TU WIEN Informatics

Algorithmen für schematische Repräsentationen

Über die Komplexität der Berechnung schematischer Repräsentationen von Transit Karten, Graphen und geometrischen Objekten

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dipl.Ing. Soeren Terziadis, B.A.
Matrikelnummer 01528974

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Inform. Dr.rer.nat. Martin Nöllenburg

Diese Dissertation haben begutachtet:

William Evans	Sabine Storandt

Wien, 30. Juni 2023

Soeren Terziadis

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Algorithms for Schematic Representations

On the complexity of computing schematic representations of transit maps, graphs and geometric objects

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dipl.Ing. Soeren Terziadis, B.A.
Registration Number 01528974

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Inform. Dr.rer.nat. Martin Nöllenburg

The dissertation has been reviewed by:

William Evans	Sabine Storandt

Vienna, 30th June, 2023

Soeren Terziadis

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Dipl.Ing. Soeren Terziadis, B.A.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe ohne Verwendung von Language Models (wie ChatGPT) oder anderen automatisierten Textgeneratoren, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 30. Juni 2023

Soeren Terziadis

Für Eine

— nach Mascha Kaléko (1934)

Acknowledgements

Throughout my Masters, my PhD and my work at TU Wien, my travels abroad and my personal life, I had the pleasure of meeting a large cast of diverse, fascinating and inspiring people. I don't hesitate when I say that this work wouldn't exist, without them. I want to use this space to acknowledge their influence, help, guidance, friendship, love and support. Therefore this section is not too long, nor is it too short, but precisely as long as it ought to be.

I thank *David Elsweiler*, *Christian Fehrmüller*, *Stefan Podlipnig* and *Martin Kronegger* whose enthusiasm sparked my interest in teaching and research. These thanks extend to my supervisor *Martin Nöllenburg* who has helped and guided me before and during my PhD in finding my way in this community.

I am grateful towards *Marc van Kreveld* and the Geometric Computing group at Universiteit Utrecht, *Giuseppe Liotta* and his group at Università degli Studi di Perugia and *Rodrigo Silviera* and the Research Group on Discrete, Combinatorial and Computational Geometry at Universitat Politècnica de Catalunya for hosting me during my research visits. I would also like to express my gratitude towards my reviewers *Will Evans* and *Sabine Storandt*, for their help- and insightful feedback on the first draft of this thesis.

I thank each and every (past and present) member of the AC group at TU Wien, who were amazing colleagues and endured countless lunch time tangents, in particular *Guangping* and *Alexander* with whom I shared an office, *Jan*, *Vaidyanathan* and *Markus* who were excellent squash partners, *Sanja* my oldest friend at TU Wien and *Jules* and *Anaïs* who, together with Melli and me, formed the best pandemic response team imaginable.

Outside of work I would like to thank *Michael Doskoczynski*, *Michael Stockert* and *Matthis Browa*, *Gwen Rippberger* and the *Dodgeball Ninjas*, and *my parents*, *siblings* and *friends and family* for their friendship and support through these last (at times very busy) years.

This is also were I would like to thank everyone who supported me while compiling this thesis with feedback, pointers and guidance. I am especially thankful to *Fabian Klute*, *Irene Parada* and *Jules Wulms* who managed to find time when really, there was none.

The end of these acknowledgments belongs to the most important person. *Melli* you have been there through everything. From the start of our adventure in Vienna until now – eight years, four courses of study, many walks through the city, countless memories, two names and one wedding later – you have supported, encouraged and believed in me. Thank you! Σ′ αγαπώ!

Kurzfassung

Schematisierungen stellen Daten auf eine Weise dar, die bestimmte Eigenschaften hervorhebt, während andere Aspekte der Daten weniger präzise dargestellt werden. Sie sind ein oft verwendetes Werkzeug, um Daten zu untersuchen, zu verstehen und sie für bestimmte Zwecke leichter zugänglich zu machen. Schematisierungen werden häufig zur Darstellung von Geodaten verwendet um Komplexität zu reduzieren und dadurch Karten und Geovisualisierungen leichter und schneller lesbar zu machen. Dies kann auf verschiedene Arten erreicht werden. So können die Grenzen zwischen Regionen für unterschiedliche Zoomstufen vereinfacht und Details weggelassen werden, die nicht sichtbar wären (z. B. in digitalen Versionen politischer Karten). Oder indem alle geraden Linien einer Karte entlang eines sehr eingeschränkten Satzes vordefinierter Orientierungen ausgerichtet werden (z.B. Liniennetzen wie U-Bahn-Karten). Oder aber durch die Darstellung von Elementen auf einer Karte als simple geometrische Formen um Vergleichbarkeit zwischen Objekten zu erleichtern (z. B. in bestimmten Kartogrammarten), um nur einige zu nennen.

