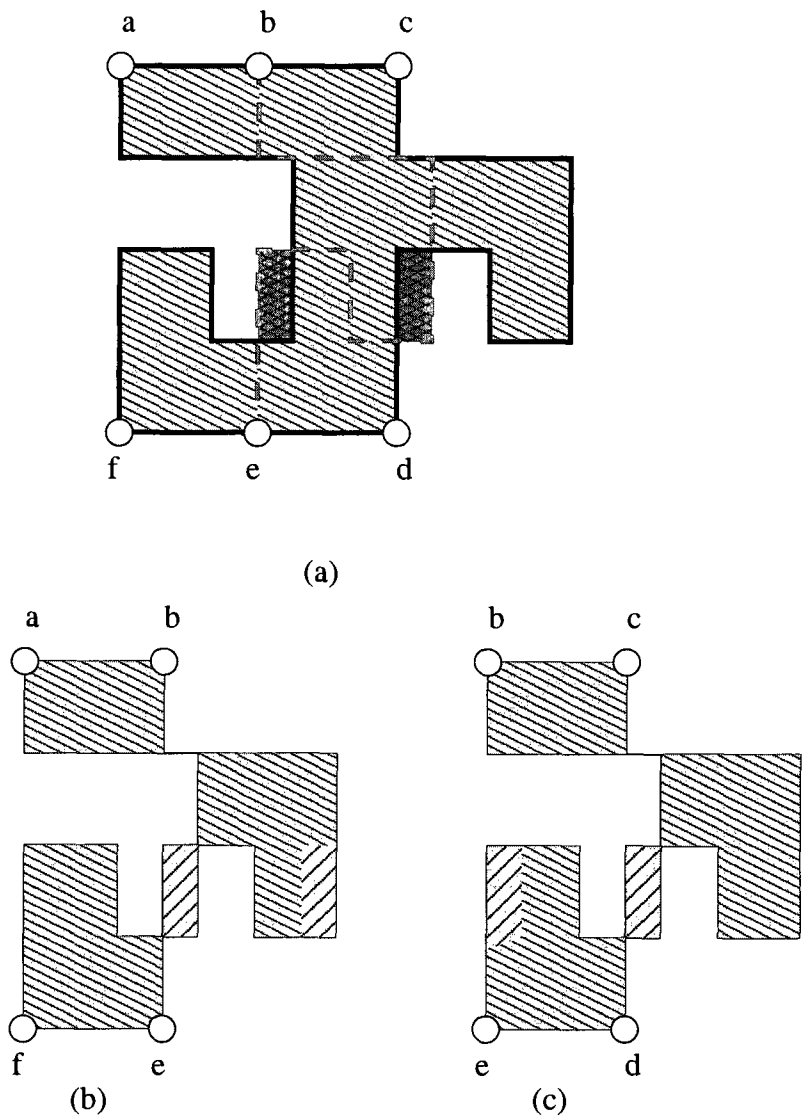

Figure 29: A simple polygon partitioned into nonsimple translationally congruent components.

$T_S(P_2[e \,..\, b])$ are subsets of δP. Therefore, points a, b, c, d, e, f belong to δP and their order is implied by T_S. Given that $a = T_S^{-1}(b)$ and $f = T_S^{-1}(e)$, we can immediately conclude that the image of $P[f \,..\, a]$ by transformation T_S is $P_1[b \,..\, e]$. Similar arguments apply to prove that $P[c \,..\, d] \cong P_1[b \,..\, e]$, $P[a \,..\, b] \cong P[b \,..\, c]$ and $P[d \,..\, e] \cong P[e \,..\, f]$. Since polylines $P[f \,..\, a]$ and $P[c \,..\, d]$ are respectively congruent to $P_1[b \,..\, e]$ and $P_2[e \,..\, b]$, then $P[f \,..\, a] \overset{\text{FLIP}}{\cong} P[c \,..\, d]$. Also, by preservation of angles, both angles $\underset{P_1}{\angle} a$ and $\underset{P_2}{\angle} c$ are congruent to $\underset{P}{\angle} b$. Since $b \in \delta P_1$ and $b \in \delta P_2$, then $\underset{P_1}{\angle} a = \underset{P_1}{\angle} b$ and $\underset{P_2}{\angle} c = \underset{P_2}{\angle} b$. It follows that $\underset{P_1}{\angle} a + \underset{P_2}{\angle} c = \underset{P_1}{\angle} b + \underset{P_2}{\angle} b$. Similarly, it can be shown that $\underset{P_1}{\angle} f + \underset{P_2}{\angle} d = \underset{P_1}{\angle} e + \underset{P_2}{\angle} e$. It follows that at least two of the points in $\{a, b, c\}$ and two of the points in $\{d, e, f\}$ are vertices of P. □

Lemma 7. *Given the preprocessing in Corollary 3 and the positions of six points (a, b, c, d, e, f) on δP, it can be checked that the points specify a solution $S = (P_1, P_2)$ to Problem 1 in constant time.*

Proof. P_1 and P_2 are properly congruent if their respective boundaries are properly congruent. Component P_1 is composed (in order) of the following alternation of polylines and angles: $P[a \,..\, b]$, $\underset{P_1}{\angle} b$, $P_1[b \,..\, e]$, $\underset{P_1}{\angle} e$, $P[e \,..\, f]$, $\underset{P}{\angle} f$, $P[f \,..\, a]$ and $\underset{P}{\angle} a$. Component P_2 is composed (in order) of the following alternation of polylines and angles: $P[b \,..\, c]$, $\underset{P}{\angle} c$, $P[c \,..\, d]$, $\underset{P}{\angle} d$, $P[d \,..\, e]$, $\underset{P_2}{\angle} e$, $P_2[e \,..\, b]$ and $\underset{P_2}{\angle} b$. Hence, if $P[a \,..\, b] \cong P[b \,..\, c]$, $P_1[b \,..\, e] \cong P[c \,..\, d]$, $P[e \,..\, f] \cong P[d \,..\, e]$, $P[f \,..\, a] \cong P_2[e \,..\, b]$, $|\overline{af}| = |\overline{be}| = |\overline{cd}|$, $\underset{P}{\angle} b = \underset{P}{\angle} a + \underset{P}{\angle} c$ and $\underset{P}{\angle} e = \underset{P}{\angle} d + \underset{P}{\angle} f$, the sextuple represents a congruent partition of P since the boundaries of P_1 and P_2 consist of respectively congruent polylines and copying the reversed string representation of $P[f \,..\, a]$ onto $\overline{be}$ is possible. Otherwise, the sextuple does not represent a solution to Problem 1. This verification can be done in constant time by Corollary 3. □

Lemma 8. *The points $\{a, b, c, d, e, f\}$ are as defined in Lemma 6. Given the position of two points of $\{a, b, c\}$ or $\{d, e, f\}$ and the preprocessing in Theorem 4, the positions of all six points (a, b, c, d, e, f) can be computed in $O(\log n)$ time.*

Proof. Suppose without loss of generality that the given pair is (a, b). The arguments are similar if (b, c), (a, c), (d, e), (d, f) or (e, f) is given. Since $P_1[b \,..\, e]$ is the split-polyline, it

cuts the perimeter into two equal components thus, $e = d_P^{-1}(b, 0.5)$. By the congruence of $P[a \,..\, b]$ and $P[b \,..\, c]$ and since a, b and c are consecutive on δP, we obtain the following equality $d_P^{-1}(b, d_P(a, b)) = c$ (in constant time if c is a vertex and in $O(\log n)$ time otherwise). From the properties of the transformation T_S, we can conclude that the circles $\bigcirc abc$ and $\bigcirc def$ are concentric with center p. Calculating p, $d = T_S(e)$ and $f = T_S^{-1}(e)$ are all constant-time operations. □

Theorem 9. *Given a simple polygon P and given that T_S, if it exists, is either a rotation or a translation, a solution $S = (P_1, P_2)$ to Problem 1 can be found in $O(n^2 \log n)$ time if and only if P can be partitioned into two congruent polygons.*

Proof. For every pair of vertices of P, we verify if it is (a, b), (b, c), (a, c), (d, e), (d, f) or (f, e) by computing the four remaining points as shown in Lemma 8. We then verify that the obtained sextuple specifies a solution to the problem as stated in Lemma 7 and if it does then copying $P[f \,..\, a]$ onto $\overline{be}$ such that $\underset{P}{\angle} a = \underset{P_2}{\angle} b$ (and $\underset{P}{\angle} c = \underset{P_1}{\angle} b$) yields a valid partition. The "if" part is trivial. The "only if" part stems from the previous lemmas; if the polygon is partitionable then by Lemma 6, (a, b, c, d, e, f) exist and appear in that order on δP. Note that the algorithm allows for degeneracies (a, b and c may be colocated for example). The split-polyline is by construction congruent to $P[f \,..\, a]$ and $P[c \,..\, d]$. The string matching checks for congruence of $P[a \,..\, b]$ and $P[b \,..\, c]$ and for congruence of $P[d \,..\, e]$ and $P[e \,..\, f]$. Therefore, P_1 and P_2 having the same polylines forming them, are congruent. The split-polyline might however intersect δP. This will result in two congruent sub-polygons P_1 and P_2 that might be nonsimple. Since we check every pair of vertices in P and we locate the four remaining points in $O(\log n)$ time for each pair, the algorithm runs in $O(n^2 \log n)$ time. □

4.6 Mirror congruence

In this section, we assume that if a solution exists then the transformation involves a glide reflection and is defined by $T_S = (g, \mathbf{v})$. Let b and e denote the endpoints of the split polyline Split(S), if it exists.

Lemma 10. *Assume that P can be nontrivially partitioned into two mirror congruent polygons then there is solution $S = (P_1, P_2)$ such that either $P_1[b\,..\,e]$ is disjoint from the polyline $T_S\,(P_1[b\,..\,e])$, $T_S\,(P_1[b\,..\,e])$ partially overlaps with $P_1[b\,..\,e]$, or $P_1[b\,..\,e]$ and $P_2[e\,..\,b]$ are line segments.*

Proof. Suppose that $T_S\,(P_1[b\,..\,e]) = P_2[e\,..\,b]$. We know that by definition $P_1[b\,..\,e] \overset{\text{FLIP}}{\cong} P_2[e\,..\,b]$. Therefore, the polyline $P_1[b\,..\,e]$ and its flip congruent $P_2[e\,..\,b]$ are mirror congruent which cannot happen unless $P_1[b\,..\,e]$ and $P_2[e\,..\,b]$ are line segments. □

In Section 4.6.1, we present an algorithm for the case where Split(S) is disjoint from T_S (Split(S)) (see Figure 30) and in Section 4.6.2, we present an algorithm for the case where they partially overlap (see Figure 31).

4.6.1 Disjoint split-polyline

In this section, we assume that if a solution exists then the split-polyline Split(S) is disjoint from its mirror image by the transformation T_S. We first show the necessary conditions for the existence of a solution in Lemma 11, namely that a solution $S = (P_1, P_2)$ can be specified by a sextuple of points on δP satisfying some properties. In Lemma 12, we show how to verify whether a given sextuple specifies a solution to Problem 1 or not. In Lemma 13, we show how, given two points of a solution sextuple, we can find the rest of the points in the sextuple in $O(\log n)$ time except when both the endpoints of Split(S) are not vertices of P. Given that (by Lemma 11) at least four points of a solution sextuple are vertices, we present an $O(n^3)$ time algorithm in Lemma 14 and Theorem 15 to solve the problem for the case discussed in this section.

For Lemmas 11, 12, 13, 14 and 25, assume that P can be nontrivially partitioned into two mirror congruent polygons P_1 and P_2 where $S = (P_1, P_2)$ and Split(S) is disjoint from T_S (Split(S)) and let $d = T_S(b)$, $c = T_S(e)$, $f = T_S^{-1}(b)$, and $a = T_S^{-1}(e)$.

Lemma 11. *The following facts hold (see Figure 30): (a, b, c, d, e, f) appear in clockwise order on δP; $P[f\,..\,a] \overset{\text{MIRROR}}{\cong} P_2[e\,..\,b]$; $P[c\,..\,d] \overset{\text{MIRROR}}{\cong} P_1[b\,..\,e]$; $P[a\,..\,b] \overset{\text{MIRROR}}{\cong} P[d\,..\,e]$; $P[b\,..\,c] \overset{\text{MIRROR}}{\cong} P[e\,..\,f]$; $P[f\,..\,a] \overset{\text{FLIP}}{\cong} P[c\,..\,d]$; $\underset{P}{\angle} a + \underset{P}{\angle} c = \underset{P_1}{\angle} e + \underset{P_2}{\angle} e$; $\underset{P}{\angle} f + \underset{P}{\angle} d = \underset{P_1}{\angle} b + \underset{P_2}{\angle} b$;*

at least two of the points in $\{a, c, e\}$ *and two of the points in* $\{b, d, f\}$ *are vertices of* P*;* $d_P^{-1}(a, 0.5) = d$*;* $d_P^{-1}(b, 0.5) = e$*; and* $d_P^{-1}(c, 0.5) = f$.

Proof. Given that $P_1[b \,..\, e]$ is a subset of δP_1 then its flip-congruent polyline $P_2[e \,..\, b]$ is a subset of δP_2. We also know that $T_S(P_1[b \,..\, e])$ and $T_S(P_2[e \,..\, b])$ are subsets of δP. Therefore, the order of $\{a, b, c, d, e, f\}$ around δP is implied by T_S. Since $a = T_S^{-1}(e)$ and $f = T_S^{-1}(b)$, then the image of $P[f \,..\, a]$ by transformation T_S is $P_2[e \,..\, b]$. Similarly, we show that $P[c \,..\, d] \overset{\text{MIRROR}}{\cong} P_1[b \,..\, e]$, $P[a \,..\, b] \overset{\text{MIRROR}}{\cong} P[d \,..\, e]$ and $P[b \,..\, c] \overset{\text{MIRROR}}{\cong} P[e \,..\, f]$. Since polylines $P[f \,..\, a]$ and $P[c \,..\, d]$ are, respectively, mirror congruent to $P_2[e \,..\, b]$ and $P_1[b \,..\, e]$ and since the mirror images of two flip-congruent polylines are themselves flip-congruent, then $P[f \,..\, a] \overset{\text{FLIP}}{\cong} P[c \,..\, d]$. Given that $a = T_S^{-1}(e)$ and $c = T_S(e)$, then $\underset{P}{\angle} a = \underset{P_2}{\angle} e$ and $\underset{P}{\angle} c = \underset{P_1}{\angle} e$. It follows that $\underset{P}{\angle} a + \underset{P}{\angle} c = \underset{P_1}{\angle} e + \underset{P_2}{\angle} e$. Similarly, we show that $\underset{P}{\angle} f + \underset{P}{\angle} d = \underset{P_1}{\angle} b + \underset{P_2}{\angle} b$. It follows that at least two of the points in $\{a, c, e\}$ and two of the points $\{b, d, f\}$ are vertices of P. Assume the preprocessing described in Theorem 4. Observe that the sequence of lengths in polyline $P[a \,..\, d]$ is composed of the sequence of lengths in polylines $P[a \,..\, b]$, $P[b \,..\, c]$ and $P[c \,..\, d]$, and also the sequence of lengths in polyline $P[d \,..\, a]$ is composed of the sequence of lengths in polylines $P[d \,..\, e]$, $P[e \,..\, f]$ and $P[f \,..\, a]$. Hence, by the respective congruence of these polylines, given the position of a, the position of d can be determined n $O(\log n)$ time by computing $d_P^{-1}(a, 0.5) = d$. The sequence of lengths in polyline $P[b \,..\, e]$ is composed of the sequence of lengths in polylines $P[b \,..\, c]$, $P[c \,..\, d]$ and $P[d \,..\, e]$, and also the sequence of lengths in polyline $P[e \,..\, b]$ is composed of the sequence of lengths in polylines $P[e \,..\, f]$, $P[f \,..\, a]$ and $P[a \,..\, b]$. Hence, by the respective congruence of these polylines, given the position of e, the position of b can be determined in $O(\log n)$ time by computing $d_P^{-1}(e, 0.5) = b$. The sequence of lengths in polyline $P[c \,..\, f]$ is composed of the sequence of lengths in polylines $P[c \,..\, d]$, $P[d \,..\, e]$ and $P[e \,..\, f]$, and also the sequence of lengths in polyline $P[f \,..\, c]$ is composed of the sequence of lengths in polylines $P[f \,..\, a]$, $P[a \,..\, b]$ and $P[b \,..\, c]$. Hence, by the respective congruence of these polylines, given the position of c, the position of f can be determined in $O(\log n)$ time by computing $d_P^{-1}(c, 0.5) = f$. □

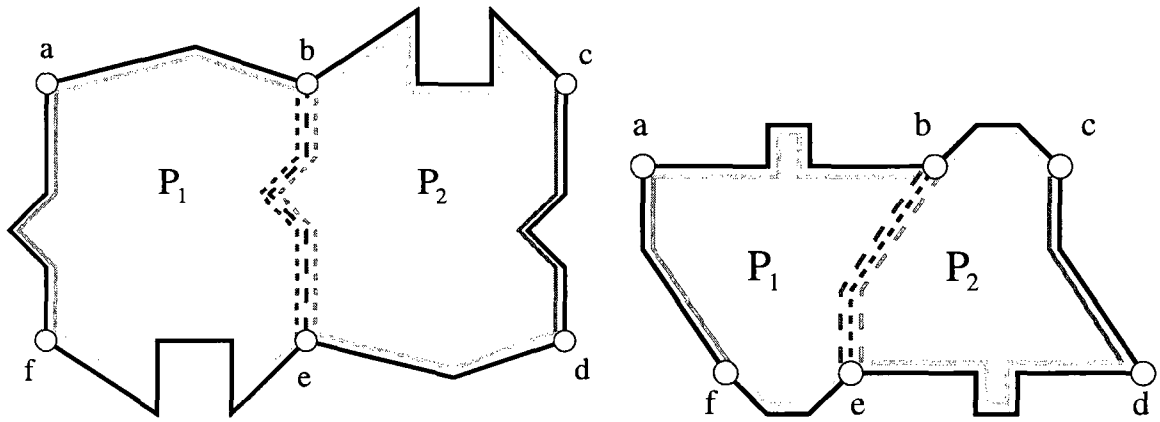

Figure 30: Polygons partitioned into two simple mirror-congruent components with a nonoverlapping split-polyline.

Lemma 12. *Given the preprocessing in Lemma 3 and the positions of six points* (a, b, c, d, e, f) *on* δP*, it can be checked that the points specify a solution* $S = (P_1, P_2)$ *for the disjoint split polyline case of Problem 1 in constant time.*

Proof. P_1 and P_2 are mirror congruent if their respective boundaries are mirror congruent. ComponentP_1 is composed (in order) of the following alternation of polylines and angles: $P[a\,..\,b]$, $\underset{P_1}{\angle} b$, $P_1[b\,..\,e]$, $\underset{P_1}{\angle} e$, $P[e\,..\,f]$, $\underset{P}{\angle} f$, $P[f\,..\,a]$ and $\underset{P}{\angle} a$. Component P_2 is composed (in order) of the following alternation of polylines and angles: $P[d\,..\,e]$, $\underset{P_2}{\angle} e$, $P_2[e\,..\,b]$, $\underset{P_2}{\angle} b$, $P[b\,..\,c]$, $\underset{P}{\angle} c$, $P[c\,..\,d]$ and $\underset{P}{\angle} d$. Hence, if $P[a\,..\,b] \overset{\text{MIRROR}}{\cong} P[d\,..\,e]$, $P[b\,..\,c] \overset{\text{MIRROR}}{\cong} P[e\,..\,f]$, $P[f\,..\,a] \overset{\text{FLIP}}{\cong} P[c\,..\,d]$, $|\overline{af}| = |\overline{be}| = |\overline{cd}|$, $\underset{P}{\angle} b = \underset{P}{\angle} d + \underset{P}{\angle} f$ and $\underset{P}{\angle} e = \underset{P}{\angle} a + \underset{P}{\angle} c$, the sextuple specifies a solution of the problem since the boundaries of P_1 and P_2 consist of respectively congruent polylines and copying the reversed string representation of $P[f\,..\,a]$ onto $\overline{be}$ is possible. Otherwise, the sextuple does not represent a congruent partition of P. This verification can be done in constant time by Corollary 3. □

Lemma 13. *The points* $\{a, b, c, d, e, f\}$ *are as defined in Lemma 11. Given the position of two points of* $\{a, c, e\}$ *or* $\{b, d, f\}$ *and the preprocessing in Theorem 4, the positions of all six points* $\{a, b, c, d, e, f\}$ *can be computed in* $O(\log n)$ *time except in the case where both* b *and* e *are not vertices of* P*.*

Proof. If (a, e) are vertices of P, then the positions of d and b are given in $O(\log n)$ time by Lemma 11. The pairs (a, e) and (b, d) form two pairs of points and their respective mirror images by T_S and hence, they are sufficient to compute the glide reflection T_S. Since $c = T_S(e)$ and $f = T_S^{-1}(b)$, c and f can then be found in constant time. Similarly, if (c, e) are vertices of P, b and f can be found in $O(\log n)$ time by Lemma 11. The pairs (b, f) and (c, e) are also two pairs of points and their mirror images and a and d can be found in constant time. If both a and e are not vertices and if both c and e are not vertices of P, then (a, c) are vertices and then d and f can be found in $O(\log n)$ time. Symmetrically, if pairs (b, d) or (b, f) are vertices, then we can find all six points. Else, similarly, (d, f) are vertices and we can then compute a and c in $O(\log n)$ time. However, in both cases ((a, c) or (d, f) are vertices) none of the obtained points form a pair of a point and its mirror image by T_S. Therefore, the remaining case is when $\{a, c, d, f\}$ are vertices of P and both b and e are not (see Lemma 14). □

An $O(n^3)$ algorithm

Lemma 14. *Given the positions of $\{a, c, d, f\}$, the fact that both b and e are not vertices (equivalent to $\{a, c, d, f\}$ being all vertices by Lemma 11) and the preprocessing in Corollary 3, the positions of b and e can be computed in $O(n)$ time.*

Proof. Since b, in this case, is not a vertex of P, $cw(b)$ and $ccw(b)$ have the same slope. Since $T_S(ccw(f)) = cw(b)$ and $T_S^{-1}(cw(d)) = ccw(b)$, then $cw_P(d)$ and $ccw_P(f)$ have the same slope. Similarly for $cw_P(a)$ and $ccw_P(c)$. For every segment s in δP such that s has the same slope as $cw_P(a)$ and $ccw_P(c)$, compute the potential b and e (the distances of b from the endpoints of the line segment that contains b should be equal respectively to $|cw_P(a)|$ and $|ccw_P(c)|$ and e is half the perimeter away from b) and check, in constant time, the congruence of polylines as stated in Lemma 12. □

Theorem 15. *Given a simple polygon P and given that* Split(S), *if it exists, is disjoint from T_S* (Split(S)), *a solution $S = (P_1, P_2)$ to Problem 1 can be found in $O(n^3)$ time if and only if P can be partitioned into two mirror congruent polygons.*

Proof. For every pair of vertices of P, we verify it is (a, e) or (c, e) or (b, d) or (b, f) by computing the four remaining points as shown in Lemma 13. We verify the solution as stated in Lemma 12. If none of the previous pairs form a pair of vertices then by Lemma 13, $\{a, c, d, f\}$ are vertices and both b and e are not. For every pair of vertices of P, we consider it is either (a, c) or (d, f) and we compute b and e as discussed in Lemma 14. If the verification succeeds in one the cases, then copying $P[f \,..\, a]$ onto $\overline{be}$ such that $\underset{P}{\angle} a = \underset{P_2}{\angle} e$ (and $\underset{P}{\angle} c = \underset{P_1}{\angle} e$) yields a valid partition. The "if" part is trivial. The "only if" part stems from the previous Lemmas; if the polygon is partitionable then by Lemma 11, a, b, c, d, e, f exist and appear in that order on δP. The split-polyline is by construction congruent to $P[f \,..\, a]$ and $P[c \,..\, d]$. The string matching checks for congruence of $P[a \,..\, b]$ and $P[d \,..\, e]$ and for congruence of $P[b \,..\, c]$ and $P[e \,..\, f]$. Therefore, P_1 and P_2, having the same polylines defining them, are congruent. The split-polyline might however intersect δP. This will result in two congruent sub-polygons P_1 and P_2 that are nonsimple. Since we check every pair of vertices in P and we locate the four remaining points in $O(n)$ time for each pair, the algorithm runs in $O(n^3)$ time. □

4.6.2 Partial overlap

In this section, we assume that if a solution S exists then the split-polyline Split(S) is partially overlapping with its mirror image by the transformation T_S. We first prove a sufficient condition for the periodicity of a string (representing a polyline) needed for the rest of the section. We then show the necessary conditions for the existence of a solution in Lemma 17, namely that a solution $S = (P_1, P_2)$ can be specified by a sextuple of points on δP that obey one of two sets of properties (which we call case 1 and case 2). In Lemma 18, we show how to verify if a given sextuple specifies a solution to Problem 1 or not. In Lemmas 19 and 20, we show how, in each one of the two cases, given two points of a solution sextuple, we can find the rest of the sextuple points. Finally, in Theorem 21, given that (by Lemma 17) at least four points of a solution sextuple are vertices, we present an $O(n^3)$ time algorithm that solves Problem 1 for the case discussed in this section.

For Lemmas 17, 18, 19 and 20, assume that P can be nontrivially partitioned into

two mirror congruent polygons P_1 and P_2 where Split(S) is partially overlapping with T_S (Split(S)) and let $T_S(e) = c$, $T_S^{-1}(b) = f$. Assume without loss of generality that the axis of glide reflection g is vertical.

Lemma 16. *Let R be a string representing a polyline, let $m(R)$ denote the representation of the mirror image of this polyline by some glide reflection and let* substr(R, i, j) *denote a substring of R from index i to index j. Given a string R such that $R = r_1 m(r_1) r_2 r_3$ for some strings r_1, r_2 and r_3 where $|r_2| \geq |r_1|$ and such that the substrings $m(r_1)r_2r_3$ and $r_1 m(r_1) r_2$ represent a polyline and the reverse of its mirror image, then R is a periodic string with period $|r_1|$.*

Proof. Given that the substrings $m(r_1)r_2r_3$ and $r_1 m(r_1) r_2$ represents a polyline and the reverse of its mirror image, then:

$$m(m(r_1)r_2r_3) = r_1 m(r_1) r_2$$

which implies that:

$$r_1 m(r_2) m(r_3) = r_1 m(r_1) r_2$$

Removing r_1, we obtain:

$$m(r_2) m(r_3) = m(r_1) r_2$$

Since $|r_2| \geq |r_1|$ and by doing the appropriate replacement of strings, we get:

$$m(r_1)\,\text{substr}(m(r_2), |m(r_1)|-1, |m(r_2)|-1) m(r_3) = m(r_1)\,\text{substr}(r_2, 0, |r_2|-|m(r_3)|-1) m(r_3)$$

Hence,

$$\text{substr}(m(r_2), |m(r_1)| - 1, |m(r_2)| - 1) = \text{substr}(r_2, 0, |r_2| - |m(r_3)| - 1)$$

Therefore r_2 (and hence R) is a periodic string with period $r_1 m(r_1)$. Note that if $|R|$ is not divisible by $|r_1 m(r_1)|$, R will end with a prefix of r_1 or $m(r_1)$. □

Lemma 17. *The following facts hold (see Figure 31):* $P[e\,..\,f] \overset{MIRROR}{\cong} P[b\,..\,c]$; $P_1[f\,..\,e] \overset{MIRROR}{\cong} P_2[c\,..\,b]$; *there exists two points* a *and* d *on* δP *such that either* $P[f\,..\,a] \overset{MIRROR}{\cong} P[c\,..\,d]$, $P[a\,..\,b] \overset{FLIP}{\cong} P[d\,..\,e]$, $\underset{P}{\angle}(2\pi - \underset{P}{\angle} d) + \underset{P}{\angle} f = \underset{P_1}{\angle} b + \underset{P_2}{\angle} b$ *and* $\underset{P}{\angle} c + \underset{P}{\angle}(2\pi - \underset{P}{\angle} a) = \underset{P_1}{\angle} e + \underset{P_2}{\angle} e$ *(this is case* 1, *see the left polygon in Figure 31), or* $P[f\,..\,a] \overset{FLIP}{\cong} P[c\,..\,d]$, $P[a\,..\,b] \overset{MIRROR}{\cong} P[d\,..\,e]$, $\underset{P}{\angle} d + \underset{P}{\angle} f = \underset{P_1}{\angle} b + \underset{P_2}{\angle} b$ *and* $\underset{P}{\angle} c + \underset{P}{\angle} a = \underset{P_1}{\angle} e + \underset{P_2}{\angle} e$ *(this is case* 2, *see the right polygon in Figure 31); if* $q = vd(b,e)/vd(f,b)$ *then for case* 1, q *is an odd integer and for case* 2, q *is even; at least two of the points in* $\{a,c,e\}$ *and two of the points in* $\{b,d,f\}$ *are vertices of* P; (a,b,c,d,e,f) *appear in clockwise order on* δP; $d_P^{-1}(a,0.5) = d$; $d_P^{-1}(b,0.5) = e$ *and* $d_P^{-1}(c,0.5) = f$.

