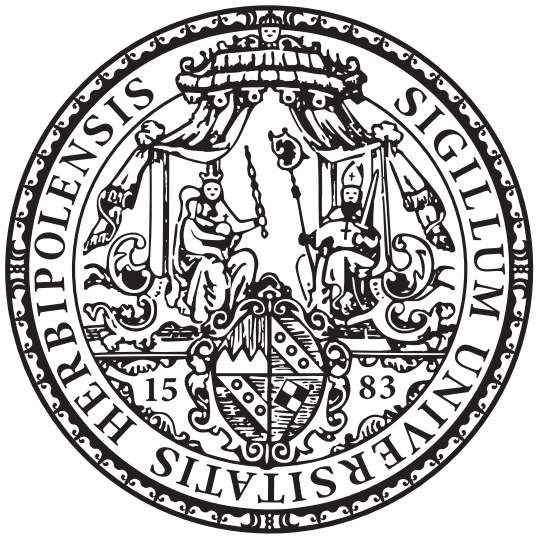
Master Thesis

On the Approximability of Orthogonal Compaction

Dominik Jilg

Date of Submission:	December 08, 2025
Advisors:	Prof. Dr. Alexander Wolff
	Prof. Dr. Marie Schmidt
	Samuel Wolf, M. Sc.



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen und Komplexität

Abstract

Many graph drawing applications rely on orthogonal layouts, in which edges consist only of horizontal and vertical segments. An orthogonal representation captures the shape of such a layout in purely combinatorial form by specifying the sequence of turns along each edge without fixing concrete coordinates. The orthogonal compaction (OC) problem then asks for a placement of all vertices and bends on a grid that realizes this representation, i.e. preserves all directions and turn orders, while minimizing the overall area of the resulting drawing.

While OC is NP-hard, even for cycles, neither APX-hardness nor constant-factor approximation algorithms are known. We study the limits of approximability for OC. We propose a family of graph instances, called snails, and show that classical heuristics for OC can perform arbitrarily poorly. Our results highlight the key role of kitty corners, which are reflex corners that point toward each other. These corners form the main source of complexity and create fundamental difficulties for designing local or greedy approximation strategies.

Building on the NP-hardness reduction from SAT by Patrignani [Pat01], we explore modifications toward a potential APX-hardness proof and identify structural difficulties. We also discuss limitations in designing approximation algorithms. We generalize OC to three dimensions and discuss how additional degrees of freedom affect approximability and potential hardness, providing a foundation for future research.

Zusammenfassung

Viele Graphanwendungen nutzen orthogonale Layouts, bei denen Kanten nur aus horizontalen und vertikalen Segmenten bestehen. Eine orthogonale Repräsentation beschreibt die Form eines solchen Layouts auf rein kombinatorischer Ebene, indem sie für jede Kante die Abfolge der Richtungsänderungen vorgibt, ohne konkrete Koordinaten festzulegen. Das orthogonale Kompaktierungsproblem (OC) besteht darin, alle Knoten und Biegungen auf einem Gitter so zu platzieren, dass diese Repräsentation respektiert wird, d.h. alle Richtungen und Biegefolgen erhalten bleiben, während die Gesamtfläche der Zeichnung minimiert wird.

Obwohl OC NP-schwer ist, selbst für einfache Graphen wie Kreise, sind weder APX-Härte noch Konstantfaktor-Approximationsalgorithmen bekannt. In dieser Arbeit untersuchen wir die Grenzen der Approximierbarkeit von OC. Dazu stellen wir eine Familie von Graphinstanzen, die sogenannten Snails, vor und zeigen, dass klassische Heuristiken für OC beliebig schlecht performen können. Unsere Ergebnisse heben die zentrale Rolle der Kitty Corners hervor, bei denen es sich um Reflexknoten handelt, die aufeinander zeigen. Diese Reflexknoten sind hauptverantwortlich für die Komplexität und verursachen grundlegende Schwierigkeiten bei der Entwicklung lokaler oder gieriger Approximationsstrategien.

Aufbauend auf der NP-Schwere-Reduktion von SAT untersuchen wir Modifikationen in Richtung eines potenziellen APX-Härtebeweises und identifizieren strukturelle Hindernisse. Außerdem diskutieren wir Limitierungen bei der Entwicklung von Approximationsalgorithmen. Wir verallgemeinern OC auf drei Dimensionen und erläutern, wie ein zusätzlicher Freiheitsgrad die Approximierbarkeit und potenzielle Härte beeinflusst, und schaffen damit eine Grundlage für zukünftige Forschungen.

Contents

1 Introduction

Graphs are one of the most fundamental structures for representing relationships between entities. From networks and transport systems to software architectures and electrical circuits, graphs provide a powerful way to model complex systems in a compact and readable form. The study of *graph drawing* is therefore concerned with finding visually effective representations of graphs that make their structure, connectivity, and hierarchy easy to understand. Readable graph layouts are essential not only for human comprehension but also because they enable efficient design, optimization, and maintenance in applications where space, time, or costs are limited, such as circuit design or infrastructure planning. Three examples of such layouts are shown in Figure 1.1.

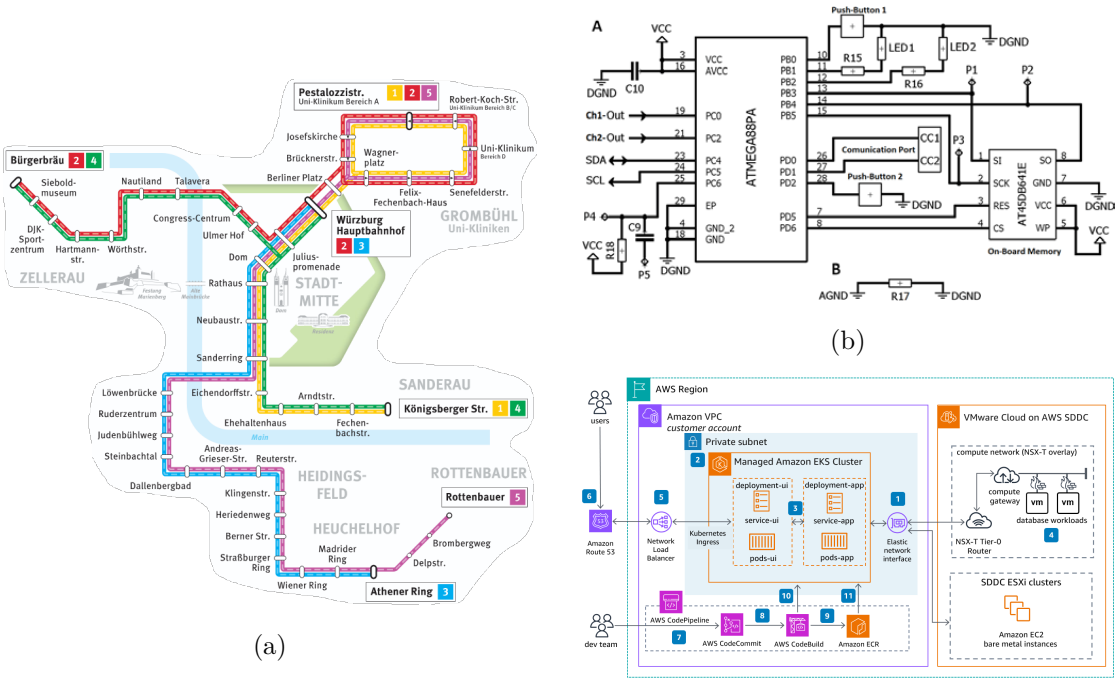


Fig. 1.1: (a) The tram network of Würzburg as an example of an infrastructure or transport system [WVV23]. (b) An electrical circuit diagram [PKB+19]. (c) A software architecture diagram [Ama24].

Among the many styles of graph drawing, *orthogonal graph drawings* play a particularly important role. In this type of graph drawings, edges are represented by a sequence of alternating horizontal and vertical segments, often placed on a grid. This drawing

convention is common in applications that require clear and organized visualizations. For example, orthogonal drawings are widely applied in software engineering, where they appear in UML and entity-relationship diagrams to highlight logical relationships between components while keeping layouts tidy and easy to follow. They are also common in urban planning and the design of metro maps or communication networks, where the regular structure helps to make navigation and topology more intuitive. Similarly, in VLSI (Very-Large-Scale Integration), a method for integrating a large number of transistors on a single semiconductor chip to create complex integrated circuits, orthogonal layouts are widely used. They help reduce wire length, minimize interference, and make efficient use of chip area by representing electrical connections with horizontal and vertical segments. For instance, minimizing the layout area directly impacts production costs, since the size of a chip determines yield and performance. As a natural consequence, algorithms that produce compact orthogonal drawings are highly desirable since they optimize resource utilization, improve usability, readability and enable scalable visualization of large and complex systems.

The computational problem underlying this task is known as the *orthogonal compaction* (OC) problem. In this problem, a given *orthogonal representation* fixes the directions of all edges and the relative positions of bends along them, but not the exact coordinates of vertices or bends. The goal is then to assign coordinates on a grid such that the drawing occupies minimal area while preserving the prescribed structure. In other words, orthogonal compaction seeks the most space efficient realization of a given layout without changing its topology or shape constraints; see Figure 1.2.

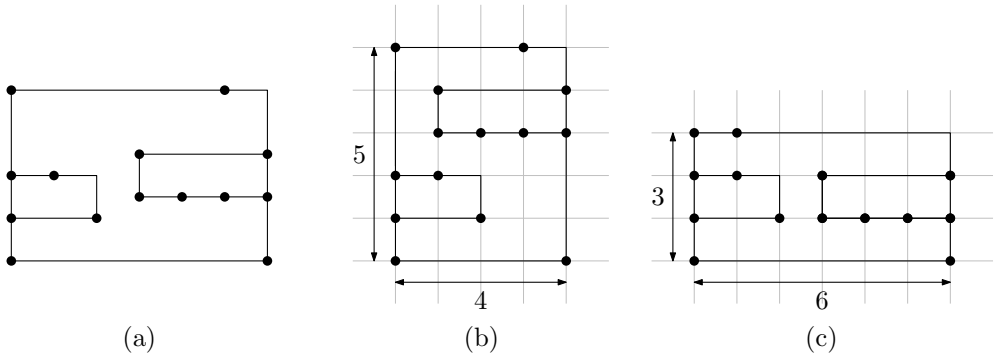


Fig. 1.2: (a) Orthogonal representation fixing edge directions and relative positions of vertices and bends. (b)-(c) Drawings of (a) occupying areas of 20 and 18 grid units, respectively.

OC is an optimization problem, whose computational complexity remains largely unresolved. While it is known that OC is NP-hard, neither its APX-hardness nor the existence of efficient approximation algorithms has been established so far. In practice, OC is typically addressed through heuristic methods, yet little is known about their approximation behavior. In this thesis, we systematically examine existing algorithms and heuristics, explore the limits of approximability, and investigate the potential APX-hardness of the problem.

Related Work. The study of orthogonal graph drawing has its roots in the Topology–Shape–Metrics (TSM) approach introduced by Tamassia [Tam87]. The central idea of TSM is to split the graph drawing process into three separate steps in order to make the problem tractable and well-structured. Note that his goal was to draw a given graph with the lowest number of bends. In the topology step, a combinatorial graph is transformed into an embedded plane graph by fixing a planar embedding. The embedding can be chosen according to various aesthetic criteria, such as minimizing edge crossings. As a preprocessing step, planarization may be applied, since not every graph is planar. Unavoidable crossings can be replaced by dummy vertices to obtain a planar representation. Similarly, graphs with vertex degrees higher than four can be handled by degree-reduction preprocessing, in which high-degree vertices are split into several vertices of degree at most four. In the shape step, the embedded plane graph is converted into an orthogonal representation by assigning bends and angles to edges and vertices in a way that respects the embedding. Tamassia proved that these two steps can be solved in polynomial time and that an orthogonal representation with a minimum number of bends can be obtained using a flow-based algorithm. The final metrics step determines the precise coordinates and edge lengths for a given orthogonal representation. The OC problem forms the core of this step, as it seeks a minimum-area layout consistent with the given orthogonal representation.

Complexity. Building on this foundation, Patrignani formally proved that the orthogonal compaction problem is NP-hard [Pat01] using a reduction from SAT via a *sliding-rectangle gadget* construction. Although he initially conjectured that the problem might be APX-hard, he later retracted this claim. More recently, Evans, Fleszar, Kindermann, Saeedi, Shin, and Wolff [EFK+22] refined these hardness results by showing that OC remains NP-hard even for very simple graph classes such as cycles, via a reduction from 3-partition. To date, neither a constant-factor approximation algorithm is known nor has APX-hardness been proven for the classic OC problem. A notable special case occurs when the input orthogonal representation is turn-regular: these representations have no so-called *kitty corners*, i.e., pairs of reflex vertices that diagonally point at each other in a face (see Figure 1.3). For these instances, the optimal compaction can be computed in linear time, which highlights that the presence of kitty corners is precisely what makes the general problem computationally hard due to the freedom of relative placements of kitty corners. As a side remark, Bannister, Eppstein, and Simons [BES12] investigated a relaxed variant of OC where changes to the embedding and the introduction of edge crossings are allowed, proving strong inapproximability results. In contrast, the classic OC problem considered in this thesis maintains a fixed orthogonal representation.

Exact algorithms. Bridgeman, Di Battista, Didimo, Liotta, Tamassia, and Vismara [BBD+00] presented the first exact algorithm for OC, showing that any turn-regular orthogonal representation can be compacted optimally in linear time. Later, Didimo, Gupta, Kindermann, Liotta, Wolff, and Zehavi [DGK+26] introduced a fixed-parameter tractable algorithm, which systematically fixes the relative positions of kitty-corner pairs and applies the exact linear-time compaction algorithm for each configuration, achieving tractability with respect to the number of kitty corners. Exact optimization has also

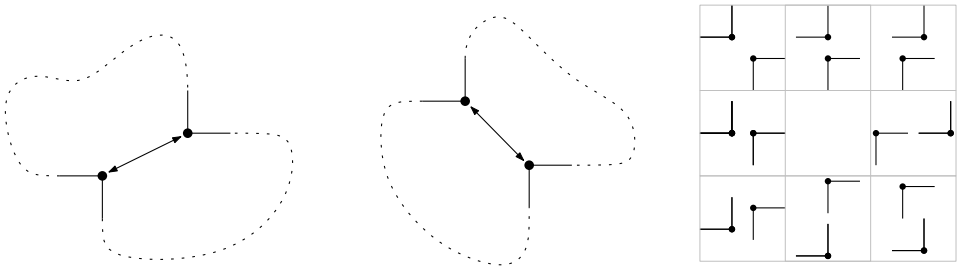


Fig. 1.3: (a)-(b) Schematic illustration of kitty corners (reflex corners of the same face pointing at each other). (c) Intuition for the hardness: the free choice of relative positions for kitty corners makes the problem difficult.

been studied via integer linear programming. Klau and Mutzel [KM99] formulated OC as an ILP and tested it on small instances with up to 300 vertices. Most instances could be solved optimally in under a second, but some required more than an hour of computation. This work demonstrated that compaction can be modeled in a mathematically exact optimization framework.

Heuristics and approximative approaches. Heuristic strategies for orthogonal compaction have been considered in several studies. Since an exact algorithm for turn-regular representation is known, a natural line of research therefore aimed at transforming arbitrary representations into turn-regular ones, and then applying the exact algorithm. In some sense, Tamassia's third step (metrics) can be interpreted in this direction as well, as it enforces rectangularization of all faces by subdividing them into rectangles. Tamassia showed that, after applying this procedure, the area of the resulting drawing is bounded by $O(n^2)$, where n denotes the number of vertices. While this bound is expressed in terms of the number of vertices rather than the optimal area, and no non-trivial bound in terms of the optimal solution is known, it provides a practical starting point for heuristic strategies. Another approach followed a heuristic route by attempting to greedily eliminate kitty corners step by step [BBD+00]. Di Battista, Garg, Liotta, Tamassia, Tassinari, and Vargiu [BGL+97] compared four different graph drawing heuristics, focusing on the overall drawing process and often allowing bends to vary. Only the TSM-based approach among these heuristics directly performs compaction for a fixed orthogonal representation. Other heuristics, such as those by Papakostas and Tollis [PT94] or by Biedl and Kant [BK98], aim at area reduction but do not rely on OC as a subroutine, since bends are not fixed. Later, Binucci and Didimo [BD05] systematically evaluated different compaction heuristics and their impact on drawing quality.

Contribution. In this thesis we construct a family of instances for which both classical heuristics for orthogonal compaction, namely the rectangularization heuristic (in the metrics step) by Tamassia [Tam87] and the greedy edge insertion heuristic that iteratively destroys kitty corner pairs introduced by Bridgeman et al. [BBD+00], perform arbitrarily poorly even for a graph that is a cycle and therefore only has one inner face.

More precisely, we show that the gap between the optimal solution and the solution produced by these heuristics can become unbounded. This result provides strong evidence that there cannot exist any trivial constant-factor approximation algorithm that merely "fixes" relative kitty corner positions in a clever way.

Beyond this, we further investigate the possible APX-hardness of the orthogonal compaction problem. Building upon Patrignani's NP-hardness reduction from SAT via sliding-rectangle gadgets, we propose modifications and extensions that might yield a full APX-hardness proof. We carefully analyze why these ideas could work in principle, but also highlight the structural obstacles that currently prevent us from turning them into a complete proof. This discussion not only outlines a promising direction for future hardness results but also clarifies the precise limitations of existing reduction techniques in this context.

In addition, we discuss potential approaches for designing approximation algorithms for the orthogonal compaction problem and identify the fundamental structural bottlenecks that make such algorithms difficult to realize. Finally, we generalize the problem to three dimensions and examine to what extent the additional degree of freedom affects its approximability and potential APX-hardness.

Outline of the Thesis. We start by introducing formal definitions, notations, and the prerequisite knowledge in Chapter 2. An overview of existing approaches is then provided, including a discussion of the Topology–Shape–Metrics paradigm, the exact linear-time algorithm for turn-regular representations, the rectangularization heuristic, the greedy insertion heuristic, and the fixed-parameter tractable algorithm for compaction in Chapter 3. A family of instances is constructed in Chapter 4 to demonstrate that the gap between heuristic solutions and the optimum can be made arbitrarily large. The thesis then investigates the potential APX-hardness of the orthogonal compaction problem, presenting proof ideas and discussing their current challenges in Chapter 5. Then we examine potential approaches for designing approximation algorithms and analyze the structural barriers that make such algorithms difficult to achieve in Chapter 6. Finally, we generalize the problem to three dimensions and discuss its potential APX-hardness in Chapter 7.

2 Preliminaries

Orthogonal Drawings and Representations. Let G be a connected planar graph of maximum degree at most four. A *planar orthogonal drawing* of G places every vertex at a distinct point of the integer grid and depicts each edge uv as a connected *polygonal chain* of horizontal and vertical line segments that starts at u, ends at v, and does not cross any other edge. A planar graph admits such a drawing if and only if its maximum degree is four. Two drawings of G are considered equivalent if they realize the same planar embedding, exhibit the same ordering of bends along each edge, and preserve the angles at every vertex. An equivalence class of drawings under this relation is called a *planar orthogonal representation*. Every drawing belonging to an orthogonal representation H is referred to as a *drawing* of H. We assume that H contains no bends, since we can simply place a dummy vertex of degree two at each bend. In this way, H can be described purely in terms of vertices and edges, together with the relative orientation of adjacent vertices, i.e., for each pair u, v of neighboring vertices, it is fixed whether u lies to the right, above, to the left, or below v. Coordinates are integral by default, and, when considering a concrete drawing, the length of an edge e in a drawing is written as $\ell(e)$.

The Orthogonal Compaction Problem.

Problem:	Orthogonal Compaction (OC)
Input:	Orthogonal representation H of a connected planar graph G.
Question:	What is a coordinate assignment for H that minimizes the area of its bounding box, i.e., the smallest axis-aligned rectangle containing the drawing?

Formally, the orthogonal compaction problem consists of assigning integer coordinates to every vertex and bend of H such that the resulting drawing is valid and the area of the bounding box is minimized. For a given instance P, we denote by $\mathcal{A}_{\text{OPT}}(P)$ the minimum possible area among all valid drawings of the underlying orthogonal representation. Besides minimizing the area of the bounding box, other objective functions have been studied, such as minimizing the maximum edge length or the total edge length of the drawing. However, in this thesis we focus on the bounding box variant.

Angle Sequences. A key concept for describing the combinatorial structure of drawings is *angle sequences*. An angle sequence is a word

$$S = s_1 \dots s_k \in \{\mathrm{L}, \mathrm{R}, \mathrm{F}\}^k,$$

where L denotes a left turn of $+90°$, R a right turn of $-90°$, and F a forward turn of $+180°$. A polygon (respectively, a polyline) realizes an angle sequence S if a counterclockwise walk along its boundary (respectively, along the polyline) produces exactly the sequence of turns S. Whenever we speak of the boundary of a face, we always traverse it counterclockwise if it is an inner face, and clockwise if it is the outer face. For convenience, within an angle sequence we write $[A \to B]$ for the subsequence starting at a turn $A \in \{\mathrm{L}, \mathrm{R}, \mathrm{F}\}$ and ending at a turn $B \in \{\mathrm{L}, \mathrm{R}, \mathrm{F}\}$, continuing cyclically if necessary. In addition to an angle sequence S, we define two derived sequences. First, let $\tilde{S}$ denote the sequence obtained from S by deleting all occurrences of the symbol F. Formally, let

$$\tilde{S} = s_{i_1} s_{i_2} \dots s_{i_m} \qquad \text{with } s_{i_j} \in \{\mathrm{L}, \mathrm{R}\}.$$

Furthermore, we define $\hat{S}$ as the sequence obtained by reversing $\tilde{S}$ and then inverting all turn directions, i.e., by replacing each L with R and vice versa. Let $\lambda(\cdot)$ be the involution defined by $\lambda(\mathrm{L}) = \mathrm{R}$ and $\lambda(\mathrm{R}) = \mathrm{L}$, and let $\mathrm{rev}(\cdot)$ denote the order reversal of a sequence. Then

$$\hat{S} = \lambda\big(\mathrm{rev}\big(\tilde{S}\big)\big).$$

Chains. To reason about relative directions and, later, about lengths and coordinates, we introduce *chains*. A chain is a sequence of consecutive edges that are aligned in the same direction according to the orientation information in H. A *horizontal chain* thus consists of edges that are all oriented from left to right, while a *vertical chain* consists of edges oriented from bottom to top. A chain is *maximal* if it cannot be extended further in either direction, that is, if it contains as many consecutive edges as possible that share the same orientation. For a chain C, we denote by $\alpha(C)$ and $\Omega(C)$ its two endpoints, corresponding to the first and last vertex of the sequence, respectively. When considering a concrete drawing of H, the *length* of a chain is defined as

$$\ell(C) = \sum_{e \in C} \ell(e),$$

that is, the sum of the lengths of its constituent edges.

Structural Properties of Faces. For a face f of an orthogonal representation H, we always assume that f is traversed in such a way that its interior lies to the left of the walk. With this convention, convex corners are always realized by left turns, and reflex corners by right turns, independent of whether f is an inner or the outer face. Let f be a face of H, traversed in this orientation. A vertex of f is called *convex* if the turn at this vertex is L, and *reflex* if the turn is R. Given two reflex vertices u and v on the boundary of f, the *rotation* $\mathrm{rot}(u, v)$ is defined as the number of convex corners

minus the number of reflex corners encountered while walking along the boundary from u (included) to v (excluded). Two vertices u and v form a pair of *kitty corners* if $\mathrm{rot}(u, v) = 2$ or $\mathrm{rot}(v, u) = 2$. Intuitively, kitty corners are pairs of reflex vertices that are separated along the boundary by exactly two net convex turns. Geometrically, two kitty corners are two reflex corners of the same face "pointing at each other". A face is called *turn-regular* if it does not contain any pair of kitty corners, and an orthogonal representation is turn-regular if all of its faces are turn-regular. We illustrate most of the introduced concepts in Figure 2.1.

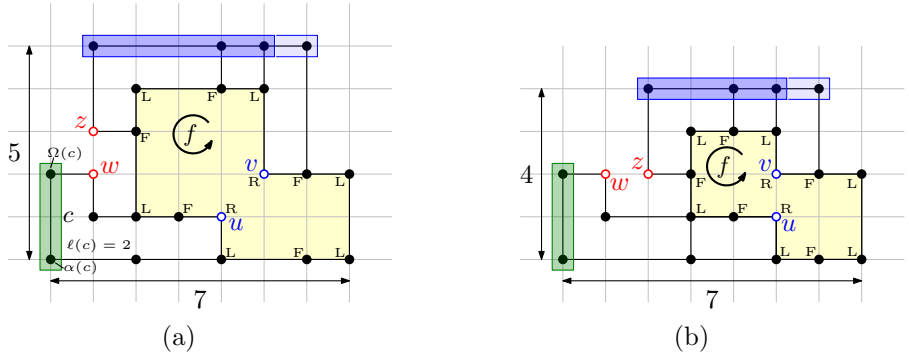


Fig. 2.1: Visualization of the introduced concepts. (a) A drawing of a non-turn-regular orthogonal representation H occupying an area of 35. Vertices u and v form a pair of kitty corners in the inner face f, while vertices w and z form a pair on the outer face. The angle sequence of the yellow-marked face f is shown. The maximal vertical chain c is highlighted in green, with its endpoints and length two, and a horizontal chain is highlighted in blue (which can be extended to a maximal horizontal chain using the light blue extension). Rotation values are $\mathrm{rot}(u, v) = \mathrm{rot}(v, u) = \mathrm{rot}(w, z) = 2$ and $\mathrm{rot}(z, w) = -6$. As an example, $[u \to v] = \mathrm{RLFLLFR}$. Note that all inner faces except f are turn-regular. (b) Another drawing of H occupying an area of 28, showing different relative positions of both kitty-corner pairs.