Die Erstellung einer schematischen Darstellung ist für eine:n Designer:in oft eine zeitintensive Aufgabe und kleine Änderungen können ein komplett neues Design erfordern. Es besteht ein steigendes Interesse daran, Teile des Designprozesses durch geeignete algorithmische Lösungen zu unterstützen oder vollständig zu automatisieren. Aus praktischer Sicht ermöglicht eine solche algorithmische Unterstützung, eine größere Menge möglicher Designs zu testen, ohne viel manuelle Arbeit zu verschwenden. Aus theoretischer Sicht zeigt sich jedoch, dass in vielen gängigen Designprozessen schwierige Probleme verborgen sind. Daher müssen automatisierte Ansätze (insbesondere wenn sie darauf abzielen, ein mehr oder weniger fertiges, qualitativ hochwertiges Endergebnis zu liefern) diese Aufgabe durch den Einsatz geeigneter Werkzeuge bewältigen.

Dies kann in Form einer Kodierung des Entwurfsproblems als mathematische Formulierung erfolgen, für die bereits vorhandene und hochoptimierte Solver existieren. Oder man kann versuchen, herauszufinden, welcher Bestandteil einen Prozesses schwer lösen lässt, indem man Eingabeinstanzen einschränkt oder zulässt, dass die Laufzeit nur mit bestimmten Parametern der Eingabe extrem skaliert, von denen in einer realen Anwendung erwartet werden kann, dass sie klein sind.

In dieser Doktorarbeit wollen wir uns verschiedenen schematische Darstellungsproblemen von zwei Seiten nähern. Einerseits untersuchen wir die Komplexität der Lösung, indem wir mithilfe Polynomialzeitreduktionen zeigen, dass ein Problem NP-schwer

oder NP-vollständig ist. Und zum anderen präsentieren wir mathematische Constraint-Programming-Formulierungen, Heuristiken, parametrisierte XP- oder FPT-Algorithmen oder polynomiale Zeitalgorithmen für eingeschränkte Eingabeklassen, die es uns ermöglich trotz der theoretischen Komplexität Lösungen für diese Probleme zu finden.

Abstract

Schematic representations display data in a way that highlights certain properties at the cost of being less accurate for other aspects of the data. They are an ubiquitous tool to explore and understand data as well as making it more immediately accessible for specific purposes. Schematic representations (or schematizations) are commonly used to display geospatial data with the goal of reducing complexity and thereby making the map more readily readable. This can be done by simplifying borders between regions for different zoom levels and omitting detail, which would not be visible (e.g. in digital versions of political maps), by orienting all straight lines of a map along a very restricted set of predefined orientations (e.g. in transit maps like metro networks) or by distorting the shape of elements on a map into simple geometric primitives to facilitate easy identification and comparability between objects (e.g. in some cartograms), to name just a few.

The creation of a schematic representation (geospatial or otherwise) is often a time-intensive task for a designer and small changes, might require large scale redesigns. There has been rising interest in supporting or completely automating parts of the design process with suitable algorithmic solutions. From a practical perspective, any such algorithmic support enables a designer to explore a larger set of possible designs, without wasting a large amount of manual work. However from a theoretical perspective it turn out that in many common design task there are difficult or intractable problems hidden. Therefore automated approaches (in particular when aiming to provide a more or less finished high-quality output) have to tackle these task by employing the appropriate tools.

This might take the form of encoding the design problem as a mathematical formulation, for which already existing and highly optimized solvers can be used. Or one can aim to identify, which part of a problem is the source for the intractability by restricting input instances or allowing the runtime to only scale problematically with certain parameters of the input, which we can expect to be small in a real world setting.