Proof. Given that $c = T_S(e)$ and $b = T_S(f)$, we conclude that $P_1[f\,..\,e] \overset{\text{MIRROR}}{\cong} P_2[c\,..\,b]$, $P[e\,..\,f] \overset{\text{MIRROR}}{\cong} P[b\,..\,c]$, $\underset{P}{\angle} f = \underset{P_2}{\angle} b$ and $\underset{P}{\angle} c = \underset{P_1}{\angle} e$.

Let v denote some angle and let $x = \underset{P}{\angle} f + \underset{P}{\angle}(2\pi - \underset{P}{\angle} b)$, $y = \underset{P_1}{\angle} b$ and $z = \underset{P_2}{\angle} e$. Since $P_1[f\,..\,e] \overset{\text{MIRROR}}{\cong} P_2[c\,..\,b]$ and $P[f\,..\,b]$ is a prefix of $P_1[f\,..\,e]$, then $T_S(P[f\,..\,b])$ is a suffix of $P_2[c\,..\,b]$. The polyline $P_1[f\,..\,e]$ starts with $P[f\,..\,b]$, angle y and $T_S(P[f\,..\,b])$. Let us denote the remaining suffix of $P_1[f\,..\,e]$ by W. The reverse of polyline $P[c\,..\,b]$ is then formed of the concatenation of $T_S(P[f\,..\,b])$, angle x and $P[f\,..\,b]$, angle v and $T_S(W)$. Let $T_S^{-1}(e)$ be denoted by e'. Let us split the subpolyline represented by W into w_1 and w_2 around e'. $P[f\,..\,e]$ is formed of the concatenation of $P[f\,..\,b]$, angle y, $T_S(P[f\,..\,b])$, angle v, w_1, angle $\underset{P_2}{\angle} e$ and w_2 and the reverse of $P[c\,..\,b]$ is formed of the concatenation of $T_S(P[f\,..\,b])$, angle x, $P[f\,..\,b]$, angle v, $T_S(w_1)$, angle z and $T_S(w_2)$. We also know that $P_2[e\,..\,b]$ is the mirror image of $P_1[f\,..\,e']$. Now, consider the string R from Lemma 16. $P[f\,..\,b]$ has the same properties as R, hence, $P_1[f\,..\,e]$ is a periodic polyline and is of the form $(P[f\,..\,b] + y + T_S(P[f\,..\,b]) + x)^k + j(P[f\,..\,b]) + r$. Similarly, the reversed string that represents $P_2[c\,..\,b]$ will be of the form $(T_S(P[f\,..\,b]) + x + P[f\,..\,b] + y)^k + jT_S(P[f\,..\,b]) + r$ where $j = 0, 1$, $k \geq 1$, $+$ is the concatenation operator for polylines and the arithmetic operator for angle x and y. Note that r is a string representation of a polyline that by Lemma 16 can be any prefix of $P[f\,..\,b]$ or $T_S(P[f\,..\,b])$. If $j = 0$ then r is a prefix of $P[f\,..\,b]$ (see Figure 31 (left)), else if $j = 1$ then r is a prefix of $T_S(P[f\,..\,b])$ (see Figure 31

(right)). Due to the periodicity of $P[b\,..\,e]$, we conclude that q is an odd number in case 1 and q is an even number in case 2. The order of $\{a, b, c, d, e, f\}$ around δP is implied.

Let d' be the start point of r on $P_1[f\,..\,e]$ and let a be the endpoint of the copy of r on $P[f\,..\,b]$ then there exists a point d on $P_2[c\,..\,b]$ such that $d = T_S(d')$. If $j = 0$ then $P[c\,..\,d] \overset{\text{MIRROR}}{\cong} P_1[d'\,..\,e]$ which implies that $P[c\,..\,d] \overset{\text{MIRROR}}{\cong} P[f\,..\,a]$, $P[d\,..\,e] \overset{\text{FLIP}}{\cong} P[a\,..\,b]$, $\underset{P_1}{\angle} b = \underset{P}{\angle}(2\pi - \underset{P}{\angle} d)$ and $\underset{P_2}{\angle} e = \underset{P}{\angle}(2\pi - \underset{P}{\angle} a)$. If $j = 1$ then $P[c\,..\,d] \overset{\text{FLIP}}{\cong} P_1[d'\,..\,e]$ which implies that $P[c\,..\,d] \overset{\text{FLIP}}{\cong} P[f\,..\,a]$, $P[d\,..\,e] \overset{\text{FLIP}}{\cong} P[a\,..\,b]$, $\underset{P_1}{\angle} b = \underset{P}{\angle}(2\pi - \underset{P}{\angle} d)$ and $\underset{P_2}{\angle} e = \underset{P}{\angle} a$. It follows that at least two of the points in $\{a, c, e\}$ and two of the points $\{b, d, f\}$ are vertices of P.

Since the sequence of lengths in polyline $P[a\,..\,d]$ is composed of the sequence of lengths in polylines $P[a\,..\,b]$, $P[b\,..\,c]$ and $P[c\,..\,d]$, and also the sequence of lengths in polyline $P[d\,..\,a]$ is composed of the sequence of lengths in polylines $P[d\,..\,e]$, $P[e\,..\,f]$ and $P[f\,..\,a]$, by the respective congruence of these polylines (in both cases 1 and 2), given the position of a, d can be found in $O(\log n)$ time by $d_P^{-1}(a, 0.5) = d$. Also, since the sequence of lengths in polyline $P[b\,..\,e]$ is composed of the sequence of lengths in polylines $P[b\,..\,c]$, $P[c\,..\,d]$ and $P[d\,..\,e]$, and also the sequence of lengths in polyline $P[e\,..\,b]$ is composed of the sequence of lengths in polylines $P[e\,..\,f]$, $P[f\,..\,a]$ and $P[a\,..\,b]$, by the respective congruence of these polylines (in both cases 1 and 2), given the position of b, e can be found in $O(\log n)$ time by $d_P^{-1}(b, 0.5) = e$. Finally, since the sequence of lengths in polyline $P[c\,..\,f]$ is composed of the sequence of lengths in polylines $P[c\,..\,d]$, $P[d\,..\,e]$ and $P[e\,..\,f]$, and also the sequence of lengths in polyline $P[f\,..\,c]$ is composed of the sequence of lengths in polylines $P[f\,..\,a]$, $P[a\,..\,b]$ and $P[b\,..\,c]$, by the respective congruence of these polylines (in both cases 1 and 2), given the position of c, f can be found in $O(\log n)$ time by $d_P^{-1}(c, 0.5) = f$. □

Lemma 18. *Given the preprocessing in Corollary 3 and the positions of six points (a, b, c, d, e, f) on δP, it can be checked that the points specify a solution $S = (P_1, P_2)$ for the partially overlapping split polyline case of Problem 1 in constant time.*

Proof. P_1 and P_2 are mirror congruent if their respective boundaries are mirror congruent. ComponentP_1 is composed (in order) of the following alternation of polylines and angles:

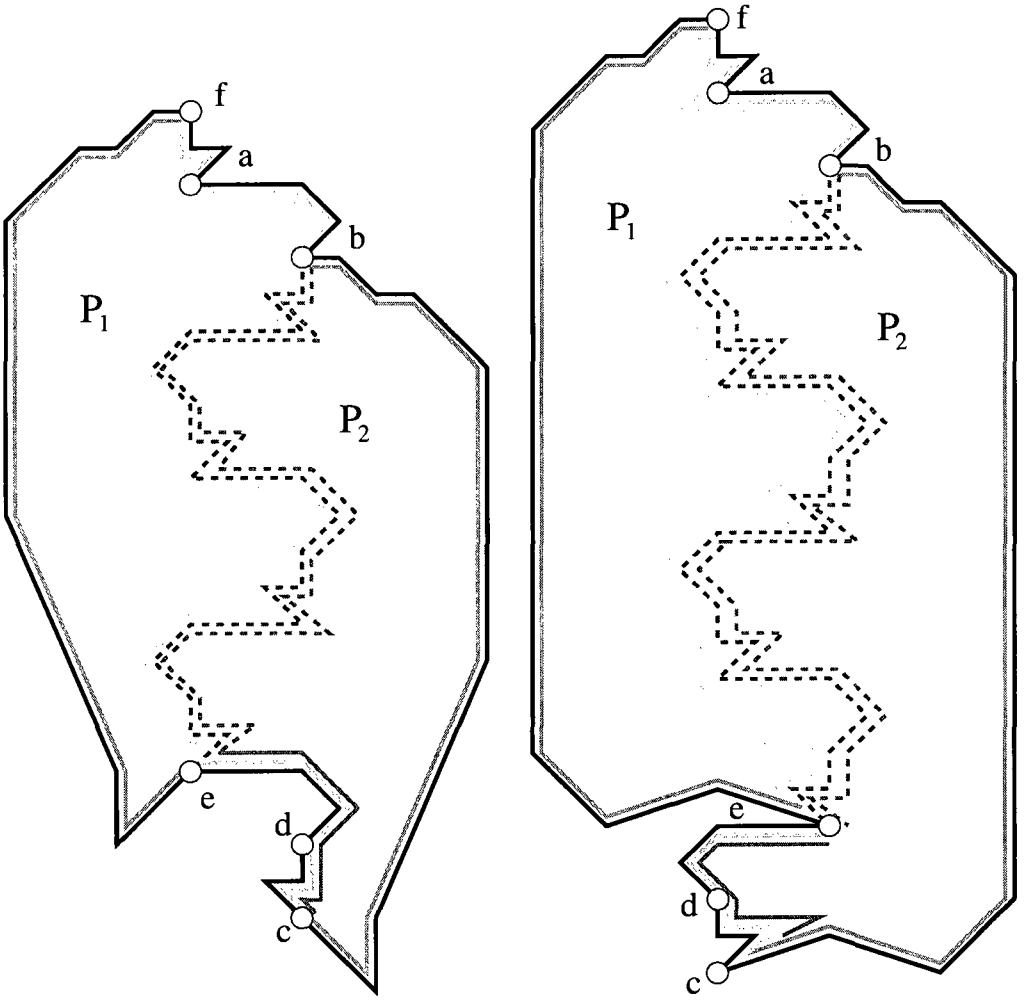

Figure 31: Polygons partitioned into two simple mirror-congruent components with an overlapping split-polyline.

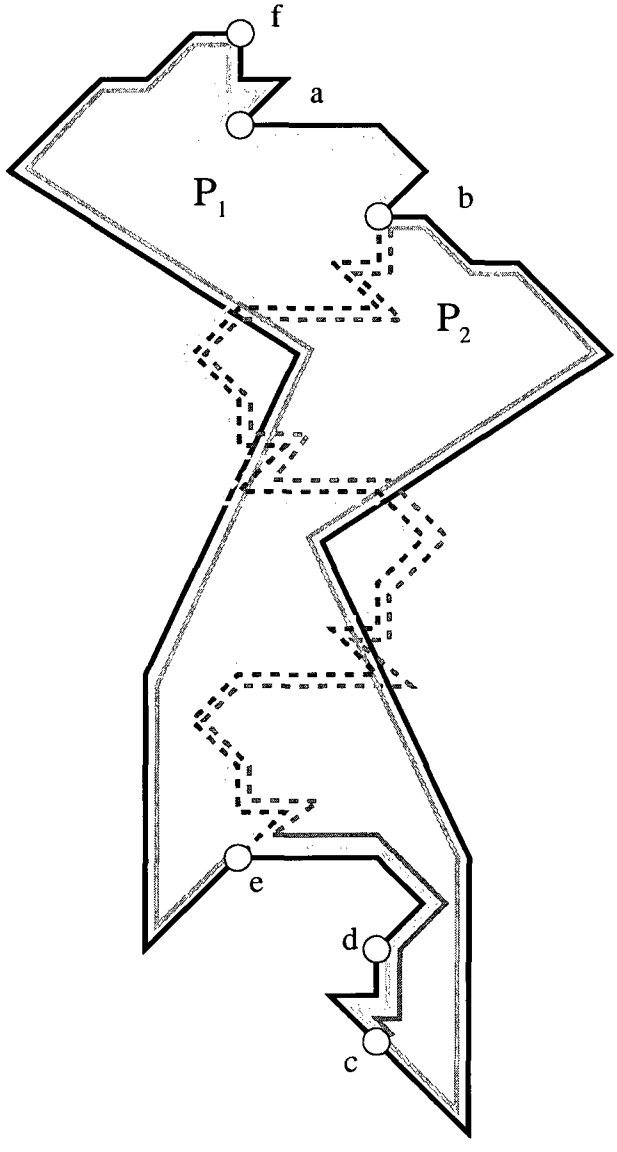

Figure 32: A simple polygon partitioned into nonsimple mirror congruent components.

$P[e\,..\,f]$, $\underset{P}{\angle} f$, $P[f\,..\,a]$, $\underset{P}{\angle} a$, $P[a\,..\,b]$, $\underset{P_1}{\angle} b$, $P_1[b\,..\,e]$ and $\underset{P_2}{\angle} e$. Component P_2 is composed (in order) of the following alternation of polylines and angles: $P[b\,..\,c]$, $\underset{P}{\angle} c$, $P[c\,..\,d]$, $\underset{P}{\angle} d$, $P[d\,..\,e]$, $\underset{P_2}{\angle} e$, $P_2[e\,..\,b]$ and $\underset{P_2}{\angle} b$. Let $m = vd(b,e)/vd(f,b)$. We need to consider two cases. First, if $P[a\,..\,b] \overset{\text{FLIP}}{\cong} P[d\,..\,e]$, $P[b\,..\,c] \overset{\text{MIRROR}}{\cong} P[e\,..\,f]$, $P[f\,..\,a] \overset{\text{MIRROR}}{\cong} P[c\,..\,d]$, $vd(f,b)|vd(b,e) \bmod vd(s)$ where $s \overset{\text{MIRROR}}{\cong} P[c\,..\,d]$, $\underset{P}{\angle}(2\pi - \underset{P}{\angle} d) + \underset{P}{\angle} f = \underset{P}{\angle} b$ and $\underset{P}{\angle} c + \underset{P}{\angle}(2\pi - \underset{P}{\angle} a) = \underset{P}{\angle} e$, we are in case 1. Else, if $P[f\,..\,a] \overset{\text{FLIP}}{\cong} P[c\,..\,d]$ and $P[a\,..\,b] \overset{\text{MIRROR}}{\cong} P[d\,..\,e]$, $P[b\,..\,c] \overset{\text{MIRROR}}{\cong} P[e\,..\,f]$, $vd(f,b)|vd(b,e) \bmod vd(s)$ where $s \overset{\text{FLIP}}{\cong} P[c\,..\,d]$, $\underset{P}{\angle} d + \underset{P}{\angle} f = \underset{P}{\angle} b$ and $\underset{P}{\angle} c + \underset{P}{\angle} a = \underset{P}{\angle} e$, we are in case 2. In both cases, the given sextuple specifies a solution of the problem since copying an alternation of $T_S(P[f\,..\,b])$ and $P[f\,..\,b]$, m times followed by a copy of $P[f\,..\,a]$ (case 1) or $P[c\,..\,d]$ (case 2) onto $\overline{be}$ will imply that $P_1[f\,..\,e] \overset{\text{MIRROR}}{\cong} P_2[c\,..\,b]$ and hence that the boundaries of P_1 and P_2 consist of respectively congruent polylines. Otherwise, the given sextuple does not represent a solution to Problem 1. This verification can be done in constant time by Corollary 3. □

Lemma 19. *The points (a,b,c,d,e,f) are as defined in Lemma 17. Given the position of any two of $\{a,c,e\}$ or $\{b,d,f\}$ and the preprocessing in Theorem 4, the positions of all six points (a,b,c,d,e,f) can be computed in $O(\log n)$ time except in the cases where either both b and e or both c and f are not vertices (Figures 33 and 34).*

Proof. By Lemma 17, at least two of $\{a,c,e\}$ and at least two of $\{b,d,f\}$ are vertices of P. If (c,e) are vertices of P, then the position of the four remaining points can be found in the following way. f and b are given in $O(\log n)$ time by Lemma 17. The pairs (b,f) and (c,e) form two pairs of points and their respective mirror images by T_S and hence, they are sufficient to compute the glide reflection. It remains to compute the positions of a and d. Let $d' = T_S^{-1}(d)$. By definition, d' is on the polyline $P_1[b\,..\,e]$. In case 1, see left of Figure 31, d can be directly computed by translating b in the direction of the glide and the norm of the translation vector is given by $(\lfloor \frac{vd(b,e)}{vd(f,b)} \rfloor + 1)vd(f,b)$. In case 2, see right of Figure 31, d' is the translate of b in the direction of the glide and the norm of translation vector is given by: $\lfloor \frac{vd(b,e)}{vd(f,b)} \rfloor vd(f,b)$. We can then compute d since its the image of d' by the glide reflection. In both cases, we can find a by $d_P^{-1}(f, d(c,d)) = a$. If (b,f) are vertices

	a	b	c	d	e	f
i	E	E	V	E	V	E
ii	E	V	E	E	E	V
iii	E	Not V	Not V	E	V	V
iv	E	V	V	E	Not V	Not V
v	V	V	Not V	V	V	Not V
vi	V	Not V	V	V	Not V	V

Table 1: Sub-Cases for the sextuple: V stands for "is a vertex", Not V for "is not a vertex" and E for "either".

of P, finding the position of the four remaining points is similar. However, if neither the pair (c, e) nor the pair (b, f) specifies a solution, then it is easy to see by Lemma 17 and a combinatorial counting that four sub-cases remains to be considered, see table 1. Sub-cases *iii* and *iv* are similar to the sub-cases above (since if (e, f) (or (b, c)) are vertices, we can compute b and c in $O(\log n)$ time by Lemma 17 (e and f). Sub-cases *v* (where both c and f are not vertices of P) and *vi* (where both b and e are not vertices of P) are the remaining ones (see Lemma 20).

□

An $O(n^3)$ algorithm

Let $1a$ and $1b$ denote two subcases of case 1. Subcase $1a$ is one where $\{a, c, d, f\}$ are vertices and $\{b, e\}$ are not and subcase $1b$ is one where $\{a, b, e, d\}$ are vertices and $\{c, f\}$ are not. Similarly $2a$ and $2b$ denote two subcases of case 2. Subcase $2a$ is one where $\{a, b, e, d\}$ are vertices and $\{c, f\}$ are not and subcase $2b$ is one where $\{a, c, d, f\}$ are vertices and $\{b, e\}$ are not. Figure 33 and Figure 34 show the four subcases.

Lemma 20. *Given the positions of $\{a, c, d, f\}$ and the preprocessing in Corollary 3, the positions of b and e can be computed in $O(n)$ time for subcases $1a$ and $1b$. Similarly, given the positions of $\{a, b, d, e\}$ and the preprocessing in Corollary 3, the positions of c and f can be computed in $O(n)$ time in subcases $2a$ and $2b$.*

Proof. In subcases $1a$ and $1b$, (see Figure 33), given the positions of $\{a, c, d, f\}$ and the fact

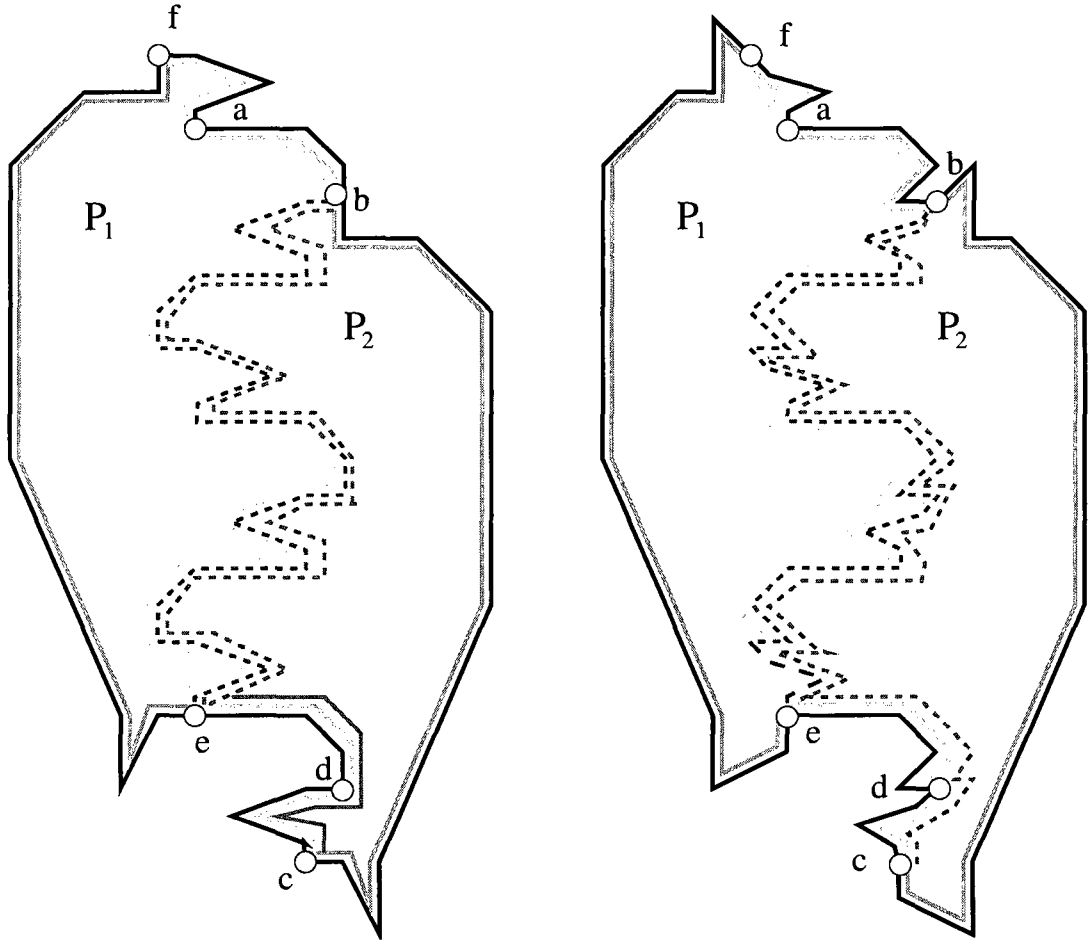

Figure 33: Subcase 1a (left) where $\{a, c, d, f\}$ are vertices and $\{b, e\}$ are not. Subcase 1b (right) where $\{a, b, e, d\}$ are vertices and $\{c, f\}$ are not.

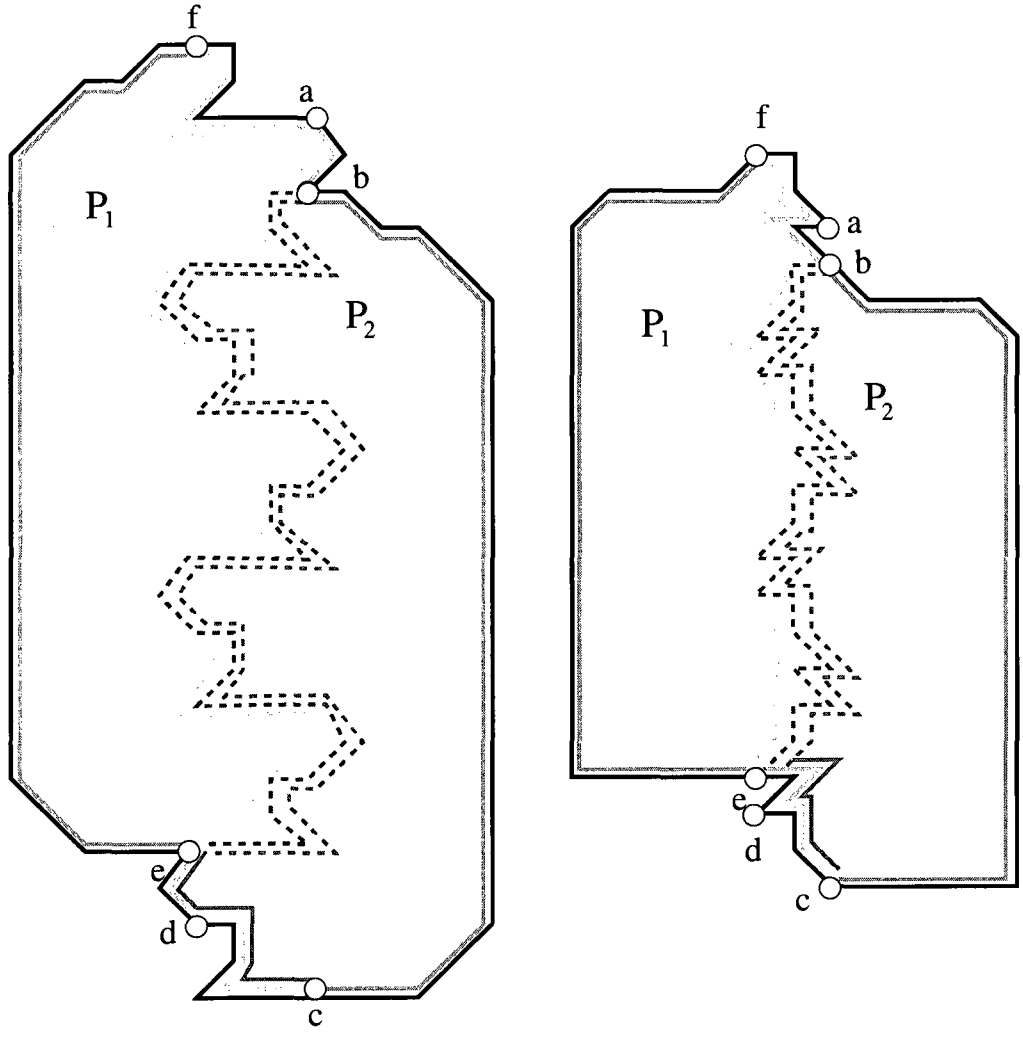

Figure 34: Subcase $2a$ (left) where $\{a, b, e, d\}$ are vertices and $\{c, f\}$ are not. Subcase $2b$ (right) where $\{a, c, d, f\}$ are vertices and $\{b, e\}$ are not.

that $P[a\,..\,b] \overset{\text{FLIP}}{\cong} P[d\,..\,e]$ by Lemma 17, we observe that $ccw_P(b)$ and $cw_P(d)$ have the same length and slope. In subcase $1a$, since b is not vertex then $cw_P(b)$ and $ccw_P(b)$ have the same slope. For every segment $s \in \delta(P)$ (clockwise from a) that has the same slope as $cw_P(d)$, compute the potential b (distance $|cw_P(d)|$ from the first encountered endpoint). The potential e can be computed in $O(\log n)$ time by Lemma 17. In subcase $1b$, b is a vertex of δP. For every vertex $p \in \delta(P)$ (clockwise from a) such that $ccw_P(p)$ has the same slope and length as $cw_P(d)$, we consider p as the potential b and we compute e in $O(\log n)$ time by Lemma 17. In subcases $2a$ and $2b$, (see Figure 34), given the positions of $\{a, b, d, e\}$ and the fact that $P[f\,..\,a] \overset{\text{FLIP}}{\cong} P[c\,..\,d]$ by Lemma 17, we observe that $ccw_P(a)$ and $cw_P(c)$ have the same length and slope. In subcase $2a$, since c is not vertex then $cw_P(c)$ and $ccw_P(c)$ have the same slope. For every segment $s \in \delta(P)$ (counterclockwise from a) that has the same slope as $ccw_P(a)$, we compute the potential c (distance $|ccw_P(a)|$ from the first encountered endpoint). The potential f can be computed in $O(\log n)$ time by Lemma 17. In subcase $2b$, c is a vertex of δP. For every vertex $p \in \delta(P)$ (counterclockwise from a) such that $cw_P(p)$ has the same slope and length as $ccw_P(b)$, we consider p as the potential c and we compute f using the function d_P. In all four cases, the validity of the solution can be checked in constant time as stated in Lemma 18. □