Complexity. We recall some notions from algorithmic complexity that will be used later. A *decision problem* asks for a yes/no answer; a canonical example is SAT, which asks whether a given Boolean formula is satisfiable. Its optimization counterpart, MAX-SAT, seeks an assignment that maximizes the number of satisfied clauses.

A *reduction* is a polynomial-time transformation that maps instances of one problem to instances of another such that solutions correspond appropriately. A *gap-preserving reduction* is a stronger form that preserves approximation ratios between instances, and is used to compare the relative difficulty of optimization problems.

A problem belongs to the class NP if a solution can be verified in polynomial time. A problem is NP-hard if every problem in NP can be reduced to it by a polynomial-time reduction, and it is NP-complete if it is both NP-hard and in NP.

The class APX consists of all optimization problems that admit a polynomial-time constant-factor approximation algorithm. A problem is called APX-hard if it is at least as hard as the hardest problems in APX under approximation-preserving reductions. If

a problem is APX-hard, then it does not admit a polynomial-time approximation scheme (PTAS) unless $P = NP$.

A parameterized problem with input size l and parameter k is *fixed-parameter tractable (FPT)* if it can be solved in time $f(k) \cdot l^{O(1)}$ for some computable function f depending only on k. Even if the problem is hard in general, it admits efficient algorithms for bounded values of the parameter.

3 Algorithms and Methods

Efficiently representing orthogonal graphs requires balancing structure, aesthetics, and practical constraints. In this chapter, we provide an overview of existing approaches, beginning with the Topology–Shape–Metrics paradigm that underlies many methods. We then discuss exact linear-time algorithms for turn-regular representations and examine two heuristics that address the general case. Finally, we review a fixed-parameter tractable approach for compaction. These methods provide a comprehensive picture of the current state of orthogonal graph drawing.

3.1 The Topology–Shape–Metrics Paradigm

Roberto Tamassia introduced the TSM-paradigm in 1987 [Tam87]. His aim was to establish a framework to produce orthogonal drawings for arbitrary input graphs with maximum degree of four. This framework relies on the principle that an orthogonal drawing can be described through three key properties: its topology, its shape, and its metrics. In the following we describe these components as done in "Graph Drawing: Algorithms for the Visualization of Graphs" [BETT99]:

(i) *Topology*: Two orthogonal drawings are topologically equal if one can be obtained from the other by continuous deformation that does not alter the sequence of edges contouring the faces of the drawing.

(ii) *Shape*: Two orthogonal drawings have the same shape if, firstly, they have the same topology and secondly, one can be obtained from the other by modifying only the length of edges without changing their angles.

(iii) *Metrics*: Two orthogonal drawings have the same metrics if they are congruent, up to a translation and/or rotation.

Each of these properties refines the previous one in describing a drawing. Specifically, if two drawings have the same metrics, then they also share the same shape, and if they share the same shape, they necessarily share the same topology. From this hierarchical description, it is natural to structure an algorithm into three successive phases which are also visualized in Figure 3.1:

(i) *Planarization*: This phase determines the topology of the drawing by fixing a planar embedding. The aim is to reduce the number of edge crossings as much as possible. One can for example extract a maximal planar subgraph and then reinsert all the remaining edges while placing dummy vertices on each crossing.

(ii) *Orthogonalization*: For a given topology, this phase determines the shape of a drawing. The output is an orthogonal representation with minimum number of bends. To accomplish this, a network flow model is formulated and minimized, yielding the angles of the edges incident to each node as well as the placement of bend points.

(iii) *Compaction*: This phase determines the final positions of vertices and bends and removes the auxiliary dummies introduced earlier. The main purpose is to minimize the overall drawing area by means of the bounding box. Edge lengths are assigned through two network flows, one handling the horizontal and the other the vertical direction. As this procedure requires all faces to be rectangular, the original algorithms first refine the drawing so that every face is rectangular, and then perform the length assignment.

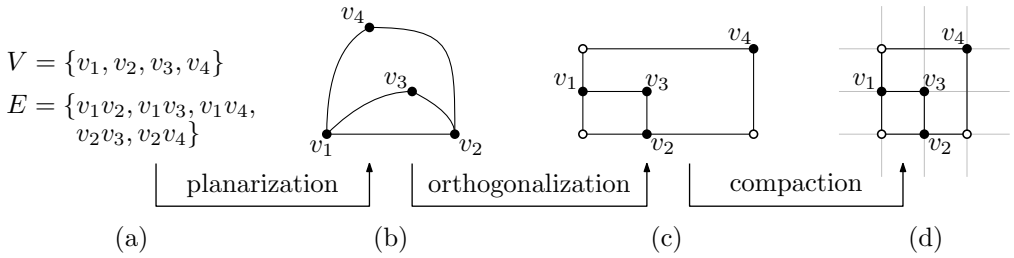


Fig. 3.1: Visualization of the TSM paradigm: (a) abstract description of a graph, (b) planar embedding produced by the planarization phase, (c) orthogonal representation (with white dummy vertices on each bend) produced by the orthogonalization phase, (d) final drawing produced by the compaction phase.

The third step, namely the compaction phase, is precisely the point at which the orthogonal compaction problem emerges. In Section 3.3 we discuss in detail one particular heuristic, referred to as HeurRR which consists of first refining the drawing so that all faces become rectangles and then applying the exact algorithm introduced in Section 3.2. This refinement procedure, known as *rectangularization*, originates directly from the TSM approach where it serves as a crucial step to make the subsequent compaction feasible. Accordingly, we will present it as an exact refinement method and explain it in depth.

3.2 Exact Linear-Time Algorithm for Turn-Regular Representations

Bridgeman et al. showed that turn-regular orthogonal representations can be compacted optimally in linear time [BBD+00]. The central idea of the optimal linear-time compaction algorithm is that the two-dimensional compaction problem can be decomposed into two independent one-dimensional problems. By constructing two directed acyclic graphs (one capturing horizontal constraints and one capturing vertical constraints) the

algorithm computes optimal x- and y-coordinates separately, which together yield an overall optimal compaction. We recall some terminology first and then continue by describing the algorithm. The original concepts were introduced in "Turn-regularity and optimal area drawings of orthogonal representations" [BBD+00]. For clarity of presentation, however, we adopt the formulation and exposition from "Parameterized Approaches to Orthogonal Compaction" [DGK+26].

Consider a directed acyclic graph (DAG) D that is plane. An *upward planar drawing* of D is a drawing that respects the embedding, places vertices at distinct points, and represents each edge as a curve that monotonically increases in the vertical direction. Such a drawing exists exactly when D can be extended to a plane st-graph, i.e. a planar digraph with a unique source and sink on the outer face [BT88].

A structural requirement for upward planarity is *bimodality*: at every internal vertex, all outgoing edges appear consecutively in the rotation around the vertex, and so do all incoming edges. Once an upward drawing exists, it uniquely fixes the left-to-right order of edges at every vertex. This information is captured in what is called an *upward planar embedding*, and a DAG together with such an embedding is referred to as an *upward plane DAG*.

Switches arise when two consecutive edges on the boundary of a face share a vertex and both point either outward (source switch) or inward (sink switch). Each face has equally many source and sink switches. From this number we define the *capacity* of a face: $n_f - 1$ for an inner face with n_f switches, and $n_f + 1$ for the outer face.

In upward drawings, every source and sink has exactly one reflex angle ($> 180°$) and the remaining angles are convex ($< 180°$). To encode this, each switch angle is labeled either L (large) or S (small). For a valid drawing, the number of large labels in a face must equal its capacity [BBLM94].

We can characterize upward embeddings purely through such labelings. Particularly, internal vertices carry no large labels, every source and sink carries exactly one, and each face has as many large labels as its capacity. A labeling that satisfies these conditions is called an *upward labeling*. Since upward labelings and upward embeddings correspond one-to-one, it is common to describe upward plane DAGs directly by their labelings. See Figure 3.2 for an example of an upward plane DAG and its labeling.

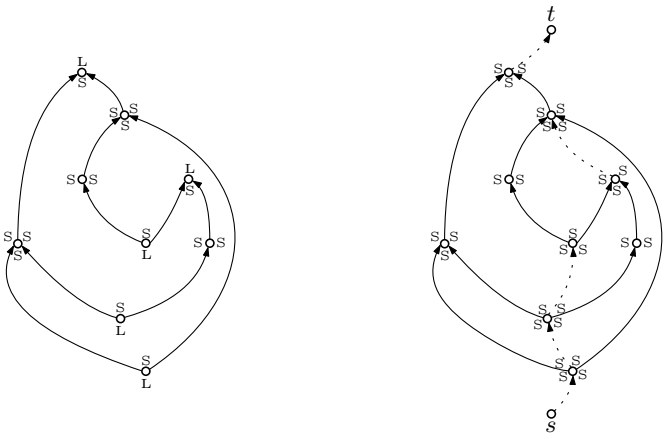
Given an upward plane DAG D, a *complete saturator* is a collection of auxiliary vertices and edges that do not belong to D itself, but serve to augment D into a plane st-graph D'. A saturator always contains a source s and sink t, both placed on the outer face of D', together with a set of *saturating edges*. Each such edge (u, v) satisfies one of the following conditions:

(i) Both u and v are source switches of the same face f, where u is assigned an S-label and v an L-label; in this case, u is said to saturate v.

(ii) Both u and v are sink switches of the same face f, with u carrying an L-label and v an S-label; here, v saturates u.

(iii) The edge connects the source s to a source switch on the outer face that has an L-label.

(iv) The edge connects a sink switch on the outer face with an L-label to the sink t.



(a) (b)

Fig. 3.2: (a) an upward plane DAG and its corresponding labeling. (b) the same DAG with a complete saturator (dotted edges).

We now return to the context of OC. The constructions of upward plane DAGs and complete saturators allow us to capture the relative horizontal and vertical positions of chains in an orthogonal representation. This provides the combinatorial framework needed to reason about coordinate assignments in the compaction problem. We denote by D_x (D_y respectively) the plane DAG whose vertices correspond to the maximal vertical (horizontal respectively) chains of a given orthogonal representation H. Two vertices of D_x are connected by an edge oriented rightward, if the vertical chains, that are represented by these vertices, are connected by a horizontal edge in H. For a vertex v of H, we denote by $c_x(v)$ ($c_y(v)$ respectively) the vertex of D_x (D_y respectively) that corresponds to the maximal vertical (horizontal respectively) chain that contains v. We now introduce some notation to capture the relative position of each pair of vertices of H. For any two vertices u and v of H with $c_x(u) \neq c_x(v)$, we write $u \rightsquigarrow_x v$ if there exists a directed path from $c_x(u)$ to $c_x(v)$ in D_x. Further we write $u \leftrightsquigarrow_x v$ if either $u \rightsquigarrow_x v$ or $v \rightsquigarrow_x u$, and $u \not\leftrightsquigarrow_x v$ if neither $u \rightsquigarrow_x v$ nor $v \rightsquigarrow_x u$ exists. Analogously we define all the above relations within D_y for any u and v with $c_y(u) \neq c_y(v)$.

Bridgeman et al. [BBD+00] proved that an orthogonal representation H is turn-regular if and only if, for every pair of vertices $u, v \in H$ either their relative order along the x-axis, the y-axis, or both is fixed, i.e. $u \leftrightsquigarrow_x v$, or $u \leftrightsquigarrow_y v$, or both. Equivalently, this characterization reflects the fact that no kitty corner pairs exist in H: for such pairs, we can still decide on their relative placement, whereas in a turn-regular representation all relative positions are already fixed. This property allows the OC problem for H to be reduced to two independent one-dimensional compaction problems, one for the x-direction and one for the y-direction, each solvable in linear time. The compaction in x-direction works as follows:

(i) Construct the DAG D_x, whose vertices represent the vertical chains of H, and augment it to a plane st-graph using a complete saturator.

(ii) Compute an optimal topological ordering $\mathcal{X}$ of D_x. Each vertex $v \in H$ then receives its x-coordinate according to $x(v) = \mathcal{X}(c_x(v))$.

The compaction in y-direction works analogously. Note that the complete saturator in each dimension is uniquely determined if and only if H is turn-regular [BBD+00]. We demonstrate the idea of the algorithm in Figure 3.3. Note that this approach only works

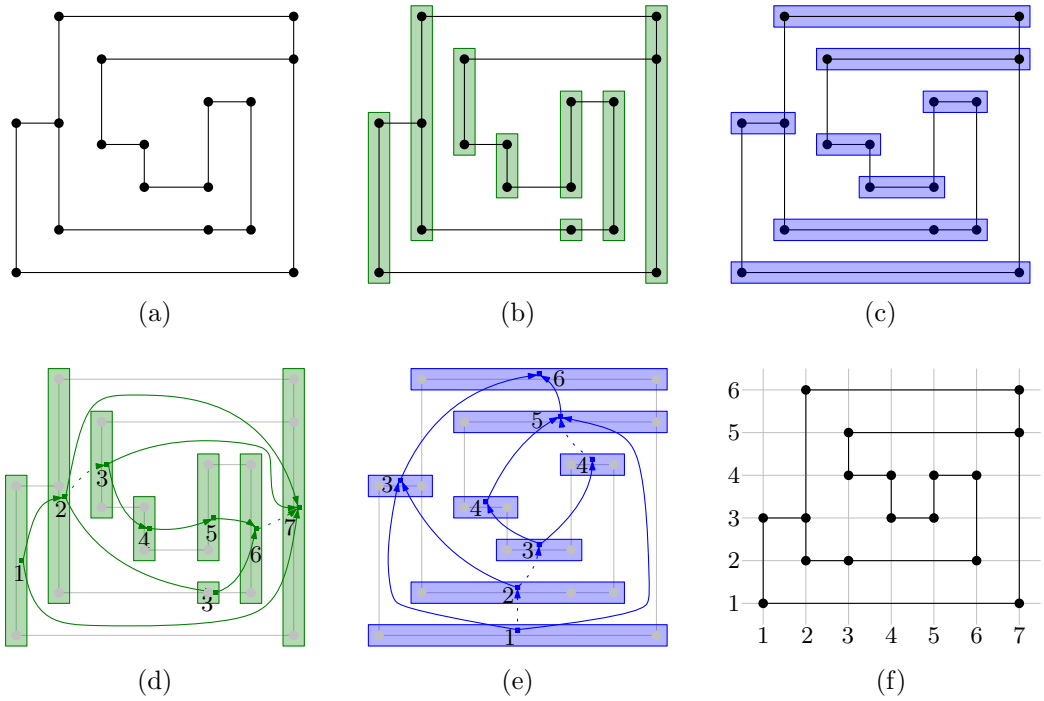


Fig. 3.3: (a) A turn-regular orthogonal representation H. (b) The maximal vertical chains of H. (c) The maximal horizontal chains of H. (d) The upward plane DAG D_x with its complete saturator (dotted edges) and an optimal topological numbering. (e) The same for D_y. (f) An optimal drawing of H where the coordinates of each vertex correspond to the topological numberings of D_x and D_y.

for turn-regular representations, since for kitty corners u, v we have $u \not\leftrightarrow_x v$ and $u \not\leftrightarrow_y v$, implying that no fixed order exists on either axis. The method fails because (i) there may be multiple complete saturators per dimension, and (ii) even combining two saturators with independent topological numberings can yield non-planar, hence invalid, drawings. This highlights that kitty corners are the natural parameter driving the hardness of the problem.

Didimo et al. [DGK+26] therefore proposed an FPT algorithm parameterized by the number of kitty corners: for each pair and each dimension, they fix the relative position and then apply the linear-time algorithm. Concretely, within a kitty corner pair one vertex may lie to the left, to the right, or on the same y-coordinate as the other. This yields

3 options per dimension, i.e. 8 possible arrangements per pair (since two vertices cannot be identical); see Figure 1.3. Since the number of possible configurations depends solely on the number of kitty corners k, and each configuration can be tested in polynomial time, this yields a fixed-parameter tractable algorithm.

3.3 Heuristic Approaches for Turn Regularization

Since an exact algorithm is available for turn-regular representations, a natural approach is to transform an arbitrary orthogonal representation into a turn-regular one, and then apply the linear-time algorithm. However, no approximation algorithm is known that guarantees turn-regularization while simultaneously bounding the drawing area for arbitrary graphs. In the literature, two heuristics have been proposed to address this task which we discuss in the following.

3.3.1 Rectangular Refinement Heuristic

This heuristic directly corresponds to a subroutine of the third step in the TSM approach and therefore was also initially proposed by Tamassia. In order to execute the two flow networks described in Section 3.1 for length assignment, all faces of the orthogonal representation must be rectangular. The refinement process therefore aims at transforming arbitrary (non-rectangular) faces into rectangles, enabling the subsequent flow computations. We now formally introduce the heuristic, following the description given in "Graph Drawing: Algorithms for the Visualization of Graphs" [BETT99].

Let G be an embedded planar graph and let H be an orthogonal representation of G. A rectangular refinement H' of H is obtained from a supergraph G' of G, where G' is constructed by (i) adding isolated vertices, (ii) subdividing an edge by inserting a vertex, or (iii) adding an edge to G. The orthogonal representation H' preserves H as a subrepresentation, while ensuring that all faces of H' are rectangular; see Figure 3.4 for an illustration.

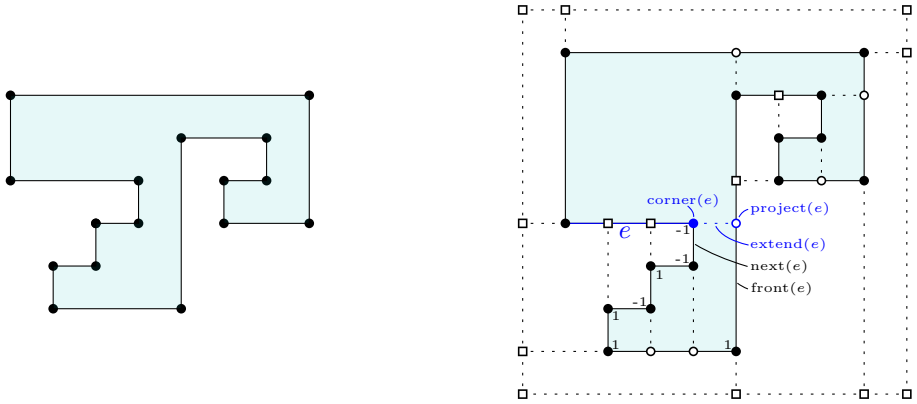
We now iteratively refine all inner faces. Let f be an inner face. If f already is rectangular, nothing has to be done. Otherwise we proceeded with the following steps:

(i) Traverse the boundary of f counterclockwise. For each edge e, denote by $\text{next}(e)$ the edge following e and by $\text{corner}(e)$ the common vertex of e and $\text{next}(e)$.

(ii) For each edge e, compute $\text{turn}(e)$: set $\text{turn}(e) = 1$ if e and $\text{next}(e)$ form a left turn (convex corner), $\text{turn}(e) = 0$ if they are aligned, and $\text{turn}(e) = -1$ if they form a right turn (reflex corner).

(iii) For each edge e, find the first edge e' encountered counterclockwise after e such that the sum of the turn values along the path from e (included) to e' (excluded) equals 1. Set $\text{front}(e) = e'$.

(iv) For each edge e with $\text{turn}(e) = -1$, insert a vertex $\text{project}(e)$ on the edge $\text{front}(e)$, and add the edge $\text{extend}(e) = (\text{corner}(e), \text{project}(e))$. Update H' such that e and $\text{extend}(e)$ are aligned at $\text{corner}(e)$.



(a) (b)

Fig. 3.4: (a) An orthogonal representation. (b) The same orthogonal representation after applying the rectangularization heuristic. White circles (squares) are the dummy vertices on the inner (outer) face added by the heuristic. The dashed lines represent the added edges and the bounding box. Note that in this example, the left "wing" of the graph is forced to extend downward due to conflicting dummy vertices introduced by the inner and outer face refinements.

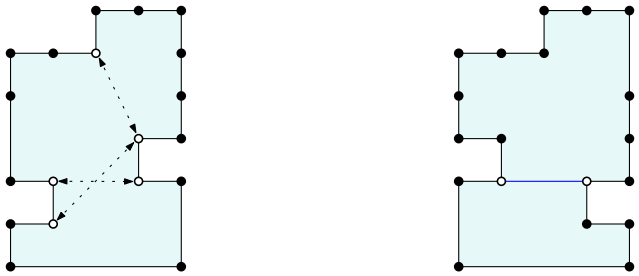
We omit the proof which can be found in "On Embedding a Graph in the Grid with the Minimum Number of Bends" [Tam87] or "Graph Drawing: Algorithms for the Visualization of Graphs" [BETT99]. The refinement of the outer face can be carried out by a slight modification of the above algorithm. Since the turn sum of the outer face is -4 (while the turn sum for every inner face is exactly $+4$) there may be edges for which $\text{front}(e)$ is not defined. Therefore we add a bounding box where all edges e with $\text{turn}(e) = -1$ and non existing $\text{front}(e)$ are projected on (see Figure 3.4). After refining all inner faces and the outer face with the procedure (and its modification for the outer face) given above, we get a supergraph G' of G along with a representation H' such that H is a subrepresentation of H' and all faces are rectangles. Therefore no more kitty corners exist and we can use the exact linear-time algorithm to get a drawing for G' and remove all dummy vertices afterwards.

Note that the order in which the edges are inserted does not matter. It is easy to see that the resulting drawing is always the same, regardless of the starting edge in the traversal of a face. This applies to all inner faces as well as to the outer face. Tamassia showed that, after rectangularizing all faces and compacting the resulting supergraph, the optimal area is bounded by $O(n^2)$, where n denotes the number of vertices of the graph (recall that, in the preprocessing step, a vertex was already placed at each bend). Note that this bound is given in terms of the number of vertices rather than the optimal area. To the best of our knowledge, no non-trivial upper bound on the area in terms of the optimal solution is known in the literature. Bridgeman et al. [BBD+00] suggested an improvement of the rectangularization heuristic by requiring that a dummy vertex on $\text{front}(e) = e'$ is only introduced when it is strictly necessary. In particular, the endpoint of e' that lies further away from e along the counterclockwise boundary traversal can

already serve as project(e). This refinement avoids undesirable effects such as those illustrated in Figure 3.4, where the left "wing" of the graph is forced to extend downward due to conflicting dummy vertices introduced by the inner and outer refinements. In such cases, the upper endpoint of e' could simply have been chosen as project(e). In Chapter 4, we construct instance families for which the gap between the optimal area and the area obtained by applying our rectangularization heuristic (which yields turn-regularization) followed by the exact compaction algorithm can become arbitrarily large. We will also discuss the improvement of the heuristic in this regard. From now on we will refer to the rectangularization heuristic as introduced by Tamassia as HeurRR.

3.3.2 Greedy Insertion Heuristic

Another natural approach to achieve turn-regularity of a given graph is to destroy kitty corner pairs by simply adding horizontal or vertical edges between pairs of kitty corners. In particular a straight edge which is randomly chosen to be horizontal or vertical is inserted in between a kitty corner pair. This is done until all faces are decomposed into smaller turn-regular faces. Note that this constitutes a much weaker structural constraint, since the resulting faces are not required to be rectangular as in HeurRR. In general, this method introduces considerably fewer dummy edges than HeurRR. Also note that the number of inserted edges is bounded by the number of kitty corner pairs but is much smaller in most cases. This is due to the fact that an insertion of one straight edge decomposes one face into two smaller faces and therefore potentially separates a former kitty corner pair. We illustrate this effect in the following figure. Throughout this



(a) (b)

Fig. 3.5: (a) A turn-regular orthogonal representation H with three kitty corner pairs highlighted by the dotted arrows. (b) The same representation with one horizontal edge inserted between one kitty corner pair which already separates and therefore eliminates all kitty corner pairs.

paper we will refer to this greedy insertion heuristic as HeurGI. We will analogously to HeurRR show that even this less restrictive technique of turn-regularization may lead to arbitrarily bad compaction results.

4 Worst-Case Behavior of Two Heuristics on a Constructed Instance Family

In this section, we investigate the worst-case behavior of the heuristics HeurRR and HeurGI. We construct a specific family of instances that is tailored to expose the limitations of both approaches. In particular, we show that the gap between the optimal solution and the solutions produced by the heuristics can be made arbitrarily large.