In this thesis we aim to approach various schematic representation problems from tow sides. On one hand we investigate the complexity of solving, by showing that a problem formulation is NP-hard or NP-complete using polynomial time reductions. And on the other hand, we provide mathematical constraint programming formulations, heuristics, parameterized XP- or FPT-algorithms or polynomial time algorithms for restricted input classes, which allow us to find solutions to (some) of these problems, in spite of their theoretical hardness.

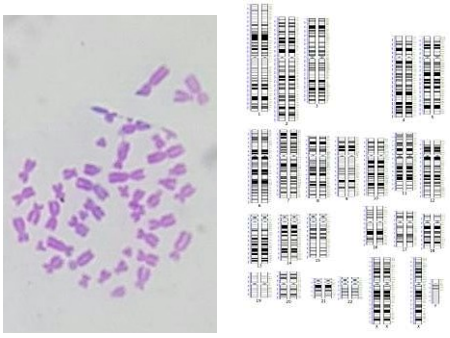
Contents

CHAPTER 1

Introduction

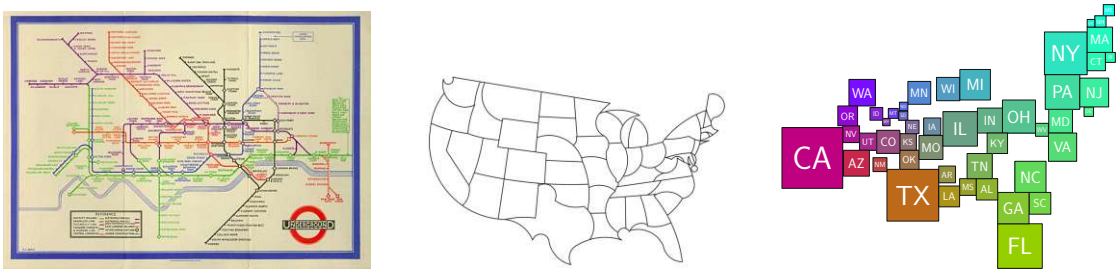
Visualizations are a fundamental tool to understand, explore and work with data. *Schematizations* (sometimes also called diagrams) of data are, in their most general sense, a visualization, which focuses on one or a few aspects of the data, while relaxing how correctly other aspects of the data are represented to enhance the readability of the aspects in focus. While the relaxed aspects are often abstracted and/or distorted, they are usually less important for the purpose of the schematization. What the particular aspect in focus is, is heavily dependent on the specific schematization, which determines the abstracted aspects and the tools used for the abstraction.

For example, Figure 1.1a (Credit: SANCHITA20. Cropped by Mikael Häggström. Used unchanged under CC BY 4.0 License) depicts a photomicrograph of human chromosomes, i.e., a realistic depiction of the actual shape of chromosomes. However a researcher might be interested in the information of the genes encoded in specific bands of such a chromosome. A schematized representation of the set of chromosomes, which sorts the chromosomes and displays them with straight parallel boundaries of similar width is shown in Figure 1.1b (used unchanged under CC0 1.0 license). This schematization emphasizes readability, in particular the order and length of encoded genes on the chromosome, while distorting actual shape and position. This thesis concerns itself with specific schematizations and some of the geometric challenges hidden within their computation. This introduction points to various related publications as illustrative examples. More thorough literature



(a) (b)

Figure 1.1: (a) Photomicrograph and (b) banding diagram of chromosomes.



(a) Harry Becks tube map. (b) Circular arc map (c) Demers cartogram

Figure 1.2: Different examples of schematic maps for geospatial data. (a) A pioneering example of schematic network maps in Henry (Harry) Becks 1933 underground map. The map is drawn in the octolinear style and is a central example for schematic transit maps. Used with permission ©TfL from the London Transport Museum collection. (b) A circular arc schematization of a political map of the United States [44]. (c) A Demers cartogram representing the states of the US with squares, whose size is relative to their population in 2016 [90].

reviews of the topics discussed in the individual chapters are included within the chapters themselves.

A very popular application area of schematization is geospatial data visualization. Geospatial data is data in which data points are associated with geographical features. This data is often visualized as a map. Examples include transit networks maps (Figure 1.2a), which emphasize the connectivity of a given network over the geographic accuracy of the depicted stations in the network, political maps (Figure 1.2b), whose geometrically complex boundaries are often simplified and abstracted using geometrically simple shapes, and some types of cartograms (Figure 1.2c), which abstract the shape of a region entirely by representing them with a singular geometric object.