Theorem 21. *Given a simple polygon P and given that* Split(S)*, if it exists, is partially overlapping with T_S* (Split(S))*, a solution $S = (P_1, P_2)$ to Problem 1 can be found in $O(n^3)$ time if and only if P can be partitioned into two congruent polygons.*

Proof. For every pair of vertices of P, we verify it is (c, e) or (b, f) or (e, f) or (b, c) by computing the remaining four points as shown in Lemma 19. For every computed sextuple, we verify (in constant time as stated in Lemma 18) if they specify a solution to the problem considering both cases 1 and case 2. If none of the previous pairs form a pair of vertices, then either $\{a, c, d, f\}$ (case 1) or $\{a, b, d, e\}$ (case 2) are vertices by Lemma 19. In subcase $1a$: for every pair of vertices, assume it is (a, c), then d and f can be computed in $O(\log n)$ time. Find b and e and do the verification as discussed in Lemma 18. In subcase $1b$: for every pair (p, s) where p is a vertex of δP and s is a line segment in δP, we verify if p is a and s is the

line segment such that $c \in s$ (c is positioned $|ccw_P(a)|$ from an endpoint of s) by computing d and f in $O(\log n)$ time, by computing b and e and doing the verification as discussed in Lemma 18. In subcase 2*a*: for every pair of vertices, assume it is (a, e), then d and b can be computed in $O(\log n)$ time. Find c and f and do the verification as discussed in Lemma 18. In subcase 2*b*: for every pair (p, s) where p is a vertex of δP and s is a line segment in δP, we verify if p is a and s is the line segment such that $e \in s$ (e is positioned $|ccw_P(a)|$ from an endpoint of s) by computing b and d in $O(\log n)$ time, by computing c and f and doing the verification as discussed in Lemma 18. Let $m = vd(b, e)/vd(f, b)$ (m should be greater than one and $vd(b, e) \bmod vd(f, b) = vd(f, a)$ for a partition to exist). If the verification succeeds in any of the previous cases then either : 1) if m is odd and we are in case 1, setting $P_1[b \,..\, e]$ to $(T_S(P[f \,..\, b]) + x + P[f \,..\, b] + y)^{\frac{m}{2}} + T_S(P[f \,..\, b]) + x + P[f \,..\, a]$ yields a valid partition, 2) if m is even and we are in case 2, setting $P_1[b \,..\, e]$ to $(T_S(P[f \,..\, b]) + x + P[f \,..\, b] + y)^{\frac{m}{2}} + y + T_S(P[f \,..\, a])$ yields a valid partition, or 3) the current sextuple does not specify a congruent partition. Note that $x = \underset{P}{\angle} f + \underset{P}{\angle}(2\pi - \underset{P}{\angle} b)$ and $y = \underset{P_1}{\angle} b$ (which is equal to $\underset{P}{\angle}(2\pi - \underset{P}{\angle} d)$ in case 1 and $\underset{P}{\angle} d$ in case 2). The "if" part is trivial. The "only if" part stems from the previous Lemmas; if P is partitionable then by Lemma 17, $\{a, b, c, d, e, f\}$ exist and appear in that order on δP. The split-polyline allows by construction for $P_1[f \,..\, e]$ to be congruent to $P_2[c \,..\, b]$. The string matching checks for congruence of $P[e \,..\, f]$ and $P[b \,..\, c]$. Therefore, P_1 and P_2 having the same polylines defining them, are congruent. The split-polyline might however intersect δP. This will result in two congruent sub-polygons P_1 and P_2 that are nonsimple (see Figure 32). □

4.6.3 Combined algorithm for the two mirror congruent cases

Theorem 22. *Given a simple polygon P, we can decide if it can be partitioned into two mirror congruent polygons and find a solution $S = (P_1, P_2)$ to Problem 1, if it exists, in $O(n^3)$ time.*

Proof. By Lemma 10, the split-polyline Split(S) is either disjoint or partially overlapping with its mirror image T_S (Split(S)). Hence, given a simple polygon P, we run the algorithm for the disjoint case from Theorem 15. If it fails, we run the algorithm for the partially

overlapping case from Theorem 21. If either of the two cases succeeds report the partition else report that P is not partitionable into two mirror congruent components. □

4.7 Ideas toward a better algorithm

4.7.1 Producing simple congruent components

Conjecture 23. *Limiting the output to simple components P_1 and P_2 will still result in an $O(n^2 \log n)$ time algorithm.*

We believe that conjecture 23 is true at least in the case of proper congruence. The idea stems from our belief that if nonsymmetric polygon P is partitionable into two proper congruent components in two different ways (i.e. using two different split-polylines) then P consists of four copies of a monomorphic tile arranged around the midpoints of the two split-polylines. A monomorphic tile is a tile that can form a monohedral tiling of the plane, in other words, it tiles the plane with copies of itself [142]. By Lemma 6, we conclude that if a polygon P is partitionable into two proper congruent subpolygons in two different ways (P_1, P_2, P_1' and P_2' pairwise congruent), there exist two sextuples (a, b, c, d, e, f) and (a', b', c', d', e', f') around δP such that: $P_1[a \,..\, b] \cong P_2[b \,..\, c]$; $P_1[f \,..\, e] \cong P_2[e \,..\, d]$; $P_1[a \,..\, f] \overset{\text{FLIP}}{\cong} P_2[c \,..\, d]$; $P_1'[a' \,..\, b'] \cong P_2'[b' \,..\, c']$; $P_1'[f' \,..\, e'] \cong P_2'[e' \,..\, d']$ and $P_1'[a' \,..\, f'] \cong P_2'[c' \,..\, d']$.

Figure 36 shows details of the properties for translationally congruent subpolygons. The following properties hold in the translation congruence case: $|\overline{ab}| = |\overline{bc}|$; $|\overline{fe}| = |\overline{ed}|$; $\{a, b, c\}$ are on a line, say (ac); $\{d, e, f\}$ are on a line, say (df); $(ac) \parallel (df)$; $acdf$ is a parallelogram; $abef$ is a parallelogram; $bcde$ is a parallelogram; the respective segments forming $P_1[a \,..\, f]$ and $P_2[c \,..\, d]$ form (pairwise) a series of parallelograms; $|\overline{a'b'}| = |\overline{b'c'}|$; $|\overline{f'e'}| = |\overline{e'd'}|$; $\{a', b', c'\}$ are on a line, say $(a'c')$; $\{d', e', f'\}$ are on a line, say $(d'f')$; $(a'c') \parallel (d'f')$; $a'c'd'f'$ is a parallelogram; $a'b'e'f'$ is a parallelogram; $b'c'd'e'$ is a parallelogram and the respective segments form (pairwise) $P_1'[a' \,..\, f']$ and $P_2'[c' \,..\, d']$ form a series of parallelograms.

Since our algorithm checks $O(n^2)$ potential positions for the sextuple (a, b, c, d, e, f) and we believe that if there is more than one candidate tuple, the input polygon P has the properties presented above, we can preprocess P to check if it obeys these properties otherwise,

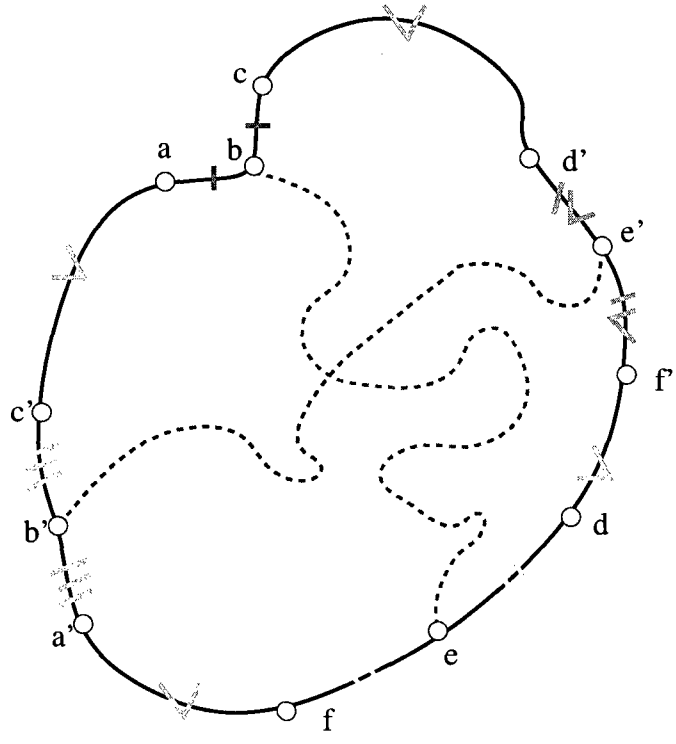

Figure 35: If a nonsymmetric polygon P is partitionable into two congruent components in two different ways then it consists of four copies of a monomorphic tile.

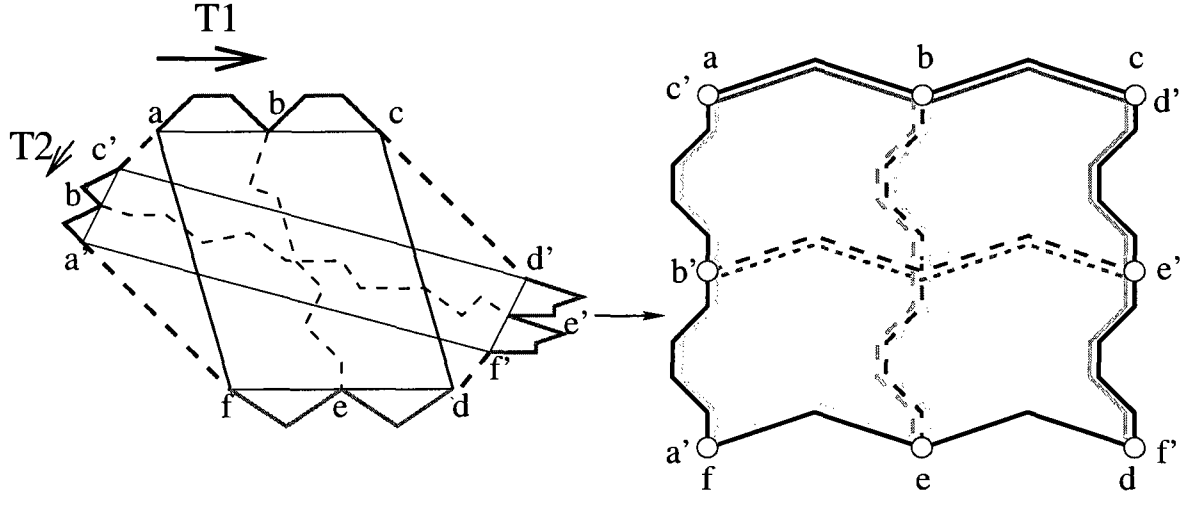

Figure 36: Properties of a simple polygon partitionable into two translationally congruent components into two different ways.

we run our algorithm on P and we test the obtained solution, if any, for nonsimplicity and output it only if it is simple. This check is linear and is done only once which does not affect the running time of our algorithm and hence, the conjecture.

4.7.2 An $O(n^2 \log n)$ algorithm for the disjoint split-polyline mirror congruence case

In his manuscript [252], Rote made the following observation about the axis of glide reflection.

Lemma 24 ([252]). *The axis of glide reflection g goes through the center of gravity g_p of P.*

This observation yields a better algorithm for the disjoint split-polyline case as shown in Lemma 25 and Theorem 26 assuming that the center of gravity of P is precomputed in linear time.

Lemma 25. *Given the positions of $\{a, c\}$, the fact that both b and e are not vertices (equivalent to $\{a, c, d, f\}$ being all vertices by Lemma 11) and the preprocessing in Theorem 4, the positions of $\{b, d, e, f\}$ can be computed in $O(\log n)$ time.*

Proof. By Theorem 4, the positions of d and f can be computed in constant time (d and f are both vertices in this case): $d_P(a, 0.5) = d$ and $d_P(c, 0.5) = f$. Observe that since $T_S(a) = e$ and $T_S(e) = c$ then the image of $\overline{ae}$ by T_S is $\overline{ec}$. Mirror congruence preserves distances, therefore $|\overline{ae}| = |\overline{ec}|$ and e belongs to the perpendicular bisector of $\overline{ac}$. Similarly, we show that b belongs to the perpendicular bisector of $\overline{df}$. The axis of glide reflection g passes through the midpoints of $\overline{ae}$, $\overline{ec}$, $\overline{bd}$ and $\overline{bf}$ and is hence parallel to $\overline{ac}$ and to $\overline{df}$. By Lemma 24, the axis of glide reflection g is determined given the positions of $\{a, c, d, f\}$. Therefore, the transformation is determined in constant time and so are the positions of points b and e. Checking if b and e belong to δP takes $O(\log n)$ time. □

Theorem 26. *Given a simple polygon P and given that* Split(S)*, if it exists, is disjoint from T_S* (Split(S))*, a solution $S = (P_1, P_2)$ to Problem 1 can be found in $O(n^2 \log n)$ time if and only if P can be partitioned into two mirror congruent polygons.*

Proof. For every pair of vertices of P, we verify if it is (a, e) or (c, e) or (b, d) or (b, f) by computing the four remaining points as shown in Lemma 13. We verify in constant time if the obtained sextuple specifies a solution of the problem as stated in Lemma 12. If none of the previous pairs form a pair of vertices then by Lemma 13, $\{a, c, d, f\}$ are vertices and both b and e are not. For every pair of pair of vertices, we consider it is (a, c) by computing $\{b, d, e, f\}$ in $O(\log n)$ time as stated in Lemma 25 and verifying if the obtained sextuple specifies a solution. If the verification succeeds in any of the cases, then copying $P[f \,..\, a]$ onto $\overline{be}$ such that $\underset{P}{\angle} a = \underset{P_2}{\angle} e$ (and $\underset{P}{\angle} c = \underset{P_1}{\angle} e$) yields a valid partition. The "if" part is trivial. The "only if" part stems from the previous Lemmas; if the polygon is partitionable then by Lemma 11, a, b, c, d, e, f exist and appear in that order on δP. The split-polyline is by construction congruent to $P[f \,..\, a]$ and $P[c \,..\, d]$. The string matching checks for congruence of $P[a \,..\, b]$ and $P[d \,..\, e]$ and for congruence of $P[b \,..\, c]$ and $P[e \,..\, f]$. Therefore, P_1 and P_2, having the same polylines defining them, are congruent. The split-polyline might however intersect δP. This will result in two congruent sub-polygons P_1 and P_2 that are nonsimple. Since we check every pair of vertices in P and we locate the four remaining points in $O(\log n)$ time for each pair, the algorithm runs in $O(n^2 \log n)$ time. □

In Lemma 20 we show how given the positions of any two of $\{a, c, e\}$ or $\{b, d, f\}$, we can find the positions of the remaining four points in logarithmic time except in subcases v and vi of Table 1. In subcase v, given any two points of $\{a, b, d, e\}$ the remaining two can be computed in logarithmic time. Similarly, in subcase vi, given any two points of $\{a, c, d, f\}$ the remaining four can computed in logarithmic time. In the former case, it remains to compute c and f while in the latter it remains to compute b and e. For subcase $1a$ (subcase $2a$), given $\{a, c, d, f\}$ ($\{a, b, d, e\}$) and given that $P[d \,..\, c]$ ($P[a \,..\, b]$) is a translated reflection of $P[f \,..\, a]$ ($P[d \,..\, e]$), the axis g of glide reflection T_S can be computed. For subcase $1b$ (subcase $2b$), given $\{a, b, d, e\}$ ($\{a, c, d, f\}$) and given that $P[d \,..\, e]$ ($P[c \,..\, d]$) is a translated copy of $P[a \,..\, b]$ along the axis g ($P[f \,..\, a]$), the axis g of glide reflection T_S can be computed as follows: g is parallel to (ae) and (bd) in subcase $1b$ (parallel to (fe) and (bc) in subcase $2b$) and by Lemma 24 goes through the center of gravity of P. However, in all previous cases,

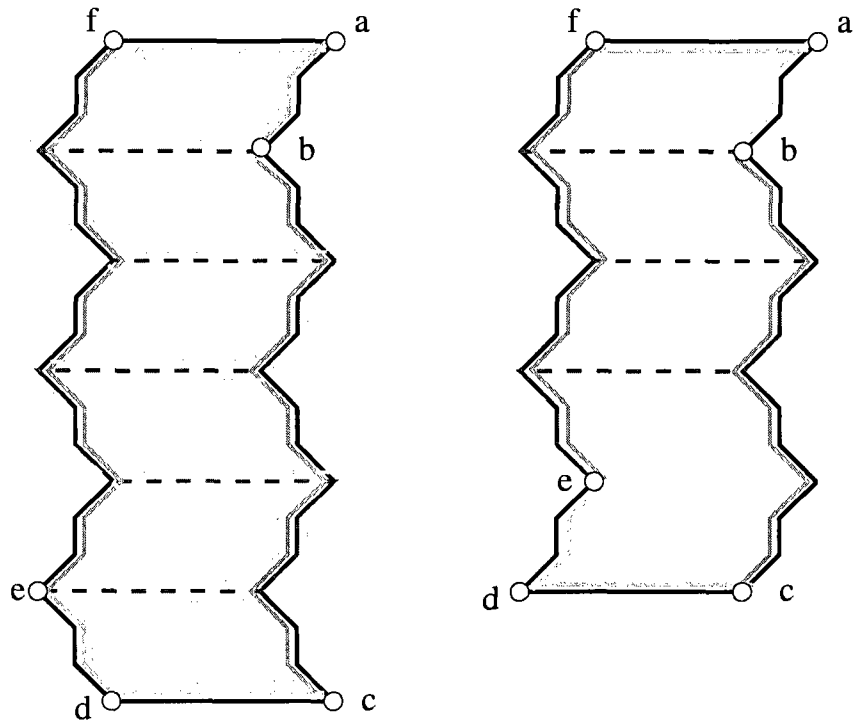

Figure 37: (a) A polygon P is partitionable into two congruent components in several different ways, (b) a polygon that is not partitionable into two congruent components.

computing g is not enough to determine the glide reflection: the translation vector remains unknown and the four computed points in each cases do not contain a pair of a point and its glide reflection. We think that it is possible to compute the remaining two points in each case without an additional linear factor in the running time. The idea stems from our belief that if there are multiple b's and e's or multiple c's and f's then the input polygon P consists of a polyline and a flip-congruent copy of it separated by two line segments and this polygon has the properties of case 1 (see Figure 37 (a)). We also conjecture that if the input polygon P has the properties of case 2 it is not partitionable (see Figure 37 (b)).

Conjecture 27. *An $O(n^2 \log n)$ time algorithm is possible for the mirror congruence case.*

4.8 Conclusion

In this chapter, we presented an $O(n^3)$ time algorithm that partitions a simple polygon P into two congruent subpolygons P_1 and P_2 if possible or reports that such a partition does not exist. P_1 and P_2 can possibly be nonsimple, i.e. self-intersecting subpolygons. We presented several conjectures that, if true, improve the running time of the algorithm. It would be interesting in the future to study a more general version of Problem 1. We conjecture Problem 28 to be **NP**-complete for polygons with holes.

Problem 28. *Given a polygon P with n vertices, compute a partition of P into k where $k \geq 3$ (properly or mirror) congruent polygons $\{P_1, P_2, \ldots, P_k\}$, or indicate such a partition does not exist.*

Another interesting problem to study would be covering a polygon with two congruent components.

Problem 29. *Given a polygon P with n vertices, compute a covering of P by two (properly or mirror) congruent polygons P_1 and P_2 or indicate such a covering does not exist.*

A polygon that can be covered with two congruent components seems to have a structure similar to a polygon that is partitionable into two congruent components. Let the *covering polylines* be the polylines that separate the polygon P into two components that cover it.

Figures 38 and 39 show four polygons covered by two translationally congruent components. If a polygon is coverable by two translationally congruent components, there seems to exist a 12-tuple of points $\{a, b, c, d, e, f, g, h, i, j, k, l\}$ on δP such that: $P[a \,..\, b] \cong P[b \,..\, c]$; $P[a \,..\, e] \cong P[b \,..\, f]$; $P[d \,..\, e] \cong P[e \,..\, f]$; $P[g \,..\, h] \cong P[h \,..\, i]$; $P[g \,..\, k] \cong P[h \,..\, l]$ and $P[i \,..\, k] \cong P[k \,..\, l]$. The covering polylines $P[k \,..\, b]$ and $P[e \,..\, h]$ are respectively congruent to $P[l \,..\, a]$ and $P[e \,..\, h]$.

Figure 40 shows three polygons covered by two rotationally congruent components. If a polygon is coverable by two rotationally congruent components, there seems to exist an octuple of points $\{a, b, c, d, e, f, g, h\}$ on δP such that: $P[a \,..\, b] \cong P[c \,..\, d]$; $P[d \,..\, e] \cong P[f \,..\, g]$. The covering polylines $P[c \,..\, d]$, $P[d \,..\, e]$, $P[e \,..\, h]$, $P[h \,..\, c]$ are such that: $P[e \,..\, a] \cong P[g \,..\, c]$ and $P[b \,..\, d] \cong P[d \,..\, f]$.

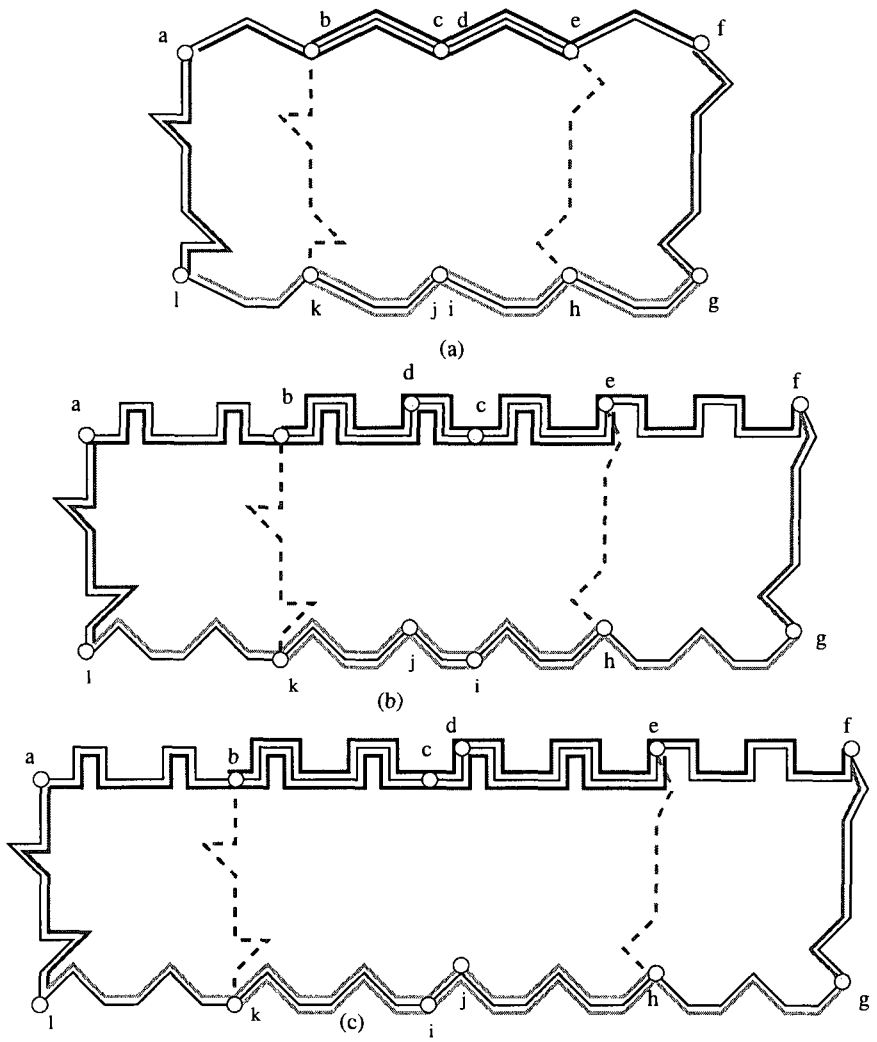

Figure 38: Three polygons (a), (b) and (c) covered each by two translationally congruent components.

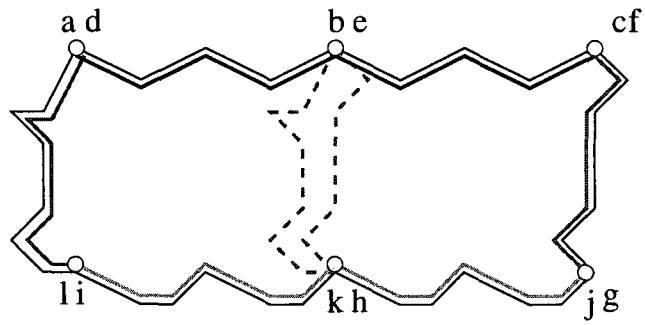

Figure 39: A polygon covered by two translationally congruent components where the endpoints of the covering polylines are colocated pairwise.

Figures 42 and 41 illustrate four polygons each covered by two mirror congruent components. If a polygon is coverable by two mirror congruent components, there seems to be three cases:

- either there exists a 12-tuple of points $\{a, b, c, d, e, f, g, h, i, j, k, l\}$ on δP such that $P[a \,..\, b] \overset{\text{MIRROR}}{\cong} P[k \,..\, j]$; $P[b \,..\, c] \overset{\text{MIRROR}}{\cong} P[j \,..\, i]$; $P[a \,..\, e] \overset{\text{MIRROR}}{\cong} P[g \,..\, k]$; $P[b \,..\, f] \overset{\text{MIRROR}}{\cong} P[h \,..\, l]$; $P[g \,..\, h] \overset{\text{MIRROR}}{\cong} P[e \,..\, d]$; $P[h \,..\, i] \overset{\text{MIRROR}}{\cong} P[c \,..\, d]$ and the covering polylines $P[k \,..\, b]$ and $P[e \,..\, h]$ are respectively mirror congruent to $P[l \,..\, a]$ and $P[f \,..\, g]$,

- there exists a sextuple of points $\{a, b, c, d, e, f\}$ on δP such that $P[a \,..\, b] \overset{\text{MIRROR}}{\cong} P[d \,..\, e]$; $P[b \,..\, c] \overset{\text{MIRROR}}{\cong} P[e \,..\, f]$ and the covering polylines $P[e \,..\, b]$ and $P[b \,..\, e]$ are respectively mirror congruent to $P[f \,..\, a]$ and $P[c \,..\, d]$,

- or there exists a quadruple of points $\{a, b, c, d\}$ on δP such that $P[d \,..\, a] \overset{\text{MIRROR}}{\cong} P[b \,..\, c]$ and the covering polylines $P[d \,..\, b]$ and $P[b \,..\, d]$ are such that $P[a \,..\, d] \overset{\text{MIRROR}}{\cong} P[c \,..\, b]$.