4.1 Construction of the Instance Family of Snails

We introduce a family of instances constructed to expose a common structural weakness in both HeurRR and HeurGI. The key issue lies in the repeated occurrence of kitty corner placements, which are not handled adequately by either heuristic, resulting in suboptimal solutions with unnecessarily large areas. We precisely describe how to define the instance family $\mathcal{P} = \{P_i \mid i \geq 1\}$ of planar, orthogonal graphs with an induced orthogonal representation. For this purpose, we first define angle sequences, which serve as blueprints for constructing the graph family. Specifically, for each $i \geq 1$, we recursively construct a polyline p_i represented by an angle sequence. At this stage, p_i (and consequently P_i) is not yet a graph or polygon, but merely a sequence of angles. We then augment p_i to a proper polygon P_i via a connector c, which can subsequently be interpreted as a planar, orthogonal graph with an induced orthogonal representation. Intuitively the idea is to construct an instance family of "growing snails", where a snail with a higher index has more windings than one with a lower index (see Figure 4.1 and Figure 4.2 for an example).

Definition 4.1 (snails)**.** *Let*

$$c = \mathrm{FLLLR} = \mathrm{FL}^3\mathrm{R} \tag{4.1}$$

$$p_1 = \mathrm{FLF}^2\mathrm{L}^3\mathrm{FR}^2 \tag{4.2}$$

$$p_i = \mathrm{F}^{4i-3}\mathrm{LF}^{5i-3}\mathrm{L}^4\mathrm{RF}^2\ p_{i-1}\ \mathrm{F}^2\mathrm{R}^3\mathrm{F}^{5(i-1)}\mathrm{RF}^{4(i-1)} \quad \text{for } i \geq 2 \tag{4.3}$$

$$P_i = p_i\ c. \tag{4.4}$$

*Let P_i be the i-*th snail.

Note that the i-th snail consists of $9i^2 + 8i - 2$ vertices. In Figure 4.1 we show how p_1 and c are drawn and also illustrate how the recursive construction works in detail. Without loss of generality we assume that the vertex that corresponds to the first F of p_i has its successor (L or F) exactly to the left. Likewise we assume that the vertex that

corresponds to the first F of c has its successor (L) exactly to the right. While this is not necessary, we do this to ensure a uniform orientation of the instances we construct. We now define crucial structures in the graph family of snails.

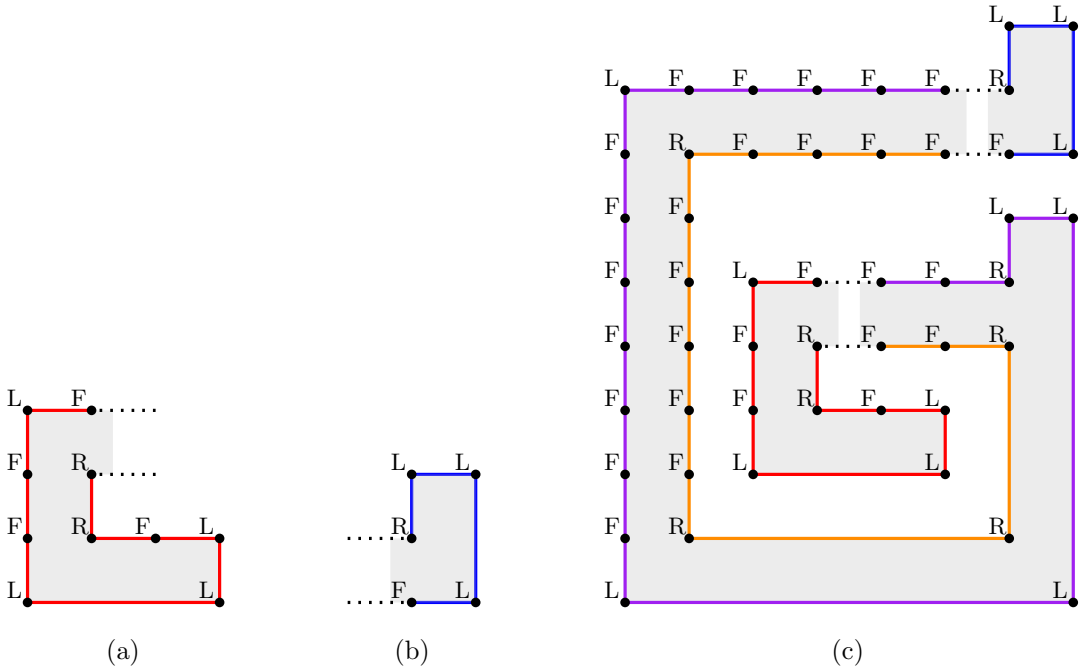


Fig. 4.1: (a) Drawing of p_1. (b) Drawing of connector c. The dashed lines between (a) and (b) indicate how P_1 is derived from p_1 and c. (c) Recursive construction and drawing of $P_2 = \underline{\mathrm{F}^5\mathrm{LF}^7\mathrm{L}^4\mathrm{RF}^2}\ \underline{p_1}\ \underline{\mathrm{F}^2\mathrm{R}^3\mathrm{F}^5\mathrm{RF}^4}\ \underline{c}$. The turn angles are measured with respect to the inner face.

Definition 4.2 (crucial chains)**.** *Let p_i be a given angle sequence as introduced in Definition 4.1. We define the following* crucial chains *within p_i:*

ol_i ***(outer left)***	*the vertical chain enclosed by the first and second occurrences of* L.
ob_i ***(outer bottom)***	*the horizontal edge enclosed by the second and third occurrences of* L.
or_i ***(outer right)***	*the vertical edge enclosed by the third and fourth occurrences of* L.
ct_{i-1} ***(connector top)***	*the horizontal edge enclosed by the fourth and fifth occurrences of* L.
cl_{i-1} ***(connector left)***	*the vertical edge enclosed by the fifth occurrence of* L *and its successor* R.

ot_{i-1} ***(outer top)***	*the horizontal chain enclosed by the first occurrence of* R *and the subsequent* L.
it_{i-1} ***(inner top)***	*the horizontal chain enclosed by the fifth-to-last and fourth-to-last occurrences of* R.
ir_i ***(inner right)***	*the vertical edge enclosed by the fourth-to-last and third-to-last occurrences of* R.
ib_i ***(inner bottom)***	*the horizontal edge enclosed by the third-to-last and second-to-last occurrences of* R.
il_i ***(inner left)***	*the vertical chain enclosed by the last two occurrences of* R.

For the sake of completeness we refer to the horizontal chain between or_1 *and* il_1 *as* ib_1. *Furthermore, when considering* P_i, *which is derived from* p_i *and the connector* c, *we define the following chains and edges:*

ot_i ***(outer top)***	*the horizontal chain enclosed by the last occurrence of* R *and the first occurrence of* L.
it_i ***(inner top)***	*the horizontal chain enclosed by the second-to-last occurrence of* R *and the subsequent* L.
cr_i ***(connector right)***	*the vertical edge enclosed by the third-to-last and second-to-last occurrences of* L.
ct_i ***(connector top)***	*the horizontal edge enclosed by the second-to-last and last occurrences of* L.
cl_i ***(connector left)***	*the vertical edge enclosed by the last occurrence of* L *and its successor* R.

Note that ct_0, cl_0, ot_0, it_0 and ir_1 do not exist by definition. All of the above definitions introduced for p_i apply analogously to higher-indexed instances. For example, since p_2 contains the structure of p_1, it follows that p_2 also includes all the elements defined within p_1, namely ol_1, ob_1, or_1, il_1. We refer to this structural preservation as *hierarchical inheritance*. In contrast, when considering P_i, the edges ot_i, it_i, cr_i, ct_i and cl_i are not subject to inheritance. This is because the connector c serves as a structural termination; it completes the construction of P_i and thus does not participate in the recursive embedding of previous patterns.

Definition 4.3 (Crucial Kitty Corners)**.** *Let* P_i *be the* i*-th snail and let* $j \in \{1, \ldots, i\}$. *Let* u_j *be the first (bottommost) vertex of* cl_j *(i.e.* $\alpha(\mathrm{cl}_j)$*) and let* v_j *be the last (topmost) vertex of* il_j *(i.e.* $\Omega(\mathrm{cl}_j)$*). We say* u_j, v_j *are* crucial kitty corners *and denote them by* C_j, K_j, *respectively.*

Note that, for each P_i, the vertices C_j and K_j form a pair of kitty corners (see Figure 4.2 for an illustration).

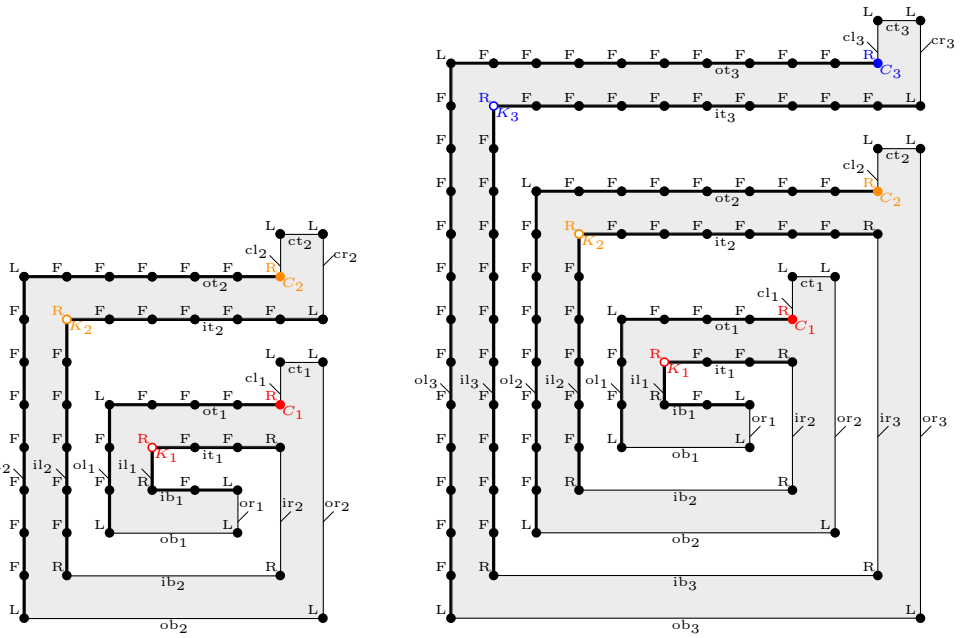


Fig. 4.2: (a) Drawing of P_2 with all crucial structures as defined in Definition 4.2 and Definition 4.3. (b) The same for P_3; note the hierarchical inheritance mentioned before. All defined chains are drawn bold; all defined edges are drawn normal. The turn angles are measured with respect to the inner face.

4.2 Optimal Drawings for the Snails

Before we use the defined graph family to prove the suboptimal behavior of HeurRR and HeurGI, we discuss how they are drawn optimally.

Theorem 4.4. *Let P_i be the i-th snail. The minimum area of P_i is $\mathcal{A}_{\mathrm{OPT}}(P_i) = 20i^2 - 9i + 1$, where the width is $W = 4i - 1$ and the height is $H = 5i - 1$.*

Proof. Consider $P_i = p_i\ c = \mathrm{F}^{4i-3}\mathrm{LF}^{5i-3}\mathrm{L}^4\mathrm{RF}^2\ p_{i-1}\ \mathrm{F}^2\mathrm{R}^3\mathrm{F}^{5(i-1)}\mathrm{RF}^{4(i-1)}\ c$. Observe that the initial part $\mathrm{F}^{4i-3}\mathrm{L}$ forms a horizontal chain h' with width at least $4i-3$ (note that h' is a subchain of ot_i). The connector c contributes at least one additional unit of width. Moreover, since h' and c must be connected via a horizontal segment (which itself requires at least one unit of width), the total minimum width needed to draw P_i is $W = 4i - 1$. Note further that the subsequent part $\mathrm{LF}^{5i-3}\mathrm{L}$ starts with a single L (the identical L as in h'), causing the path to turn and thus producing a vertical chain ol_i with height at least $5i - 2$. By a similar argument as before, the connector requires at least one additional unit of height. This yields a total minimum height of $H = 5i - 1$ needed to draw P_i; see Figure 4.3. We provide a drawing for each P_i that attains these bounds by assigning the following edge lengths. For each $j \in \{1, \dots, i\}$, we set $\ell(\mathrm{ob}_j) = 4j - 1$ and $\ell(\mathrm{or}_j) = 5j - 4$. Moreover, for each $j \in \{2, \dots, i\}$, we set $\ell(\mathrm{ib}_j) = 4j - 3$ and $\ell(\mathrm{ir}_j) = 5j - 7$. Additionally we set $\ell(\mathrm{cr}_i) = 2$. All remaining edges are assigned unit length. □

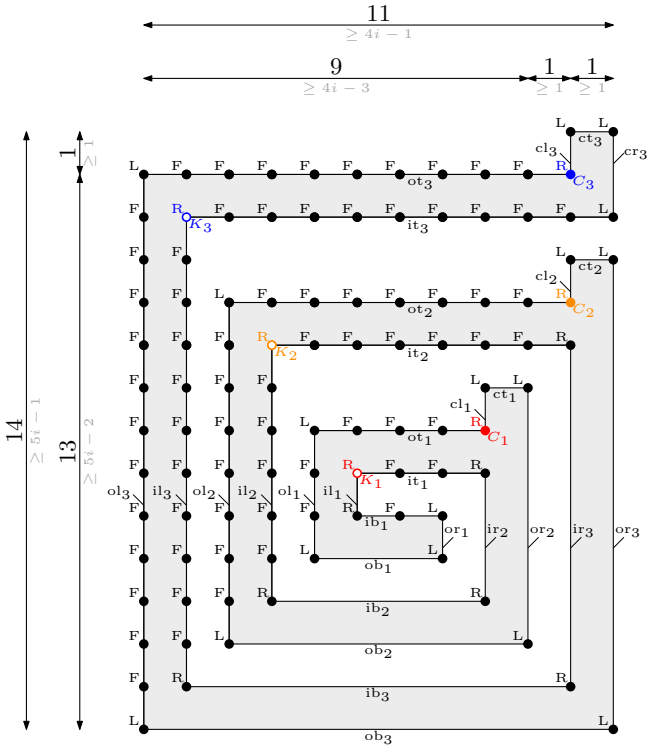


Fig. 4.3: Optimal drawing of P_3. The turn angles are measured with respect to the inner face.

Intuitively the snails are designed in such a way that, when "rolled up", they fit together compactly, allowing for an optimal drawing with the discussed area.

4.3 Performance of HeurRR on the Snails

We now investigate the performance of HeurRR on the defined graph family $\mathcal{P}$. Recall that HeurRR consists of three sub-steps, where, for the sake of simplicity, we regard the application of the exact linear-time compaction algorithm as the third sub-step:

(i) Refinement of all interior faces (*here*: only one inner (gray) face).

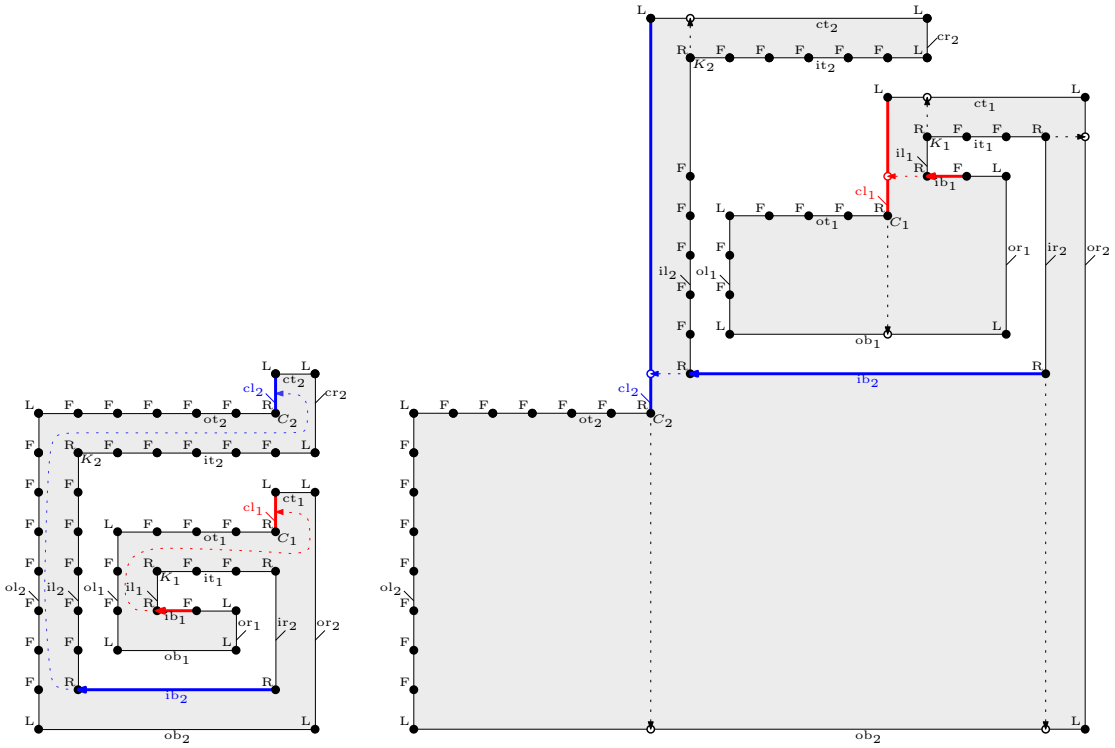
(ii) Refinement of the outer face.

(iii) Assignment of length value by applying the exact linear-time algorithm.

In the following, we describe in detail how the three steps of the heuristic behave on our graph family.

Refinement of the inner face. We provide a schematic view of the refinement process applied to the inner face of P_2 in Section 4.3. Informally speaking, the refinement of the inner face causes the spiral to expand and stretch apart. Note that the insertion of dummy vertices does not alter the definition of edges and chains. For instance, if a

dummy vertex is inserted on the edge cl_j, the designation cl_j still refers to the same original endpoints. While the edge may now include a dummy vertex, its delimiting vertices remain unchanged. Figure 4.4 (a) shows the initial structure of P_2 drawn opti-



(a) (b)

Fig. 4.4: sub-step (1) of HeurRR: Refinement of the inner (gray) face. The turn angles are measured with respect to the inner face.

mally. The critical edges are dummy edges extending the edges ib_j on cl_j for $1 \le j \le i$ and thus responsible for the expansion during the refinement. They are highlighted as red and blue dotted arrows. This follows the refinement pattern described earlier, as introduced by Tamassia. Figure 4.4 (b) displays the critical edges inserted during the refinement (red/blue dotted arrows), the resulting expansion, and all additional dummy edges (black dotted arrows) as well as all inserted dummy vertices (disks with white interior). We state this formally in the following lemma.

Lemma 4.5. *The refinement of the inner face applied to P_i places every il_j to the right and fully above C_j for all $j \in \{1, \ldots, i\}$.*

Proof. Recall that Tamassia's refinement algorithm works by starting at a reflex vertex (which contributes –1) and traversing counterclockwise, adding –1 for each reflex vertex (R) and +1 for each convex vertex (L), until the cumulative sum reaches +1. The next edge in this traversal is then the target for the dummy edge to be added from the original starting point. Further recall that the order in which dummy edges are inserted

is irrelevant in Tamassia's algorithm. This allows us to focus solely on the critical edges, as they are not subject to any particular insertion order. Since only left or right turns are relevant, we omit all F-turns in our sequence representations. Observe that

$$\tilde{P}_i = (\mathrm{L}^5\mathrm{R})^{i-1}\mathrm{L}^4\mathrm{R}^2(\mathrm{R}^4)^{i-1}\mathrm{L}^3\mathrm{R} \tag{4.5}$$

$$= (\mathrm{LLLLLR})^{i-1}\mathrm{LLLLRR}(\mathrm{RRRR})^{i-1}\mathrm{LLLR} \tag{4.6}$$

holds for all $i \geq 1$. The starting point of all critical edges that are responsible for the expansion, are the lower endpoints $\alpha(\mathrm{il}_j)$ of the introduced vertical chains il_j. These lower endpoints correspond to every fourth R in the block $\mathrm{RR}(\mathrm{RRRR})^{i-1}$, starting from the first, i.e. positions of the form $4(j-1)+1$ for $j \in \{1,\ldots,i\}$ in $\mathrm{RR}(\mathrm{RRRR})^{i-1}$. Note further that the first occurrence corresponds to the lower end of il_1, the second to the lower end of il_2, and so on (see Figure 4.5 for an illustration). Starting from each of the i selected R's, we traverse the sequence forward, subtracting 1 for each R and adding 1 for each L, continuing cyclically if needed. We stop when the cumulative sum first reaches +1 and mark that position. For $j \in \{1,\ldots,i-1\}$ the cumulative sum of the j-th selected R reaches +1 exactly after the last L (and thus immediately before the R) in the $(i-j)$-th (LLLLLR)-block. The R in the $(i-j)$-th block LLLLLR corresponds exactly to C_j. It remains to discuss the last R, i.e. $j = i$. For this last selected R the cumulative sum reaches +1 exactly after the last L (and thus immediately before the R) in the (LLLR)-block (which is the block corresponding to the connector c). It is easy to see that this R corresponds to C_i. For all $j \in \{1,\ldots,i\}$ the edge preceding C_j in counter-clockwise order is cl_j. Due to the construction, each cl_j lies vertically above C_j. Thus horizontal dummy edges are inserted between the mentioned R vertices (i.e. the lower endpoints of the vertical chains il_j) and dummy vertices on cl_j. The graph construction permits this edge only if il_j is located to the right and fully above of C_j, which proves the claim. □

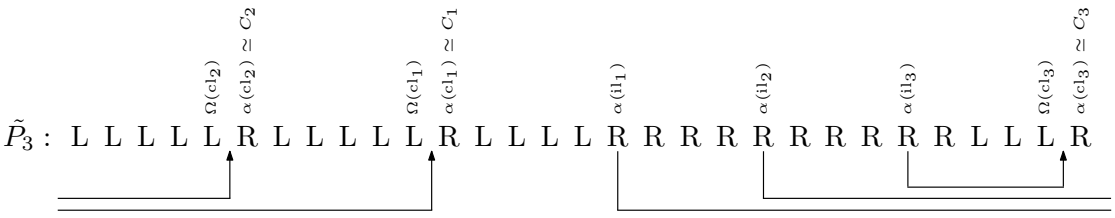
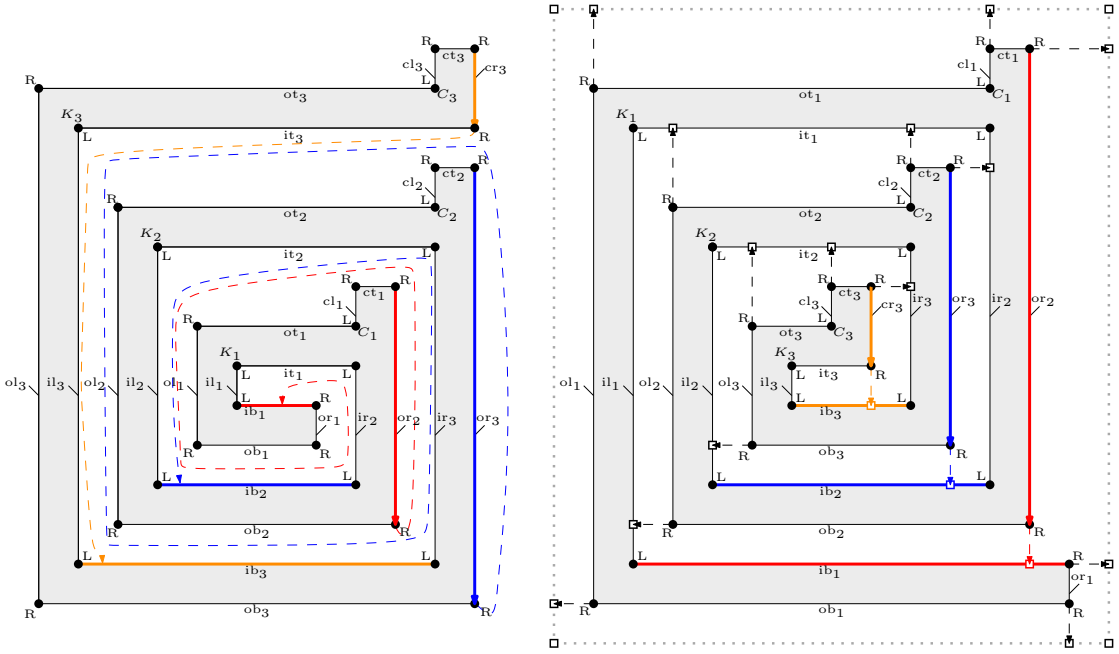


Fig. 4.5: Illustration of $\tilde{P}_3$ mentioned in Lemma 4.5: The arrows start at the reflex vertices and point to the edges (lying between two consecutive points) where a dummy edge needs to be added. These are precisely the edges where the rotation sum first reaches +1.

Note that only the critical refinement steps, that are responsible for the expansion, have been formally proven; see Section 4.3 for a visual illustration. It is easy to see that following the structural modifications introduced by critical steps, all remaining reflex vertices on the inner face can be trivially rectified. The analysis is now continued by considering the second step of HeurRR.