Schematic representations of transit networks are the focus of the first part of this thesis. Transit networks are present in almost every densely populated area in the world and include bus, tram and metro networks. We will mostly focus on metro networks here (often also called tube or subway system). Common questions passengers try to answer with a schematic map are “how do I get from one station to the next?” and “how long do I have left in my travel?” (a task measured by the number of stations rather than the actual distance) [109]. The archetypical design of a schematic representation of such a transit network to answer such questions was pioneered in the London Underground Map by Henry Beck (more commonly called Harry Beck) in 1933 (Figure 1.2a). His design draws the stations as circles and connects them with lines representing connecting railway tracks. However the connections, which in the actual network can include turns and curves are abstracted to straight line edges restricted to be parallel to one of four directions (horizontal, vertical and the two 45° diagonals). Based on the maximum of eight connections, that are possible for a single station, this design is called octolinear [1].

[1]The nomenclature oct**o**linear is not unique. The alternative word oct**i**linear is based on the word

The metrolines are represented as colored polylines following a path through the network. This design abstracts the geographical positions of stations, both absolute (geographical features like the river Thames are also abstracted if present at all) and relative (distances between adjacent stations are uniform). The uniformity of distances between stations models that distances in the network are usually measured by number of stops. Harry Beck already stated some design criteria and in recent years multiple attempts have been made to formalize these criteria [95, 132, 133].

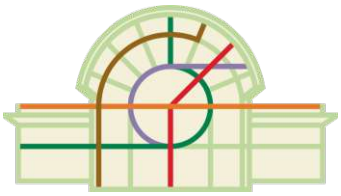
The octolinear design has obvious connections to a well establish research topic in the area of graph drawing, namely orthogonal drawings, which in the same terminology could be called tetralinear drawings (only using horizontal and vertical edges). However while bend minimal topology preserving orthogonal drawings can be computed in polynomial time using flow networks as shown in Tamassia's seminal work [119], the additional two directions make this problem a lot harder. Specifically the problem becomes NP-hard [96], which intuitively means that we do not expect to find an efficient algorithm for this problem, which scales nicely with the size of the input (for a slightly more thorough introduction, we refer to the next section, which introduces theoretical concepts used in this thesis). Since Beck's original design, various alternatives in drawing style have also been investigated. Orthoradial layouts [93, 92, 9] place stations on the intersection of concentric circles and radially emanating rays from their center. Edges are placed either on radial segments of these circles or straight line segments of the rays. Other styles also working with curved elements include curvylinear methods based on Bézier curves [53] and circular arc schematization [67, 61, 44].

A direct generalization of octo-/tetralinear designs is the k-linear design, which allow edges to be parallel to one of k possible directions. They are the focus of Chapter 3 of this thesis.

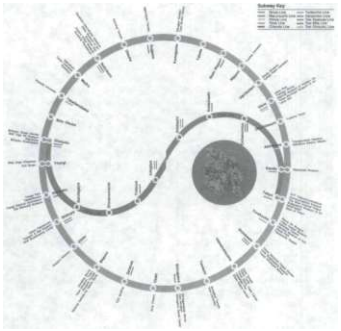
Computational metro map research focuses on creating a layout of a given a network that conforms to a specified drawing style and optimizes some formulated quality criteria which vary throughout different models. Additionally official metro-maps contain other components, that are important for a usable schematic transit map. Some are design elements, which are beneficial for the user (legends, pictographic representations of landmarks, color choices, styling of bends, intersections, interchanges, one-way stations, etc. [133]). These elements are usually added as a post-processing step. Station labels are an important feature, which have been considered as an algorithmic optimization problem either as a secondary step assuming a fixed topological layout of the network [91] or even in the layout process [97]. Various geometric challenges can be hidden in the placement of additional information on a map. For example, Niedermann and Haunert [91] use a conflict graph between labels, encoding overlap between them and then employing maximal independent set algorithms to compute the set of labels that can be displayed.

rect**i**linear drawings (graph drawings with only horizontal and vertical edges). In contrast oct**o**linear is based on the common prefix octo- indicating a relation to the number 8 based on greek word οχτώ. We decided on oct**o**linear for reasons of uniformity, since the corresponding designs using six directions are more commonly called hex**a**linear rather than hex**i**linear, based on the common prefix hexa-.