The structure of the covering seems to indicate that a polynomial time algorithm—similar to the partition algorithm—is possible for covering a polygon with two congruent polygons. Therefore, we conjecture the following:

Conjecture 30. *Covering a polygon with two congruent components is polynomial.*

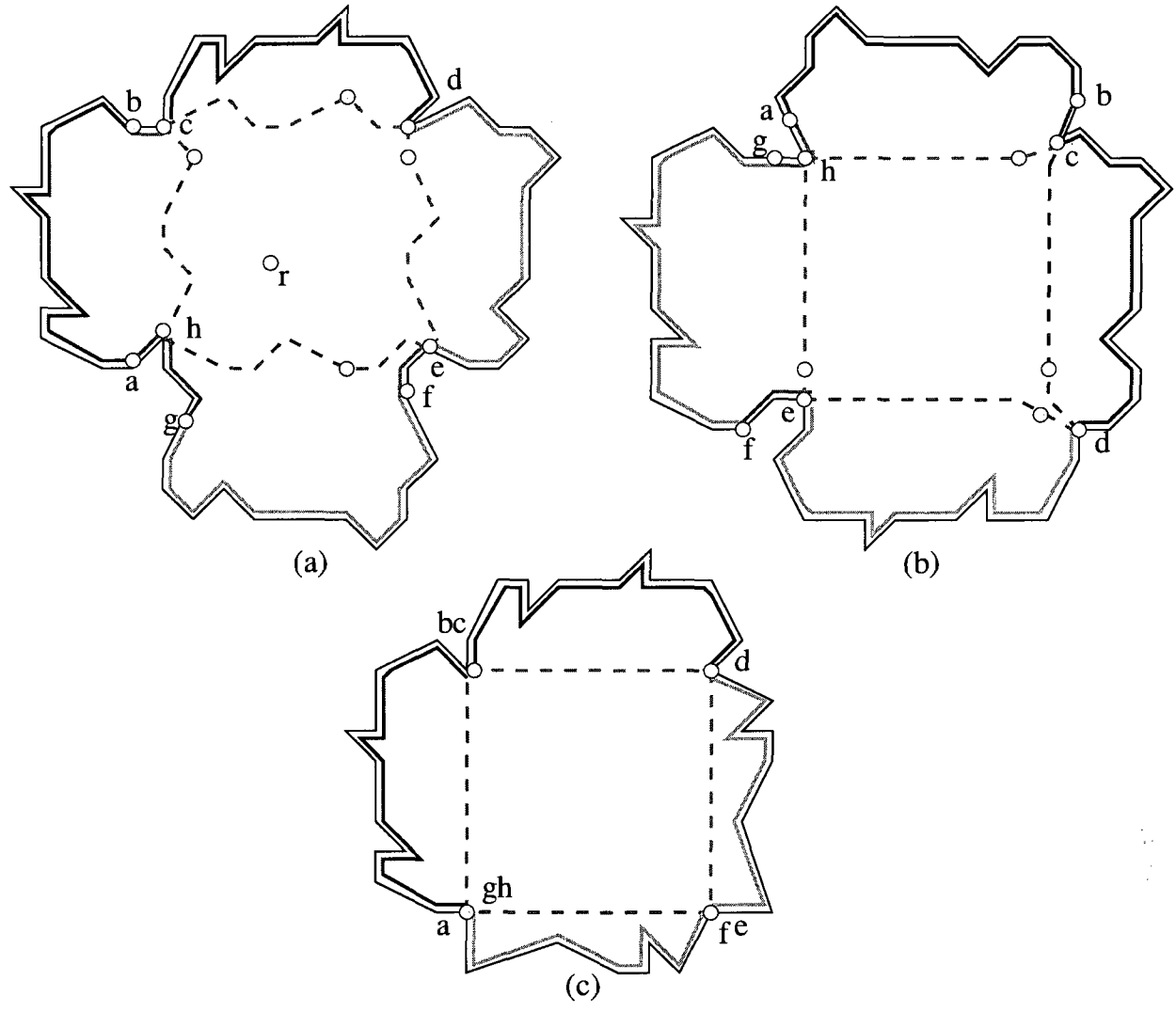

Figure 40: Three polygons (a), (b) and (c) each covered by two rotationally congruent components.

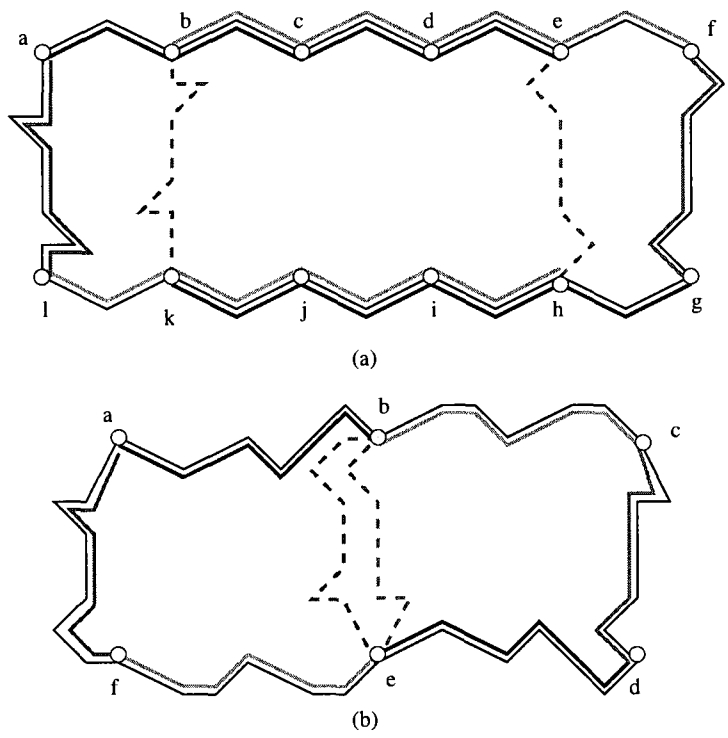

Figure 41: Two polygons each covered by two mirror congruent components.

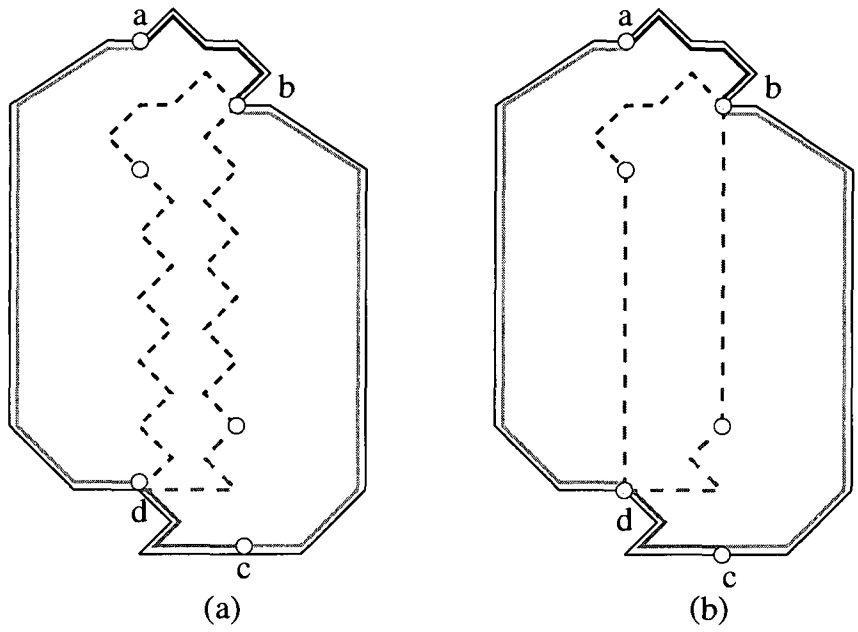

Figure 42: Two polygons each covered by two mirror congruent components where the endpoints of the covering polylines are colocated pairwise.

Chapter 5

Partitioning squares into equal area components with minimum ink

5.1 Introduction

In this chapter, we explore experimentally the partition of squares (and partially rectangles) into k equal area components while minimizing the perimeter of the cuts that is, in other words, an equal area minimum ink partition for some constant $k \geq 1$. As we saw in Chapter 3, several variants of the problems have been discussed in the literature. We summarize the previous results in two tables: Table 2 for general and convex polygon partition and Table 3 for rectangle and square partition. In Table 3, we follow Nagamochi and Abe [240] and Beaumont et al.'s [33] definitions for the different rectangle and square partition problems: *PERI-SUM* denotes the total perimeter minimization problem where the sum of the perimeters of the resulting rectangles or squares in the partition is minimized; *CUT* denotes the minimization of the perimeter of the cut lines and is equivalent to PERI-SUM; *PERI-MAX* denotes the maximum perimeter minimization problem where the maximum perimeter of the resulting rectangles or squares is minimized; and *ASPECT-RATIO* denotes

the maximum aspect ratio minimization problem where the maximum aspect ratio of the resulting rectangles or squares is minimized.

Most of the previous results in Tables 2 and 3 solve problems that are quiet different than the one we explore in this chapter. Hert and Lumelsky partition arbitrary simple and nonsimple polygons and do not consider equal areas (except for the **NP**-hardness proof) [30]. Guardia and Hurtado consider Steiner free partition of convex bodies [143]. Note that Steiner free partition for rectangles and squares becomes quickly impossible (for values of $k \geq 3$). Akiyama et al. consider equal area and equal perimeter partition of convex polygons [9,10]. None of these works consider optimizing the perimeter. Kong et al. consider the PERI-MAX but not CUT and PERI-SUM that are of interest to us [179, 180]. Although the more recent works of Nagamochi and Abe [240] and Beaumont et al. [33] focus on rectangle and square partition and although they minimize the perimeter (PERI-SUM), they consider areas that are not necessarily equal. Bose et al. partition rectangles and squares into equal area components while minimizing the perimeter of the cuts (the CUT problem) but they mainly consider straight line orthogonal cuts [51, 54]. The authors restrict their attention to orthogonal cuts. They prove that if k is a perfect square then orthogonal cuts are optimal. A recursive algorithm that is exponential in the number of cuts but guaranteed to cut a rectangle optimally into k equal areas is given. Two approximation algorithms are described that output a near optimal partitioning of rectangles and prisms into equal area components. A third approximation algorithm that runs in $O(1)$ time and that finds near optimal cuts of a unit square is also presented. Their optimal partitioning of the square in $2, 3, 4, 5$ and 6 components using straight lines cuts is shown in Figure 43. The closest problems to the one we consider are posed by Koutsoupias et al. [181] and Bose et al. [55]. Allowing straight line and curved cuts, Koutsoupias et al. present a PTAS for partitioning a simple polygon into equal area components while minimizing the perimeter of the cuts [181]. They also give an $O(n \log n)$ time algorithm for convex polygons. However, the number of areas is restricted to two and the areas are allowed to be disconnected. Bose et al. quarter the square optimally (with some assumptions about the optimal solution), i.e. partition it into $k = 4$ equal area components with minimum perimeter cuts [55]. Assuming symmetry, the optimal 4-partition

Polygon	Description	Minimum Ink	Algorithm	Reference
Arbitrary	Not necessarily Equal Area	Yes	**NP**-hard	[30]
Arbitrary	Not necessarily Equal Area	No	$O(pn)$	[30]
Convex	No Steiner points	No	No	[143]
Convex	Equal Area, Equal Perimeter	No	No	[9, 10]
Convex	Two, not necessarily connected	Yes	$O(n \log n)$	[181]
Simple	Two, not necessarily connected	Yes	PTAS	[181]

Table 2: Area partitioning results.

of the square using straight line cuts and sections of circular arcs are given. The conjectured to be optimal 3-partition of the square using straight line (not necessarily orthogonal) cuts and sections of circular arcs is presented [52]. Our work is an extension of this latter work. Our goal is to investigate a characterization of the optimal solution for partitioning the unit square into k equal area components while minimizing the total perimeter of the cuts, where $3 \leq k \leq 10$. We use only straight line cuts (which may be orthogonal) or only circular cuts (which may be straight lines as well). The solution for partitioning the (1×2) rectangle into k equal area components is investigated as well, where $3 \leq k \leq 6$. We start by describing the set of experiments we have accomplished in Section 5.2. In Section 5.3, we list the results we obtained and finally in Section 5.4, we strengthen the experimental result for the 5-partition of the square by obtaining it analytically.

5.2 Experiments description

The characterizations of the determined best solutions for partitioning the unit square and the (1×2) rectangle into equal area components were obtained experimentally using Surface Evolver. Surface Evolver is an interactive program for the study of surfaces shaped by surface tension and subject to various constraints [57]. The Surface Evolver's role in this context is to evolve the surface toward minimal energy: minimal perimeter of the internal cuts in our case. There are two main commands in Surface Evolver useful for our purposes: the iteration command, g, which evolves the surface toward its minimal energy and the vertex popping command, o, which creates new vertices, Steiner points, in the interior of

Polygon	k Components	Equal Area	Obj. Func.	Algorithm	Reference
Square	Squares	Yes	P-M	$O(k^6)$	[179]
Rectangle	Rectangles	Yes	P-M	$O(k^6)$	[180]
Rectangle	Rectangles	Yes	CUT	$O(4^k)$	[54]
Square	Rectangles	Yes	CUT	$1+\frac{1}{2(\sqrt{k}-1)}$-approx	[54]
Rectangle	Rectangles	Yes	CUT	$1+\frac{1}{2(\sqrt{k}-1)}$-approx	[51]
Square	Rectangles	No	P-S	**NP**-complete, $\frac{7}{4}$-approx	[33]
Square	Rectangles	No	P-M	**NP**-complete, $\frac{2}{\sqrt{3}}$-approx	[33]
Square	$k=4$, arbitrary*	Yes	P-S	No	[55]
Rectangle	Rectangles	No	P-M	1.25-approx	[240]
Rectangle	Rectangles	No	P-S	$\frac{2}{\sqrt{3}}$-approx	[240]
Rectangle	Rectangles	No	A-R	bounded aspect ratio	[240]

Table 3: Area partitioning results where "P-M", "P-S", "A-R" abbreviate PERI-MAX, PERI-SUM and ASPECT-RATIO respectively and * indicates a partition with straight line or circular cuts.

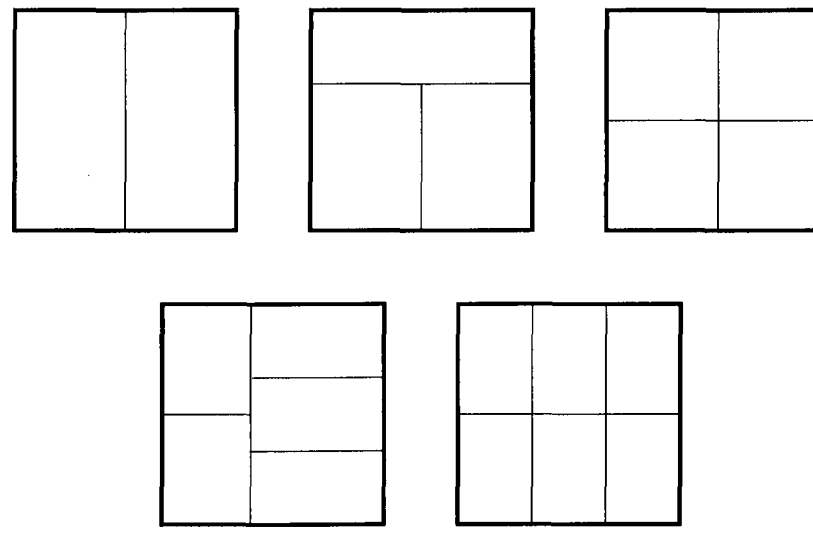

Figure 43: Optimal orthogonal straight line cut partitions of the square into $2, 3, 4, 5$ and 6 components.

square or the rectangle if their creation would allow a more minimal solution.

The experiments necessitate having an idea of the initial topology of the solution. A comprehensive set of starting cases is required where each case is given as an input datafile to Surface Evolver. Let us consider the square partitioning first. For the number of partitions k, where $3 \leq k \leq 10$, we consider the case where all the k partitioned regions are adjacent to the boundary of the square. For $4 \leq k \leq 10$, we consider the additional case where only $k-1$ regions are adjacent to the boundary of the square and one region is in the interior. For $6 \leq k \leq 10$, we consider one more case where $k-2$ regions are adjacent to the boundary of the square and two regions are in the interior. For $k = 9$ and $k = 10$, we add the case where three regions are in the interior and $k-3$ regions are adjacent to the sides of the square. The number of edges touching the boundary varies for each of the cases and we determine all the possible ways to arrange the varying number of edges on the sides of the square. We eliminate possible duplicates occurring due to rotational or mirror symmetry. For each possible starting partition, we allow Surface Evolver to minimize the perimeter of the cuts while permitting it to pop vertices if necessary. Note that in the case of two or more regions in the interior, we try the two possibilities of connecting them by either a common edge or common vertex. It is worth mentioning as well that we examined the case where we start with triangular faces for the interior regions and the minimization process would pop new vertices to find a minimal solution (up to $k-1$ Steiner points for the instance of one internal region for example).

The same procedure described above was followed for the (1×2) rectangle. However, the number of topological starting combinations to try in the case of the rectangle was higher because of the loss of symmetry, since all the sides of the original are no longer the same length. For the circular cuts, the same procedure was followed while allowing Surface Evolver to refine the internal edges. Note that the refinement process was not started from the straight line cuts that we have determined to be the best but rather all combinations were tried again in this case. It turned out for the cases considered in this chapter, the circular cut solutions were simple refinements of the best straight line solutions.

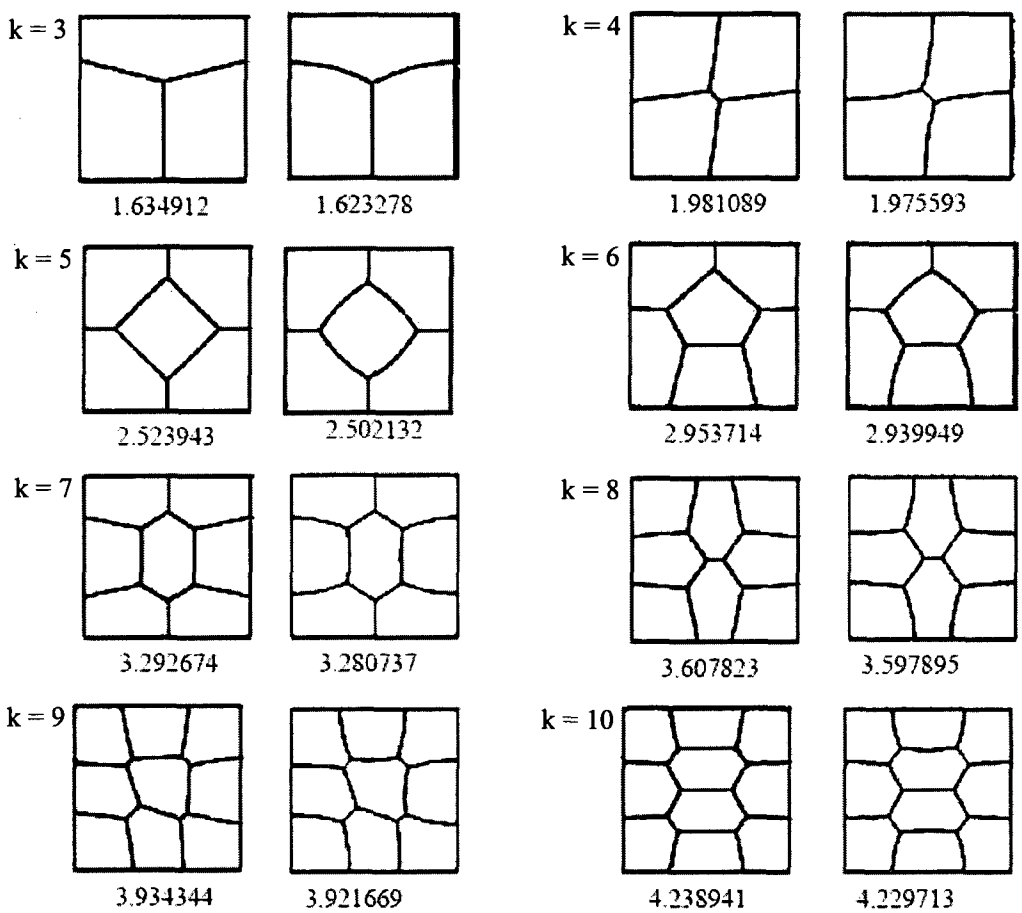

Figure 44: The determined best partitions of the unit square.

5.3 Results

To summarize the results, Figure 44 shows a list of what we have determined to be the best solutions for the three to ten equal area partitions of the unit square for both straight lines and circular cuts. Each partition is labelled with the corresponding number of partitions k and the sum of the perimeters of the cuts. Figure 45 exhibits the found best straight line partitions of the (1×2) rectangle for the cases where $3 \leq k \leq 6$.

Examining the comprehensive set of experimental results obtained, the following observations can be made regarding the optimal solution:

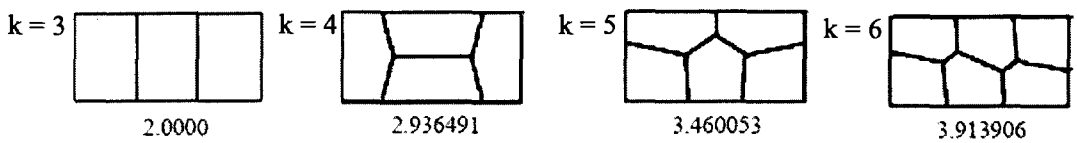

Figure 45: The determined best partitions of the (1×2) rectangle.

Observation 31. *Total circular cuts perimeter is less than total straight line cuts perimeter.*

Observation 32. *All Steiner points in the interior of the square (and the rectangle) are of degree* 3.

Observation 33. *For circular cuts, the interior angles at a Steiner point are* $120°$ *(this has been confirmed by using analytical representations of the best solutions for* $k = 3, 4, 5$*). The interior angles of edges with vertices on the boundary of the original square are* $90°$ *(again, this has been confirmed for* $k = 3, 4, 5$*).*

Observation 34. *All partitions have mirror symmetry: either with respect to vertical or horizontal lines or with respect to the diagonal. In addition, several partitions have a* $180°$ *rotational symmetry where the same partition is obtained after rotating the square by* $180°$*. Note that for* $k = 5$ *the partition has* $90°$ *symmetry as well.*

Based on the previous set of observations, we make the following conjecture:

Conjecture 35. *For a large number of components in partitions, we expect a "honeycomb" pattern shape - a tiling of approximately regular hexagons.*

We agreed on Conjecture 35 with Simon who has unpublished results on this problem [267]. This conjecture is further supported by the relatively recent proof—by Hales—of the very long standing "Honeycomb Conjecture" (attributed to the Greek mathematician Pappus of Alexandria). The honeycomb conjecture, now the honeycomb proof, states that for any equal area partition of the plane, the regular hexagonal grid is the one with minimum perimeter [148].

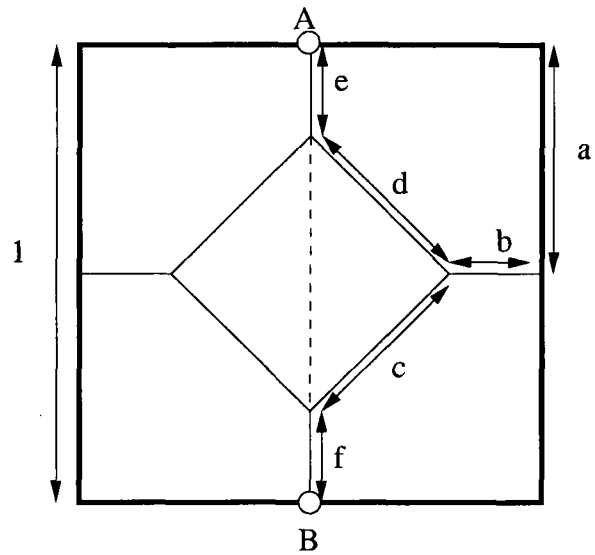

Figure 46: 5-partition of the unit square.

5.4 Analytical backing

The goal of this section is to strengthen the result obtained from Surface Evolver for the five partition by obtaining it analytically. We consider the partition of the unit square into five equal areas as shown in Figure 46.

The perimeter of the cuts l can be expressed as $l = 2b + 2c + 2d + e + f$.

The areas of the regions on the right of the line (AB) (the top-right, bottom-right and middle one) can be expressed as follows and are known respectively to be

$$area1 = (1 - e - f)\frac{(\frac{1}{2} - b)}{2}$$

$$area2 = \frac{ab}{2} + \frac{e}{4} + \sqrt{s(s-d)(s-\sqrt{e^2+\frac{1}{4}})(s-\sqrt{a^2+b^2})}$$

where

$$s = \frac{d + \sqrt{e^2 + \frac{1}{4}} + \sqrt{a^2 + b^2}}{2}$$

$$area3 = \frac{(1-a)b}{2} + \frac{f}{4} + \sqrt{s'(s'-c)(s'-\sqrt{f^2+\frac{1}{4}})(s'-\sqrt{(1-a)^2+b^2})}$$

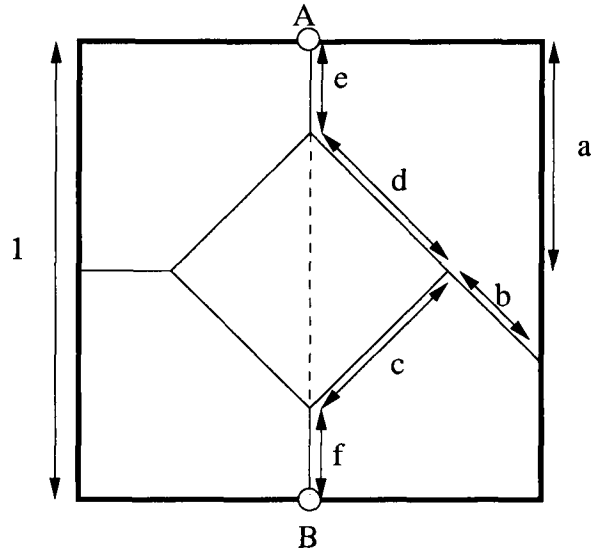

Figure 47: 5-partition of the unit square while allowing b to move.

where

$$s' = \frac{c + \sqrt{f^2 + \frac{1}{4}} + \sqrt{(1-a)^2 + b^2}}{2}$$

Assuming symmetry along the line (AB) and assuming b to move horizontally, the optimization of the perimeter function can be reduced to the optimization of a function of the three parameters a, e and f.

If we remove the assumption for b and let it move as in Figure 47, the angle between a and b is introduced as a new parameter. Again, we reduce the problem to the optimization of a function of three parameters a, e and f while varying the angle explicitly. The value is minimized when b is horizontal as was assumed above.

In both cases, the result is the same as the numerical approximation given by Surface Evolver: 2.523943.

5.5 Conclusion

Tables 4 and 5 display a summary of the perimeter of the cuts with a comparison with previously published results. Table 4 shows the results for nonorthogonal straight line cuts

Table 4: Straight line partitions.

k	Previous Result	Surface Evolver Approximation	Our Analytical Result
3	1.63282 [52]	1.63491250321	1.634912503
4	1.9810890 [55]	1.98108902015	1.981089020
5	–	2.52394331793	2.523943328
6	–	2.95371443853	–
7	–	3.29267460641	–
8	–	3.60782394229	–
9	–	3.93434474798	–
10	–	4.23894158524	–

Table 5: Circular cut partitions.

k	Previous Result	Surface Evolver Approximation
3	1.592 [52]	1.62327887438
4	–	1.97559321003
5	–	2.50213220778
6	–	2.93994972004
7	–	3.28073781333
8	–	3.59789581746
9	–	3.92166941092
10	–	4.22971391094

while Table 5 is reserved for circular cuts. For $k = 3$ and for both straight line and circular cuts, the perimeter of the cuts we obtained are greater than the perimeter obtained in [52], although the positioning and arrangement of the straight line cuts are the same as for our solution. For $k = 4$, our results coincide with the previous ones [55], for both the best solution obtained from Surface Evolver and the analytical calculation.

Chapter 6

Packing

6.1 Introduction

In this chapter, we are interested in packing the maximum number of axis-aligned squares in a simple polygon. Fowler et al. proved the problem to be **NP**-complete for polygons with holes [121]. The reduction is based on the construction of intersection graphs. Given $X = \{x_1, x_2, \cdots, x_n\}$ objects to pack, assign a vertex v_i for each object. An edge (v_i, v_j) exists if the corresponding objects overlap. Considering a finite number of positions for the objects, the problem reduces to a maximum independent set problem on a general graph, which is known to be **NP**-complete. Hochbaum and Maass designed a PTAS for the version of the problem where the input polygons are orthogonal grid polygons and where the squares are $(k \times k)$ squares [155]. Their algorithm is based on the *shifting strategy* for covering and packing problems. The strategy allows to bound the error of the simple divide-and-conquer approach by applying it repetitively and then choosing the best solution. Let l be the shifting parameter, their algorithm works as follows: the input polygon is divided into vertical strips of width k and groups of l consecutive strips are considered. Repeating the shifting l times results in the starting partition. For each vertical partition, the same strategy is applied horizontally and a local enumeration algorithm is applied to obtain optimal solutions for the resulting $(lk \times lk)$ squares. The running time of this algorithm is $O(k^2 l^2 n^{l^2})$ where n is the number of grid squares inside the boundary of the given polygon and its approximation

ratio is $(1 + \frac{1}{l})^2$. Baur and Fekete address a similar problem which consists of packing k $(L \times L)$ squares into an input polygon—possibly with holes—such that L is as big as possible. The authors prove that, unless $\mathbf{P} = \mathbf{NP}$, there is no polynomial time algorithm that finds a solution within more than $\frac{13}{14}$ of the optimum for orthogonal polygons and they give a polynomial algorithm that solves their problem with a $\frac{2}{3}$-approximation ratio. They also conjecture that the problem we are interested in is polynomial for simple orthogonal polygons [31].