Refinement of the outer face. We begin by illustrating the effect of the outer refinement using P_3 as a simplified example. We then outline all critical steps in detail and explain why the argument applies to all snails. Note that the outer refinement is not



(a) (b)

Fig. 4.6: sub-step (2) of HEURRR: Refinement of the outer (white) face. The turn angles are measured with respect to the outer face.

affected by the inner refinement. For clarity, we leave the inner face unrefined to better illustrate the effect of the outer refinement. Additionally, we omit all F-nodes and, as a result, do not necessarily draw the edges and chains to scale. These adjustments, however, do not influence the effect of the outer refinement. Figure 4.6 (a) shows the initial structure of P_3. Furthermore, the critical edges that induce the reversal of the winding direction during the outer refinement are highlighted in red, blue and orange. Here, critical edges denote the dummy edges that are extended onto the edges ib_j for $1 \leq j \leq i$. The dashed arrow indicate the edges to which a dummy edge is added, originating from the tail of the arrow. Figure 4.6 (b) displays the critical edges inserted during the refinement (red/blue/orange dashed arrows), the resulting structure with a reversed winding direction, all additional dummy edges (black dashed arrows) as well as all inserted dummy vertices (squares with white interior). Furthermore, the outer refinement introduces a bounding box which is highlighted by gray dotted lines. At a conceptual level, the outer refinement causes the snail graph to wind in the opposite direction: the end that initially lies on the outside ends up on the inside, and vice versa.

Lemma 4.6. *The refinement of the outer face of P_i enforces the x-coordinates of the outer right chains as follows: $x(\mathrm{cr}_i) < x(\mathrm{or}_i) < x(\mathrm{or}_{i-1}) < \cdots < x(\mathrm{or}_1)$. Similarly,*

the refinement determines the y-coordinates of the inner bottom chains according to the following order: $y(\mathrm{ib}_1) < y(\mathrm{ib}_2) < \cdots < y(\mathrm{ib}_i)$.

Proof. As in the proof of Lemma 4.5, we determine for each critical edge the point at which the cumulative turn sum reaches +1 for the first time. Note that we now traverse the sequence in clockwise order, as we are considering the outer face. Unlike before, we must take the turning angles of the outer face into account. From this perspective, each left turn on the inner face corresponds to a right turn on the outer face (and vice versa). Therefore, consider

$$\hat{P}_i = \mathrm{LR}^3(\mathrm{L}^4)^{i-1}\mathrm{L}^2\mathrm{R}^4(\mathrm{LR}^5)^{i-1} \tag{4.7}$$

$$= \mathrm{LRRR}(\mathrm{LLLL})^{i-1}\mathrm{LLRRRR}(\mathrm{LRRRRR})^{i-1}. \tag{4.8}$$

We focus on the critical edges whose starting points are the lower endpoints of the edges or_j (i.e. $\alpha(\mathrm{or}_j)$) for $j \in \{2, \ldots, i\}$ first. These lower endpoints correspond to every fifth R in the block $(\mathrm{LRRRRR})^{i-1}$, starting from the third, i.e. positions of the form $6(j-1)+4$ for $j \in \{1, \ldots, i-1\}$ in $(\mathrm{LRRRRR})^{i-1}$. Further, note that the first occurrence corresponds to $\alpha(\mathrm{or}_2)$, the second to $\alpha(\mathrm{or}_3)$, and so on. Starting from each of the $i-1$ selected R's, we traverse the sequence $\hat{P}_i$, continuing cyclically if needed. For $j \in \{1, \ldots, i-1\}$ the cumulative sum of the j-th selected R reaches +1 exactly after the $(4(i-j)+2)$-th L in the $(\mathrm{LLLL})^{i-1}\mathrm{LL}$-block. The $(4(i-j)+2)$-th L in the $(\mathrm{LLLL})^{i-1}\mathrm{LL}$-block corresponds exactly to the right endpoint of ib_j (i.e. $\alpha(\mathrm{ib}_j)$). Therefore a vertical dummy edge originating from each $\alpha(\mathrm{or}_j)$ for $j \in \{2, \ldots, i\}$ to ib_{j-1} is inserted; see Figure 4.7 for an illustration. It remains to discuss the right turn corresponding to $\alpha(\mathrm{cr}_i)$. It is easy to see that this particular R is the third R in $\hat{P}_i$ and therefore its cumulative sum reaches +1 exactly after the second L succeeding that R. This L corresponds to the left endpoint of ib_i (i.e.$\alpha(\mathrm{ib}_i)$), so a vertical dummy edge originating from $\alpha(\mathrm{cr}_i)$ to ib_i is inserted. The insertion of the discussed critical edges yields the reversed winding of each P_i and thus the desired relative positions of the edges, as claimed. □

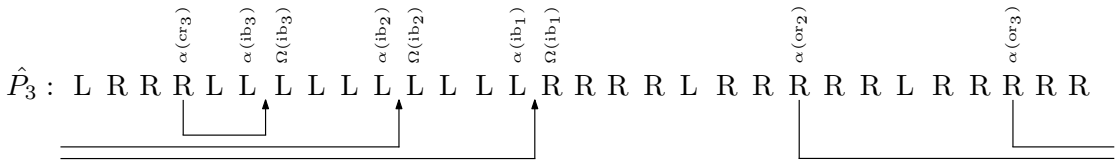


Fig. 4.7: Illustration of $\hat{P}_3$ mentioned in Lemma 4.6: The arrows start at the reflex vertices and point to the edges (lying between two consecutive points) where a dummy edge needs to be added. These are precisely the edges where the rotation sum first reaches +1.

We proceed by stating two lemmas.

Lemma 4.7. *For each* $j \in \{2, \ldots, i\}$ *the dummy edge inserted from* $\Omega(\mathrm{ol}_j)$ *to* it_{j-1} *forces all vertices that are on* it_{j-1} *(except the right end vertex* $\Omega(\mathrm{it}_{j-1})$ *of* it_{j-1}*) to lie to the left of the inserted dummy vertex.*

Proof. Since we again traverse in a clockwise direction as in Lemma 4.6, it is easy to see that for the specified starting points (the upper endpoints of ol_j for $j \in \{2, \ldots, i\}$), the sum increases to $+1$ upon reaching the right endpoint of it_{j-1}. As a result, a dummy node is inserted at the immediate successor edge (in clockwise direction). □

Lemma 4.8. *For each $j \in \{2, \ldots, i\}$ the dummy edge inserted from $\alpha(\mathrm{ob}_j)$ to il_{j-1} forces all vertices that are on il_{j-1} (except the upper end vertex $\Omega(\mathrm{il}_{j-1})$ of il_{j-1}) to lie to the bottom of the inserted dummy vertex.*

Proof. Analogous to Lemma 4.7. □

Note that only the critical refinement steps have been formally proven. All other refinement steps are omitted at this point but can be visually inferred from Figure 4.6 (b). It is easy to see that, following the structural modifications introduced by critical steps, all remaining reflex vertices on the outer face can be trivially rectified; either by inserting a dummy vertex directly on the boundary of the outer face, or by placing a dummy vertex on the newly introduced bounding box. In Figure 4.8 we show the combined refinement of the inner and outer face on the example of P_3, along with the resulting shape of the orthogonal graph drawing.

Assignment of length values. In the following, we do not describe how the exact compaction algorithm assigns length values. Instead, we derive a lower bound on the area that any such assignment must respect. We do this by carefully examining the instances after the two refinement steps. This lower bound is in fact tight, as it exactly matches the optimum area with respect to the suboptimal choice of kitty corners that arise from the refinement steps. This area analysis finally allows us to quantify the suboptimal behavior of HeurRR. We now analyze the minimal required width and height of any layout of this instance separately. This is done by identifying non-overlapping horizontal and vertical regions ($\mathcal{H}_1$–$\mathcal{H}_6$ and $\mathcal{V}_1$–$\mathcal{V}_4$ in Figure 4.8), each of which requires a dedicated unit of space in the overall layout. Here, a non-overlapping region is defined as a set of chains that do not share any interior coordinates along the relevant axis: for horizontal regions, no two chains in the same region share an interior x-coordinate (they may touch at boundaries if adjacent); for vertical regions, no two chains share an interior y-coordinate. Each non-overlapping region thus requires a dedicated unit of space in the corresponding dimension of any layout. We state this in the following two lemmas.

Lemma 4.9. *The minimal width of each drawing of the i-th snail produced by* HeurRR *is at least $4i^2 + 6i - 3$.*

Proof. Consider the following lower bounds for the non-overlapping horizontal regions for some P_i (with $i \geq 2$):

$$\mathcal{H}_1 : \quad \ell(\mathrm{ot}_1) + 1 \tag{4.9}$$

$$\mathcal{H}_2 : \quad \sum_{j=1}^{i-1} \ell(\mathrm{it}_j) \tag{4.10}$$

$$\mathcal{H}_3 : \quad \sum_{j=2}^{i} \big(\ell(\mathrm{ot}_j) + 1\big) \tag{4.11}$$

$$\mathcal{H}_4 : \quad \ell(\mathrm{it}_i) + 1 \tag{4.12}$$

$$\mathcal{H}_5 : \quad 2i - 3 \tag{4.13}$$

$$\mathcal{H}_6 : \quad 2 \tag{4.14}$$

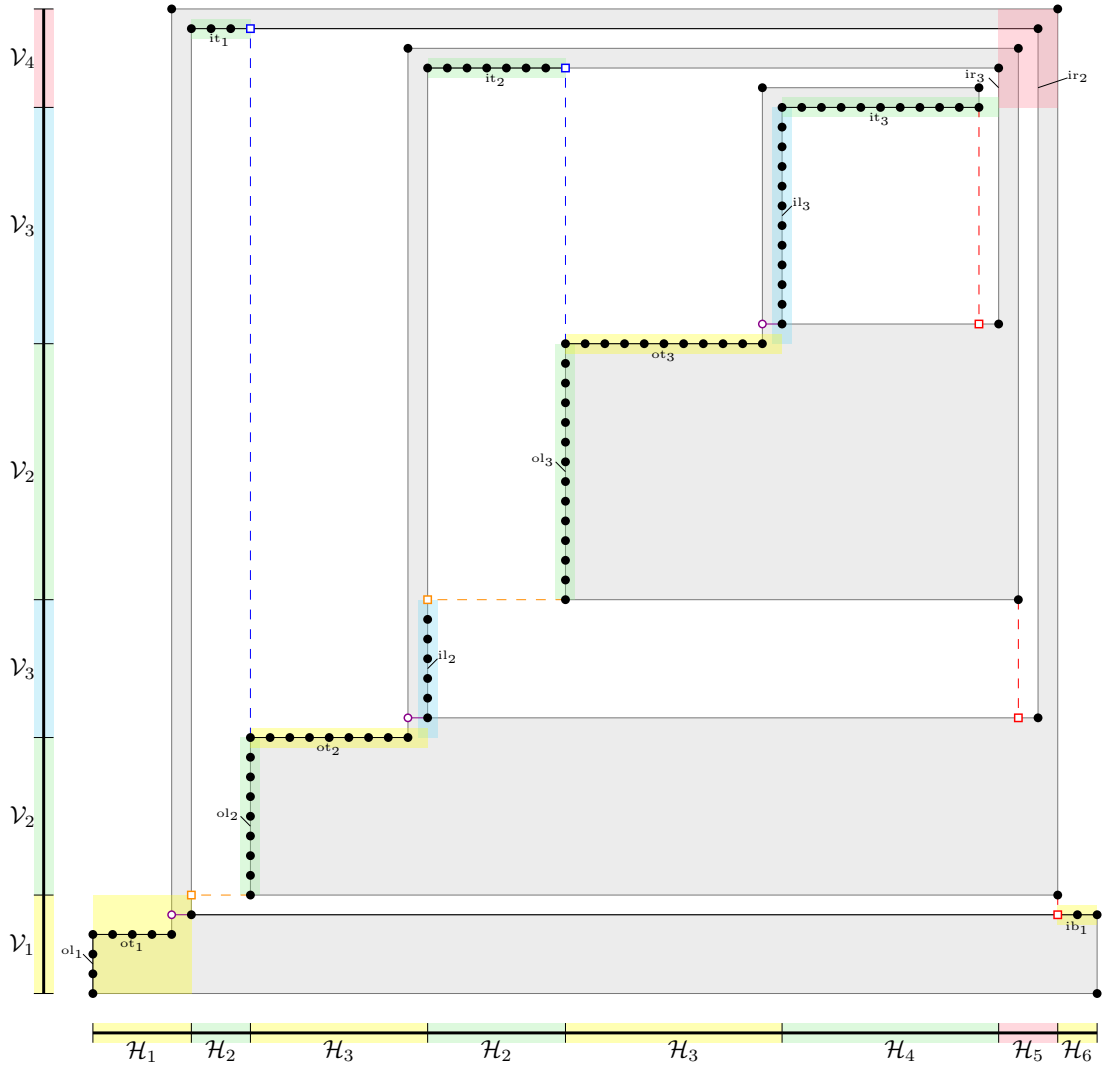


Fig. 4.8: Drawing produced by HeurRR for the instance P_3. Purple edges with purple circles indicate dummy edges and vertices from Lemma 4.5. Red, blue, and orange dashed edges with square markers correspond to dummy edges and vertices from Lemma 4.6, Lemma 4.7, and Lemma 4.8, respectively. Furthermore each horizontal region (labeled $\mathcal{H}_1$ through $\mathcal{H}_6$) corresponds to a lower bound contribution to the overall width, while each vertical band (labeled $\mathcal{V}_1$ through $\mathcal{V}_4$) contributes to the lower bound of the height.

Explanations:

$\mathcal{H}_1$ Region $\mathcal{H}_1$ must include the horizontal chain ot_1 and requires at least one additional unit of width for the purple dummy edge.

$\mathcal{H}_2$ This lower bound holds because the blue dummy vertices can be seen as the right endpoints of it_j for $j \in \{1, \dots, i-1\}$, according to Lemma 4.7. Hence, the green-marked segments contributing to $\mathcal{H}_2$ each have length at least $\ell(\mathrm{it}_j)$.

$\mathcal{H}_3$ Region $\mathcal{H}_3$ includes each of the horizontal chains ot_j for $j \in \{2, \dots, i\}$, and requires one additional unit of width for each purple dummy edge lying to the right of the respective ot_j.

$\mathcal{H}_4$ Region $\mathcal{H}_4$ must include the horizontal chain it_i and needs one extra unit of width for placing the vertical edge ir_i to the right of it_i.

$\mathcal{H}_5$ This bound results from including the vertical chains ir_j for $j \in \{2, \dots, i\}$. Between every ir_j and ir_{j-1} for $j \in \{i, \dots, 3\}$, there must be a vertical edge or_j, and or_2 must lie to the right of ir_2. Therefore, the minimum width of region $\mathcal{H}_5$ is $2i-3$.

$\mathcal{H}_6$ The red dummy vertex can be interpreted as the left endpoint of ib_1, due to Lemma 4.6 (and $\ell(\mathrm{ib}_i) = 2$).

Combining all the above regions, we obtain a global lower bound on the required width for P_i as the sum of the individual contributions. With a slight abuse of notation, we now write $\mathcal{H}_1, \dots, \mathcal{H}_6$ to denote these lower bounds, although they originally refer to the corresponding regions.

$$\begin{aligned}
\text{min width} &\geq \mathcal{H}_1 + \mathcal{H}_2 + \mathcal{H}_3 + \mathcal{H}_4 + \mathcal{H}_5 + \mathcal{H}_6 && (4.15)\\
&= \ell(\mathrm{ot}_1) + 1 + \sum_{j=1}^{i-1} \ell(\mathrm{it}_j) + \sum_{j=2}^{i} \big(\ell(\mathrm{ot}_j) + 1\big) + \ell(\mathrm{it}_i) + 1 + 2i - 3 && (4.16)\\
&= \sum_{j=1}^{i} \big(\ell(\mathrm{it}_j) + \ell(\mathrm{ot}_j)\big) + 3i && (4.17)\\
&\qquad \left[\text{since } \ell(\mathrm{it}_j) = \begin{cases} 4j-1 & \text{for } j < i \\ 4j-2 & \text{for } j = i \end{cases}, \quad \ell(\mathrm{ot}_j) = \begin{cases} 4j & \text{for } j < i \\ 4j-2 & \text{for } j = i \end{cases}\right]\\
&= 4i^2 + 6i - 3, && (4.18)
\end{aligned}$$

which yields the claim. □

Lemma 4.10. *The minimal height of each drawing of the i-th snail that is generated by* HeurRR *is at least* $5i^2 + 2i - 1$.

Proof. Consider the following lower bounds for the non-overlapping vertical regions for some P_i (with $i \geq 2$):

$$\mathcal{V}_1 : \quad \ell(\mathrm{ol}_1) + 2 \tag{4.19}$$

$$\mathcal{V}_2 : \quad \sum_{j=2}^{i} \ell(\mathrm{ol}_j) \tag{4.20}$$

$$\mathcal{V}_3 : \quad \sum_{j=2}^{i} \big(\ell(\mathrm{il}_j) + 1\big) \tag{4.21}$$

$$\mathcal{V}_4 : \quad 2i - 1 \tag{4.22}$$

Explanations:

- $\mathcal{V}_1$ Region $\mathcal{V}_1$ must include the vertical chain ol_1 and requires at least two additional units of height for the purple and orange dummy vertices.
- $\mathcal{V}_2$ Region $\mathcal{V}_2$ includes each of the vertical chains ol_j for $j \in \{2, \ldots, i\}$.
- $\mathcal{V}_3$ This lower bound holds because the orange dummy vertices can be seen as the upper endpoints of il_j for $j \in \{2, \ldots, i\}$, according to Lemma 4.8. Hence, the blue-marked segments contributing to $\mathcal{V}_3$ each have length at least $\ell(\mathrm{il}_j) + 1$, since the purple dummy vertices enforce the requirement of one additional unit of height below each il_j for $j \in \{2, \ldots, i\}$.
- $\mathcal{V}_4$ This bound results from including the horizontal chains it_j for $j \in \{1, \ldots, i\}$. Between every it_j and it_{j-1} for $j \in \{i, \ldots, 2\}$, there must be a horizontal chain ct_j, and ct_1 must lie to the top of it_1. Therefore, the minimum height of region $\mathcal{V}_4$ is $2i - 1$.

Combining all the above regions, we obtain a global lower bound on the required height for P_i as the sum of the individual contributions. With a slight abuse of notation, we now write $\mathcal{V}_1, \ldots, \mathcal{V}_4$ to denote these lower bounds, although they originally refer to the corresponding regions.

$$\text{min height} \geq \mathcal{V}_1 + \mathcal{V}_2 + \mathcal{V}_3 + \mathcal{V}_4 \tag{4.23}$$

$$= \ell(\mathrm{ol}_1) + 2 + \sum_{j=2}^{i} \ell(\mathrm{ol}_j) + \sum_{j=2}^{i} \big(\ell(\mathrm{il}_j) + 1\big) + 2i - 1 \tag{4.24}$$

$$= \sum_{j=1}^{i} \big(\ell(\mathrm{il}_j) + \ell(\mathrm{ol}_j)\big) + 3i - 1 \tag{4.25}$$

$$\Big[\text{since } \ell(\mathrm{il}_j) = 5j - 4, \quad \ell(\mathrm{ol}_j) = 5j - 2 \quad \text{for } j \in \{1, \ldots, i\}\Big]$$

$$= 5i^2 + 2i - 1, \tag{4.26}$$

which yields the claim. □

As illustrated in Figure 4.8, the introduced lower bounds are indeed tight for the length assignment generated by HeurRR. We proceed by summarizing the main results.

Theorem 4.11. *The drawing of P_i using the refinements of* HeurRR *requires a drawing area of at least $\mathcal{A} = 20i^4 + 38i^3 - 7i^2 - 12i + 3 \in \Theta(i^4)$, where the width is $W = 4i^2 + 6i - 3 \in \Theta(i^2)$ and the height is $H = 5i^2 + 2i - 1 \in \Theta(i^2)$.*

Proof. By applying Lemma 4.9 and Lemma 4.10 the claim follows directly. □

We can now prove the suboptimal behavior of HeurRR by comparing the optimal area of P_i with the area required by applying HeurRR to P_i.

Theorem 4.12 (Area Gap of HeurRR on P_i)**.** *For the family of instances P, the quotient between the area required by* HeurRR *and the optimal area is $\Theta(i^2)$; that is,*

$$\frac{\mathcal{A}_{\text{HeurRR}}(P_i)}{\mathcal{A}_{\text{Opt}}(P_i)} = \Theta(i^2). \tag{4.27}$$

Proof. The assertion is an immediate consequence of Theorem 4.4 and Theorem 4.11. □

We conclude this section by discussing the improvement of HeurRR proposed by Bridgeman et al. [BBD+00]. Recall that this modification refines the rectangularization heuristic by introducing a dummy vertex only when it is strictly necessary. With this adjustment, we address the gaps discussed in Lemma 4.7 and Lemma 4.8, which produce the regions b in the horizontal and C in the vertical direction. These gaps disappear since the vertices on the chains are no longer pushed toward the boundary as a consequence of the fixed insertion of dummy vertices. However, the horizontal regions a and c, as well as the vertical regions B arising from the inner refinement, remain unaffected. Consequently, the modified heuristic achieves a constant improvement in the overall area for our family of snails, but it is easy to see that the asymptotic growth of the quotient $\mathcal{A}_{\text{HeurRR}}/\mathcal{A}_{\text{Opt}}$ still lies in $\Theta(i^2)$.

4.4 Performance of HeurGI on the Snails

Recall that HeurGI operates by iteratively resolving kitty corner pairs through the insertion of straight-line edges. As our instances always consist of exactly one inner face, we decompose the heuristic into three sub-steps:

(i) Decomposition of all interior faces (*here*: only one inner (gray) face).
In this step, each inner face is recursively decomposed by adding straight edges (chosen randomly as horizontal or vertical) between two kitty corners, until the face consists of smaller (not necessarily rectangular) turn-regular subfaces.

(ii) Decomposition of the outer face.
Following the same principle, straight edges are recursively inserted between kitty corners of the outer face, resulting in a subdivision into turn-regular regions.

(iii) Assignment of length value by applying the exact linear-time algorithm.

In the following, we describe (analogously to Section 4.3) how the first two sub-steps of the heuristic behave on our graph family.

Decomposition of the inner face. We provide a schematic view of the decomposition process applied to the inner face of P_2 in Figure 4.9 (b). We then describe all critical steps in detail and explain why the argument holds for all instances of the constructed family. Figure 4.9 (a) shows the initial structure of P_2 drawn optimally. Furthermore, the kitty

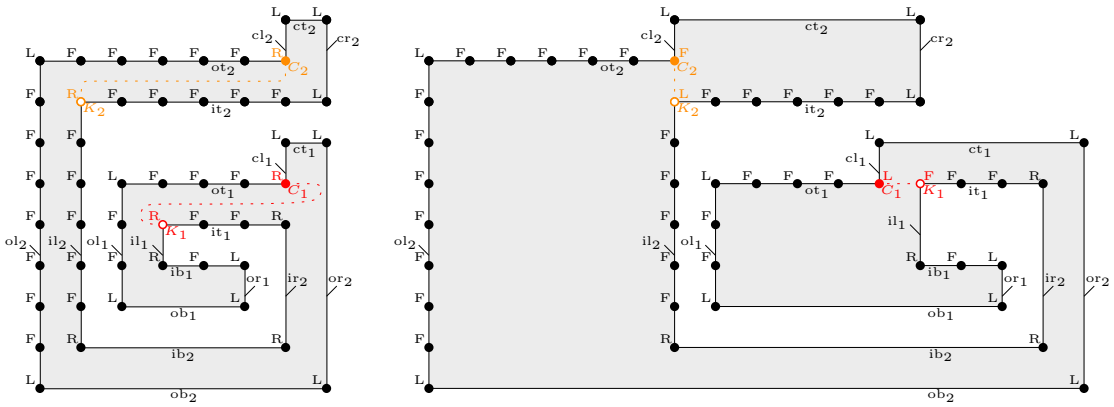


Fig. 4.9: Sub-step (1) of HEURGI: Decomposition of the inner (gray) face. The turn angles are measured with respect to the inner face.

corner pairs that are responsible for the expansion are highlighted in red and orange. The dotted lines indicate the dummy edges. Figure 4.9 (b) displays the inserted edges between the kitty corner pairs (red/orange dotted lines), and the resulting expansion. Observe that a horizontal edge was added between K_1 and C_1, while a vertical edge connects K_2 and C_2. Informally speaking, the decomposition of the inner face increases the width of the snail, as kitty corner pairs that are originally far apart in an optimal drawing are forced to lie adjacent (either vertically or horizontally).