(a) Moscow metro (Lebedev Studio and ©ГУП «Московский метрополитен»)



(b) SMW 2019 logo (courtesy of Maxwell J. Roberts, tubemapcentral.com)



(c) Japanese train map (credit [104] via [7])[2].

Figure 1.3: Examples of schematic transit maps, which include shapes that (a) do not conform to the chosen style like the circular shapes in the otherwise octolinear metro map of Moscow or (b-c) follow a predefined shape entirely.

Here we want to focus on a particular component of real world metro maps, namely the inclusion of shapes that deliberately go against the defined drawing style. This can be done with the purpose of further abstraction, e.g., drawing circular lines as actual circles within an octolinear layout like in the official Moscow metro map (Figure 1.3a) or for artistic or advertisement reasons, by including logos either partially as in the Schematic Mapping Workshop 2019 logo (Figure 1.3b) or distorting the entire map to a shape, e.g., the stylized train map in the shape of the yin and yang symbol (Figure 1.3c). The inclusion of arbitrary shapes into otherwise style-conforming metro maps is the topic of Chapter 4.

The application of transit map schematization is a large focus of the first part of this thesis, as evidenced by Chapters 3 and 4. However there are other applications of schematization, which are worth investigating in their own right. The second half of this thesis, i.e., Chapters 5, 6 and 7 discuss various such applications. Chapter 5 in particular is of interest here since it falls in both categories, thereby forming a bridge between the topics, since it is concerned with a theoretical question about separating geometric objects using simple geometric separators, but at the same time it can be motivated partially again from transit map schematization, where tariff zones or other area highlighting can be interpreted as a schematic variant of such a geometric separator.

Tariff zones (see Figure 1.4 for an example of tariff/fare zones in the London subway

[2] Avelar and Hurni [7] provide the link http://www.ika.ethz.ch/publications/avelar, which as of the writing of this thesis is not accessible anymore.

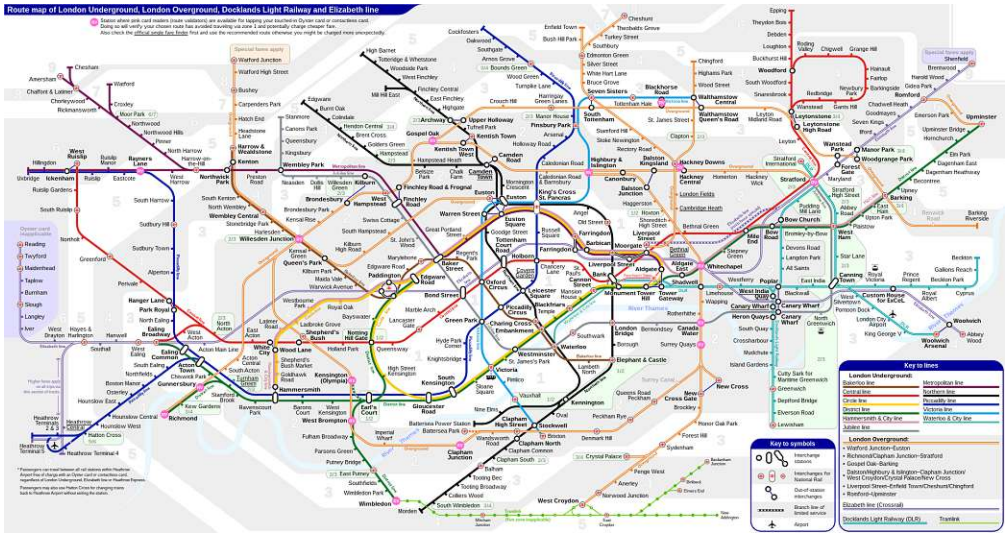
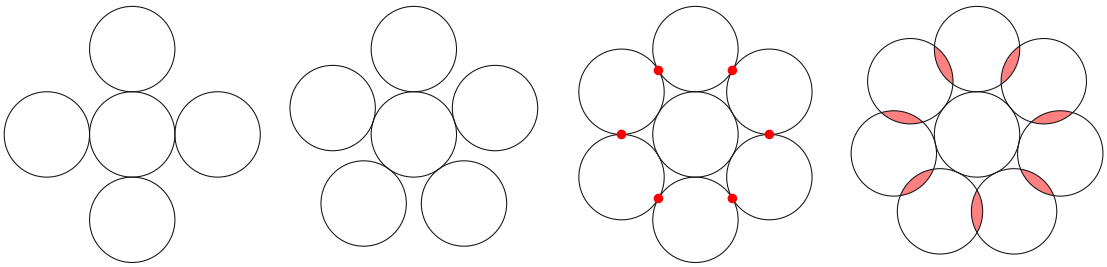