Here is a formulation of the problem at hand:

Problem 36. *Given a simple polygon P and an $(L \times L)$ square S, maximize the number of axis-aligned copies of S that can be packed in P.*

The chapter is organized as follows. Section 6.2 contains several general definitions needed for the rest of the chapter. In Section 6.3, we present three polynomial time algorithms for packing the maximum number of unit squares in three classes of orthogonal polygons: the staircase polygons, the pyramids (or double staircase polygons) and Manhattan skyline polygons. In Section 6.4, we study a special case of the problem for general orthogonal polygons. We conclude in Section 6.5 by posing the problem in the latter section from a graph theory perspective.

6.2 Definitions

Let the polygon in which the maximum number of squares is packed be called the *container* polygon. In the following, we embed container polygons on unit square grids. Two $(k \times k)$ squares are said to have the same *alignment* in a packing if the coordinates of their respective corners are ik apart on the grid and they are said to be have the same horizontal (vertical) alignment if their respective $x(y)$-coordinates are jk on the grid apart where i and j are integers. Figure 48 illustrates the different alignments of unit squares on a square grid.

An area of the container where no square is packed is called a *p-hole* (packing hole). An *internal* p-hole is one that is not adjacent to the boundary of the container. Figure 49 shows two examples of p-holes shaded in blue.

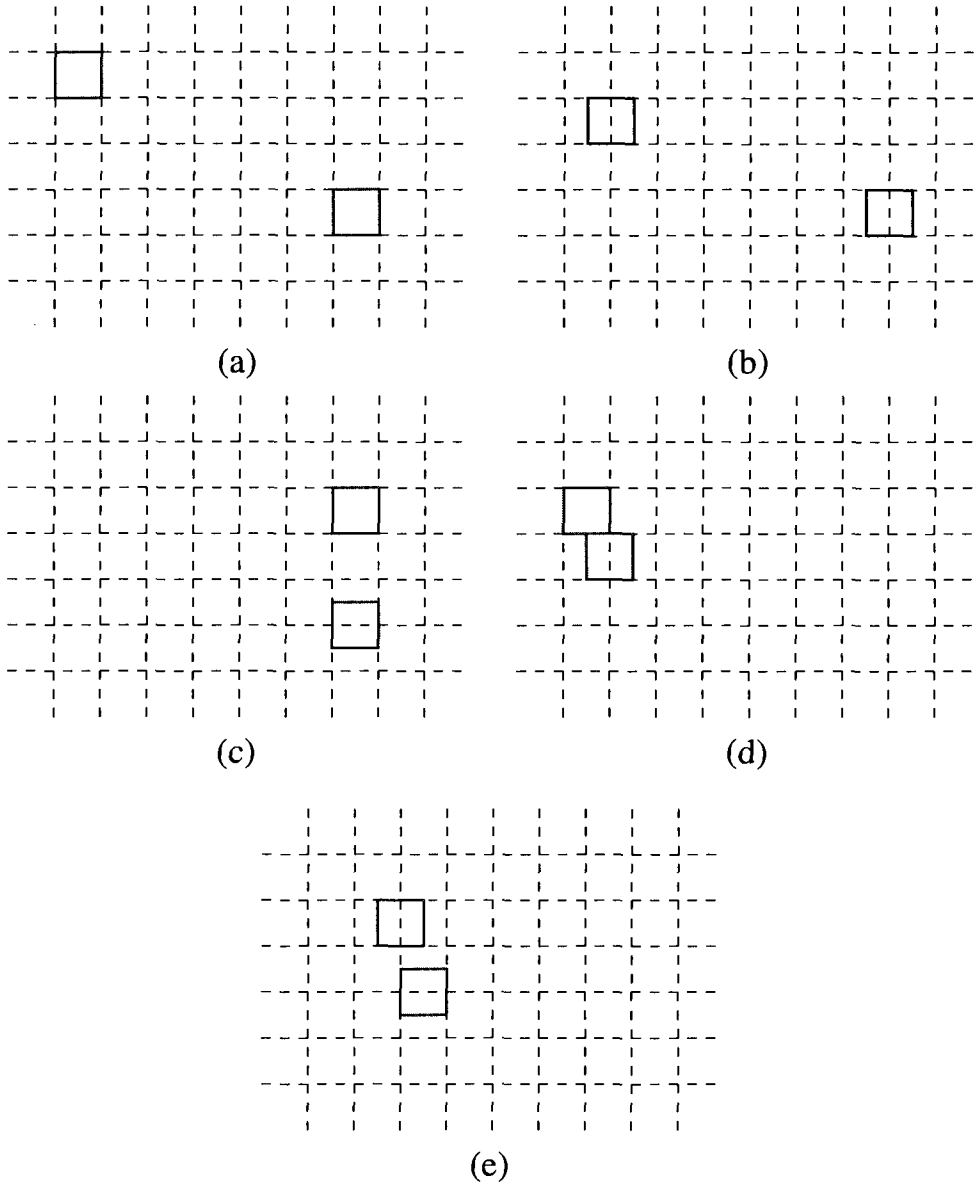

Figure 48: Different alignments of unit squares: (a) and (b) two squares with same alignment, (c) two squares with same vertical alignment only, (d) two squares with same horizontal alignment only (e) two squares with different alignments.

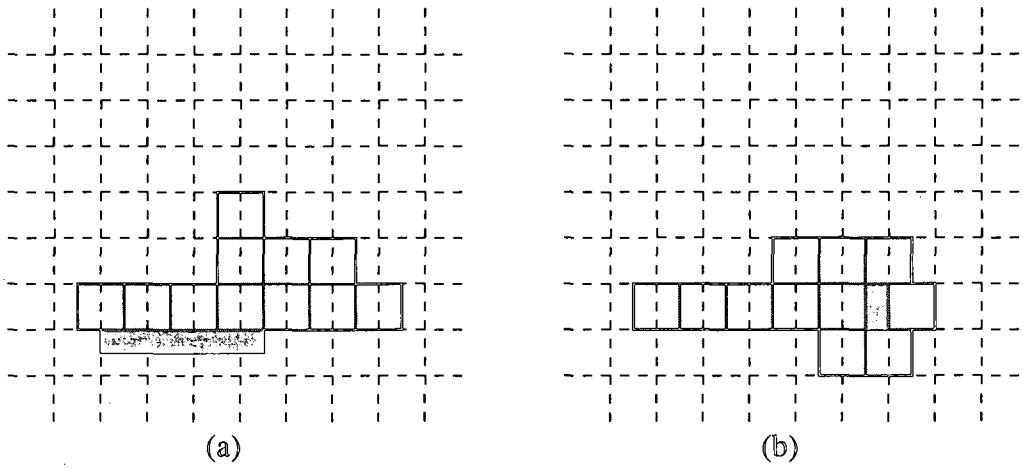

Figure 49: (a) A p-hole (shaded in blue) on the boundary, (b) an internal p-hole (shaded in blue).

We orient our orthogonal polygons such that their edges are parallel to the axes. As a part of our techniques to prove some characteristics of an optimal solution, we apply *gravity* to the squares in the packing. Applying gravity consists of displacing a square, when possible, into a lower adjacent p-hole. Figure 50 (a) and (b) show respectively an optimal solution for a given polygon and the application of gravity to the squares in the solution. The p-holes are shaded in red.

A *Manhattan skyline* polygon is an orthogonal polygon monotone with respect to one of its edges, call it *base*. A *pyramid* polygon (or *double staircase* polygon) is a Manhattan skyline with no dents. A *staircase* polygon is a pyramid monotone with respect to both the x and y axes, i.e. it has two bases. The monotone polyline linking the two bases in a staircase polygon is called a *stair* polyline. Figures 54, 52 and 51 show respectively examples of a Manhattan skyline polygon, a pyramid and a staircase polygon. Let U-edge denote a dent of a Manhattan skyline polygon P and let L-edge denote a horizontal edge with one reflex endpoint. Each horizontal edge e is defined by its y-coordinate y_e and its length l_e.

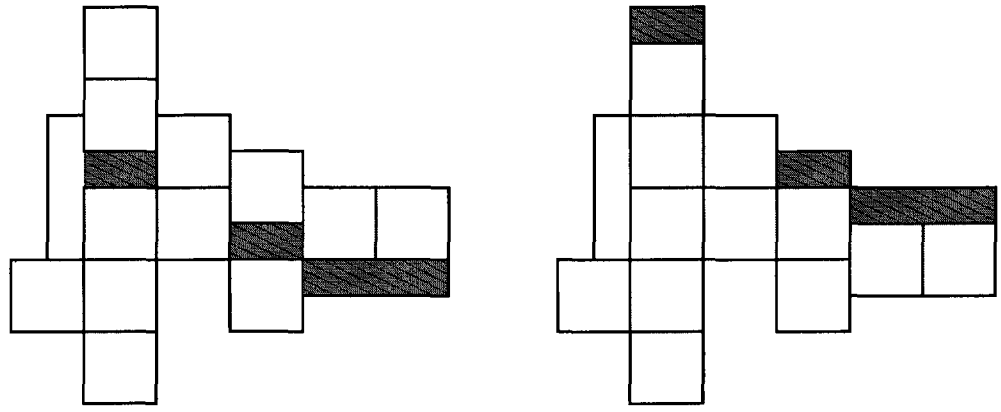

Figure 50: (a) An optimal solution for packing an orthogonal polygon with 2×2 squares, (b) gravity applied to the squares in the solution.

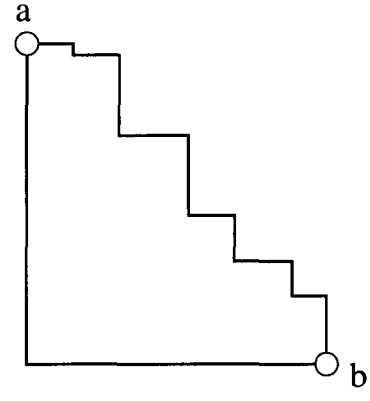

Figure 51: An example of a staircase polygon and a stair polyline $P[a \,..\, b]$.

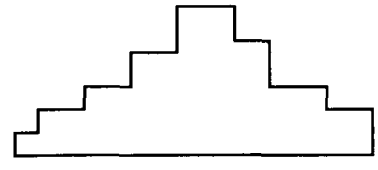

Figure 52: An example of a pyramid.

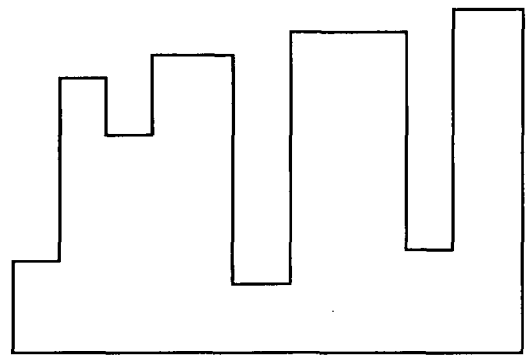

Figure 53: An example of a Manhattan skyline polygon.

6.3 Polynomial time algorithms for special cases

6.3.1 A lemma and an observation

Observation 37 and Lemma 38 are important to our polynomial time algorithms.

Observation 37. *An $(l \times w)$ rectangle can be optimally packed with unit squares in $O(1)$ time. The optimal number of squares is $(\lfloor l \rfloor \times \lfloor w \rfloor)$.*

Lemma 38. *Flooring the y-coordinate of U-edges and L-edges does not affect the optimality of a solution (for the problem of packing the maximum number of squares in staircase, pyramid and Manhattan skyline polygons).*

Proof. Flooring the y-coordinate y_e of an U-edge or L-edge e consists of replacing it with $\lfloor y_e \rfloor$ as shown in Figure 54 (b) for a Manhattan skyline polygon. It implies deleting from the given polygon P rectangles of the form $(l_e \times (y_e - \lfloor y_e \rfloor))$. If the given polygon P has only U- and L-edges with integer coordinates then flooring them does not bring any changes to the polygon nor to the optimal solution and we are done. Otherwise, assume that that every solution has at least $k \geq 1$ squares intersecting the to-be-deleted rectangles. Consider the solution with the minimum number k of such squares. Figure 55 shows two examples of squares intersecting the to-be-deleted rectangles: (a) shows one near an U-edge and (c) shows one near an L-edge. Consider, without loss of generality, the case of an U-edge. The

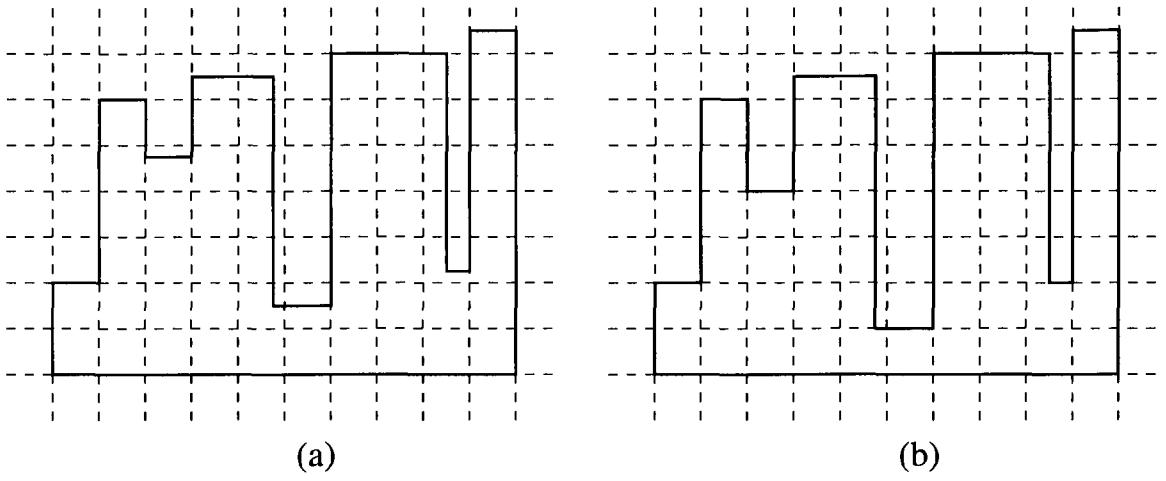

Figure 54: (a) A Manhattan skyline polygon, (b) flooring the y-coordinates of its U- and L-edges.

other case is similar. The distance d from the square to edge to horizontal base of the polygon P is fractional and hence there is a p-hole below the square. Therefore, the square can be slid down, a contradiction to the fact that k was the minimum of number of such squares. □

6.3.2 An $O(n)$ algorithm for packing unit squares in staircase polygons

In this section, we present a polynomial time algorithm for the following problem:

Problem 39. *Given a staircase polygon P, find the maximum number of unit squares that can be packed in it.*

We orient the polygon P such that its lower leftmost corner is the intersection of its two bases and we assume that this corner is a vertex of a square grid. We number the rows of the grid starting from the horizontal edge incident to this corner. We define the right wall of a row to be the edge of the stair polyline with the smallest x-coordinate that is partially or fully contained in the row. We first prove that there always exists an optimal solution with a square in the lower left corner of a staircase polygon (unless the maximum is zero). Then we prove that there always exists an optimal solution where all the packed squares are

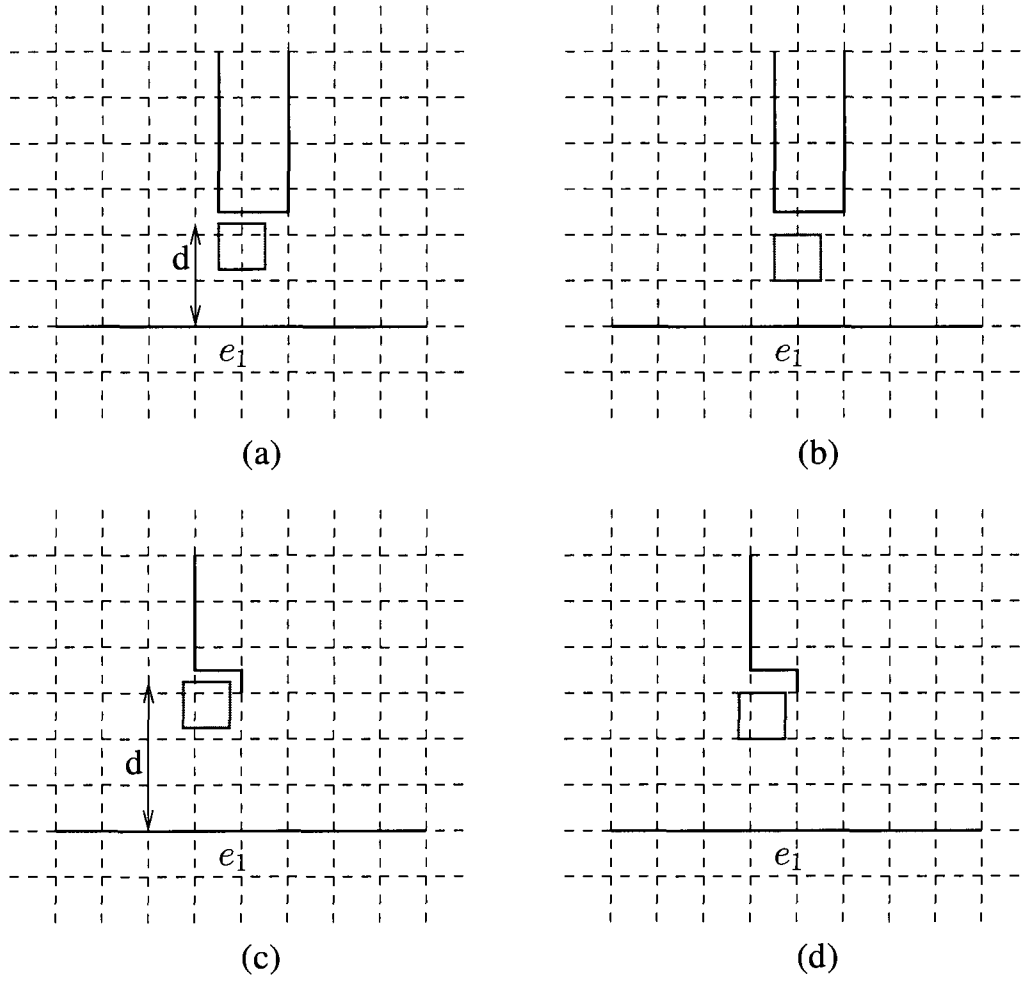

Figure 55: Flooring the y-coordinates of U-edges (a) and L-edges (b) does not affect the number of squares in an optimal solution.

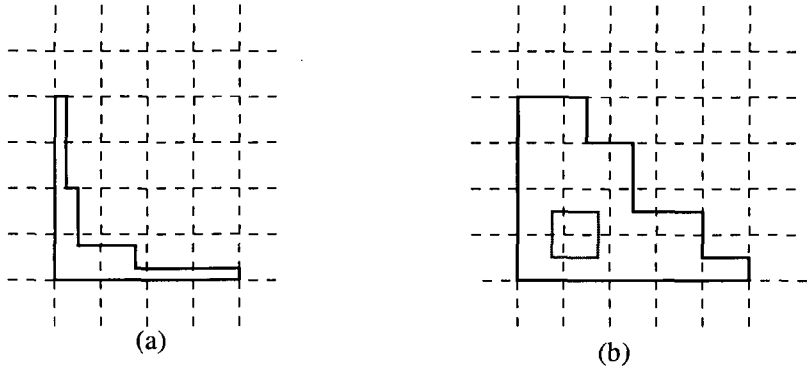

Figure 56: (a) A staircase polygon where the maximum number of unit squares is zero, (b) the red square can be displaced to occupy the corner.

on the grid and therefore that a greedy algorithm is possible.

Lemma 40. *Every optimal solution has an equivalent optimal solution with a square in the lower left corner unless the optimal number of squares is zero (see Figure 56 (a)).*

Proof. Consider the grid square at this corner and consider any feasible solution. If, in this solution, this grid square is empty then the solution is not optimal. Therefore, as shown in Figure 56, the lower leftmost grid square is intersected by a packed (red) square. Since the grid square can be intersected by no other packed square, the red square can be slid to touch the corner. □

Lemma 41. *There always exists an optimal solution such that all packed squares are aligned on the grid.*

Proof. Assume that every optimal solution has at least $k \geq 1$ squares not aligned on the grid. Consider the solution with the minimum number k of nonaligned squares. Let z be the grid cell that is: (1) closest to the corner (in manhattan distance) and (2) not covered with one square. z, which is shown in blue in Figure 57, is either: covered with no squares; partially covered with one square; or partially or totally covered by two or more squares. The first case contradicts optimality, since z can be covered with an additional square. The second case contradicts the fact that all optimal solutions have at least k squares not aligned

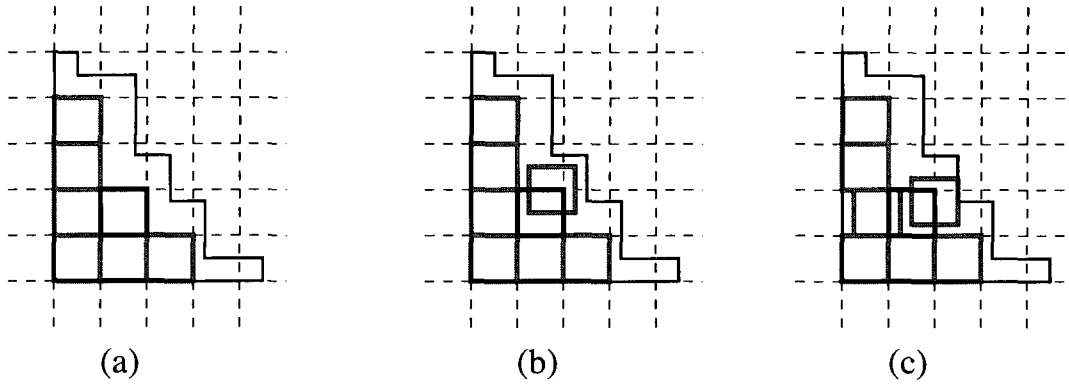

Figure 57: The closest grid cell to the corner not covered with a square is either: (a) covered with no squares, (b) partially covered with one square, (c) partially or totally covered by two or more squares.

on the grid since it is possible to align the square partially covering z to fully cover z giving a solution of the same size with k-1 squares not aligned on the grid. The third case gives an obvious contradiction. It is geometrically impossible because z has the side of the polygon or grid-aligned squares to the left and below (otherwise it is not the closest to the corner). □

Theorem 42. *An optimal solution is obtained in $O(n)$ time by greedily filling a staircase polygon P starting from the lower leftmost corner. All squares in this solution are aligned on the grid.*

Proof. Let $x_{r,i}$ be the x-coordinate of the right wall of row i. Let x_v be the x-coordinate of the vertical base of the polygon P and y_h the y-coordinate of the horizontal base of the polygon P. The algorithm is as follows: set $max = 0$. Choose the L-edge e with the smallest y-coordinate y_e. If j is the row in which this L-edge is contained then $max \leftarrow max + \lfloor x_{r,j} - x_v \rfloor \times \lfloor y_e - y_h \rfloor$ (by Observation 37). Let $y_h = y_e$ and iterate. Finally, add the maximum number of squares obtained in the last remaining rectangle to max. Report max as the maximum number of squares that can be packed in P.

The running time of this algorithm depends on the number of horizontal edges on the boundary of P. This number is bounded by n. Therefore the algorithm, which does constant work each horizontal edge, runs in $O(n)$ time.

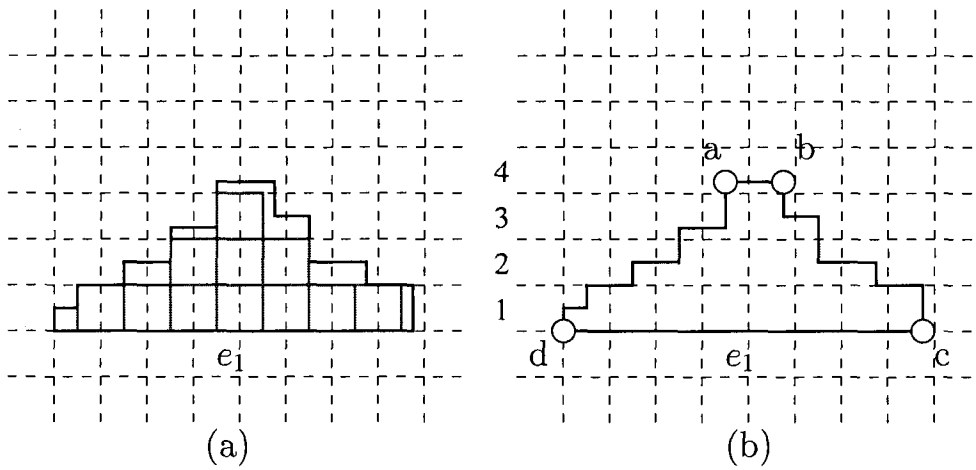

Figure 58: (a) An optimal solution for a pyramid, (b) the grid numbering starting at e_1 and the left and right walls for row 2.

Assume that the solution obtained by the algorithm is not optimal, i.e. there exists an optimal solution with one or more squares. By Lemma 41, this solution has an equivalent one where all packed squares are on the grid. Therefore, there exists a row in the latter solution with at least one more square, which is not possible since the algorithm packs the maximum number of squares in a row.

□

6.3.3 An $O(n)$ algorithm for packing unit squares in pyramids (or double staircase polygons)

In this section, we present a polynomial time algorithm for the following problem:

Problem 43. *Given a pyramid P, find the maximum number of unit squares that can be packed in it.*

We orient P such that it is monotone with respect to the horizontal axis and we lay the horizontal edge of the polygon P with the minimum y-coordinate on a square grid. Call this edge e_1. We number the rows of the grid by their level starting from this edge as shown in Figure 58 (b). Let a and b be the endpoints of the line segment with the largest y-coordinate

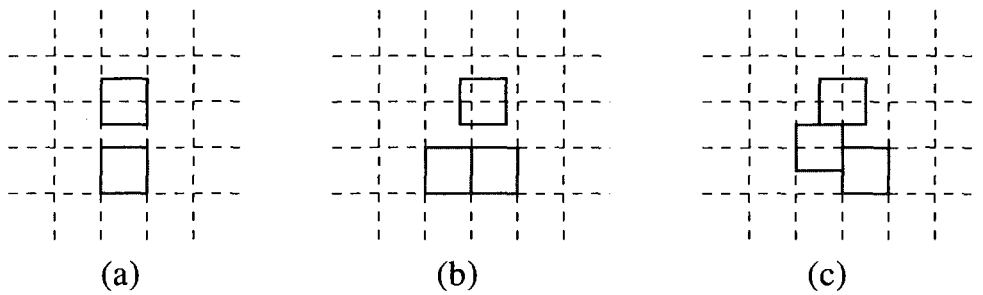

Figure 59: (a) A p-hole where squares adjacent to it have the same vertical alignment, (b) a p-hole where squares adjacent to it do not have the same vertical alignment and the upper adjacent square is not blocked, (c) a p-hole where squares adjacent to it do not have the same vertical alignment and the upper adjacent square is blocked.

in P and let c and d be the endpoints of e_1. Define the left chain of the polygon P to be the polyline $P[d\,..\,a]$ and the right chain of P to be the polyline $P[b\,..\,c]$ (see Figure 58). Define the left wall of a row to be the edge of the left chain with the greatest x coordinate and that is partially or fully contained in the row. Define the right wall of a row to be the edge of the right chain with the smallest x coordinate and that is partially or fully contained in the row. Figure 58 (b) shows the left and right walls of row 2.