Lemma 4.13. *Every decomposition of the inner face of P_i requires inserting one (horizontal or vertical) edge between each pair C_j and K_j, for all $j \in \{1, \ldots, i\}$, regardless of the insertion order.*

Proof. First, we prove that each pair (C_j, K_j) satisfies the kitty corner property. Recall that a face is traversed in counterclockwise direction, and the rotation $\mathrm{rot}(u, v)$ between two reflex vertices u and v along the boundary counts the number of convex corners (L) minus the number of reflex corners (R), including u but excluding v. Two vertices form a pair of kitty corners if $\mathrm{rot}(u, v) = 2$ or $\mathrm{rot}(v, u) = 2$. We therefore again consider the turn sequence

$$\tilde{P}_i = (\mathrm{L}^5\mathrm{R})^{i-1}\mathrm{L}^4\mathrm{R}^2(\mathrm{R}^4)^{i-1}\mathrm{L}^3\mathrm{R}, \tag{4.28}$$

where we again omit all F-nodes. Note that for $j \in \{1, \ldots, i-1\}$ the j-th R in $\tilde{P}_i$ corresponds to C_{i-j} and the $(4j-2)$-th R in the block $\mathrm{R}^2(\mathrm{R}^4)^{i-1}$ corresponds to K_j. It follows that we can extract the part from C_j to K_j for $j \in \{1, \ldots, i-1\}$ as

$$[C_j \to K_j] = \mathrm{R}(\mathrm{L}^5\mathrm{R})^{j-1}\mathrm{L}^4\mathrm{R}^2(\mathrm{R}^4)^{j-1}, \tag{4.29}$$

where the first R is C_j and the last R is K_j. It is now easy to see that $\mathrm{rot}(C_j, K_j) = 2$, since $C_j \to K_j$ includes $(5j-3)$ R's (while including the first but excluding the last) and $(5j-1)$ L's. We now observe that the sequences $[C_j \to K_j]$ are nested in the sense that the vertices appear in the order

$$C_{i-1}, C_{i-2}, \ldots, C_1, K_1, \ldots, K_{i-2}, K_{i-1} \tag{4.30}$$

along $\tilde{P}_i$. Consequently, the angle sequence $[C_k \to K_k]$ contains all angle subsequences $[C_j \to K_j]$ for $j < k$. Now consider the insertion of an edge between a fixed pair (C_k, K_k). When traversing counter-clockwise from C_j to K_j for any $j > k$, the inserted edge causes the traversal to skip the entire angle subsequences between C_k and K_k. That is, the angle subsequences $[C_k \to K_k]$ is replaced by either LF (in the case of a horizontal edge) or FL (in the case of a vertical edge). Figure 4.9 (b) illustrates how the insertion of an edge causes the angle subsequences between C_k and K_k to be skipped. Since the total turn count of the original angle sequence $[C_k \to K_k]$ (now including the last R) is 1, and since both LF and FL also contribute a turn sum of 1, the overall rotation $\mathrm{rot}(C_j, K_j)$ remains unchanged for all $j > k$. Thus, inserting a connecting edge between any pair (C_k, K_k) preserves the kitty corner property of all subsequent pairs (C_j, K_j) with $j > k$. It remains to discuss C_i and K_i. Note that C_i (K_i) corresponds to the second-to-last (last) R in $\tilde{P}_i$. Therefore, $\mathrm{rot}(C_i, K_i) = 2$, since $[C_i \to K_i]$ is simply the final block $\mathrm{RL}^3\mathrm{R}$ in $\tilde{P}_i$. As HeurGI is designed to eliminate all kitty corner pairs, and it is straightforward to verify that the inner face contains no such pairs other than (C_j, K_j) for $j \in \{1, \ldots, i\}$, while the inserted edges do not interfere with one another, the claim follows. □

Lemma 4.14. *The decomposition of the inner face places every chain* it_j *to the right of the chain* ot_j *for all* $j \in \{1, \ldots, i\}$ *regardless of whether the edge between* C_j *and* K_j *is being inserted vertically or horizontally.*

Proof. Since C_j is the right endpoint of ot_j and K_j is the left endpoint of it_j, there are only two ways to connect C_j and K_j with an orthogonal edge. In both cases (vertical or horizontal edge insertion) it_j lies to the right of ot_j; see Figure 4.9 (b). □

Decomposition of the outer face. Before analyzing the effect of the outer face decomposition, we first show that by applying HeurGI to any P_i ($i \geq 2$, since P_1 has no kitty corner pairs on the outer face), exactly one edge is inserted into the outer face, and that this single edge eliminates all existing kitty corner pairs on the outer face. We start with two preparation lemmas.

Lemma 4.15. *Let* $\hat{P}_i = \mathrm{LR}^3(\mathrm{L}^4)^{i-1}\mathrm{L}^2\mathrm{R}^4(\mathrm{LR}^5)^{i-1}$ *be given as in Lemma 4.6. For a kitty corner pair* $(\mathrm{R}_{start}, \mathrm{R}_{end})$ *on the outer face* $[\mathrm{R}_{start} \to \mathrm{R}_{end}]$ *has to contain the whole* $(\mathrm{L}^4)^{i-1}\mathrm{L}^2$ *block.*

Proof. As already described in Lemma 4.6 each R (L respectively) is a reflex (convex respectively) vertex on the outer face. Now consider an arbitrary kitty corner pair $(\mathrm{R}_{start}, \mathrm{R}_{end})$ and the associated angle subsequence $[\mathrm{R}_{start} \to \mathrm{R}_{end}]$. Since the turn sum starts at -1 due to the initial R_{start}, we need at least three more left turns than right turns in order to reach $\mathrm{rot}(\mathrm{R}_{start}, \mathrm{R}_{end}) = 2$. Observe that in the sequence $\hat{P}_i$ each L outside of the block $(\mathrm{L}^4)^{i-1}\mathrm{L}^2$ is followed by at least three R's. Consequently these L's alone cannot accumulate sufficient positive contribution to achieve a total turn sum of two. It follows that any $[\mathrm{R}_{start} \to \mathrm{R}_{end}]$ yielding a rotation sum of 2 must include the block $(\mathrm{L}^4)^{i-1}\mathrm{L}^2$, as claimed. □

Lemma 4.16. *In the outer face of P_i, the insertion of a single edge (either horizontal or vertical) between the two vertices of any kitty corner pair destroys all kitty corner pairs. That is, after inserting such an edge, the outer face contains no further kitty corner pairs.*

Proof. Let $(\mathrm{R}_{start}, \mathrm{R}_{end})$ be an arbitrary kitty corner pair on the outer face of P_i. By Lemma 4.15, the path $[\mathrm{R}_{start} \to \mathrm{R}_{end}]$ must contain the full turn block $(\mathrm{L}^4)^{i-1}\mathrm{L}^2$. We insert a new edge between R_{start} and R_{end}, either horizontally or vertically, thereby splitting the outer face into two new faces. Let us denote the path $[\mathrm{R}_{start} \to \mathrm{R}_{end}]$ by Q, and its complement on the outer face by $\overline{Q} = [\mathrm{R}_{end} \to \mathrm{R}_{start}]$. After the insertion, both R_{start} and R_{end} become part of three distinct faces: the original inner face and the two newly created faces. The original right turns at these vertices (each contributing a reflex turn of -1) are destroyed and replaced with one left turn each on the two new outer faces. More precisely, the new structure at these two vertices now exhibits either an FL or LF pattern, depending on the orientation of the inserted edge. This is illustrated in Figure 4.10 (b) and Figure 4.10 (c). In this sense, the FL and LF replacements are two convex turns that structurally eliminate the former pair of reflex turns defining the kitty corner.

Let Q^* denote the updated version of Q, i.e. the path $[\mathrm{R}_{start} \to \mathrm{R}_{end}]$ after inserting the new edge and replacing the original RR pair at the endpoints with either FL or LF. Define $\overline{Q^*}$ analogously. In both paths, the total contribution to the turn sum increases by $+3$ (since $-1 - 1 \to 0 + 1$). Note that $\mathrm{rot}(\mathrm{R}_{start}, \mathrm{R}_{end}) = 2$, which excludes the endpoint R_{end}. Throughout, $\mathrm{rot}(\cdot)$ denotes the full rotation sum along the specified path, including the final vertex. Thus, $\mathrm{rot}(Q) = 1$ and $\mathrm{rot}(Q^*) = 4$. Since the total rotation of $\hat{P}_i$ is -4, it follows that $\mathrm{rot}(\mathrm{R}_{end}, \mathrm{R}_{start}) = -6$, $\mathrm{rot}(\overline{Q}) = -7$ and $\mathrm{rot}(\overline{Q^*}) = -4$.

Suppose for the purpose of contradiction that a second kitty corner pair $(\mathrm{R}_1, \mathrm{R}_2)$ appears on the modified outer face (i.e. not in the original inner face). Then, by Lemma 4.15, the path $[\mathrm{R}_1 \to \mathrm{R}_2]$ must again contain the full block $(\mathrm{L}^4)^{i-1}\mathrm{L}^2$. This follows from the fact (which we denote $(\star)$) that the replacement of R_{start} and R_{end} by FL or LF only introduces isolated L turns. All remaining turn sequences still satisfy the following: every single L (outside the large L block) is followed by at least one R, and any new potential LL arising from the replacement is followed by at least two R turns. This makes it impossible to generate a total rotation sum of $+2$ without including the large L block. Since this block appears only once (in Q^*), the complementary path $\overline{Q^*}$ cannot contain another kitty corner pair.

Let $Q' \subseteq Q^*$ be the subpath from R_1 to R_2 and let $\overline{Q'}$ be the complementary subpath in Q^* such that $Q^* = Q' \cup \overline{Q'}$. Suppose $\mathrm{rot}(R_1, R_2) = 2$, i.e. $\mathrm{rot}(Q') = 1$. Then the complementary subpath satisfies $\mathrm{rot}(R_2, R_1) = \mathrm{rot}(Q^*) - \mathrm{rot}(R_1, R_2) = 2$, i.e. $\mathrm{rot}(\overline{Q'}) = 1$. By the same reasoning ($\star$), this is impossible unless $\overline{Q'}$ contains the large L block, which it does not. Thus, our assumption leads to a contradiction. Therefore, no new kitty corner pair can exist after the insertion of one edge. □

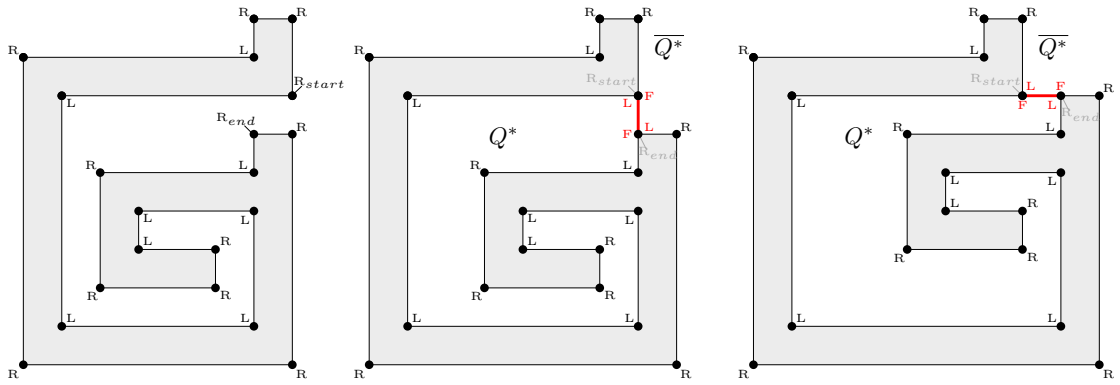


Fig. 4.10: Illustration of an edge inserted on the outer face between two kitty corners. In (b), a vertical edge yields an FL pattern in Q and in $\overline{Q}$; in (c), a horizontal edge reverses the two roles. The turn angles are measured with respect to the outer face.

In the following, we aim to analyze the structure of the kitty corners that appear on the outer face of P_i, i.e. we want to identify all existing pairs of kitty corners. This analysis will later allow us to conclude that, regardless of which edges HeurGI selects, we can always demonstrate suboptimality.

Lemma 4.17. *Every P_i has $6(i-1)$ distinct kitty corner pairs on the outer face.*

Proof. We prove the claim by induction on i. For each P_i, there are six types of kitty corner pairs that can occur on the outer face. For each $j \in \{2, \ldots, i\}$, these six types are defined by combinations of edge endpoints, where $\alpha(\cdot)$ denotes the lower endpoint of an edge and $\Omega(\cdot)$ denotes the upper endpoint. For clarity, we will denote cr_i as or_{i+1} from this point onward, reflecting its role in the recursive construction. The six distinct types of kitty corner pairs are represented by the symbols (pentagon, diamond, circle, cross, square, and star) shown below. Note that P_1 contains no kitty corner pairs. Starting from $i = 2$, each additional step in the construction introduces exactly one new kitty corner pair of each of the six types.

⬠ $(\Omega(\mathrm{cl}_j), \alpha(\mathrm{or}_{j-1}))$	○ $(\alpha(\mathrm{or}_{j+1}), \Omega(\mathrm{ol}_{j-1}))$	□ $(\alpha(\mathrm{ol}_j), \Omega(\mathrm{or}_{j-1}))$
◇ $(\Omega(\mathrm{or}_{j+1}), \alpha(\mathrm{ol}_{j-1}))$	× $(\alpha(\mathrm{or}_{j+1}), \Omega(\mathrm{cl}_{j-1}))$	☆ $(\Omega(\mathrm{ol}_j), \alpha(\mathrm{or}_{j-1}))$

Base Case ($i = 2$). As before, since we are considering the outer face, each L contributes $+1$ and each R contributes -1 to the rotation count. Thus, we analyze the sequence $\hat{P}_2$ shown in the Figure 4.11.

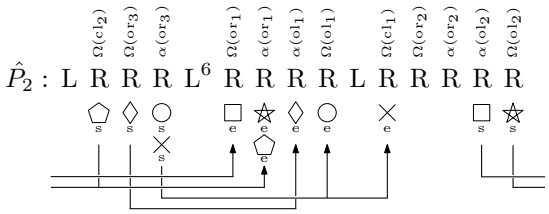


Fig. 4.11: Each kitty corner pair is annotated with a symbol and marked by its start (s) and end (e) position. The arrows indicate the rotation path on the outer face that connects the start to the end of each pair.

Since $\mathrm{rot}(s, e) = 2$ in all six cases, each of the marked pairs indeed forms a valid kitty corner pair. Note that no more than these six kitty corner pairs can exist on the outer face of P_2.

Inductive Hypothesis. Assume for some $i \geq 2$, that P_i has exactly $6(i-1)$ distinct kitty corner pairs on the outer face ($i-1$ of each type).

Inductive Step. We now consider P_{i+1}. First of all notice that due to the recursive construction, P_{i+1} is derived from P_i by cutting off the connector c and adding one more winding around P_i; see Figure 4.12.

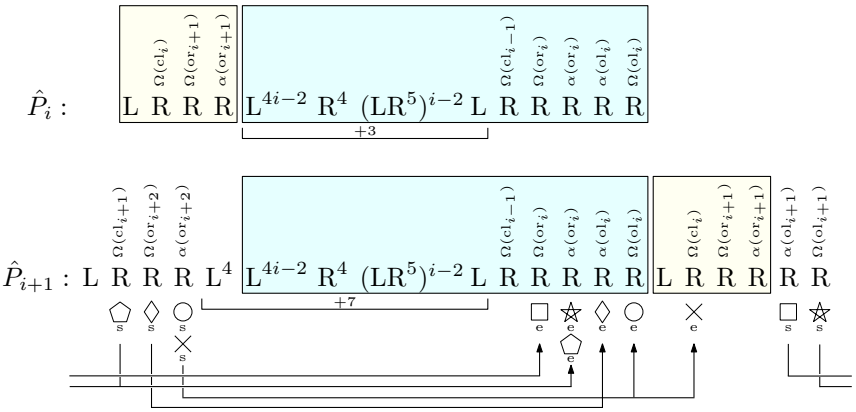
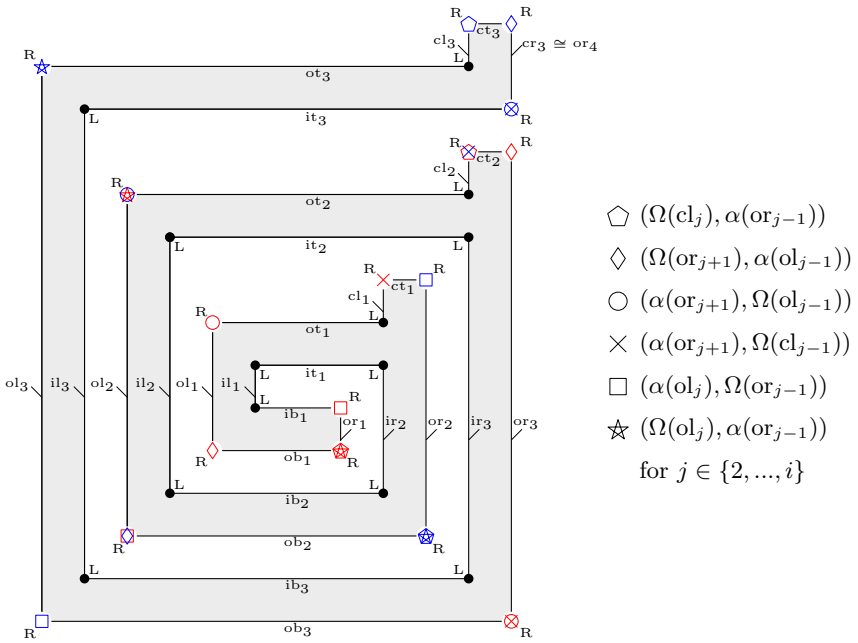


Fig. 4.12: The figure shows $\hat{P}_i$ and $\hat{P}_{i+1}$ as part of the recursive construction, highlighting the new winding added in $\hat{P}_{i+1}$. The blue and yellow blocks indicate repeated segments of the previous layout, showing how kitty corner pairs propagate in the recursive step. Each kitty corner pair is annotated with a symbol and marked by its start (s) and end (e) position. The arrows indicate the rotation path on the outer face that connects the start to the end of each pair.

The rotation sum of the new winding is the same as the rotation sum of c, namely two. Therefore P_{i+1} inherits all $6(i-1)$ kitty corner pairs from P_i. It remains to show that P_{i+1} has exactly six new kitty corner pairs (in relation to P_i). This is obvious, since

the six new kitty corner pairs are again annotated as in the base case and each rotation sum is two. □



(a)

Fig. 4.13: Illustration of the six types of kitty corner pairs in P_3, with those corresponding to $j = 2$ highlighted in red and those with $j = 3$ in blue. The turn angles are measured with respect to the outer face.

We give a visual example of the discussed kitty corners on the outer face in Figure 4.13.

Assignment of length values. In analogy to our approach for HeurRR, we now provide a lower bound on the area after applying HeurGI. Since we have shown that there are six classes of kitty corner pairs on the outer face, and that inserting either a vertical or horizontal edge between any pair from any class eliminates all such pairs, we now analyze the area of the resulting drawing. This analysis is based on the choice of a specific pair on the outer face, taking into account the refinement of both the inner and outer faces. Recall that HeurGI inserts horizontal or vertical edges between kitty corner pairs randomly. For the purposes of our area analysis, we will consider only vertical edge insertions. It is straightforward to verify that this choice does not significantly affect the overall area. In total, $i + 1$ edges are inserted into P_i: i edges in the inner face and one in the outer face. If only horizontal edges were inserted on the inner face, the resulting drawing would gain at most i units in width (in contrast to all edges are inserted vertical) while saving at most i units in height; see Figure 4.9 (b) and Figure 4.10 (c). However, as we will later show that the algorithm produces drawings of width $\Theta(i^2)$ and total area

$\Theta(i^3)$, so this trade-off is asymptotically negligible. The same reasoning applies if one allows a mixture of horizontal and vertical insertions: any gain in one dimension is offset by a corresponding loss in the other, and in light of the asymptotic area growth, such local adjustments do not result in a significant reduction of the total area. Furthermore, flipping the single edge on the outer face from vertical to horizontal would only result in a constant, and therefore negligible, change.

We now analyze the minimal required width and height of any layout of this instance after applying HeurGI. We restrict our attention to the ✯-class of kitty corner pairs for the outer face, where all refinement edges are inserted vertically. As before, this is done by identifying non-overlapping horizontal and vertical regions, each demanding a dedicated unit of space.

Lemma 4.18. *When connecting two kitty corners* $(\Omega(\mathrm{ol}_j), \alpha(\mathrm{or}_{j-1}))$ *of the ✯-class, all* ot_j *are horizontally disjoint for all* $j \in \{1, \ldots, i\}$.

Proof. This follows from Lemma 4.14 and the fact that by adding the vertical or horizontal edge $(\Omega(\mathrm{ol}_j), \alpha(\mathrm{or}_{j-1}))$ for $j \in \{1, \ldots, i\}$, ot_{j-1} is always forced to lie to the left of ot_j. □

Lemma 4.19. *When connecting two kitty corners* $(\Omega(\mathrm{ol}_j), \alpha(\mathrm{or}_{j-1}))$ *of the ✯-class, the vertical chain* ol_{j-1} *always lies above all vertical chains* ol_k *with* $k > j-1$ *for all* $j \in \{1, \ldots, i\}$.

Proof. It is easy to see that ol_{j-1} has to lie above ol_j. Due to the snail-property, all higher indexed ol_k have to lie to the right and below ot_j, which yields the claim. □

Lemma 4.20. *The minimal width of the drawing of the* i*-th snail produced by* HeurGI*, where the algorithm chooses the star-class of kitty corners on the outer face to be connected by a vertical line and all refinement edges are inserted vertically, is at least* $2i^2 + 9i - 6$.

Proof. Consider the following lower bounds for the non-overlapping horizontal regions for some P_i (with $i \geq 2$):

$$\mathcal{H}_1 : \quad \sum_{j=1}^{i} \ell(\mathrm{ot}_j) \tag{4.31}$$

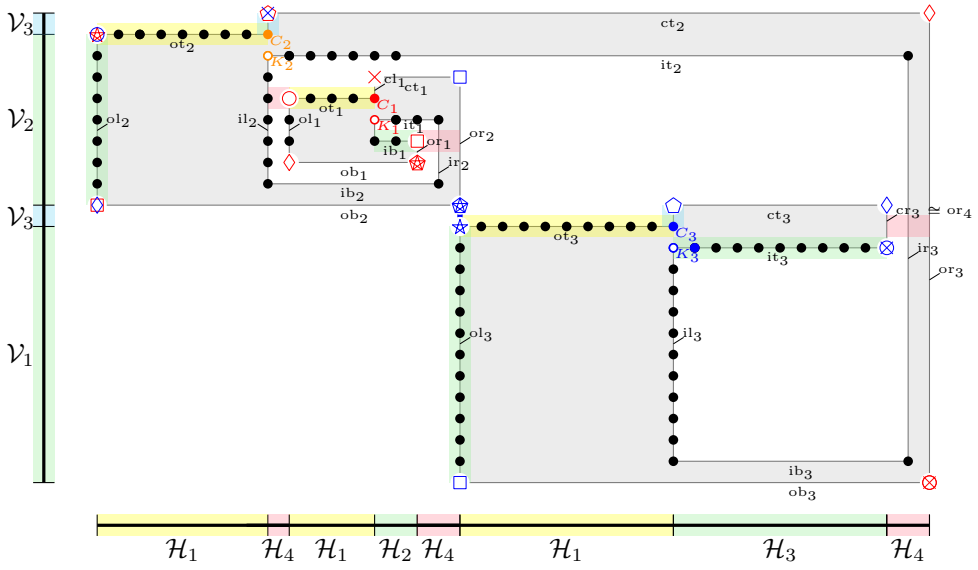
$$\mathcal{H}_2 : \quad 2 \tag{4.32}$$

$$\mathcal{H}_3 : \quad \ell(\mathrm{it}_i) \tag{4.33}$$

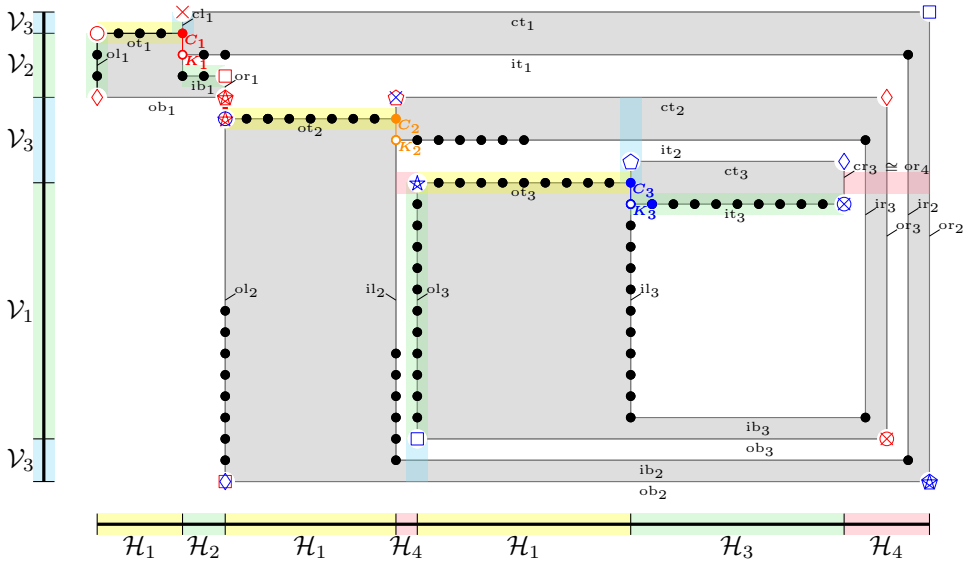
$$\mathcal{H}_4 : \quad 3i - 4 \tag{4.34}$$

Explanations:

$\mathcal{H}_1$ Region $\mathcal{H}_1$ contains all horizontal chains ot_j for $j \in 1, \ldots, i$. This follows from the fact that the vertical edges in the inner face strictly separate the chains ot_j from the chains it_j, and thus also from each other; see Lemma 4.14 and Lemma 4.18.