Lemma 44. *There always exists an optimal solution such that all packed squares are vertically aligned on the grid.*

Proof. Assume that gravity has been applied to every optimal solution and that every optimal solution has at least $k \geq 1$ squares not vertically aligned on the grid. Consider the solution with the minimum number k of nonaligned squares. Let z be the p-hole that is closest to e_1 by vertical distance. The adjacent squares to z either have the same horizontal alignment as shown in Figure 59 (a) or have different alignments. In the latter case, the upper square adjacent to the p-hole can be either blocked or not (by another square) as shown in Figure 59 (b) and (c) respectively. In the first two cases, the upper adjacent square can be slid into z which contradicts the fact that gravity was applied. In the third case, the distance d of the blocking square (see the green square in Figure 60) to the edge e_1 is either fractional or integer as shown in Figure 60 (a) and (d) respectively. In the case d is

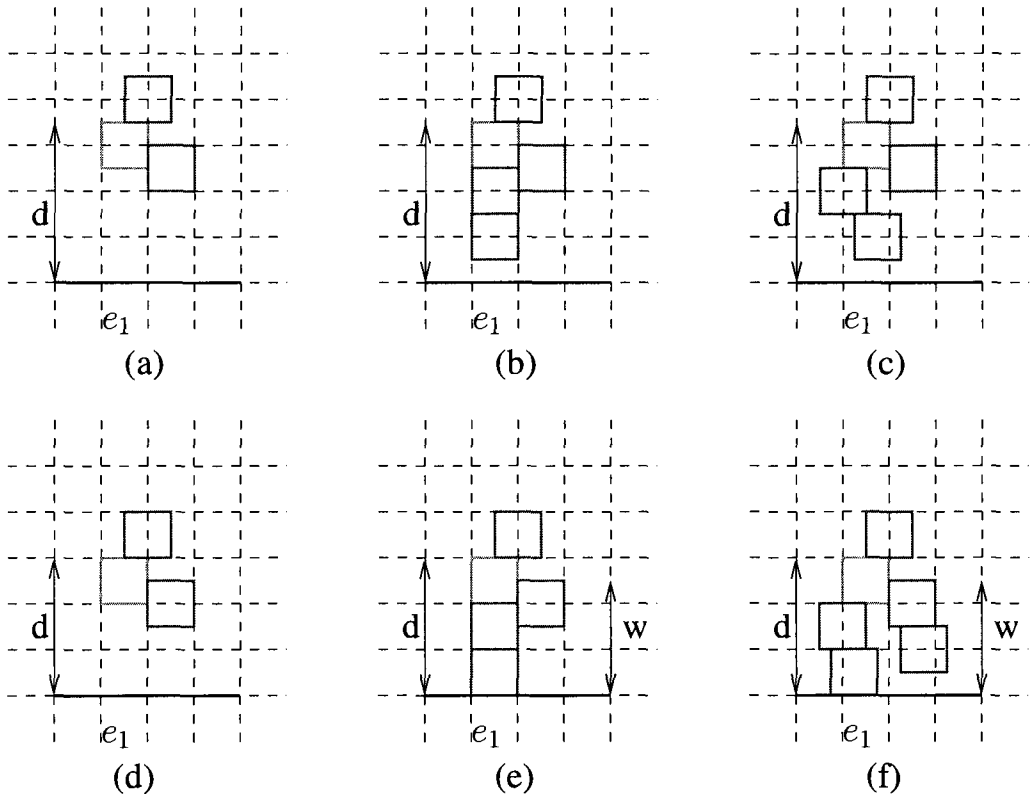

Figure 60: (a) A p-hole where squares adjacent to it do not have the same vertical alignment and the upper adjacent square is blocked by the green square, (b) the squares below the green square have either the same alignment or (c) different alignments.

fractional, there will be a p-hole "below" the blocking square since e_1 is laid on the integer grid. This contradicts the fact that z was the closest to e_1. In the case d is an integer, the blocking square has an integer number of squares "below" it (for the solution to be optimal). However, in this case, the lower square adjacent to the hole has distance w to e_1 where w is fractional. Hence, there is a p-hole below this square lower than z, a contradiction (d' being fractional implies also the nonoptimality of the solution). □

It is important to note that Theorem 45 implies Theorem 42.

Theorem 45. *An optimal solution is obtained in $O(n)$ time by greedily filling a pyramid P starting from e_1. All squares in this solution are vertically aligned on the grid.*

Proof. Let $x_{l,i}$, $x_{r,i}$ be the x-coordinates of the left and the right walls of row i, respectively. Let y_h the y-coordinate of the horizontal base of the polygon P. The algorithm is as follows: set $max = 0$. Choose the L-edge e with the smallest y-coordinate y_e. If j is the row in which this L-edge is contained then $max \leftarrow max + \lfloor x_{r,j} - x_{l,j} \rfloor \times \lfloor y_e - y_h \rfloor$ (by Observation 37). Let $y_h = y_e$ and iterate. Finally, add the maximum number of squares obtained in the last remaining rectangle to max. Report max as the maximum number of squares that can be packed in P.

The running time of this algorithm depends on the number of horizontal edges on the boundary of P. This number is bounded by n. Therefore the algorithm, which does constant work for each horizontal edge, runs in $O(n)$ time.

Assume that the solution obtained by the algorithm is not optimal, i.e. there exists an optimal solution with one or more squares. By Lemma 44, this solution has an equivalent one where all packed squares are vertically aligned on the grid. Therefore, there exists a row in the latter solution with at least one more square, which is not possible since the algorithm packs the maximum number of squares in a row.

□

6.3.4 An $O(n)$ algorithm for packing unit squares in Manhattan skyline polygons

In this section, we present a polynomial time algorithm for the following problem.

Problem 46. *Given a Manhattan skyline polygon P, find the maximum number of unit squares that can be packed in it.*

We orient P such that it is monotone with respect to the horizontal axis and we lay the horizontal edge of P with the minimum y-coordinate, say e_1, on a square grid.

It is important to note that Theorem 47 implies Theorem 45 and Theorem 42.

Theorem 47. *An optimal solution is obtained in $O(n)$ time by greedily filling a Manhattan skyline polygon P starting from e_1 (see Figure 61). All squares in this solution are vertically aligned on the grid.*

Proof. The algorithm is as follows: choose the U-edge or the L-edge e with the smallest y-coordinate y_e. The grid row in which e is contained has a left wall and a right wall. Consider the rectangle R_e defined by the intersection of the line $y = y_e$ with these two walls and the bottom edge of the polygon. Fill R_e optimally with unit squares in constant time. Remove R_e from the polygon and repeat until there are no more U- or L- edges. The removal of the rectangle might result in several subpolygons for the recursive steps. Recurse on all of them. At the end, we will be left with (vertical) rectangles which can be filled in constant time. Figure 54 shows the algorithm being run on an example Manhattan skyline polygon.

The running time of this algorithm depends on the number of horizontal edges on the boundary of P. This number is bounded by n. Therefore the algorithm, which does constant work for each horizontal edge, runs in $O(n)$ time.

Lemma 38 showed that flooring the U- and L- edges does not affect the optimality of a solution. The proof that vertically aligning squares on the grid gives an optimal solution is similar to the one of Theorem 45. □

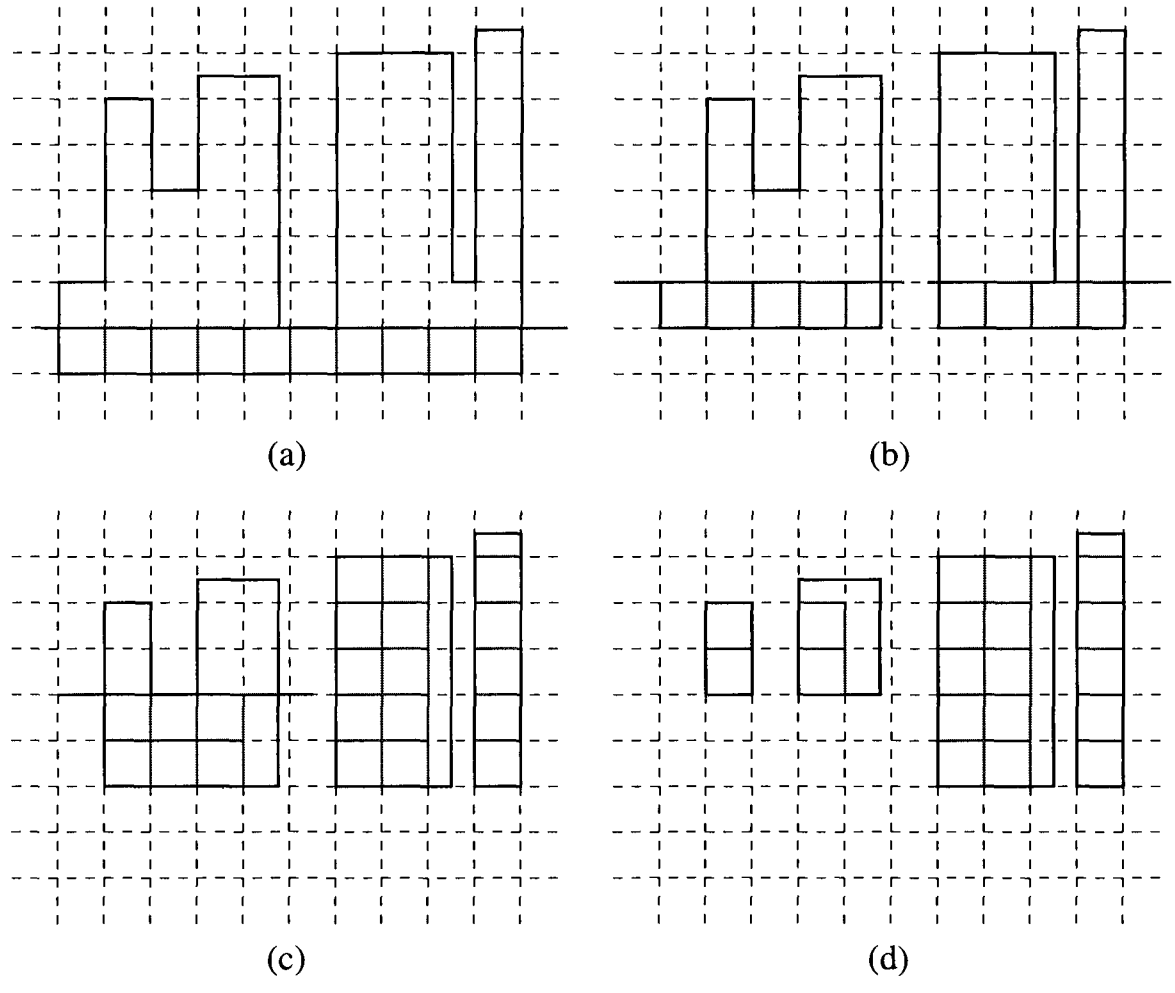

Figure 61: Applying the algorithm to an example Manhattan skyline polygon.

6.4 Toward a polynomial time algorithm for packing (2×2) squares in orthogonal grid polygons

6.4.1 Definitions

An *orthogonal grid* polygon is an orthogonal polygon whose vertices have integer coordinates. In this section, we consider packing (2×2) squares in such polygons. The packed squares are considered to be on the grid (the proof that there is always an optimal solution with the squares on the grid is similar to the proofs in the previous section). Therefore, the *alignment* of a packed square is defined by the parity of its corners' coordinates. Every square can have one of four alignments on the grid: an even-even alignment, an odd-odd alignment, an odd-even alignment and even-odd alignment as shown respectively in Figure 62 (a), (b), (c) and (d). Let y_{max} be the y-coordinate of the vertex of P with the maximum y-coordinate and let x_{min} be the x-coordinate of the vertex of P with the minimum x-coordinate. We embed P on a grid where the (0,0) point is the (x_{min}, y_{max}) point.

The p-holes exist due to the adjacency of the different alignments; Figure 63 shows the adjacency of alignments. We assume that all p-holes are of the form $(1 \times X)$ (horizontal) or $(X \times 1)$ (vertical) where X is an integer. T-like and L-like p-holes (see Figure 69 (a)) are divided into their horizontal and vertical parts. A p-hole is said to be odd or even depending on the parity of X.

6.4.2 A binary integer program (BIP)

In order to characterize optimal solutions, we present a system that takes, through a graphical interface, an orthogonal grid polygon as input and outputs an optimal solution with the maximum number of (2×2) squares that can be packed in the polygon. The system models the problem with an binary integer program (BIP) that is then solved with SCIP (Solving Constraint Integer Program), a non-commercial mixed integer programming solver [1]. An optimal solution is constructed by setting a binary variable to 1 for each corresponding (2×2) square in the solution.

Here is a description of the BIP model of the problem. We have one family of variables.

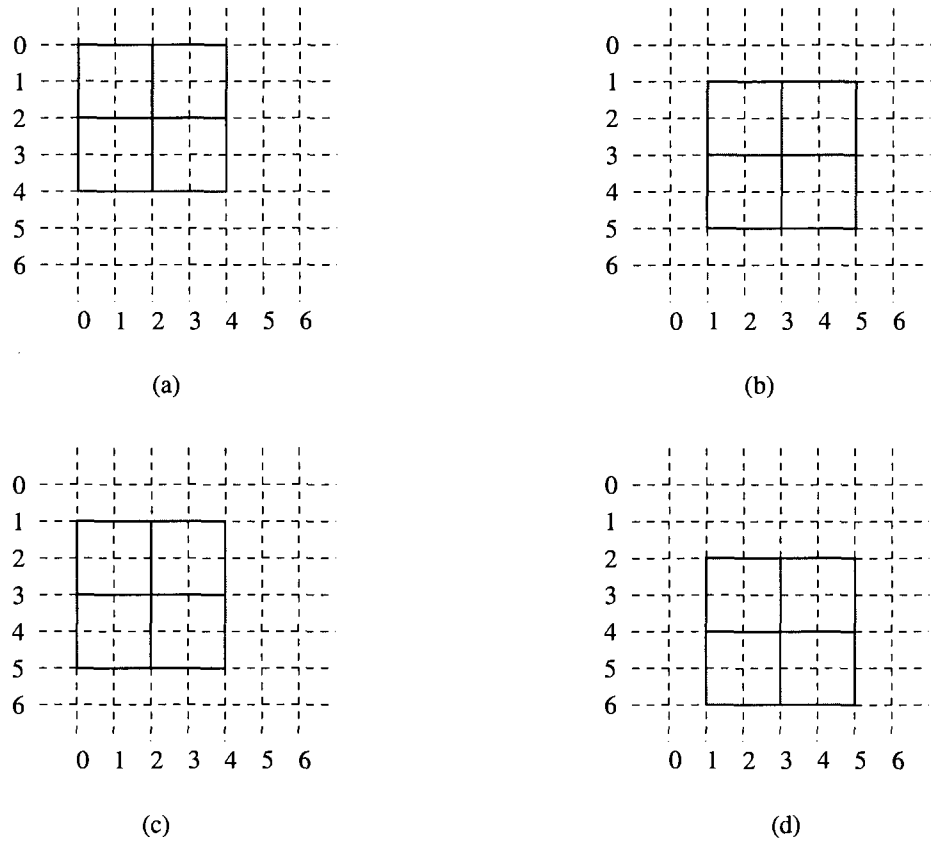

Figure 62: The four different possible alignments of (2×2) squares on the square grid: (a) even-even, (b) odd-odd, (c) odd-even, (d) even-odd.

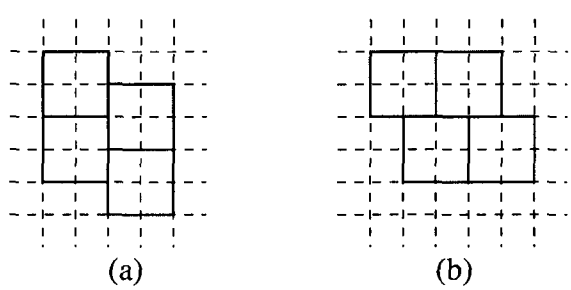

Figure 63: The adjacency of different alignments.

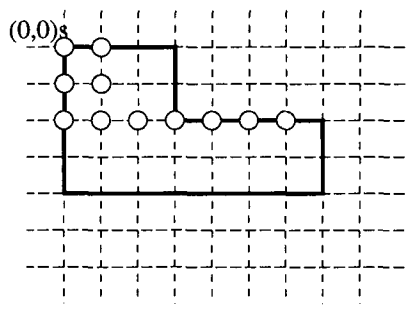

Figure 64: An example of an input polygon: the points corresponding to variables are marked in yellow.

Let O be the set of coordinates for all the upper left corners of (2×2) squares contained in P (not necessarily in the solution). For each such (2×2) square, define a variable x_{ij} representing its upper left corner where $(i, j) \in O$.

$$x_{ij} = \begin{cases} 1 & \text{if the corresponding square is in the solution.} \\ 0 & \text{otherwise} \end{cases}$$

Here is the model :

$$\max_{x_{ij}} \sum_{(i,j) \in O} x_{ij}$$

$$x_{ij} + x_{(i+1)j} + x_{i(j+1)} + x_{(i+1)(j+1)} \leq 1 \quad \forall\, (i,j), (i+1,j), (i,j+1), (i+1,j+1) \in O \quad (3)$$

where (3) ensures that there are no overlapping squares in the solution.

Let the polygon in Figure 64 be an example input polygon embedded on the grid shown. The set $O = \{(0,0), (0,1), (1,0), (1,1), (2,0), (2,1), (2,2), (2,3), (2,4), (2,5)\}$ and hence the set of corresponding variables is $\{x_{00}, x_{01}, x_{10}, x_{11}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}\}$.

The binary integer program produced by the system for the example polygon is the following:

```
Maximize
obj: x0_0+x0_1+x1_0+x1_1+x2_0+x2_1+x2_2+x2_3+x2_4+x2_5
Subject To
c1:x0_0+x1_0+x0_1+x1_1<= 1
c2:x0_1+x1_1<= 1
c3:x1_0+x2_0+x1_1+x2_1<= 1
c4:x1_1+x2_1+x2_2<= 1
c5:x2_0+x2_1<= 1
c6:x2_1+x2_2<= 1
c7:x2_2+x2_3<= 1
c8:x2_3+x2_4<= 1
c9:x2_4+x2_5<= 1
c10:x2_5<= 1
binary
x0_0 x0_1 x1_0 x1_1 x2_0 x2_1 x2_2 x2_3 x2_4 x2_5
End
```

The solution produced by the solver is shown in Figure 65. Other examples of solutions produced by the system are shown in Figures 67 and 66.

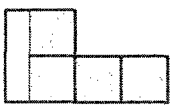

Figure 65: A screenshot of an optimal solution for a given polygon.

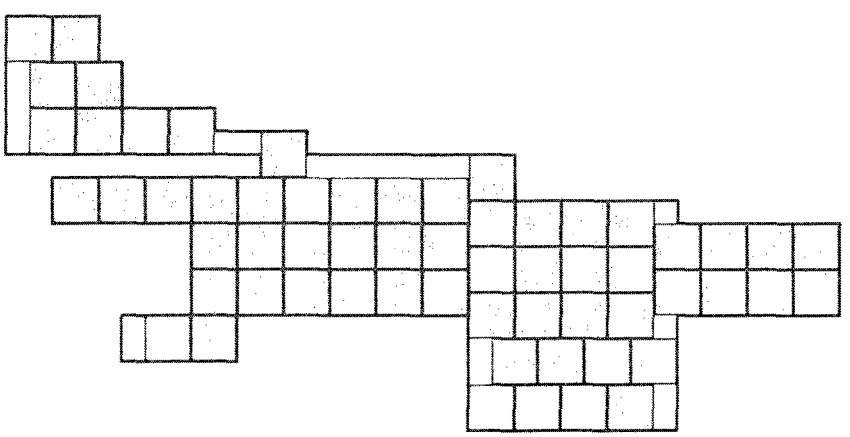

Figure 66: Another screenshot of an optimal solution for a given polygon.

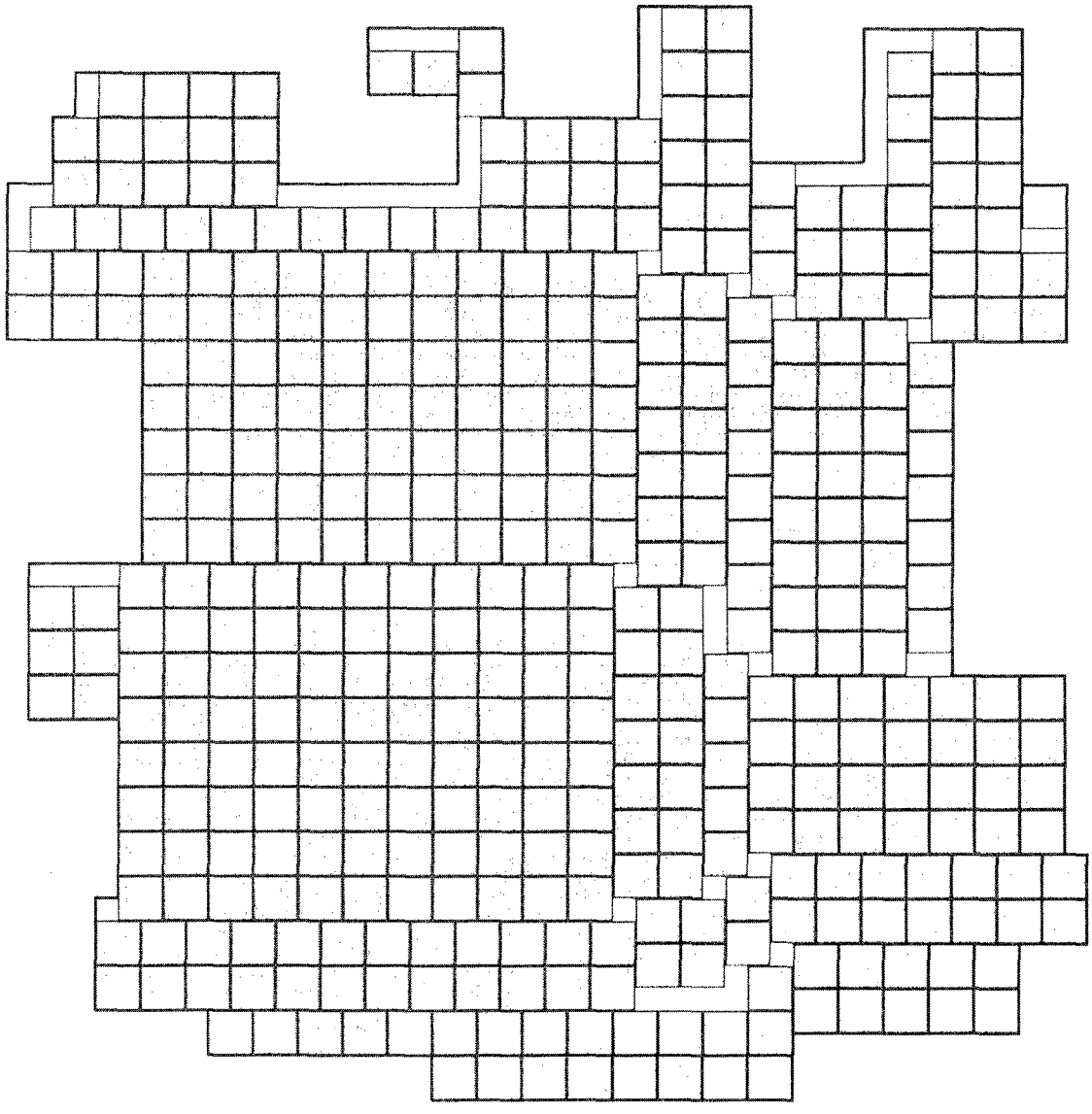

Figure 67: A third screenshot of an optimal solution for a given polygon.

6.4.3 An interesting observation and an interesting lemma

Observing the optimal solutions obtained using the system described in Section 6.4.2 led us to the following observation and lemma.

Observation 48. *A (1×1) p-hole occurs due to the adjacency of the four possible different alignments of squares as shown in Figure 68.*

Lemma 49. *There always exists an optimal solution such that all the p-holes that are not adjacent to $\delta(P)$ are (1×1) grid squares.*

Proof. Assume that every optimal solution has a number of internal p-holes and that gravity has been applied to the squares in these solutions. Consider the solution with the minimum area of internal p-holes. If the squares in this solution have one grid alignment then there are no internal p-holes and we are done. Consider that the squares have several grid alignments, see Figure 63 for an example. If the solution does not have internal p-hole, we are done. Otherwise, let z be the lowest horizontal p-hole that is not a (1×1) square; z is shaded in red in all the following figures. By our definition of p-holes, z is a $(1 \times X)$ p-hole where X is either odd or even. If z is odd then it is always possible to displace $(X-1)/2$ squares from the upper row adjacent to z in the direction of z, see Figure 69 for examples of odd p-holes (on the left) and the corresponding displacements (on the right). This contradicts the fact that we applied gravity to the solution. Hence, z is an even p-hole.

If z is even and $X > 2$ then it is then always possible to displace at least $(X-2)/2$ squares from the upper row adjacent to z in the direction of z, see Figure 70 (a) and (b) for

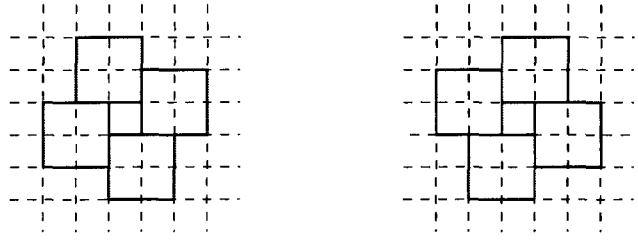

Figure 68: The two different ways in which a (1×1) p-hole occurs.

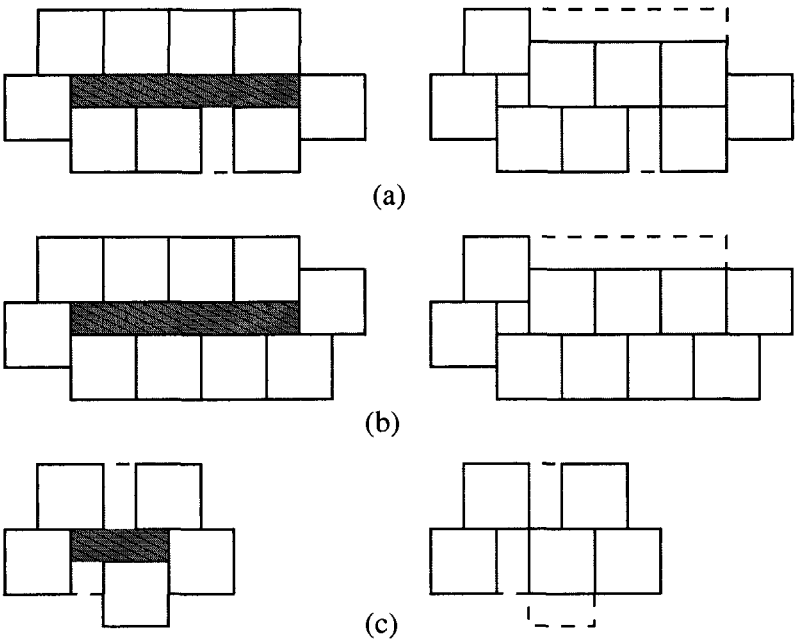

Figure 69: Examples of odd p-holes.

examples of p-holes and their corresponding displacements . If $X = 2$ and the squares from the upper row adjacent to the horizontal side of z have the same horizontal alignment as the squares from the lower row then displacing one square in the direction of z is always possible which contradicts the gravity assumption.

If $X = 2$ and the squares adjacent to the horizontal side of z have different horizontal alignments, such as in Figure 70 (c), then we need to show that there is still a displacement that decreases the area of internal p-holes. We chose z to be the lowest horizontal p-hole, hence situations similar to those in Figure 71 that allow lower p-holes (shown in red) are not possible (the solution shown in Figure 71 (a) is not even optimal unless the lower p-hole is replaced by the boundary).

Color the squares in z's column in green and assume that there are no vertical p-holes ($(X \times 1)$ p-holes) in the solution without loss of generality. If the column of green squares continues to the boundary with same horizontal alignment (see Figure 72 (a)), we can displace the entire column up and decrease the area of internal p-holes, a contradiction to

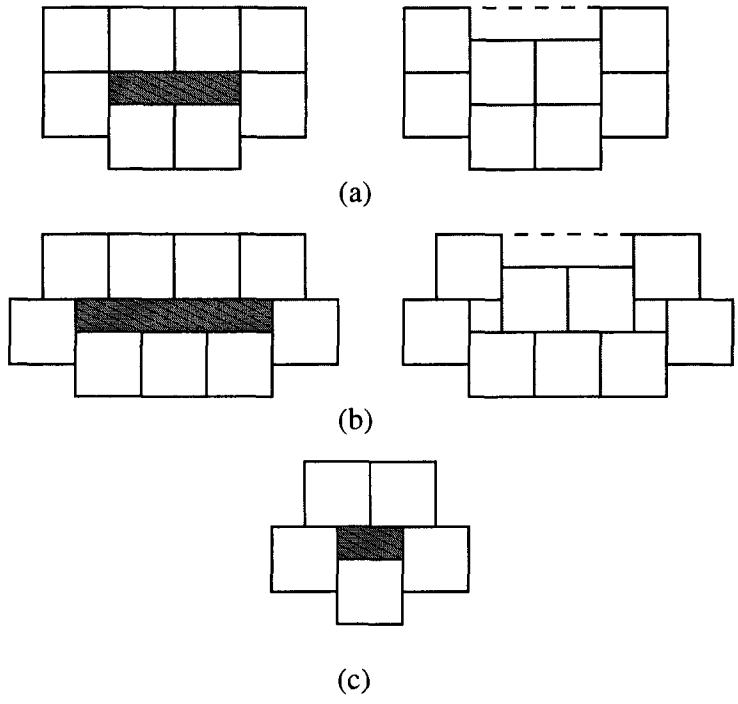

Figure 70: Examples of even p-holes.