(a)



(b)

Fig. 4.14: (a) Drawing produced by HeurGI for the instance P_3, where all edges are inserted vertically and the ✯-class of kitty corner pairs is chosen on the outer face for $j = i = 3$, i.e. $(\Omega(\mathrm{ol}_3), \alpha(\mathrm{or}_2))$. Each horizontal region (labeled $\mathcal{H}_1$ through $\mathcal{H}_4$) corresponds to a lower bound contribution to the overall width, while each vertical band (labeled $\mathcal{V}_1$ through $\mathcal{V}_3$) contributes to the lower bound of the height. (b) The same for $j = i - 1$, i.e. the kitty corner pair $(\Omega(\mathrm{ol}_2), \alpha(\mathrm{or}_1))$ on the outer face.

$\mathcal{H}_2$ The chain ib_1 always contributes to the width and $\ell(\mathrm{ib}_1) = 2$. This is because ib_1 is always non-overlapping and in between ot_1 and ot_j when connecting $(\Omega(\mathrm{ol}_j), \alpha(\mathrm{or}_{j-1}))$.

$\mathcal{H}_3$ The chain it_i always contributes to the width and $\ell(\mathrm{it}_i) = 4i - 2$; see Lemma 4.14.

$\mathcal{H}_4$ When adding the edge $(\Omega(\mathrm{ol}_j), \alpha(\mathrm{or}_{j-1}))$ we get $i - j + 1$ windings to the right of it_i each contributing a width of 2. We also get $j - 2$ windings to the right of ib_1, each contributing a width of 2. At last, we have $i - 2$ gaps with width 1 between the spirals (see Figure 4.14).

Combining all the above regions, we obtain a global lower bound on the required width for P_i as the sum of the individual contributions. With a slight abuse of notation, we now write $\mathcal{H}_1, \ldots, \mathcal{H}_4$ to denote these lower bounds, although they originally refer to the corresponding regions.

$$\text{min width} \geq \mathcal{H}_1 + \mathcal{H}_2 + \mathcal{H}_3 + \mathcal{H}_4 \tag{4.35}$$

$$= \sum_{j=1}^{i} \ell(\mathrm{ot}_j) + \ell(\mathrm{ib}_1) + \ell(\mathrm{it}_i) + 3i - 4 \tag{4.36}$$

$$\left[\text{since } \ell(\mathrm{it}_j) = \begin{cases} 4j-1 & \text{for } j < i \\ 4j-2 & \text{for } j = i \end{cases}, \quad \ell(\mathrm{ot}_j) = \begin{cases} 4j & \text{for } j < i \\ 4j-2 & \text{for } j = i \end{cases}\right]$$

$$= 2i^2 + 9i - 6, \tag{4.37}$$

which yields the claim. □

Lemma 4.21. *The minimal width of the drawing of the i-th snail produced by* HeurGI*, where the algorithm chooses the star-class of kitty corners on the outer face to be connected by a vertical line and all refinement edges are inserted vertically, is at least* $10i-7$.

Proof. Consider the following lower bounds for the non-overlapping vertical regions for some P_i (with $i \geq 2$), where j indicates which pair is chosen, i.e. $(\Omega(\mathrm{ol}_j), \alpha(\mathrm{or}_{j-1}))$ for $j \in \{2, \ldots, i\}$:

$$\mathcal{V}_1 \quad \ell(\mathrm{ol}_i) \tag{4.38}$$

$$\mathcal{V}_2 \quad \ell(\mathrm{ol}_{j-1}) \tag{4.39}$$

$$\mathcal{V}_3 \quad 5(i - j) + 2 \tag{4.40}$$

Explanations:

$\mathcal{V}_1$ Region $\mathcal{V}_1$ must include the vertical chain ol_i.

$\mathcal{V}_2$ Region $\mathcal{V}_1$ includes the vertical chain ol_{j-1}; see Lemma 4.19.

$\mathcal{V}_3$ When adding the edge $(\Omega(\mathrm{ol}_j),\ \alpha(\mathrm{or}_{j-1}))$ we get $i-j$ windings below ol_i each contributing a height of 2. We also get $i-j$ windings above ol_i each contributing a height of 3. At last, we have cl_{j-1} and cl_i each contributing a height of 1.

Combining all the above regions, we obtain a global lower bound on the required height for P_i as the sum of the individual contributions. With a slight abuse of notation, we now write $\mathcal{V}_1, \ldots, \mathcal{V}_3$ to denote these lower bounds, although they originally refer to the corresponding regions.

$$\begin{aligned} \text{min height} &\geq \mathcal{V}_1 + \mathcal{V}_2 + \mathcal{V}_3 && (4.41)\\ &= \ell(\mathrm{ol}_i) + \ell(\mathrm{ol}_{j-1}) + 5(i-j) + 2 && (4.42)\\ &\qquad \Big[\text{since} \quad \ell(\mathrm{ol}_j) = 5j-2 \quad \text{for } j \in \{1,\ldots,i\}\Big] \\ &= 10i - 7, && (4.43) \end{aligned}$$

which yields the claim. □

As illustrated in Figure 4.14, the introduced lower bounds are indeed tight for the length assignment generated by HeurGI, when the heuristic chooses vertical segments and the ✯-class. By analogous calculations, one can show that for all choices of kitty corners on the outer face, the width of the drawing grows in $\Theta(i^2)$, while the height grows in $\Theta(i)$. We omit the full proof here; however, based on the earlier considerations regarding vertical versus horizontal insertion and the illustrative analysis of the pentagram-class kitty corners on the outer face, the result follows.

Corollary 4.22. *The drawing area of P_i produced by the refinements of* HeurGI *is in $\Theta(i^3)$, with width in $\Theta(i^2)$ and height in $\Theta(i)$.*

We can now prove the suboptimal behavior of HeurGI by comparing the optimal area of P_i with the area required by applying HeurGI to P_i.

Theorem 4.23 (Area Gap of HeurGI on P_i)**.** *For the family of instances P, the quotient between the area required by* HeurGI *and the optimal area is $\Theta(i)$; that is,*

$$\frac{\mathcal{A}_{\text{HeurGI}}(P_i)}{\mathcal{A}_{\text{Opt}}(P_i)} = \Theta(i). \tag{4.44}$$

Proof. The assertion is an immediate consequence of Lemma 4.20, Lemma 4.21 and Theorem 4.4. □

5 Challenges in Establishing APX-Hardness

As mentioned in the introduction, no efficient approximation techniques for OC are known so far. This raises the natural question whether the problem is APX-hard. In this chapter, we discuss challenges in establishing such a hardness result, outline proof ideas, and analyze their limitations.

5.1 Motivation

As demonstrated in Chapter 4, the choice of kitty corners can lead to arbitrarily poor compaction outcomes. In particular, we have shown that both proposed heuristics, HeurRR and HeurGI, may perform arbitrarily badly in the worst case. The absence of any known approximation algorithm, combined with the possibility of arbitrarily poor compaction under different choices, suggests that orthogonal compaction might indeed be APX-hard. While both designing an approximation algorithm and proving APX-hardness remain unresolved, our focus here lies on the latter. This choice is motivated by two main considerations. First, although OC has been known to be NP-hard for more than 25 years, no approximation scheme or constant factor algorithm has been found even for very restricted cases. This long-standing gap suggests that the difficulty comes from the structure of the problem itself rather than from a lack of clever algorithmic ideas. Second, proving APX-hardness would not only help to explain why existing heuristics and approximation approaches fail but would also define a clear theoretical limit on what any polynomial time algorithm can achieve. Therefore, before spending more effort on designing an approximation algorithm, it makes sense to first ask whether such an algorithm could exist at all.

5.2 Proof Attempts

Before we proceed by discussing how we could prove APX-hardness, we recall the NP-hardness proof of Patrignani, who used sliding rectangle gadgets and a reduction from SAT. The Boolean Satisfiability Problem (SAT) asks whether there exists an assignment of true/false values to variables that makes a given Boolean formula evaluate to true. It is a fundamental NP-complete problem widely used for reductions in computational complexity proofs.

Theorem 5.1. *OC is NP-hard.*

Proof sketch. We do not formally proof every detail but give a sketch of the proof introduced by Patrignani [Pat01] similar to Didimo et al. [DGK+26]. The key concept is

a reduction from SAT to OC. Given a SAT instance Φ with n variables and m clauses, Patrignani specified how one can create an orthogonal representation H_Φ admitting an orthogonal grid drawing of size $w_\Phi \cdot h_\Phi$ if and only if Φ is satisfiable.

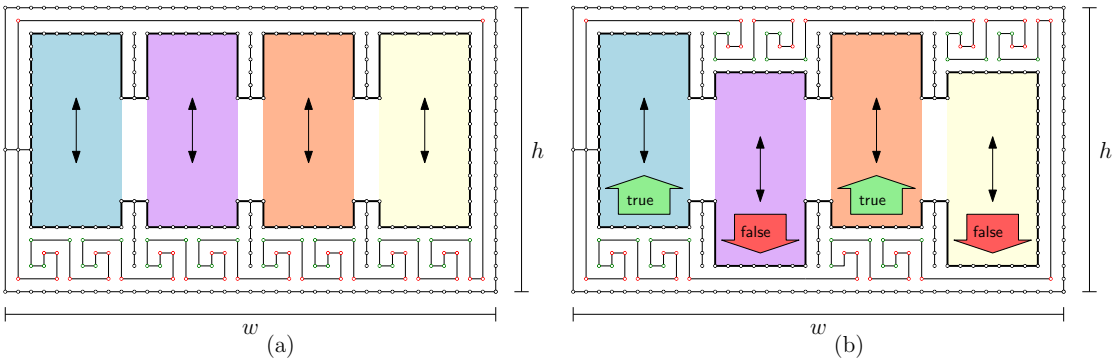


Fig. 5.1: (a) The sliding rectangle gadgets introduced by Patrignani [Pat01] together with the belt (red and green vertices) and a bounding frame. (b) Illustration of the role of each sliding rectangle: when the corresponding variable is set to true, the rectangle slides upward; if the variable is false, it slides downwards. The belt ensures that an intermediate position is suboptimal.

Every variable is represented by a sliding rectangle gadget inside a frame (see the colored rectangles in Figure 5.1). The core insight of the approach is that the sliding rectangles can either be slid upwards (corresponding to the variable being true) or downwards (corresponding to the variable being false). No intermediate position is possible without increasing the overall height. This restriction is enforced by a belt consisting of alternating groups of four convex vertices (red) and four reflex vertices (green) while traversing the belt counterclockwise. Importantly, this belt does not alter the orthogonal representation itself but provides the structural mechanism that enables the rectangles to slide up or down.

Every clause is represented as a chamber through the variable rectangles as illustrated in Figure 5.2 (a). Each variable-clause rectangle is equipped with blocking segments regarding the occurrence of the variable in that clause; see Figure 5.2 (b). Each clause is equipped with a pathway consisting of $(2n-1)$ A's; see Figure 5.2 (c).

The essential observation is that a clause can be drawn efficiently, meaning that each variable-clause rectangle can be realized in the intended size as shown in Figure 5.2 (b), if and only if there is at least one variable that satisfies it. This follows from the fact that we have only $(2n-1)$ A's available, and in each variable–clause rectangle, two A's are consumed whenever the variable does not satisfy the clause or does not appear in it. Only correct assignments can be arranged using just one A; see Figure 5.2 (a) for an example. Note that if, for instance, all variables satisfy the clause, no issue arises, since two A's can still be fitted in a variable-clause rectangle when the assignment is correct. We can thus conclude that drawings that fit perfectly within the $w_\Phi \cdot h_\Phi$ rectangle, that is, efficient and gap-free drawings, are possible if and only if the formula is satisfiable, which completes the reduction. □

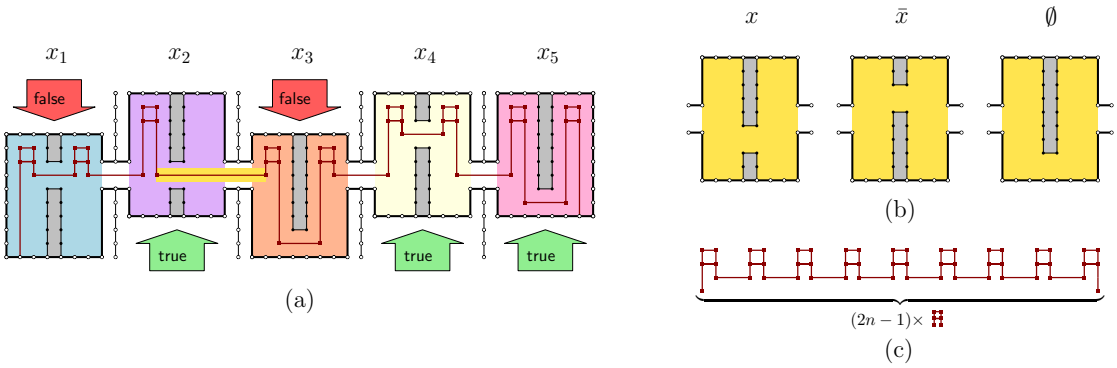


Fig. 5.2: (a) A clause gadget as introduced by Patrignani [Pat01] for the clause $\bar{X_1} \vee X_2 \vee \bar{X_4}$. Each rectangle is a part of the whole sliding rectangle of the corresponding variable. Inside each variable-clause rectangle the corresponding blocker segments are drawn in gray. We highlight the segment that encodes the satisfied assignment of x_2 for the clause (yellow crossbar). (b) The blocker segments depending on whether a variable appears in a clause as positive, negative, or absent. (c) The pathway consisting of $(2n-1)$ A's.

A natural approach to derive an APX-hardness result is to use this NP-hardness reduction as a foundation. Since Håstad has shown that MAX-SAT is APX-hard [Hås01], it is plausible that our compaction problem inherits this property under an appropriate gap-preserving reduction. In this context we aim to achieve the following: If a SAT instance Φ is not satisfiable, every drawing of the corresponding representation H_Φ requires more area than $w_\Phi \cdot h_\Phi$. To establish APX-hardness one would need that each clause requiring repair contributes a multiplicative factor to the total area, rather than an additive increase. By repair we mean that even the best variable assignment which satisfies the maximum possible number of clauses still leaves some clauses unsatisfied. These unsatisfied clauses require additional area, and the corresponding increase in area is what we refer to as repairing. In the following we discuss several approaches and modifications of the reduction introduced by Patrignani.

5.2.1 Copying the Gadgets in Patrignani's Proof

A first idea to strengthen the existing NP-hardness reduction towards APX-hardness is to replicate the construction multiple times, which is a common technique to transform decision problems into optimization problems with provable approximation bounds. One could arrange multiple of the orthogonal representations H_Φ as introduced in Theorem 5.1 in a $k \times k$ grid for a constant parameter k. We illustrate this idea in Figure 5.3. Unfortunately, in the case of OC this approach does not suffice. That is because the area increase required for unsatisfiable SAT instances does not scale proportionally with the instance size. For a SAT formula Φ with n variables and m clauses, the corresponding representation H_Φ has width $O(n)$ and height $O(m)$. As illustrated in Figure 5.4, any unsatisfiable instance can always be repaired by adding an additional area of $O(n)$. This follows from the fact that our objective function measures the total area of the bounding

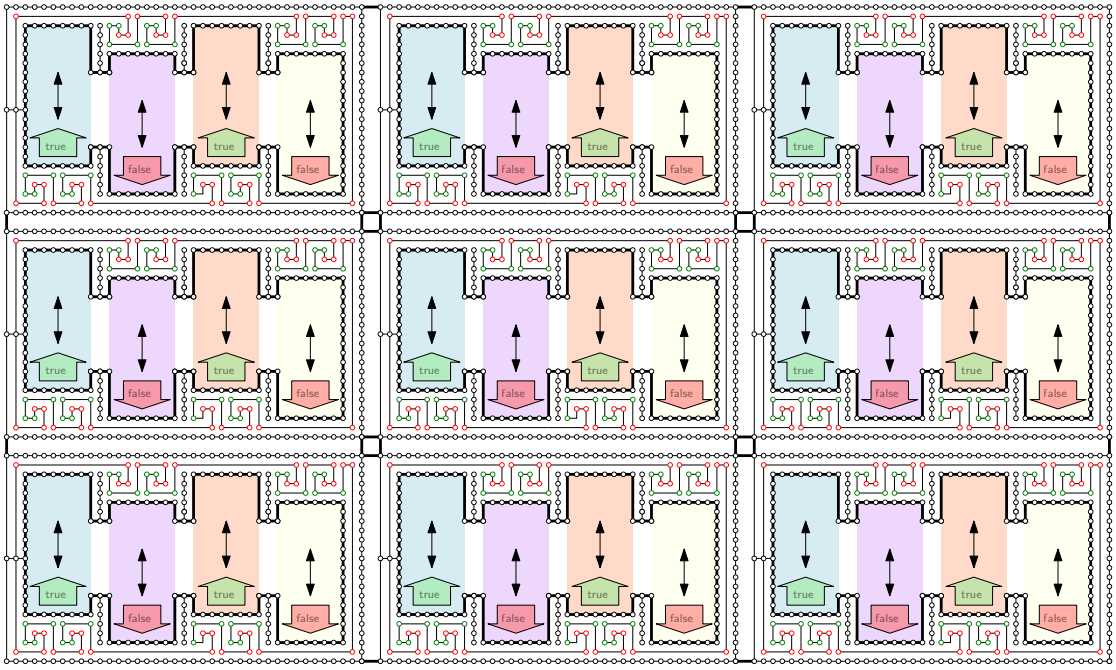


Fig. 5.3: (a) Multiple instances of H_Φ attached together in an 3×3 grid.

box. When a clause is unsatisfied, one has to extend one of the variable–clause rectangles vertically, allowing the clause to proceed with only $(2n-1)$ A's in the pathway. As a consequence, this breaks the belt structure that keeps the sliding rectangles in place. The main observation is that one can satisfy an unsatisfied clause by paying three units of height (exactly the height of the belt-snails) while paying for this height across the entire width of the drawing due to the global nature of the bounding box. Consequently, multiple unsatisfied clauses can be repaired without incurring further cost in terms of additional area, since the vertical space above the sliding rectangles has already been paid for. This observation highlights why the current construction does not yet yield APX-hardness: repairing all unsatisfied clauses requires only $c \cdot O(n)$ additional area for a sufficiently large constant c, which is asymptotically negligible compared to the area $O(nm)$. Even replicating the instance in a $k \times k$ grid does not resolve this imbalance in order of magnitude and as a consequence the total area of the combined instance still dominates the additive increase resulting from potential repairs.

5.2.2 Observation: The Problem with Area-Exploding Gadgets

We proceed by attempting to abstractly describe the properties that a crucial missing structure would need to satisfy. We essentially need the existence of a structure or gadget that can be drawn compactly but is forced, through clever constraints, to "explode" with respect to its area otherwise. In our SAT-reduction we seek a mechanism that (i) admits a compact drawing if the formula is satisfiable, but (ii) produces an "exploding" area

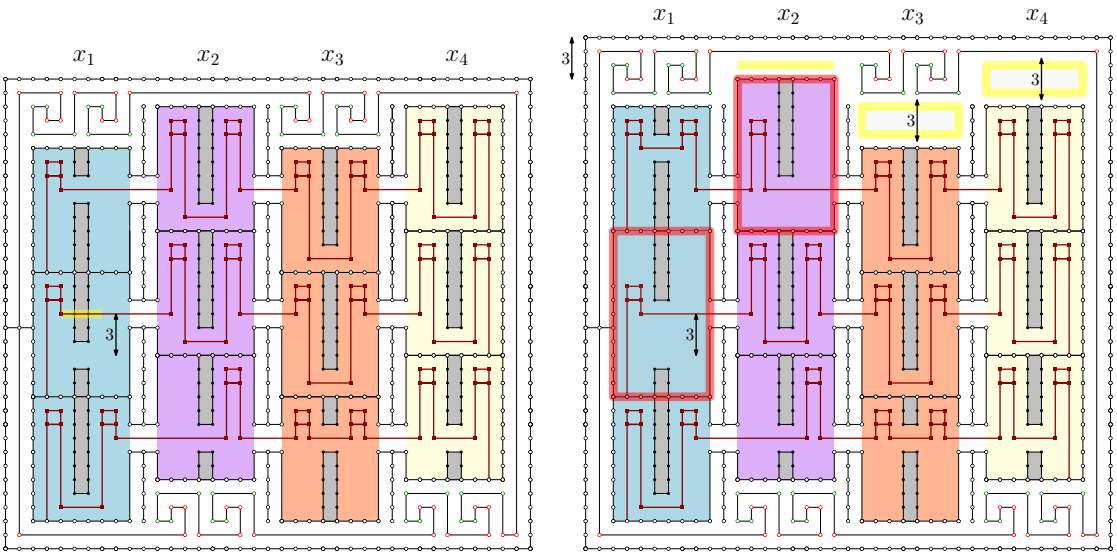


Fig. 5.4: (a) H_Φ for the unsatisfiable $\Phi = \bar{x}_1 \wedge x_1 \wedge (x_2 \vee \bar{x}_3 \vee x_4)$. The shown assignments prevent $C_2 = x_1$ to be fulfilled, highlighted by the absence of the $(2n)$-th A, which would be required. (b) The same repaired H_Φ. Note the extension of the variable–clause rectangle corresponding to x_1 in C_2. The vertical space caused by "height paid across width" is highlighted in yellow. Note that the sliding rectangles can move freely once the belt structure is opened by adding three units of height.

if the formula is unsatisfiable. To illustrate, one can think of a key-lock mechanism: if the SAT formula is satisfiable, the key fits perfectly into the lock. If the formula is unsatisfiable, the key does not fit, which increases the area of the construction roughly proportional to the length of the key or lock; see Figure 5.5 (a) and Figure 5.5 (c).

Here we already address a fundamental limitation. Since every orthogonal representation inherently aims for the smallest possible area, such "exploding" key-lock mechanisms are difficult to achieve. In particular, when introducing additional constraints or blockers, a drawing that minimizes the area behaves "lazily" and always applies the smallest possible adjustment. For example increasing the height in Figure 5.5 (b) by only two units already suffices, eliminating any need for the key to be pushed out of the lock. Abstractly speaking, in an orthogonal representation one can always enforce a minimum size for a structure (for instance, through chains of many vertices). What is missing, is the ability to also enforce upper bounds on the size. This is simply impossible in the current framework, as every representation can "create the space it needs". It is precisely this ability to freely allocate space that prevents the lock from being forced to remain tight and, consequently, from pushing the key outwards.

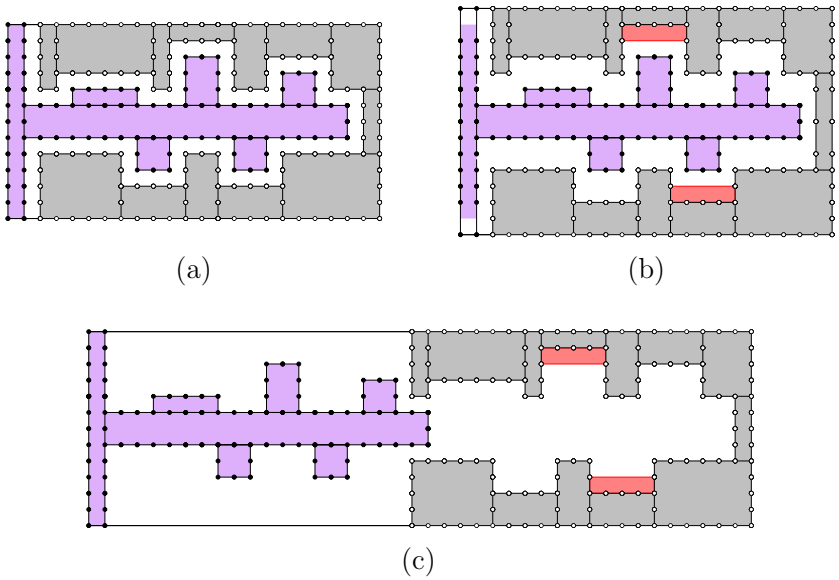


Fig. 5.5: (a) Illustration of a potential key-lock mechanism drawn optimally. (b) The same representation drawn optimally after the insertion of blocker constraints (red rectangles). (c) The intended but not enforceable effect: the key does not fit, potentially forcing a larger area by the key getting pushed outwards.