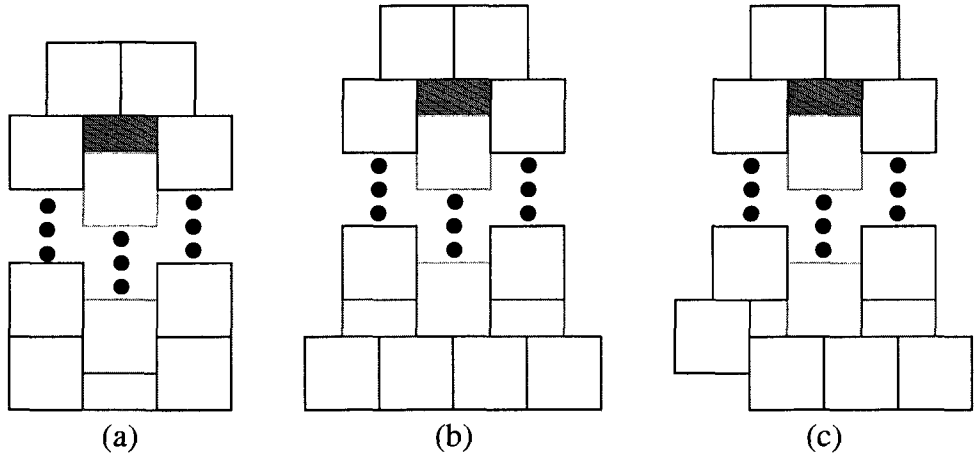

Figure 71: (1×2) p-holes that are impossible after the application of gravity.

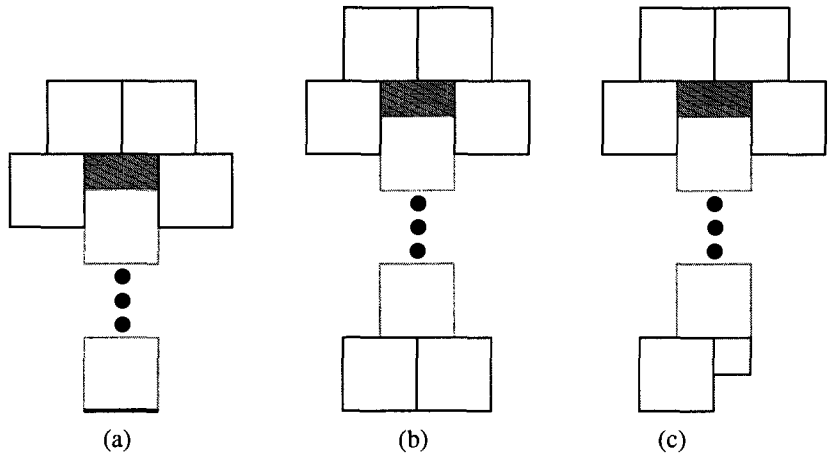

Figure 72: Cases for a (1×2) p-hole.

the fact that the solution has the minimum area of internal p-holes. Hence, the column of green squares hits a row of squares with a different horizontal alignment. Two cases are possible as shown in Figure 72 (b) and (c).

In both cases, the distance d (in grid squares) shown in Figure 73 (left figures of both (i) and (ii)) is odd. Therefore, the red squares shown on the right are both empty (but there is no (1×2) because this defeats the fact that we applied gravity). If we displace all the green squares up to fill z, we will obtain in both cases a larger p-hole (z will merge with either two or one (1×1) p-holes). We have to show that this new p-hole can be displaced to the boundary. Let us look at even p-holes (odd p-holes have similar logic).

In Figure 74 and 75, the dotted line delimits a "sub-area" of the packing and the bold line denotes the boundary. The green squares are assumed to be movable to fill the p-hole above them. Note that the size of the chosen rows of squares (3) is just an example. The argument relies on the fact that at any point during the displacement of a p-hole to the boundary, only few cases can occur: a row of squares can be followed either by a row with same alignment (see Figure 74 (a)), or a row with a different alignment. In latter case, the row can have more, less or the same number of squares. In the case it has more or less squares (and due to the application of gravity), it can have only one more (or one less) square (see Figure 74 (b) and (c)). In all three cases (see Figure 74 (b) and (c) and (d), the

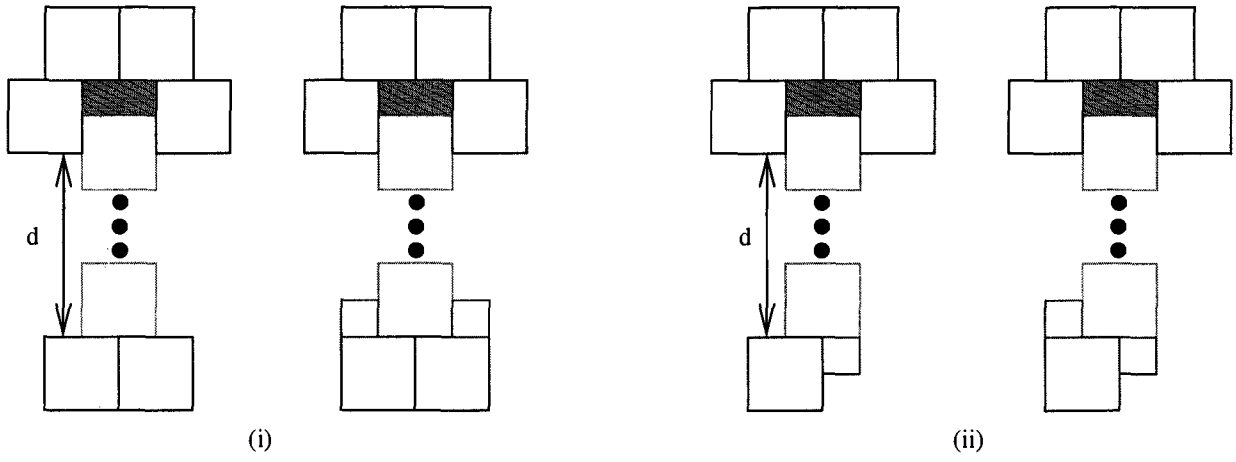

Figure 73: Subcases for a (1×2) p-hole.

red squares are empty or the row is adjacent to the boundary as shown in Figure 75. As we can see in Figure 74, being able to displace the green squares allows for the displacement of the blue squares up without modifying the total area of p-holes in this "sub-area" and thus, the p-hole can move further toward the boundary. The same logic applies for case (c) in Figure 75. The only cases where a p-hole can decrease in size while being displaced (the total area staying the same) are cases (a), (b) and (d) in Figure 75. However, in these three cases, the p-hole is already on the boundary and we are done.

□

6.5 Conclusion

In this chapter, we presented three polynomial time algorithms for packing the maximum number of unit squares in three subclasses of orthogonal polygons: the staircase polygons, the pyramids and Manhattan skyline polygons. We also studied the structure of an optimal solution for packing (2×2) squares in grid orthogonal polygons. In particular, Lemma 49 shows that p-holes have a certain structure in an optimal solution. This promising structure seems to support the long standing conjecture on this problem [31]: the existence of a polynomial time algorithm to solve it.

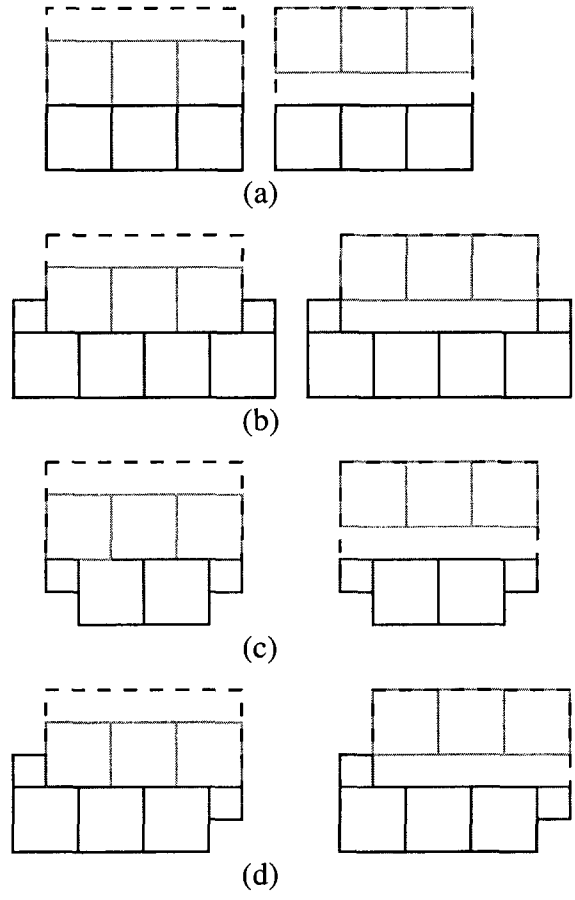

Figure 74: The different cases that can occur while displacing p-holes (1).

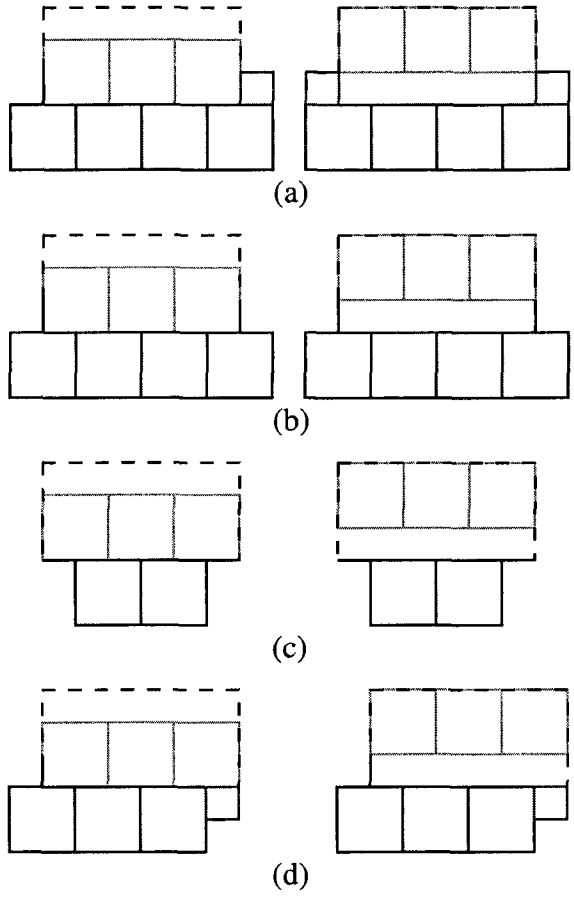

Figure 75: The different cases that can occur while displacing p-holes (2).

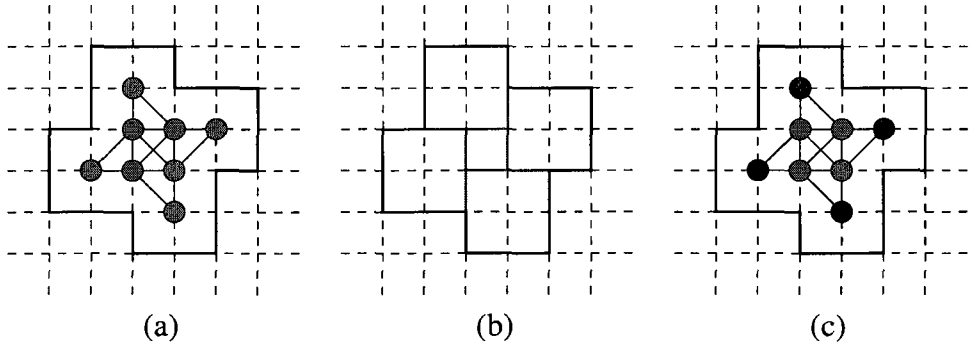

Figure 76: (a) An orthogonal polygon and its corresponding intersection graph, (b) the only optimal solution for the given polygon, (c) the equivalent independent set (vertices in blue) of the graph.

Since the problem we explored in Section 6.4 is discrete in nature, it is possible to study it from a graph theory perspective by transforming it into a maximum independent set problem on intersection graphs. Figure 76 shows the graph corresponding to an example polygon, the only optimal solution for the given polygon and the maximum independent corresponding to that solution. The family of graphs corresponding to packing in grid orthogonal polygons has several characteristics (e.g. $\Delta(G) = 8$, $\omega(G) = 4$). Therefore, the problem that is of interest to us can be posed differently: is maximum independent set on such graphs polynomially solvable?

Another interesting open problem is triangle packing. Very little is known about this problem. Variations such as packing triangles in a strip, wedge or polygon are all open.

It would also be interesting to explore packing with rotations. Except from packing the strip with rectangles with 90° rotations [108, 161, 230] and the heuristics to pack identical boxes in a "car trunk" [103], most of the work assume axis-parallel objects. Allowing rotations is interesting in many applications (like in the case of the trunk) and might give better solutions as shown by Erdös and Graham for the maximum packing of squares with unit squares [109].

Bibliography

[1] T. Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007. `http://opus.kobv.de/tuberlin/volltexte/2007/1611/`.

[2] P.K. Agarwal and N.H. Mustafa. Independent set of intersection graphs of convex objects in 2D. *Computational Geometry: Theory and Applications*, 34(2):83–95, 2006.

[3] P.K. Agarwal and M-T. Shing. Oriented aligned rectangle packing problem. *European Journal of Operational Research*, 62(2):210–220, 1992.

[4] P.K. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11(3-4):209–218, 1998.

[5] A. Aggarwal. *The Art gallery problem: Its variations, applications and algorithmic aspects.* PhD thesis, John Hopkins University, 1984.

[6] A. Aggarwal and B. Chazelle. Efficient algorithm for partitioning a polygon into star-shaped polygons. Technical report, IBM T.J. Watson Research Center, 1984.

[7] A. Aggarwal, S.K. Ghosh, and R.K. Shyamasundar. Computational complexity of restricted polygon decompositions. In Godfried Toussaint, editor, *Computational Morphology*, pages 11–11. North-Holland, 1988.

[8] O. Aichholzer, C. Huemer, S. Kappes, B. Speckmann, and C.D. Tóth. Decompositions, partitions, and coverings with convex polygons and pseudo-triangles. *Graphs and Combinatorics*, 23(5):481–507, 2007.

[9] J. Akiyama, G. Nakamura, E. Rivera-Campo, and J. Urrutia. Perfect divisions of a cake. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 114–115, 1998.

[10] J. Akiyama, G. Nakamura, E. Rivera-Campo, and J. Urrutia. Radial perfect partitions of convex sets in the plane. *Revised Papers from the Japanese Conference on Discrete and Computational Geometry*, 1763:1–13, 1998.

[11] M.O. Albertson and C.J. O'Keefe. Covering regions with squares. *SIAM Journal on Algebraic and Discrete Methods*, 2(3):240–243, 1981.

[12] H. Alt and F. Hurtado. Packing convex polygons into rectangular boxes. *Revised Papers from the Japanese Conference on Discrete and Computational Geometry*, 2098:67–80, 2001.

[13] T. Asano, T. Asano, and H. Imai. Partitioning a polygonal region into trapezoids. *Journal of the ACM*, 33(2):290–312, 1986.

[14] J. Augustine, S. Banerjee, and S. Irani. Strip packing with precedence constraints and strip packing with release times. In *Proceedings of the 18th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 180–189, 2006.

[15] L.J. Aupperle. Covering regions by squares. Master's thesis, University Saskatchewan, 1987.

[16] L.J. Aupperle, H.E. Conn, J.M. Keil, and J. O'Rourke. Covering orthogonal polygons with squares. In *Proceeding of the 26th Allerton Conference Communication, Control, and Computing*, pages 97–106, 1988.

[17] F. Aurenhammer. Weighted skeletons and fixed-share decomposition. *Computational Geometry: Theory and Applications*, 40(2):93–101, 2008.

[18] D. Avis and G.T. Toussaint. An efficient algorithm for decomposing a polygon into star-shaped polygons. *Pattern Recognition*, 13(6):395–398, 1981.

[19] Y. Azar and L. Epstein. On two dimensional packing. *Journal of Algorithms*, 25(2):290–310, 1997.

[20] C.L. Bajaj and T.K. Dey. Convex decomposition of polyhedra and robustness. *SIAM Journal on Computing*, 21(2):339–364, 1992.

[21] A.L. Bajuelos, A.P. Tomas, and F. Marques. Partitioning orthogonal polygons by extension of all edges incident to reflex vertices: lower and upper bounds. *Lecture Notes in Computer Science*, 3045:127–136, 2004.

[22] B.S. Baker, D.J. Brown, and H.P. Katseff. A $\frac{5}{4}$ algorithm for two-dimensional packing. *Journal of Algorithms*, 2(4):348–368, 1981.

[23] B.S. Baker, E.G. Coffman, and R.L. Rivest. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855, 1980.

[24] V. Bálint. Two packing problems. *Discrete Mathematics*, 178(1-3):233–236, 1998.

[25] N. Bansal, X. Han, K. Iwama, M. Sviridenko, and G. Zhang. Harmonic algorithm for 3-dimensional strip packing problem. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1197–1206, 2007.

[26] R. Bar-Yehuda and E. Ben-Chanoch. A linear time algorithm for covering simple polygons with similar rectangles. *International Journal of Computational Geometry and Applications*, 6(1):79–102, 1996.

[27] G. Barequet, M. Dickerson, and D. Eppstein. On triangulating three-dimensional polygons. In *Proceedings of the 12th Annual Symposium on Computational Geometry*, pages 38–47, 1996.

[28] F.W. Barnes. How many $(1 \times 2 \times 4)$ bricks can you get into an odd box? *Discrete Mathematics*, 133(1-3):55–78, 1994.

[29] F.W. Barnes. Best packing of rods into boxes. *Discrete Mathematics*, 142(1-3):271–275, 1995.

[30] H. Bast and S. Hert. The area partitioning problem. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 163–171, 2000.

[31] C. Baur and S.P. Fekete. Approximation of geometric dispersion problems. *Algorthmica*, 30:450–470, 2001.

[32] O. Beaumont, V. Boudet, F. Rastello, and Y. Robert. Matrix-matrix multiplication on heterogeneous platforms. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1033–1051, 2001.

[33] O. Beaumont, V. Boudet, F. Rastello, and Y. Robert. Partitioning a square into rectangles: **NP**-completeness and approximation algorithms. *Algorithmica*, 34:217–239, 2002.

[34] P. Belleville. Computing two-covers of simple polygons. Master's thesis, Mcgill University, 1991.

[35] P. Belleville. On restricted boundary covers and convex three-covers. In *Proceedings of the 5th Canadian Conference on Computational Geometry*, pages 467–472, 1993.

[36] P. Belleville. A study of convex covers in two or more dimensions. Master's thesis, Simon Fraser University, 1995.

[37] A. Below, J.A. De Loera, and J. Richter-Gebert. Finding minimal triangulations of convex 3-polytopes is **NP**-hard. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 65–66, 2000.

[38] M. Benkert, J. Gudmundsson, C. Knauer, E. Moet, R. van Oostrum, and A. Wolff. A polynomial-time approximation algorithm for a geometric dispersion problem. *Lecture Notes in Computer Science*, 4112:166–175, 2006.

[39] P. Berman and B. DasGupta. Approximating rectilinear polygon cover problems. In *Proceedings of the 4th Canadian Conference on Computational Geometry*, pages 229–235, 1992.

[40] P. Berman and B. DasGupta. Complexities of efficient solutions of the rectilinear polygon cover problems. *Algorithmica*, 17(4):331–356, 1997.

[41] P. Berman, B. DasGupta, S. Muthukrishnan, and S. Ramaswami. Improved approximation algorithms for rectangle tiling and packing. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 427–436, 2001.

[42] M. Bern. Compatible tetrahedralizations. In *Proceedings of the 9th Annual Symposium on Computational Geometry*, pages 281–288, 1993.

[43] M. Bern. Triangulations and mesh generation. In J.E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 563–582. CRC Press, 2004.

[44] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In D-Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*. World Scientific, 1992.

[45] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In D.S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 296–345. PWS Publishing Company, 1997.

[46] S. Bespamyatnikh. Packing two disks in a polygon. *Computional Geometry: Theory and Applications*, 23(1):31–42, 2002.

[47] T.C. Biedl, E.D. Demaine, M.L. Demaine, A. Lubiw, and G.T. Toussaint. Hiding disks in folded polygons. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 36–37, 1998.

[48] R.P. Boland and J. Urrutia. Polygon area problems. In *Proceedings of the 12th Canadian Conference on Computational Geometry*, pages 159–162, 2000.

[49] R.P. Boland and J. Urrutia. Partitioning polygons into tree-monotone and y-monotone subpolygons. *Lecture Notes in Computer Science*, 2669:985–994, 2003.

[50] R. Boppana and M.M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. In *Proceedings of the 2nd Scandinavian Workshop on Algorithm Theory*, pages 13–25, 1990.

[51] P. Bose, J. Czyzowicz, E. Kranakis, D. Krizanc, and D. Lessard. Near optimal-partitioning of rectangles and prisms. In *Proceedings of the 11th Canadian Conference on Computational Geometry*, pages 162–165, 1999.

[52] P. Bose, J. Czyzowicz, E. Kranakis, D. Kriznac, and A. Maheshwari. Cutting circles and squares in equal area pieces. *Geombinatorica, XI-1*, 2001.

[53] P. Bose, J. Czyzowicz, E. Kranakis, and A. Maheshwari. Algorithms for packing two circles in a convex polygon. *Revised Papers from the Japanese Conference on Discrete and Computational Geometry*, 2098:93–103, 2000.

[54] P. Bose, J. Czyzowicz, and D. Lessard. Cutting rectangles in equal area pieces. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 94–95, 1998.

[55] P. Bose, E.D. Demaine, J. Iacono, and S. Langerman. Quartering a square optimally. In *Abstracts of the Japan Conference on Discrete and Computational Geometry*, 2002.

[56] P. Bose, P. Morin, and A. Vigneron. Packing two disks into a polygonal environment. *Journal of Discrete Algorithms*, 2(3):373–380, 2004.

[57] K. Brakke. Surface evolver. *Experimental Mathematics*, 1(2):141–165, 1999.

[58] A. Brandstädt. The jump number problem for biconvex graphs and rectangle covers of rectangular regions. In *Proceedings of the International Conference on Fundamentals of Computation Theory*, pages 68–77, 1989.

[59] D. Bremner and T.C. Shermer. Point visibility graphs and o-convex cover. *International Journal of Computational Geometry and Applications*, 10(1):55–71, 2000.

[60] S. Cabello. Approximation algorithms for spreading points. *Journal of Algorithms*, 62(2):49–73, 2007.

[61] P. Cappanera. A survey on obnoxious facility location problems. Technical report, University of Pisa, 1999.

[62] S. Carlsson, B.J. Nilsson, and S.C. Ntafos. Optimum guard covers and m-watchmen routes for restricted polygons. *Algorithms and Data Structures*, 519:367–378, 1991.

[63] L.G. Casado, T. Csendes, I. García, and P. G. Szabó. Lower bounds for equal circles packing in a square problem using the TAMSASS-PECS stochastic algorithm. In *Proceedings of the International Workshop on Global Optimization*, 1999.

[64] L.G. Casado, I. García, P.G. Szabó, and T. Csendes. Equal circles packing in a square II: New results for up to 100 circles using TAMSASS-PECS algorithm. *Optimization Theory: Recent Developments from Matrahaza*, 2001.

[65] L.G. Casado, P. G. Szabó, and I. García Fernandez. Packing up to 100 equal circles in a square. Technical report, Universidad de Almeria, 1999.

[66] S. Chaiken, D. Kleitman, M. Saks, and J. Shearer. Covering regions by rectangles. *SIAM Journal of Algebraic Discrete Methods*, 2(4):394–410, 1981.

[67] A. Chalcraft. Perfect square packings. *Journal of Combinatorial Theory Series A*, 92(2):158–172, 2000.

[68] T.M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms*, 46(2):178–189, 2003.

[69] T.M. Chan. A note on maximum independent sets in rectangle graphs. *Information Processing letters*, 89(1):19–23, 2004.

[70] T.M. Chan. Three problems about simple polygons. *Computational Geometry: Theory and Applications*, 35(3):209–217, 2006.

[71] B. Chandra and M.M. Halldursson. Approximation algorithms for dispersion problems. *Journal of Algorithms*, 38(2):438–465, 2001.

[72] R. Chandrasekaran and A. Daughety. Location on tree networks: P-centre and n-dispersion problems. *Mathematics of Operations Research*, 6:50–57, 1981.

[73] B. Chazelle. Convex decompositions of polyhedra. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 70–79, 1981.

[74] B. Chazelle. A theorem on polygon cutting with applications. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 339–349, 1982.

[75] B. Chazelle. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM Journal on Computing*, 13(3):488–507, 1984.

[76] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Computational Geometry*, 6(5):485–524, 1991.

[77] B. Chazelle and D. Dobkin. Decomposing a polygon into its convex parts. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, pages 38–48, 1979.

[78] B. Chazelle and L. Palios. Triangulating a non-convex polytype. In *Proceedings of the 5th Annual Symposium on Computational Geometry*, pages 393–400, 1989.

[79] D.Z. Chen, X. Hu, Y. Huang, Y. Li, and J. Xu. Algorithms for congruent sphere packing and applications. In *Proceedings of the 17th Annual Symposium on Computational Geometry*, pages 212–221, 2001.

[80] Y. Cheng, S.S. Iyengar, and R.L. Kashyap. A new method of image compression using irreducible covers of maximal rectangles. *IEEE Transactions on Software Engineering*, 14(5):651–658, 1988.

[81] F.Y.L. Chin, S.P.Y. Fung, and C.A. Wang. Approximation for minimum triangulation of convex polyhedra. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 128–137, 2001.

[82] M. Chlebík and J. Chlebíková. Approximation hardness of optimization problems in intersection graphs of d-dimensional boxes. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 267–276, 2005.

[83] V. Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, 18:39–41, 1975.

[84] V. Chvátal. *Linear Programming.* Freeman, 1983.

[85] E.G. Coffman, M.R. Garey, D.S. Johnson, and R.E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

[86] H.E. Conn and J. O'Rourke. Some restricted rectangles covering problems. Technical report, John Hopkins University, 1987.

[87] H.E. Conn and J. O'Rourke. Minimum weight quadrilaterilization in $O(n^3 \log n)$ time. In *Proceedings of the 28th Allerton Conference on Communication, Control and Computing*, pages 788–797, 1990.

[88] H.T. Croft, K.J. Falconer, and R.K. Guy. *Unsolved Problems in Geometry.* Springer-Verlag, 1991.

[89] J. Csirik and G.J. Woeginger. Shelf algorithms for online-strip packing. *Information Processing Letters*, 63(4):171–175, 1997.

[90] J.C. Culberson and R.A. Reckhow. Covering polygons is hard. *Journal of Algorithms*, 17(1):2–44, 1994.

[91] M. Damian and J. O'Rourke. Partitioning regular polygons into circular pieces II: Nonconvex partitions. Unpublished Draft.

[92] M. Damian and J. O'Rourke. Partitioning regular polygons into circular pieces I: Convex partitions. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 43–46, 2003.

[93] M. Damian-Iordache. Exact and approximation algorithms for computing α-fat decompositions. In *Proceedings of the 12th Canadian Conference on Computational Geometry*, pages 93–96, 2002.

[94] M. Damian-Iordache and S.V. Pemmaraju. Computing optimal α-fat and α-small decompositions. In *Proceedings of the 12th Annual Symposium on Discrete Algorithms*, pages 338–339, 2001.

[95] K. Daniels and V. Milenkovic. Multiple translational containment. part I: An approximate algorithm. *Algorithmica*, 19(1-2):148–182, 1997.

[96] W. Fernandez de la Vega and V. Zissimopoulos. An approximation scheme for strip packing of rectangles with bounded dimensions. *Discrete Applied Mathematics*, 82(1-3):93–101, 1998.

[97] E.D. Demaine, J.S.B. Mitchell, and J. O'Rourke. The Open Problems Project. http://maven.smith.edu/ orourke/TOPP/.

[98] V.J. Dielissen and A. Kaldewaij. Rectangular partition is polynomial in two dimensions but **NP**-complete in three. *Informaion Processing Letters*, 38(1):1–6, 1991.