5.2.3 Mitigating the Problem of Free Space: Snail Curtains

In the following, we try to mitigate the problem that the reparation of unsatisfied clauses creates free space (see empty spaces highlighted in yellow in Figure 5.4). In particular, we need to ensure that each repair operation in one column does not create space to allow other columns to fix unsatisfied clauses with no further area increase. The idea is to construct a structure above the representation H_Φ that behaves like a fluid, which we call *snail curtain*. By this, we mean that if the height of one sliding rectangle is increased vertically, this additional height is distributed evenly across the entire width of the structure and thus potentially increases the height of the drawing with every unsatisfied clause. In other words, if one sliding rectangle needs to be increased by r in height, the total height of the entire construction increases only by approximately r/n, where n denotes the number of variables, i.e. the number of sliding rectangles. To achieve this, we must scale the sizes of the variable–clause rectangles accordingly, since the vertical increase of a single sliding rectangle has to be distributable across the entire width, and r/n must be an integer. Note that we have to adjust the height of the belt as well, since an increased height of the variable–clause rectangles and the resulting larger distance between the true and false positions of the sliding rectangles also requires a higher belt structure. We highlight the rescaling in Figure 5.6.

We now describe the structure of the snail curtain. It consists of $3n$ columns and n rows, built above the construction using the same snail elements as in the belt. The idea is that any local repair in one column pushes exactly n rows and three columns of the

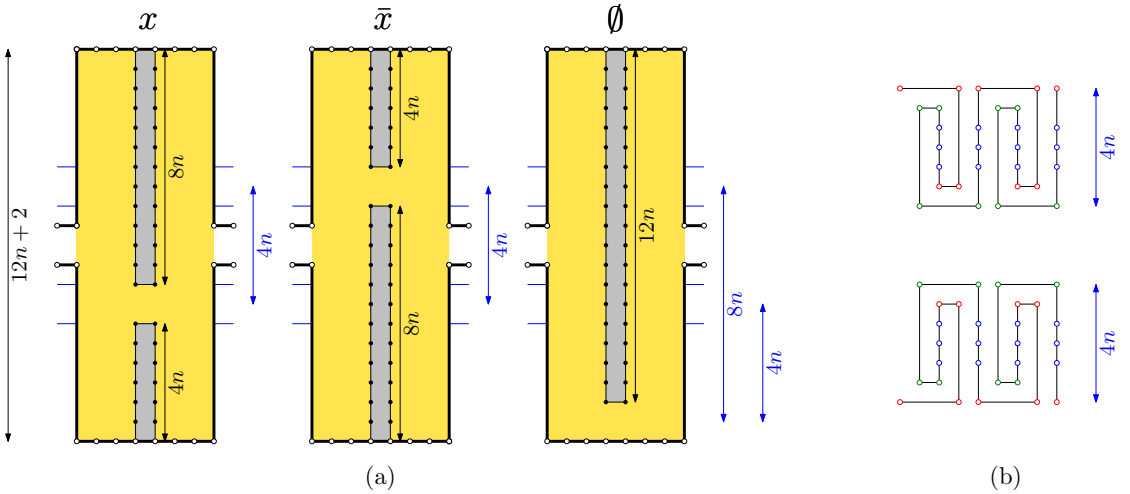


Fig. 5.6: (a) The rescaled versions of the variable-clause rectangles. The blue lines indicate the true and false positions. The height of each of the variable-clause rectangles is in $O(n)$, while the width remains constant size. (b) The rescaled version of the belt by adding $4n-3$ blue scaling vertices between every switch from convex vertices (red) to reflex vertices (green) and vice versa.

snail curtain upwards. This upward shift of n rows can then be evenly redistributed as a single additional row across the entire width of the curtain. Consequently, no empty spaces are created. Also note that we add one additional belt element between each pair of variables to ensure that the same width is always pushed upwards. This modification does not affect the functionality of the underlying construction. We illustrate the fluid behavior of the snail curtain in Figure 5.7.

We have now ensured that each repair operation in one column does not create space to allow other columns to fix unsatisfied clauses with no further area increase. However, one problem remains that we cannot yet control: a literal may appear in many clauses. For example, if x_1 appears positively in all clauses $C_1, \ldots, C_{m/3}$ and negatively in all remaining clauses, then all clauses can be satisfied by sliding the corresponding sliding rectangle downward (i.e., setting the variable to false) and subsequently repairing only the upper third by shifting that part of the rectangle upward. Hence, we cannot control how many clauses can be repaired with one repair. As a consequence, we still cannot enforce a multiplicative increase in area dependent on the number of unsatisfied clauses, since one repair within a single column may still be sufficient. We address this issue and propose a corresponding fix in the following section.

5.2.4 The 3D-Manhattan Problem as a Foundation

Building on Håstad's initial APX-hardness proof of MAX-SAT, many SAT variants have since been analyzed with respect to their approximability. As a foundation, we adopt the SAT variant used in the three dimensional Minimal Manhattan Network Problem (MMN3D) reduction by Muñoz, Seibert, and Unger, who proved that the MMN3D

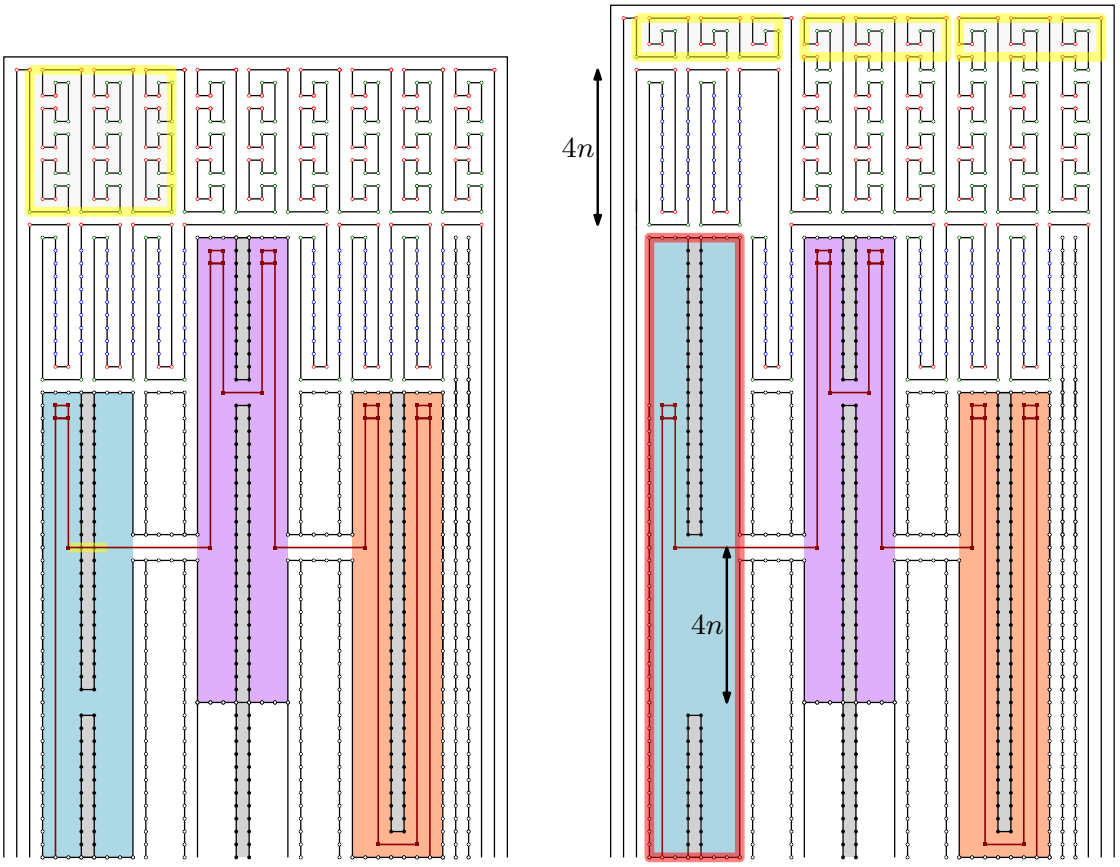


Fig. 5.7: (a) An illustration of the modification of the reduction H_Φ introduced by Patrignani. The modification includes the rescaled variable-clause rectangles and belt, the introduced belt elements and blocker structures between each pair of variables, and the mentioned snail curtain above the construction (here: three rows). (b) The sketched repair of the first variable in the first clause with the even distribution of the pushed snails across the whole width highlighted in yellow.

problem is APX-hard [MSU09]. In their work, they construct a geometric reduction from a SAT variant to the MMN3D problem, employing clause and variable gadgets to achieve a gap-preserving reduction. The MMN3D problem consists of connecting points in space by shortest Manhattan paths, and because those paths always form "staircase-like" connections between points, it can be interpreted as an orthogonal graph in three dimensions. This analogy to our problem suggests that their approach could serve as a natural foundation for our own work. The SAT variant they employed to establish the APX-hardness reduction is (3,B2)-SAT. In this variant, each clause contains exactly three literals, and each variable appears exactly twice positively and twice negatively, for a total of four occurrences. For a natural parameter l, any instance of this type therefore consists of $3l$ variables and $4l$ clauses. We recall the hardness result for the

corresponding MAX-SAT problem of this SAT variant, which was proved by Bermann, Karpinski, and Scott [BKS03].

Theorem 5.2. *There exists a family of MAX-*$(3, B2)$*-SAT instances, containing* $1016l$ *clauses for some* l*, such that it is NP-hard to distinguish between instances where at least* $(1016 - \varepsilon)l$ *clauses can be satisfied and those where at most* $(1015 + \varepsilon)l$ *clauses can be satisfied (for arbitrary small* $\varepsilon < 1/2$*).*

The advantages of this SAT variant are obvious since the resulting reduction is sparse for large instances, as each clause contains only three variables and each variable appears in exactly four clauses. We intend to exploit this property in our own OC reduction. To overcome the mentioned problem of multiple repairs being possible within a single sliding rectangle, and to avoid having both positive and negative occurrences of a variable represented within the same sliding rectangle, we adopt a strategy inspired by the construction used in the MMN3D reduction [MSU09]. In their reduction, each variable is represented by four distinct variable gadgets. Following this idea, we separate the positive and negative occurrences of each variable in our own construction. In particular, for every variable x_j, we introduce two sliding rectangles, X_j and $\bar{X}_j$, which are synchronized by a modification to the belt, to avoid that both the positive and the negative occurrences of one variable satisfy clauses simultaneously. We illustrate this structural idea in Figure 5.8.

5.2.5 Another Rescaling of the Variable-Clause Rectangles

Our ultimate goal is to enforce a multiplicative increase in area for each unsatisfied clause, thereby achieving a gap-preserving reduction from the family of MAX-(3,B2)-SAT instances introduced in Theorem 5.2. Recall that, within this family, it is NP-hard to distinguish between instances in which at most $(1015+\varepsilon)l$ clauses can be satisfied and those in which at least $(1016 - \varepsilon)l$ clauses can be satisfied, for arbitrarily small $\varepsilon < 1/2$. Hence, in the corresponding no-instances, at least εn clauses remain unsatisfied. To transfer this hardness gap to our OC problem, we aim to enforce that each unsatisfied clause necessarily triggers an additional amount of area by forcing the addition of one row of the snail curtain per repair operation. Since we have already modified our construction such that at most two clauses can be repaired within a single column, any no-instance from the family of Theorem 5.2 would still require at least $\varepsilon l/2$ independent repairs, each contributing a proportional increase in total area.

To make this mechanism feasible, one final modification is required. In its current form, the variable–clause rectangles corresponding to empty literals could still repair clauses as well. This would again enable the entire repair process to take place within only a constant number of columns, thereby undermining the intended multiplicative area growth. To prevent such undesired shortcuts, we introduce an additional rescaling step. This rescaling ensures that empty variable–clause rectangles cannot be utilized for repairs, thereby guaranteeing that every necessary fix contributes to the total area increase by enforcing that repairs can only occur through actual variable occurrences within clauses; see Figure 5.9. In particular, we scale our variable-clause rectangles in a

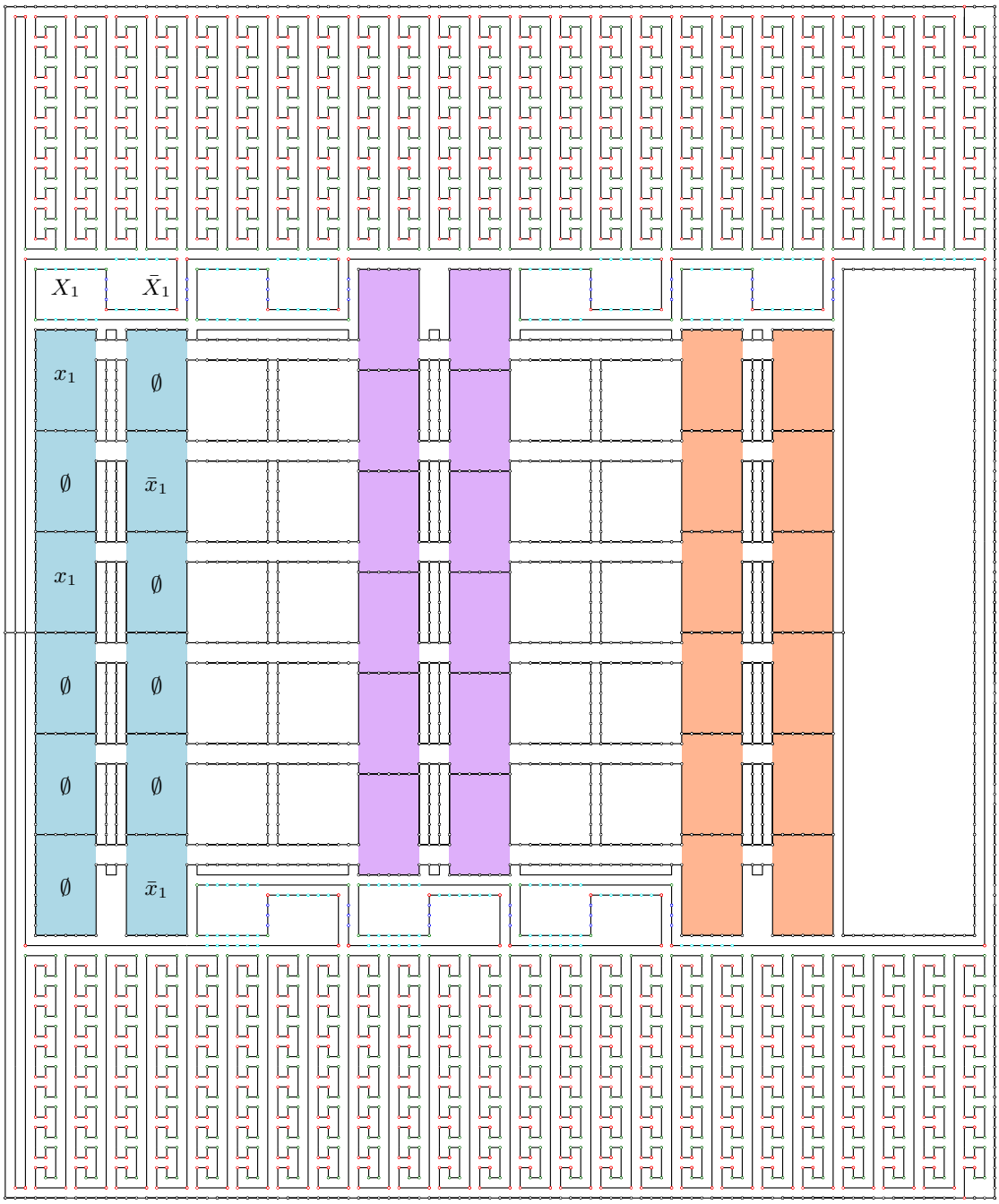


Fig. 5.8: An illustration of the advanced modification of the reduction H_Φ. The modification includes the splitting of each variable into its positive and negative literals. Both sliding rectangles for each variable are synchronized by widening the belt snails similar to the heightening as introduced in Figure 5.6. Note that we use the structure of any (3,B2)-SAT instance which results in any sliding rectangle containing only two variable literals as shown for x_1. Note that the variable-clause rectangles are not drawn to scale.

way that the repair of a variable-clause rectangle that corresponds to an empty literal "costs" at least as much as repairing all the εn unsatisfied clauses only through variable-clauses that correspond to actual literals. Putting all modifications and adjustments

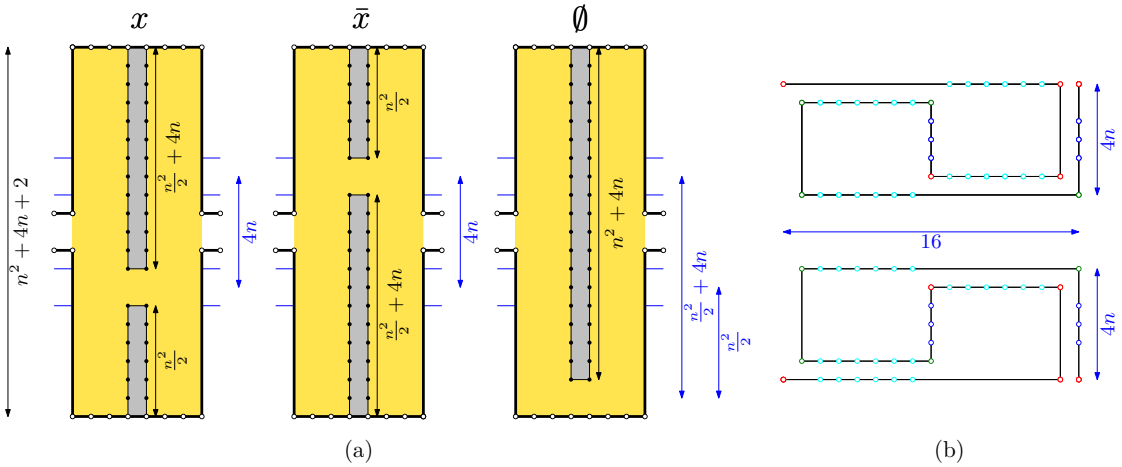


Fig. 5.9: (a) The further rescaled versions of the variable–clause rectangles. The blue lines indicate the true and false positions. The height of each variable–clause rectangle is now in $O(n^2)$, while the width remains of constant size. (b) The correspondingly rescaled version of the belt, obtained by inserting $4n-3$ blue scaling vertices between each transition from convex (red) to reflex (green) vertices, and six light-blue vertices between every pair of consecutive red or consecutive green vertices.

together, we have now achieved that each repair operation which can fix at most two clauses, namely the two associated with one variable column, necessarily adds exactly one additional row to the snail curtain. Consequently, the total area increase of the drawing is directly proportional to the number of unsatisfied clauses. For the family of instances defined in Theorem 5.2, this means that, for a parameter l, at least $\varepsilon l/2$ curtain rows are required in the No-instances. Each such row has a constant height of four units and spans the entire width of the construction.

5.2.6 Limiting Factors

Despite this progress, one crucial limitation remains. Although we successfully achieved a proportional area increase per unsatisfied clause, the order of magnitude in which this increase occurs is too small compared to the overall scale of the construction. In our modified reduction, the height of the entire representation H_Φ for a (3,B2)-SAT instance with $n = 3l$ variables and $m = 4l$ clauses grows asymptotically as $O(m \cdot n^2)$, i.e. $O(l^3)$ in total height. The overall width of the construction remains in $O(n) = O(l)$, resulting in a total area of $O(l^4)$. However, the area increase induced by unsatisfied clauses is much smaller. Since each repair adds one curtain row of constant height spanning the full width, the total additional area grows only with the number of required repairs, that is, in $O(\varepsilon l \cdot m) = O(l^2)$. Thus, even though we obtain a clear distinction between satisfiable and unsatisfiable instances, the resulting gap in total area is asymptotically

negligible since the additive increase of $O(l^2)$ is dominated by the overall area of $O(l^4)$. In other words, while we can conceptually relate the number of unsatisfied clauses to an area increase, the geometric scaling of our construction is too coarse to yield a significant multiplicative gap. To establish APX-hardness, the area increase would need to grow on the same asymptotic scale as the total area itself, i.e, in $O(l^4)$. Achieving this remains the central challenge for any future refinement of the reduction. We now analyse precisely where and why the loss of order of magnitude occurs in the modified construction.

The first source of cost in order of magnitude stems from the second rescaling step. By inflating the variable–clause rectangles to height $\Theta(n^2)$ we force any attempt to repair an $\varnothing$-variable–clause rectangle to be as expensive as repairing all clauses, and thus such a repair will never be performed in an optimal drawing. This rescaling successfully removes one class of undesirable shortcuts, but it does so at the price of increasing the relevant height scale by a factor of $\Theta(n)$. This local increase in clause height does not translate into a corresponding growth of the total area gap we aim to create. In other words, the global area measure does not aggregate these costs in the way we need. However, we believe that this particular imbalance in order of magnitude can be resolved. Previously, we argued that it is necessary to prevent $\varnothing$-variable–clause rectangles from repairing unsatisfied clauses too cheaply. To achieve this, we enforced that repairing via such a rectangle should be as expensive as repairing all unsatisfied clauses through variable–clause rectangles corresponding to actual variable occurrences. In the following, we sketch a modification to the construction that addresses this issue without inflating the height, but instead by increasing the width by one order of magnitude. The underlying idea is that this additional growth in width directly contributes to the repair process, since each repair introduces one snail row spanning the entire width. Although we do not provide a formal proof here, we outline the concept. The modification concerns both the variable–clause rectangles and the connecting pathways. Recall that we split the variables into two columns representing positive and negative occurrences, resulting in $2n$ sliding rectangles and, consequently, $2n - 3$ $\varnothing$-variable–clause rectangles due to the usage of (3,B2)-SAT instances. We now adjust each clause by increasing the number of A's per clause to $12n - 8$. An $\varnothing$-variable–clause rectangle requires two A's regardless of the truth assignment. In contrast, a variable–clause rectangle corresponding to an actual variable occurrence consumes $4n - 2$ A's when the clause is unsatisfied, but only two when it is satisfied. Since each clause is limited to $12n - 8$ A's, not all variable occurrences can remain unsatisfied simultaneously. Even repairing all $\varnothing$-variable–clause rectangles does not suffice. Consequently, each unsatisfied clause is forced to be repaired through the variable occurrences it actually contains. We illustrate this principle in Figure 5.10. As mentioned, the increase in order of magnitude in width introduced by this modification contributes to every scaling factor, thereby resolving one of the two asymptotic imbalances that occurred. What remains is the second imbalance, which, unfortunately, appears unresolvable to the best of our knowledge.