[99] D. Dobkin, D.L. Souvaine, and C.J. van Wyk. Decomposition and intersection of simple polygons. *Algorithmica*, 3(1):473–485, 1988.

[100] K.A. Dowsland. An exact algorithm for the pallet loading problem. *European Journal of Operational Research*, 31:78–84, 1987.

[101] P. Eades. Symmetry finding algorithms. In G.T. Toussaint, editor, *Computational Morphology*, pages 41–51. North Holland, 1988.

[102] H. Edelsbrunner, J. O'Rourke, and E. Welzl. Stationing guards in rectilinear art galleries. *Computer Vision, Graphics and Image Processing*, 28(2):167–176, 1984.

[103] F. Eisenbrand, S. Funke, A. Karrenbauer, J. Reichel, and E. Schömer. Packing a trunk: now with a twist! In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, pages 197–206, 2005.

[104] F. Eisenbrand, S. Funke, J. Reichel, and E. Schömer. Packing a trunk. In *Proceedings of the 11th Annual European Symposium on Algorithms*, pages 618–629, 2003.

[105] D. El-Khechen and T. Fevens. Minimal perimeter cuttings of squares and rectangles into equal area pieces. In *Abstracts of the Japan Conference on Discrete and Computational Geometry*, 2004.

[106] D. El-Khechen, T. Fevens, J. Iacono, and G. Rote. Partitioning a polygon into two congruent pieces. In *Abstracts of the Kyoto International Conference on Computational Geometry and Graph Theory*, 2007.

[107] D. El-Khechen, T. Fevens, J. Iacono, and G. Rote. Partitioning a polygon into two mirror congruent pieces. In *Proceedings of the 20th Canadian Conference on Computational Geometry*, pages 131–134, 2008.

[108] L. Epstein and R. van Stee. This side up! *ACM Transaction on Algorithms*, 2(2):228–243, 2006.

[109] P. Erdös and R.L. Graham. On packing squares with equal squares. *Journal of Combinatorial Theory Series A*, 19:119–123, 1975.

[110] E. Erhan. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990.

[111] K. Erikson. Splitting a polygon into two congruent pieces. *The American Mathematical Monthly*, 103(5):393–400, 1996.

[112] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric graphs. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 671–679, 2001.

[113] H. Everett, W. Lenhart, M. Overmars, T.C. Shermer, and J. Urrutia. Strictly convex quadrilateralizations of polygons. In *Proceedings of the 4th Canadian Conference on Computational Geometry*, pages 77–83, 1992.

[114] H. Exeler. Das homogene packproblem in der betriebswirtschaftlichen praxis. *European Journal of Operational Research*, 1988.

[115] M. Farach and S. Muthukrishnan. Perfect hashing for strings: formalization and algorithms. *Combinatorial Pattern Matching*, 1075:130–140, 1996.

[116] H. Y. F. Feng and T. Pavlidis. Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition. *IEEE Transactions on Computing*, 24(6):636–650, 1975.

[117] L. Ferrari, P.V. Sankar, and J. Sklansky. Minimal rectangular partitions of digitized blobs. *Computer Vision, Graphics and Image Processing*, 28:58–71, 1984.

[118] A.V. Fishkin, O. Gerber, K. Jansen, and R. Solis-Oba. On packing squares with resource augmentation: maximizing the profit. In *Proceedings of the 2005 Australasian Symposium on Theory of Computing*, pages 61–67, 2005.

[119] F. Fodor. Densest packing of nineteen congruent circles in a circle. *Geometriae Dedicata*, 74(2):139–145, 2004.

[120] A. Fournier and D.Y. Montuno. Triangulating simple polygons and equivalent problems. *ACM Transactions on Graphics*, 3(2):153–174, 1984.

[121] R. Fowler, M. Paterson, and S. Tanimoto. Optimal packing and covering in the plane are **NP**-complete. *Information Processing Letters*, 12:133–137, 1981.

[122] D.S. Franzblau. Performance guarantees on a sweep-line heuristic for covering rectilinear polygons with rectangles. *SIAM Journal on Discrete Mathematics*, 2(3):307–321, 1989.

[123] D.S. Franzblau and D.J. Kleitman. An algorithm for covering polygons with rectangles. *Information Control*, 63(3):164–189, 1986.

[124] E. Friedman. Erich friedman packing center. http://www.stetson.edu/ efriedma/packing.html.

[125] E. Friedman. Packing unit squares in squares: A survey and new results. *The Electronic Journal of Combinatorics*, 7, 2002.

[126] S.P.Y. Fung, F.Y.L. Chin, and C. Keung Poon. Approximating the minimum triangulation of convex 3-polytopes with bounded degrees. *Computational Geometry Theory and Applications*, 32(1):1–12, 2005.

[127] Z. Füredi. The densest packing of equal circles into a parallel strip. *Discrete Computational Geometry*, 6(2):95–106, 1991.

[128] M.R. Garey, D.S. Johnson, F.P. Preparata, and R.E. Tarjan. Triangulating a simple polygon. *Information Processing Letters*, 7:175–179, 1978.

[129] S. Gerdjikov and A. Wolff. Pseudo-convex decomposition of simple polygons. In *Abstracts of the 22nd European Workshop on Computational Geometry*, pages 13–16, 2006.

[130] L. Gewali, M. Keil, and S. Ntafos. On covering orthogonal polygons with star-shaped polygons. *Information Sciences*, 65(1–2):45–63, 1992.

[131] L. Gewali and Ntafos. Minimum covers for grids and orthogonal polygons by periscope guards. In *Proceedings of the 2nd Canadian Conference on Computational Geometry*, pages 358–361, 1990.

[132] S.K. Ghosh. Approximation algorithms for art gallery problems. In *Proceedings of Canadian Information Processing Society Congress*, 1987.

[133] I. Golan. Performance bounds for orthogonal, oriented two-dimensional packing algirithms. *SIAM journal on Computing*, 10:571–582, 1981.

[134] T. Gonzalez, M. Razzazi, M-T. Shing, and S-Q. Zheng. On optimal guillotine partitions approximating d-box partitions. *Computational Geometry: Theory and Applications*, 4(1):1–12, 1994.

[135] T. Gonzalez, M. Razzazi, and S-Q. Zheng. An efficient divide-and-conquer approximation algorithm for partitioning into d-boxes. *International Journal of Computational Geometry and Applications*, 3(4):417–428, 1993.

[136] T. Gonzalez and S-Q. Zheng. Bounds for partitioning rectilinear polygons. In *Proceedings of the 1st Annual Symposium on Computational Geometry*, pages 281–287, 1985.

[137] T. Gonzalez and S-Q. Zheng. Approximation algorithms for partitioning a rectangle with interior points. *Algorithmica*, 5(1):11–42, 1990.

[138] K.D. Gourley and D.M. Green. A polygon-to-rectangle conversion algorithm. *IEEE Computer Graphics*, 3(1):31–36, 1983.

[139] R.L. Graham and B.D. Lubachevsky. Repeated patterns of dense packings of equal disks in a square. *The Electronic Journal of Combinatorics*, 3, 1996.

[140] R.L. Graham, B.D. Lubachevsky, K.J. Nurmela, and P.R.J. Ostergaard. Dense packings of congruent circles in a circle. *Discrete Mathematics*, 181(1-3):139–154, 1998.

[141] D.H. Green. The decomposition of polygons into convex parts. *Advances in Computing Research*, 1:235–259, 1983.

[142] B. Grünbaum and G.C. Shephard. *Tiling and Patterns.* Freeman, 1987.

[143] R. Guardia and F. Hurtado. On the equipartitions of convex bodies and convex polygons. In *Proceedings of the 16th European Workshop on Computational Geometry*, 2000.

[144] J. Gudmundsson, T. Husfeldt, and C. Levcopoulos. Lower bounds for approximate polygon decomposition and minimum gap. *Information Processing Letters*, 81(3):137–141, 2002.

[145] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear time algorithms for visibility and shortest path problems inside simple polygons. *Algorithmica*, 2(1-4):1–13, 1987.

[146] O. Gunther. Minimum k-partitioning of rectilinear polygons. *Journal of Symbolic Computation*, 9(4):457–483, 1990.

[147] E. Györi, F. Hoffmann, K. Kriegel, and T.C. Shermer. Generalized guarding and partitioning for rectilinear polygons. *Computatinal Geometry: Theory and Applications*, 6(1):21–44, 1996.

[148] T.C. Hales. The Honeycomb Conjecture. *Discrete Computational Geometry*, 25:1–22, 2001.

[149] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, pages 105–142, 1999.

[150] A. Hegedus. Algorithms for covering polygons by rectangles. *Computer Aided Design*, 14:257–260, 1982.

[151] L. Heinrich-Litan and M.E. Lübbecke. Rectangle covers revisited computationally. *Journal of Experimental Algorithmics*, 11:1–21, 2006.

[152] J. Hershberger and J. Snoeyink. Convex polygons made from few lines and convex decompositions of polyhedra. In *Proceedings of the 3rd Scandinavian Workshop on Algorithm Theory*, pages 376–387, 1992.

[153] S. Hert and V.J. Lumelsky. Polygon area decomposition for multiple-robot workspace division. *International Journal of Computional Geometry and Application*, 8(4):437–466, 1998.

[154] S. Hertel and K. Mehlhorn. Fast triangulation of the plane with respect to simple polygons. *Information and Control*, 64(1-3):52–76, 1985.

[155] D.S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 35(1):130–136, 1985.

[156] J. Hoffman. http://www.msri.org/publications/sgp/jim/geom/cmc/library/cube/ mainc.htm.

[157] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, and R.E. Stearns. NC-approximation schemes for **NP**- and **PSPACE**-hard problems for geometric graphs. *Journal of Algorithms*, 26(2):238–274, 1998.

[158] H. Imai and T. Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4(4):310–323, 1983.

[159] H. Imai and T. Asano. Efficient algorithms for geometric graph search problems. *SIAM Journal on Computing*, 15(2):478–494, 1986.

[160] K. Jansen and R. Solis-Oba. An asymptotic approximation algorithm for 3D-strip packing. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 143–152, 2006.

[161] K. Jansen and R. van Stee. On strip packing with rotations. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 755–761, 2005.

[162] K. Jansen and G. Zhang. On rectangle packing: maximizing benefits. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 204–213, 2004.

[163] J. Januszewaki. Packing similar triangles into a triangle. *Periodica Mathematica Hungarica*, 46(1):61–65, 2003.

[164] D. Jennings. On packing unequal rectangles in the unit square. *Journal of Combinatorial Theory Series A*, 68(2):465–469, 1994.

[165] D. Jennings. On packings of squares and rectangles. *Discrete Mathematics*, 138(1-3):293–300, 1995.

[166] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computing and Systems Sciences*, 9(3):256–278, 1974.

[167] J. Kahn, M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM Journal of Algebraic and Discrete Methods*, 4(2):194–206, 1983.

[168] J.M. Keil. *Decomposing a polygon into Simpler Components.* PhD thesis, University of Toronto, 1983.

[169] J.M. Keil. Decomposing a polygon into simpler components. *SIAM Journal on Computing*, 14(4):799–817, 1985.

[170] J.M. Keil. Minimally covering a horizontally convex orthogonal polygon. In *Proceedings of the 2nd Annual Symposium on Computational Geometry*, pages 43–51, 1986.

[171] J.M. Keil. Covering orthogonal polygons with non-piercing rectangles. *International Journal of Computational Geometry and Applications*, 7(5):473–484, 1997.

[172] J.M. Keil. *Polygon Decomposition*, chapter 11. Elsevier Science B. V, 2000.

[173] J.M. Keil and J.R. Sack. Minimum decompositions of polygonal objects. In Godfried Toussaint, editor, *Computational Morphology*, pages 197–216. North-Holland, 1988.

[174] J.M. Keil and J. Snoeyink. On the time bound for convex decomposition of simple polygons. *International Journal on Computational Geometry and Applications*, 12(3):181–192, 2002.

[175] C. Kenyon and E. Rémila. Approximate strip packing. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 31–36, 1996.

[176] C. Kenyon and E. Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operation Research*, 25(4):645–656, 2000.

[177] S. Khanna, S. Muthukrishnan, and M. Paterson. On approximating rectangle tiling and packing. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, 1998.

[178] S.K. Kim, C-S. Shin, and T-C. Yang. Placing two disks in a convex polygon. *Information Processing Letters*, 73:33–39, 2000.

[179] T. Y. Kong, D.M. Mount, and M. Werman. The decomposition of a square into rectangles of minimal perimeter. *Discrete Applied Mathematics*, 16(3):239–243, 1987.

[180] T.Y. Kong, D.M. Mount, and A.W. Roscoe. The decomposition of a rectangle into rectangles of minimal perimeter. *SIAM Journal on Computing*, 17(6):1215–1231, 1988.

[181] E. Koutsoupias, C.H. Papadimitriou, and M. Sideri. On the optimal bisection of a polygon (extended abstract). In *Proceedings of the 6th Annual Symposium on Computational Geometry*, pages 198–202, 1990.

[182] V.S. Anil Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 445–454, 1999.

[183] H.C. Lee and T.C. Woo. Determining in linear time the minimum area convex hull of two polygons. *IIE Transactions*, 20(4):338–345, 1988.

[184] J.Y.-T. Leung, T.W. Tam, C.S. Wong, G.H. Young, and F.Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.

[185] C. Levcopoulos. A fast heuristic for covering polygons by rectangles. *Lecture notes in Computer Science*, 199:269–278, 1985.

[186] C. Levcopoulos. Fast heuristics for minimum length rectangular partitions of polygons. In *Proceedings of the 2nd Annual Symposium on Computational Geometry*, pages 100–108, 1986.

[187] C. Levcopoulos. Minimum length and thickest-first rectangular partitions of polygons. Technical report, Department of Computer and Information Science, Linkoping University, 1986.

[188] C. Levcopoulos. Improved bounds for covering general polygons with rectangles. *Lecture Notes in Computer Science*, 287:95–102, 1987.

[189] C. Levcopoulos and J. Gudmundsson. Approximation algorithms for covering polygons with squares and similar problems. In *Proceedings of the International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 27–41, 1997.

[190] C. Levcopoulos and J. Gudmundsson. Close approximation of minimum rectangular coverings. *Journal of Combinatorial Optimization*, 3:437–452, 1999.

[191] C. Levcopoulos and A. Lingas. Bounds on the length of convex partitions of polygons. In *Proceedings of the 4th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 279–295, 1984.

[192] C. Levcopoulos and A. Lingas. Covering polygons with the minimum number of rectangles. *Lecture Notes in Computer Science*, 166:63–72, 1984.

[193] C. Levcopoulos, A. Lingas, and J-R. Sack. Algorithms for minimum length partitions of polygons. *BIT*, 27:474–479, 1987.

[194] K. Li and K-H. Cheng. On three-dimensional packing. *SIAM Journal on Computing*, 19(5):847–867, 1990.

[195] K. Li and K-H. Cheng. Heuristic algorithm for online packing in three dimensions. *Journal of Algorithms*, 13(4):589–605, 1992.

[196] J-M. Lien and N.M. Amato. Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, pages 121–131, 2007.

[197] A. Lingas. The power of non-rectilinear holes. In *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, pages 369–383, 1982.

[198] A. Lingas. Heuristics for minimum edge length rectangular partitions of rectilinear figures. In *Proceedings of the 9th International Colloquium Automata, Languages and Programming*, pages 369–383, 1983.

[199] A. Lingas, R. Pinter, R. Rivest, and A. Shamir. Minimum edge length partitioning of rectilinear polygons. In *Proceedings of the 20th Allerton Conference on Communication, Control and Computation*, pages 53–63, 1982.

[200] A. Lingas and V. Soltan. Minimum convex partition of a polygon with holes by cuts in given directions. In *Proceedings of the 7th International Symposium on Algorithms and Computation*, pages 315–325, 1996.

[201] A. Lingas, A. Wasylewicz, and P. Żyliński. Note on covering monotone orthogonal polygons with star-shaped polygons. *Information Procesing Letters*, 104(6):220–227, 2007.

[202] A. Lingas, A. Wasylewicz, and P. Zylinski. Linear-time 3-approximation algorithm for the r-star covering problem. *Lecture Notes in Computer Science*, 4921:157–168, 2008.

[203] 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 5th Annual Symposium on Computational Geometry*, pages 344 – 353, 1989.

[204] W.T. Liou, J.J. Tan, and R. C.T. Lee. Covering convex rectilinear polygons in linear time. *International Journal of Computational geometry and applications*, 1(2):137 – 185, 1991.

[205] W. Lipski. Finding a manhattan path and related problems. *Networks*, 13:399–409, 1983.

[206] W. Lipski. An $O(n \log n)$ manhattan path algorithm. *Information Processing Letters*, 19:99–102, 1984.

[207] W. Lipski, E. Lodi, F. Luccio, C. Mugnai, and L. Pagli. On two-dimensional data organization II. *Fundamentae Informaticae*, 2:245–260, 1979.

[208] R. Liu and S. Ntafos. On decomposing a polygon into uniformly monotone parts. *Information Processing Letters*, 27:85–89, 1988.

[209] R. Liu and S. Ntafos. On partitioning rectilinear polygons into star-shaped polygons. *Algorithmica*, 6(1):771–800, 1991.

[210] M. Locatelli and U. Raber. Packing equal circles in a square: I. solution properties. Unpublished manuscript, 1998.

[211] M.A. Lopez and D.P. Mehta. Efficient decomposition of polygons into l-shapes with application to VLSI layouts. *ACM Transactions Design Automation of Electronic Systems*, 1(3):371–395, 1996.

[212] B.D. Lubachevsky and R.L. Graham. Dense packings of equal disks in an equilaterial triangle: From 22 to 34 and beyond. *The Electronic Journal of Combinatorics*, 2, 1995.

[213] B.D. Lubachevsky, R.L. Graham, and F.H. Stillinger. Pattern and structures in disk packings. *Periodica Mathematica Hungarica*, 34(1-2):123–142, 1997.

[214] A. Lubiw. Decomposing polygonal regions into convex quadrilaterals. In *Proceedings of the 1st Annual Symposium on Computational Geometry*, pages 97–106, 1985.

[215] A. Lubiw. The boolean basis problem and how to cover some polygons by rectangles. *SIAM Journal on Discrete Mathematics*, 3(1):98–115, 1990.

[216] F. Maire. Polyominos and perfect graphs. *Information Processing Letters*, 50(2):57–61, 1994.

[217] C.D. Maranas, C.A. Floudas, and P.M. Pardalos. New results in the packing of equal circles in a square. *Discrete Mathematics*, 142(1-3):287–293, 1995.

[218] H. Martini and V. Soltan. Minimum convex partition of a polygon by guillotine cuts. *Discrete and Computational Geometry*, pages 291–305, 1998.

[219] W.J. Masek. Some **NP**-complete set covering problems. Manuscript, 1979.

[220] A. Meir and L. Moser. On packing squares and cubes. *Journal of Combinatorial Theory*, 5:126–134, 1968.

[221] J.B.M Melissen. Densest packing of congruent circles in an equilateral triangle. *The American Mathematical Monthly*, 100(10):916–925, 1993.

[222] J.B.M. Melissen. Densest packing of eleven congruent circles in a circle. *Geometriae Dedicata*, 50(1):15–25, 1994.

[223] J.B.M. Melissen and P.C. Schuur. Packing 16, 17 or 18 circles in an equilateral triangle. *Discrete Mathematics*, 145(1-3):333–342, 1995.

[224] V. Milenkovic. Multiple translational containment. part II: Exact algorithms. *Algorithmica*, 19(1-2):183–218, 1997.

[225] V. Milenkovic. Rotational polygon overlap minimization. In *Proceedings of the 13th Annual Symposium on Computational Geometry*, pages 334–343, 1997.

[226] V. Milenkovic. Rotational polygon containment and minimum enclosure. In *Proceedings of the 14th Annual Symposium on Computational Geometry*, pages 1–8, 1998.

[227] V. Milenkovic. Rotational polygon overlap minimization and compaction. *Computational Geometry: Theory and Applications*, 10(4):305–318, 1998.

[228] V. Milenkovic. Rotational polygon containment and minimum enclosure using only robust 2D constructions. *Computational Geometry: Theory and Applications*, 13(1):3–19, 1999.

[229] F.K. Miyazawa and Y. Wakabayashi. An algorithm for the three-dimensional packing problem with asymptotic performance analysis. *Algorithmica*, 18(1):122–144, 1997.

[230] F.K. Miyazawa and Y. Wakabayashi. Packing problems with orthogonal rotations. *Lecture Notes in Computer Science*, 2976:359–368, 2004.

[231] F.K. Miyazawa and Y. Wakabayashi. Two- and three-dimensional parametric packing. *Computers Operations Research*, 34(9):2589–2603, 2007.

[232] D. Moirtra. Finding a minimal cover for binary images: An optimal parallel algorithm. *Algorithmica*, 6(1–6):624–657, 1991.

[233] J. Mookherje and N. Prabhakaran. Spatial decomposition of a tumor into a minimum number of spherical components. In *Proceedings of the 1992 ACM-SIGAPP Symposium on Applied Computing*, pages 988–992, 1992.

[234] J.W. Moon and L. Moser. Some packing and covering theorems. *Colloquium Mathematicum*, 17:103–110, 1967.

[235] R. Motwani. Lecture notes on approximation algorithms- volume I. Book in preparation.

[236] R. Motwani, A. Raghunathan, and H. Saran. Covering orthogonal polygons with star polygons: the perfect graph approach. In *Proceedings of the 4th Annual Symposium on Computational Geometry*, pages 211–223, 1988.

[237] R. Motwani, A. Raghunathan, and H. Saran. Perfect graphs and orthogonally convex covers. *SIAM Journal Discrete Mathematics*, 2(3):371–392, 1989.

[238] M. Müller-Hannemann and K. Weihe. Minimum strictly convex quadrangulations of convex polygons. In *Proceedings of the 13th Annual Symposium on Computational Geometry*, pages 193–202, 1997.

[239] K. Mulmuley. A fast planar partition algorithm, I. *Journal of Symbolic Computation*, 10:253–280, 1990.

[240] H. Nagamochi and Y. Abe. An approximation algorithm for dissecting a rectangle into rectangles with specified areas. *Discrete Applied Mathematics*, 155(4):523–537, 2007.

[241] S. Nahar and S.K. Sahni. Fast algorithm for polygon decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 7(4):473–483, 1988.

[242] J. Neliben. New approaches to the pallet loading problem. Technical report, RWTH Aachen, Lehrstuhl fur Angewandte Mathematik, 1993.

[243] T. Ohtsuki. Minimum dissection of rectilinear regions. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 1210–1213, 1982.

[244] J. O'Rourke, I. Pashchenko, and G. Tewari. Partitioning orthogonal polygons into fat rectangles. In *Proceedings of the 13th Canadian Conference on Computational Geometry*, pages 133–136, 2001.

[245] J. O'Rourke and K. Supowit. Some **NP**-hard decomposition problems. *IEEE Transactions on Information Theory*, 29(2):181–190, 1983.

[246] J. O'Rourke and G. Tewari. The structure of optimal partitions of orthogonal polygons into fat rectangles. *Computational Geometry: Theory and Applications*, 28(1):49–71, 2004.

[247] M.M. Paulhus. An algorithm for packing squares. *Journal of Combinatorial Theory Series A*, 82(2):147–157, 1998.

[248] B. Ram. The pallet loading problem: A survey. *International Journal of Production Economics*, 28(2):217–225, 1992.

[249] S.S. Ravi, D.J. Rosenkrantz, and G.K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

[250] R.A. Reckhow and J. Culberson. Covering a simple orthogonal polygon with a minimum number of orthogonally convex polygons. In *Proceedings of the 3rd Annual Symposium on Computational Geometry*, pages 268–277, 1987.

[251] T. Richardson. Optimal packing of similar triangles. *Journal of Combinatorial Theory Series A*, 69(2):288–300, 1995.

[252] G. Rote. Some thoughts about decomposing a polygon into two congruent pieces, 1997. Unpublished Draft, `page.mi.fu-berlin.de/~rote/Papers/postscript/Decomposition+of+a+polytope+into+two+congruent+pieces.ps`.

[253] K.F. Roth and R.C. Vaughan. Inefficiency in packing squares with unit squares. *Journal of Combinatorial Theory Series A*, 24:170–186, 1978.

[254] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete and Computational Geometry*, 7(3):227–253, 1992.

[255] J.R. Sack. An $O(n \log n)$ algorithm for decomposing rectilinear polygons into convex quadriltaerals. In *Proceedings of the 20th Allerton Conference on Communication, Control and Computing*, pages 21–30, 1982.

[256] J.R. Sack and G.T. Toussaint. A linear-time algorithm for decomposing rectilinear polygons into convex quadriltaerals. In *Proceedings of the 19th Allerton Conference on Communication, Control and Computing*, pages 21–30, 1981.

[257] J.R. Sack and G.T. Toussaint. Guard placement in rectilinear polygons. In G.T. Toussaint, editor, *Computational Morphology*, pages 153–175. North-Holland, 1988.

[258] B. Schachter. Decomposition of polygons into convex sets. *IEEE Transactions on Computing*, 27(11):1078–1082, 1978.

[259] I. Schiermeyer. Reverse-fit: a 2-optimal algorithm for packing rectangles. *Lecture Notes in Computer Science*, 855:290–299, 1994.

[260] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry: Theory and Applications*, 1(1):51–64, 1991.

[261] M.I. Shamos. *Computational Geometry.* PhD thesis, Yale University, 1978.

[262] M.I. Shamos, 2004. Personal Communication.

[263] M.I. Shamos and F.P. Preparata. *Computational Geometry: An introduction.* Springer, 1985.

[264] M. Shapira and A. Rappoport. Shape blending using the star-skeleton representation. Manuscript, 1994.

[265] T.C. Shermer. A linear time algorithm for bisecting a polygon. *Information Processing Letters*, 41:135–140, 1992.

[266] T.C. Shermer. On recognizing unions of two convex polygons and related problems. *Pattern Recognition Letters*, 14(9):737–745, 1993.

[267] H. Simon, 2004. Personal Communication.

[268] D.D. Sleator. A 2.5 optimal algorithm for packing in two dimensions. *Information Processing Letters*, 10(1):37–40, 1980.

[269] A. Soifer. Packing triangles in triangles. *Geombinatorics*, 8(4):110–115, 1999.

[270] V. Soltan and A. Gorpinevich. Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles. *Discrete Computational Geometry*, 9(1):57–79, 1993.

[271] A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997.

[272] P.G. Szabó, T. Csendes, L.G. Casado, and I. García. Equal circles packing in a square I- problem setting and bound for optimal solution. *Optimization Theory: Recent Developments from Matrahaza*, pages 191–206, 2001.

[273] M. Tanase and R.C. Veltkamp. Polygon decomposition based on the straight line skeleton. In *Proceedings of the 19th Annual Symposium on Computational Geometry*, pages 58–67, 2003.

[274] K. Tang, C. C.L. Wang, and D.Z. Chen. Minimum area convex packing of two convex polygons. *International Journal of Computional Geometry and Applications*, 16(1):41–74, 2006.

[275] A.G. Tarnowsky. Exact polynomial algorithm for special case of the two-dimensional cutting stock problem: A guillotine pallet loading problem. Technical report, Belarusian State University,, 1992.

[276] G.F. Tóth. *Packing and covering*, pages 19–41. CRC Press, 1997.

[277] G.T. Toussaint. Quadrangulations of planar sets. In *Proceedings of the 4th International Workshop on Algorithms and Data Structures*, pages 218–227, 1995.

[278] M. van Kreveld. On fat partitioning, fat covering and the union size of polygons. *Computational Geometry: Theory and Applications*, 9(4):197–210, 1998.

[279] M. van Kreveld and I. Reinbacher. Good news partitioning a simple polygon by compass directions. In *Proceedings of the 19th Annual Symposium on Computational Geometry*, pages 78–87, 2003.

[280] C.A. Wang, B.-T Yang, and B. Zhu. On some polyhedra covering problems. *Journal of Combinatorial optimization*, 4(4):437–447, 2000.

[281] C. Worman. Decomposing polygons into r-star or α-boundable subpolygons. Master's thesis, University of Saskatchewan, 2004.

[282] C. Worman and J.M. Keil. Polygon decomposition and the orthogonal art gallery problem. *International Journal of Computational Geometry and Applications*, 17(2):105–138, 2007.

[283] H. Yamazaki, K. Sakanushi, and Y. Kajitani. Optimum packing of convex polygons by a new data structure sequence-table. In *Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems*, pages 821–824, 2000.