The second source of cost in order of magnitude stems from the global nature of the bounding-box objective. Once the bounding box has been enlarged vertically in one column, all other columns may use that vertical slack at no further cost. Particularly, a single repair increases the height of a column by about $4n$ (in our scaled geome-

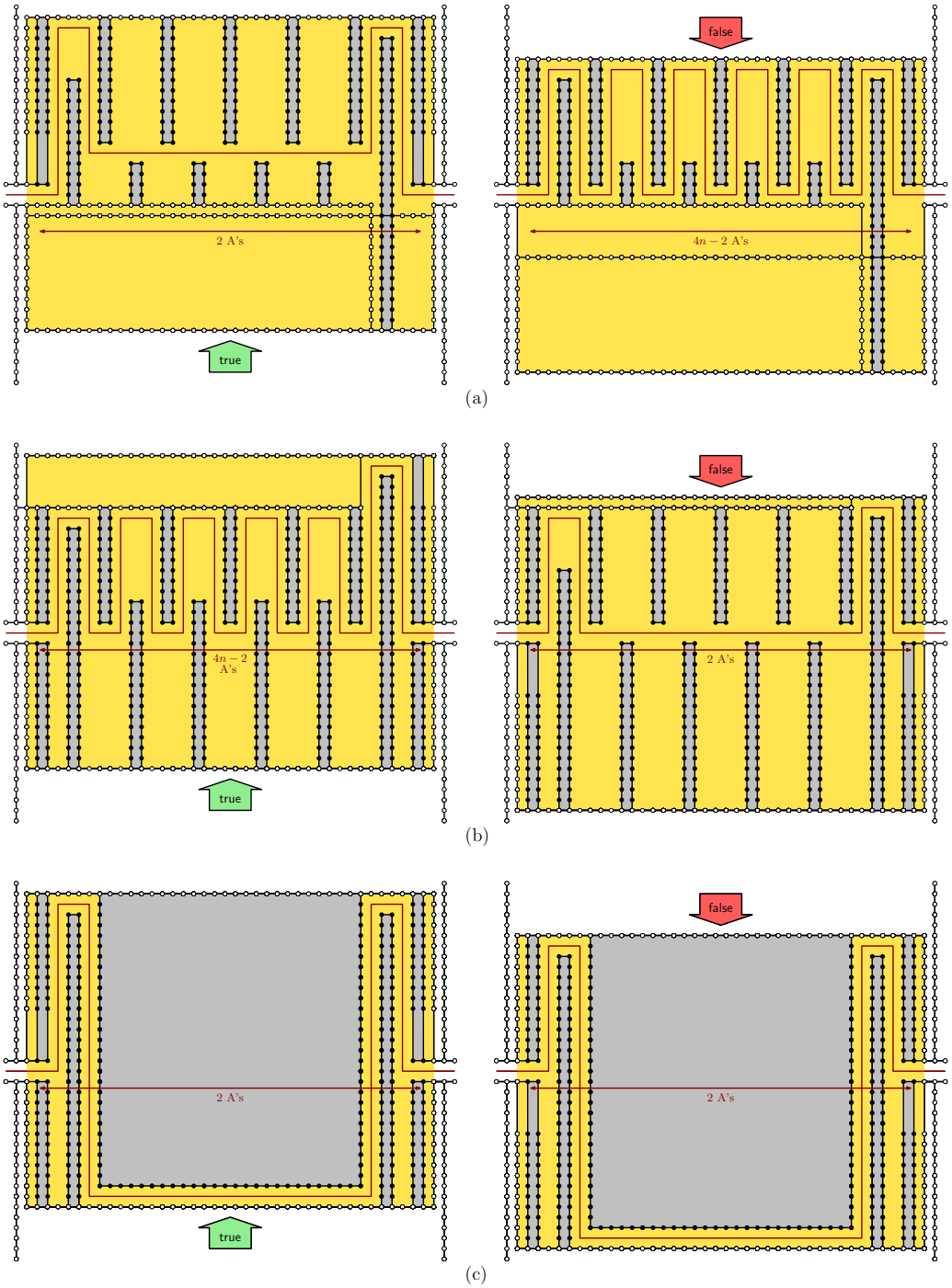


Fig. 5.10: (a) A variable-clause rectangle for a positive literal when the corresponding variable is set to true (left) or false (right). (b) The same for a negative literal. Notice that in the case of an unsatisfied variable occurrence, the repair consumes $4n-2$ A's, whereas in the satisfied case, only 2 A's are sufficient. (c) The $\varnothing$-variable–clause rectangle, representing a variable that does not appear in the clause.

try), but the snail curtain redistributes this vertical expansion evenly across the whole width (consisting of n variables). As a result, the per-column height increase becomes $\Theta(4n/n) = \Theta(1)$, so the global additive height increase is only constant, and therefore the additional area contributed by many repairs remains asymptotically negligible. This is exactly the point where "one order of magnitude is lost". We denote this phenomenon the *MAX vs. SUM* problem. The bounding-box objective computes essentially a maximum in each dimension, whereas our reduction would require that local repairs sum up to a large global penalty. Consequently, a construction that enforces many individual penalties does not necessarily produce a large total-area gap under the bounding-box metric. In summary, the combination of the snail-curtain redistribution and the bounding-box objective creates a structural bottleneck. Local repairs are smoothed out into constant global height increments, preventing the additive accumulation of penalties required for a gap-preserving reduction. Together with the key–lock phenomenon described above, this observation currently marks the endpoint of the techniques presented in this work. Overcoming it seems to require a fundamentally different mechanism to force local penalties to sum under the global cost. This effect is intimately connected to the key–lock observation discussed earlier. Intuitively, we would like to build gadgets whose areas are tight from both below and above. In other words we aim for constructions that are compact when everything fits, but forced to expand in a way that accumulates across gadgets when some constraints fail. However, in orthogonal representations there is no mechanism to enforce non-trivial upper bounds on local sizes since any gadget may simply expand by the smallest possible amount that repairs the violation, and that local expansion can be reused globally because the bounding box measures only extremes.

6 Challenges in Designing Approximation Algorithms

While the previous chapter focused on potential hardness results, the complementary question concerns the existence of efficient approximation algorithms for OC.

At first glance, OC appears promising for geometric approximation techniques. The problem involves minimizing area under planar embedding constraints, a setting in which local optimization or greedy strategies often succeed. For instance, a greedy algorithm achieves a 2-approximation for the Minimum Manhattan Network problem in the plane [GSZ11], and a 3-approximation in three dimensions [MSU09]. Furthermore, there does exist a PTAS for the euclidean Traveling Salesperson Problem (TSP), based on local dynamic programming and geometric partitioning [Aro98]. These works indicate that comparable geometric strategies may apply to our setting, such as the partitioning principle in the PTAS for the euclidean TSP, grid-based refinements from the Manhattan network literature, or more generally the recursive decomposition techniques underlying many geometric approximation schemes. Conversely, as already pointed out by Patrignani [Pat01], the main difficulty lies in the fact that the property of having minimum area is inherently global, since it concerns the entire drawing and does not necessarily reflect in any of its parts; see Figure 6.1 for an illustration. This constitutes a natural bottleneck, since local optima are not necessarily aligned with the global optimum. We have also shown in Chapter 4 that small local decisions, particularly in the relative

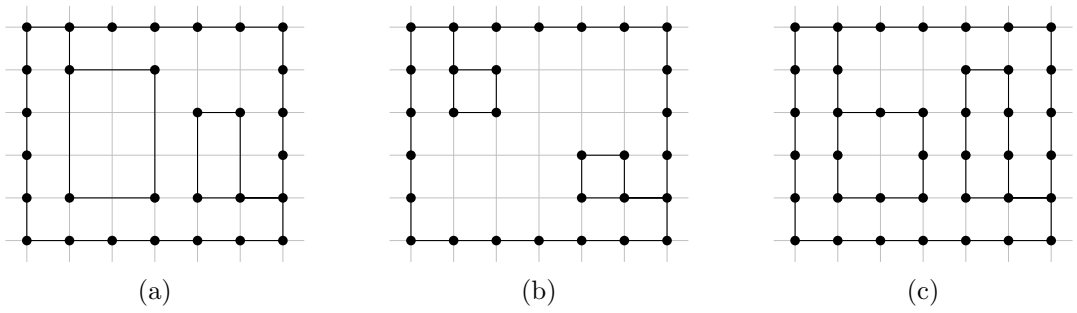


Fig. 6.1: (a) An orthogonal representation with an optimal drawing but not optimally drawn subgraphs. (b) The same representation but with minimum area of the subgraphs. (c) Another representation where the optimal drawing of the subgraphs is indeed necessary for an global optimal drawing.

placement of kitty corners, can lead to globally unbounded errors. We now strengthen this observation in the context of the local–global relationship and demonstrate that even for very small graphs, two locally suboptimal choices may result in a globally opti-

mal outcome, whereas seemingly optimal local choices can produce a suboptimal global result. This observation is illustrated in Figure 6.2.

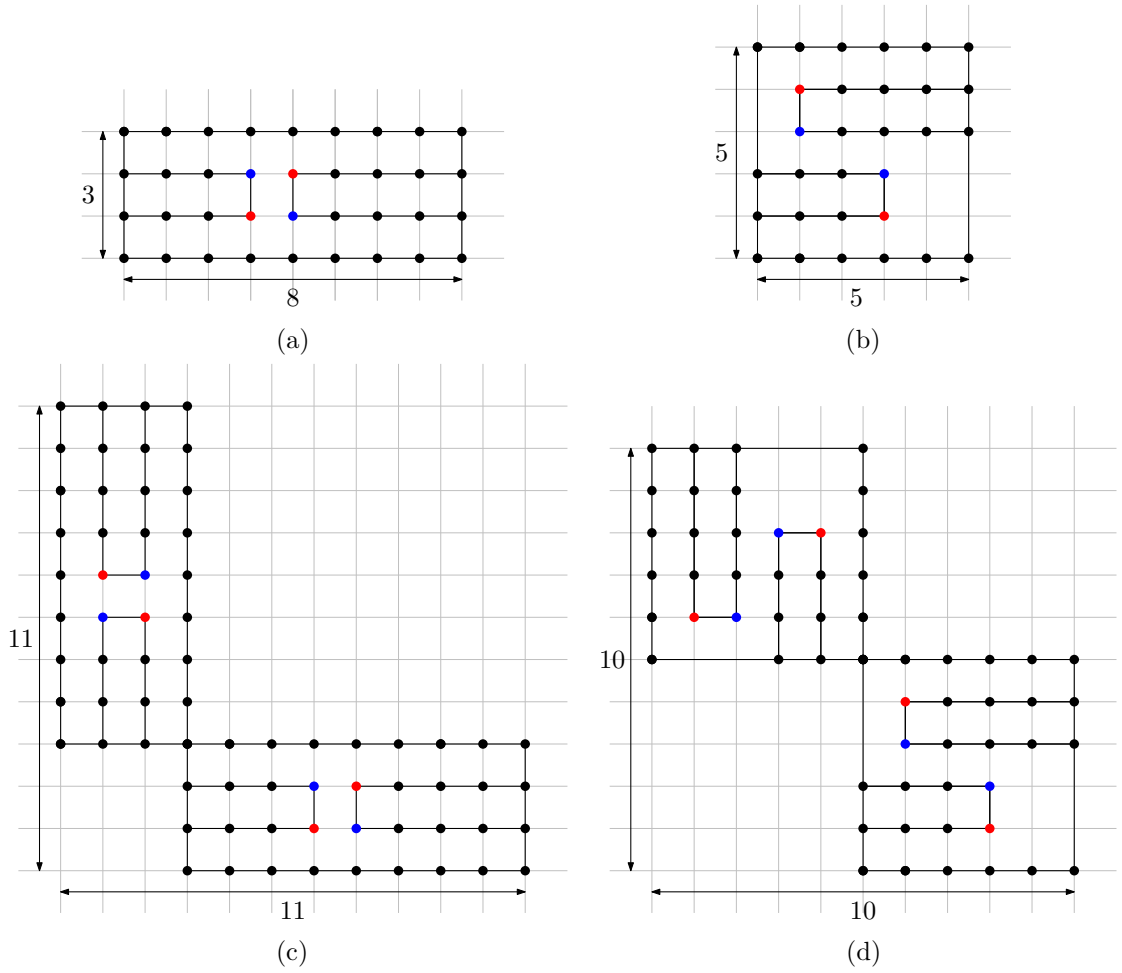


Fig. 6.2: (a) An orthogonal representation with an optimal drawing occupying an area of 24. (b) The same representation with a drawing occupying an area of 25. (c) An orthogonal representation composed of two copies of the subgraph from (a), drawn such that each subgraph is locally optimal. (d) A globally optimal drawing achievable only when the subgraphs are drawn suboptimally. The two kitty-corner pairs responsible for the different local choices are highlighted in red and blue.

In summary, the fundamental obstacle in designing an approximation algorithm for OC lies in the discrepancy between local and global optimality. This structural bottleneck prevents the application of standard local or greedy optimization paradigms. Moreover, the natural parameter driving the complexity of OC are kitty corners, since an exact linear-time algorithm exists for turn-regular representations. Unfortunately, no alternative approach appears more promising than operating on these corner configurations by either fixing them or destroying them (as in the proposed heuristics in Chapter 3).

Together, these two observations, i.e., the global–local discrepancy and the absence of any known structural characterization beyond the turn-regular case, strictly limit the potential for fundamentally different approaches towards designing an approximation algorithm for OC.

7 Orthogonal Compaction in Three Dimensions

In the previous chapters we have established the challenges that come with showing APX-hardness and designing approximation algorithms for the orthogonal compaction problem. This is motivated from the success of the MMN problem, where Muñoz et al. were able to show APX-hardness for the 3D version, while it is still unresolved whether the problem is APX-hard in 2D. Due to the orthogonality constraint, MMN3D can be interpreted as a problem on orthogonal graphs, which makes it conceptually comparable to OC both in 2D and 3D. While MMN in two dimensions is known to be NP-hard [CGS11], neither a PTAS nor APX-hardness has been established. However, an APX-hardness proof exists for the three-dimensional version [MSU09], and constant-factor approximation algorithms are known for both the 2D and 3D variants (see [GSZ11],[MSU09]). This suggests that the additional dimension may provide structural freedom that enables the encoding of richer combinatorial information in reduction gadgets. For orthogonal compaction, moving to three dimensions is a natural generalization, as the problem is inherently geometric.

Let now G be a connected planar graph of maximum degree at most six. A *3D orthogonal drawing* of G places every vertex at a distinct point in three-dimensional integer space and represents each edge uv as a polygonal chain consisting of axis-aligned line segments connecting u and v, without intersecting any other edge. Two 3D drawings of G are considered equivalent if they preserve the relative orientation of incident edges at each vertex along all three axes and maintain the ordering of bends along each edge. An equivalence class of 3D drawings under this relation is called a *3D orthogonal representation*. Every drawing belonging to a 3D orthogonal representation H is referred to as a *drawing* of H. As in the planar case, we assume that H contains no bends by placing a dummy vertex of degree two at each bend if necessary.

The three-dimensional Orthogonal Compaction Problem.

Problem:	Three Dimensional Orthogonal Compaction (OC3D)
Input:	3D orthogonal representation H of a connected graph G.
Question:	What is a three-dimensional coordinate assignment for H that minimizes the volume of its bounding box, i.e., the smallest axis-aligned cuboid containing the drawing?

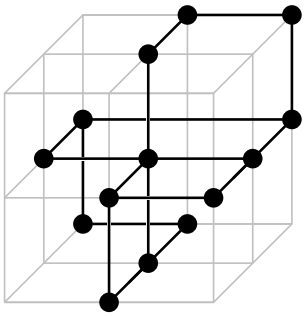


Fig. 7.1: A three-dimensional orthogonal representation with an optimal drawing.

Building on the ideas from the MMN3D case, we also attempted to "lift" Patrignani's proof to three dimensions. As elaborated in Chapter 5, one of the challenges is the lack of structural constraints. For example, fixing one part of the construction may create unintended space that enables further repairs (see Figure 5.4). To counter this, we explored a 3D variant of the 2D construction in which we insert a constant number of parallel layers behind the main plane and connect them in a carefully controlled way (see Figure 7.2 for a sketch of the idea). The goal was to place so-called *rigid blockers* behind each clause, thereby enforcing additional structure in the graph and reducing the freedom available for unwanted repairs.

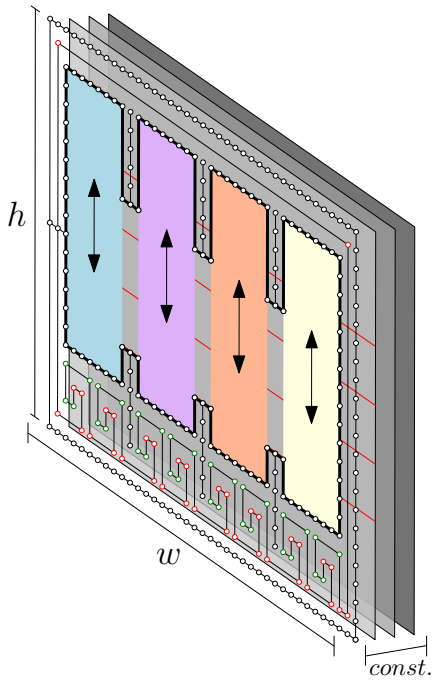


Fig. 7.2: A sketch of a layered construction. The red lines represent rigid blockers, which are shown as simple line segments for clarity but may correspond to more elaborate geometric configurations.

While this layered construction introduces additional geometric rigidity and restricts some repair operations, it does not eliminate the underlying flexibility of the embedding. In other words, the three-dimensional setup does not fundamentally change the nature of the problem. This becomes apparent when contrasted with MMN: although the 3D version offers more geometric space to encode constraints, the core difference between MMN and OC remains the essential bottleneck identified in Chapter 6, namely the discrepancy between global and local optimality. In MMN, every pair of points must be connected by a shortest Manhattan path, which is a strictly local constraint. In contrast, orthogonal compaction optimizes only the total bounding box (or bounding cuboid in 3D), without imposing structural restrictions on subcomponents. Consequently, this structural bottleneck directly carries over from the planar to the spatial case. Together with the observation that the key–lock principle introduced in Section 5.2.2 generalizes to three dimensions without essential modification, we conclude that, with the methods used in this work, extending OC to three dimensions does not provide any direct advantage in overcoming the identified structural limitations in order to show APX-hardness.

8 Conclusion and Future Work

This thesis provides an overview of the approximation perspective on the OC problem. We began by tracing its historical and theoretical development, with a focus on computational complexity. Although OC has been known to be NP-hard for more than 25 years, neither an APX-hardness proof nor a constant-factor approximation algorithm has been established to date. We reviewed the most relevant algorithmic approaches, including the exact linear-time algorithm for turn-regular representations and the two classical heuristics, namely HeurRR and HeurGI.

To examine their limitations, we introduced a family of graphs referred to as snails and demonstrated that both heuristics can perform arbitrarily poorly. The approximation gap between their results and the optimal area can become unbounded, even though the instances consist only of simple cycles with one inner and one outer face.

We revisited Patrignani's NP-hardness reduction from SAT and used it as a conceptual foundation to explore possible extensions toward an APX-hardness proof. To this end, we proposed four modifications to his construction that could preserve an approximation gap from SAT within the OC framework. While these modifications do not yet yield a complete proof, they help to identify the central obstacles. The first is the MAX vs. SUM discrepancy. OC minimizes the bounding box, which depends solely on the extreme coordinates of a drawing, rather than aggregating local costs. The second is the key–lock observation. OC lacks an inherent mechanism to "tighten" structures, as each subgraph essentially creates the space it needs, largely independent of the rest of the drawing. Although the proposed modifications do not fully overcome these challenges, they contribute valuable structural insight. They reveal both the potential and the limits of current reduction techniques and provide a conceptual foundation for future attempts to establish APX-hardness.

In the subsequent discussion, we examined possible directions for designing approximation algorithms. Here, the situation appears similarly constrained. The natural parameter that governs the complexity of OC are the kitty corners. Any known approach to approximation relies on manipulating these configurations, either by fixing or destroying them, as done in existing heuristics. However, since both heuristics can yield arbitrarily poor results, there seems to be no straightforward way to achieve a guaranteed approximation ratio. Another structural challenge lies in the discrepancy between local and global optimality. The minimal area of a drawing is a purely global property, while local improvements do not necessarily translate into better overall solutions. This inherent mismatch strongly limits the applicability of local or greedy optimization paradigms.

Finally, we generalized the problem to three dimensions and discussed whether the additional degree of freedom might help to overcome the structural limitations observed

in the two-dimensional case. Although higher-dimensional analogues, such as the three-dimensional Minimum Manhattan Network problem, exhibit richer structure and known approximation results, the fundamental obstacles of OC persist. The bounding-box objective and the absence of local structural constraints remain the decisive barriers.

Future Work. The insights from this thesis suggest several directions for future research on orthogonal compaction. A central question is whether the approaches explored here could eventually lead to a formal APX-hardness proof. It remains open whether moving to three dimensions is necessary or sufficient to overcome the structural limitations identified, or if entirely new techniques are required. It is also worth investigating whether there exist alternative foundational problems better suited than SAT for reductions, and whether the current approaches can be adapted to transfer an approximation gap.

Another remaining open topic is the development of approximation algorithms or even a PTAS. Could there be new structural or methodological approaches that handle kitty corners differently, allowing the use of the linear-time algorithm for turn-regular representations in novel ways, or even enabling local or greedy strategies? It may also be worthwhile to explore other classes of graphs beyond turn-regular graphs that permit meaningful structural statements about approximation. In addition, it could be useful to look at parameters other than kitty corners, or to change the objective function itself. For example measuring the actual occupied area instead of the bounding box area might help to overcome the MAX vs. SUM limitation identified in this work. A more detailed analysis of the fixed parameter tractable algorithm could also yield tighter bounds or new insights to support the development of improved approximation strategies.

These questions suggest that the orthogonal compaction problem still holds many open challenges, and that progress will likely require a combination of new conceptual ideas, structural insights, and careful algorithmic design.

Bibliography

[Ama24] Amazon Web Services: Read graphs, diagrams, tables, and scanned pages using multimodal prompts in amazon bedrock. aws.amazon.com, 2024. Accessed on October 11, 2025.

[Aro98] Sanjeev Arora: Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998, 10.1145/290179.290180.

[BBD+00] Stina S. Bridgeman, Giuseppe Di Battista, Walter Didimo, Giuseppe Liotta, Roberto Tamassia, and Luca Vismara: Turn-regularity and optimal area drawings of orthogonal representations. *Comput. Geom.*, 16(1):53–93, 2000, 10.1016/S0925-7721(99)00054-1.

[BBLM94] Paola Bertolazzi, Giuseppe Di Battista, Giuseppe Liotta, and Carlo Mannino: Upward drawings of triconnected digraphs. *Algorithmica*, 12(6):476–497, 1994, 10.1007/BF01188716.

[BD05] Carla Binucci and Walter Didimo: Experiments on area compaction algorithms for orthogonal drawings. In *Proc. Canadian Conference on Computational Geometry (CCCG)*, pages 113–116, 2005. `http://www.cccg.ca/proceedings/2005/39.pdf`.

[BES12] Michael J. Bannister, David Eppstein, and Joseph A. Simons: Inapproximability of orthogonal compaction. *J. Graph Algorithms Appl.*, 16(3):651–673, 2012, 10.7155/JGAA.00263.

[BETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999, ISBN 0-13-301615-3.

[BGL+97] Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, and Francesco Vargiu: An experimental comparison of four graph drawing algorithms. *Comput. Geom.*, 7:303–325, 1997, 10.1016/S0925-7721(96)00005-3.

[BK98] Therese Biedl and Goos Kant: A better heuristic for orthogonal graph drawings. *Comput. Geom.*, 9(3):159–180, 1998, 10.1016/S0925-7721(97)00026-6.

[BKS03] Piotr Berman, Marek Karpinski, and Alex D. Scott: Approximation hardness of short symmetric instances of MAX-3SAT. *Electron. Colloquium Comput.*

Complex., TR03-049, 2003. https://eccc.weizmann.ac.il/eccc-reports/2003/TR03-049/index.html.

[BT88] Giuseppe Di Battista and Roberto Tamassia: Algorithms for plane representations of acyclic digraphs. *Theor. Comput. Sci.*, 61:175–198, 1988, 10.1016/0304-3975(88)90123-5.

[CGS11] Francis Y. L. Chin, Zeyu Guo, and He Sun: Minimum manhattan network is np-complete. *Discret. Comput. Geom.*, 45(4):701–722, 2011, 10.1007/S00454-011-9342-Z.

[DGK+26] Walter Didimo, Siddharth Gupta, Philipp Kindermann, Giuseppe Liotta, Alexander Wolff, and Meirav Zehavi: Parameterized approaches to orthogonal compaction. *J. Comput. Syst. Sci.*, 155:103692, 2026, 10.1016/J.JCSS.2025.103692.

[EFK+22] William S. Evans, Krzysztof Fleszar, Philipp Kindermann, Noushin Saeedi, Chan-Su Shin, and Alexander Wolff: Minimum rectilinear polygons for given angle sequences. *Comput. Geom.*, 100:101820, 2022, 10.1016/J.COMGEO.2021.101820.

[GSZ11] Zeyu Guo, He Sun, and Hong Zhu: Greedy construction of 2-approximate minimum manhatten networks. *Int. J. Comput. Geom. Appl.*, 21(03):331–350, 2011, 10.1142/S0218195911003688.

[Hås01] Johan Håstad: Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001, 10.1145/502090.502098.

[KM99] Gunnar W. Klau and Petra Mutzel: Optimal compaction of orthogonal grid drawings. In Gérard Cornuéjols, Rainer E. Burkard, and Gerhard J. Woeginger (editors): *Proc. Integer Programming and Combinatorial Optimization (IPCO)*, volume 1610 of *Lect. Notes Comput. Sci.*, pages 304–319. Springer, 1999, 10.1007/3-540-48777-8_23.

[MSU09] Xavier Muñoz, Sebastian Seibert, and Walter Unger: The minimal manhattan network problem in three dimensions. In Sandip Das and Ryuhei Uehara (editors): *Proc. Algorithms and Computation (WALCOM)*, volume 5431 of *Lect. Notes Comput. Sci.*, pages 369–380. Springer, 2009, 10.1007/978-3-642-00202-1_32.

[Pat01] Maurizio Patrignani: On the complexity of orthogonal compaction. *Comput. Geom.*, 19(1):47–67, 2001, 10.1016/S0925-7721(01)00010-4.

[PKB+19] Mahboubeh Parastarfeizabadi, Abbas Z. Kouzani, Jaclyn Beckinghausen, Tao Lin, and Roy V. Sillitoe: A programmable multi-biomarker neural sensor for closed-loop DBS. *IEEE Access*, 7:230–244, 2019, 10.1109/ACCESS.2018.2885336.

[PT94] Achilleas Papakostas and Ioannis G. Tollis: Improved algorithms and bounds for orthogonal drawings. In Roberto Tamassia and Ioannis G. Tollis (editors): *Graph Drawing, DIMACS International Workshop (GD)*, volume 894 of *Lect. Notes Comput. Sci.*, pages 40–51. Springer, 1994, 10.1007/3-540-58950-3_355.

[Tam87] Roberto Tamassia: On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987, 10.1137/0216030.

[WVV23] WVV: Liniennetzpläne in Würzburg, 2023. `https://www.wvv.de/mobil-b2c/liniennetzplan-wuerzburg/`, Accessed on October 11, 2025.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Würzburg, den 08.12.2025

.....
Dominik Jilg