Figure 1.4: Map of the modern london subway system indicating tariff (or fare) zones. While zones are generally nested, some boundary zones (like 2/3 and 3/4) are complex shapes, and split up into smaller contiguous areas. Used unchanged under CC BY-SA 4.0 license.

system) are a common feature of maps that visualize transit systems of larger geographical areas. These are geometric shapes, which enclose a subset of stations in a shape (which itself is often schematized in a style similar to the actual map) to indicate travel fares for journeys in the network. Such areas can be simple, nested or even disconnected. Automated methods usually do not consider the placement of such tariff zones when computing a layout. When considering this task as a post-processing step, this becomes a special case of the much more fundamental problem, of finding the most simple (an possibly schematized) separator between two sets of points. In Chapter 5 we consider a variant of this fundamental underlying geometric problem. We generalize the contained points to a given set of polygons of one color, which need to be separated from polygons of a different color using a low complexity polygon. These low complexity polygons can be thought of a the most simplified separator between the two sets.

In the following two chapters we move slightly away from the motivation of transit network schematization and onto schematic representations of other concepts. In particular we focus on representations of graphs using simple geometric shapes in the plane in Chapter 6 and images emerging from the intersection pattern of curves, which are used as solutions for puzzles in Chapter 7.

Transit networks are examples of a specific class of graphs. However more general classes of graphs can also be visualized in a schematic way. A well-known possibility to visualize and represent graphs are geometric contact and intersection representations. In such a representation every vertex is represented by a (usually simple) geometric object, which is placed in the plane. Edges are represented by contact or intersection



(a) $\delta = 4$ (b) $\delta = 5$ (c) $\delta = 6$ (d) $\delta = 7$

Figure 1.5: Unit disk contact representations of a (a) 4- and (b) 5-star. A (c) 6-star and therefore also any larger star like (d) a 7-star can not be represented this way since a leaf vertex inevitably contacts at least one other leaf.

of the geometric objects of the two vertices of the edge. For a set of geometric objects $\mathcal{A}$, such a representation is then called an $\mathcal{A}$-contact representation or $\mathcal{A}$-intersection representation. Usually the set simply consists of a single object of fixed size (e.g. unit disk contact representations) or congruent copies of varying sizes (e.g. disk contact representations). In fact if we consider the graph G on all regions of a map, which contains an edge between neighboring regions, then the map itself can be viewed as a contact representation of G using the geometric shapes of the regions. The graph G is called the *dual graph* of the map. If we restrict the map to simpler shapes than the geographically correct outlines, we obtain a schematic representation, which is commonly used for some variants of cartograms, e.g., using circles (Dorling cartograms [46]) or squares (Demers cartograms [90]).

Contact and intersection representation are a very well-studied topic in computational geometry and graph theory. There are fundamental connections between graph theoretic or algorithmic concepts and contact or intersection representations. For example, it is well known that graphs, which can be represented with touching disks in the plane are exactly planar graphs [74] and that any planar graph admits a contact representation with low complexity polygons (at most 8 corners, which are sometimes necessary) [137, 4].

On the algorithmic side contact and intersection graphs are used for their properties that are guaranteed through the existence of their geometric representation. Such properties can be, for example, a bound on the maximum degree of a vertex (in unit disk contact representation it is at most 6, as shown in Figure 1.5) or that vertices can easily be separated into strips of disconnected subgraphs if they can be represented by unit sized objects. These properties can then be leveraged to obtain more efficient algorithmic results on other hard problems, like a 2-approximation of maximum independent set in unit disk graphs [40], a problem which is Poly-APX-complete in general graphs [16]. Unit disk intersection graphs are also used to model communication networks between communication nodes with uniform range [75] and even quantum computing [123].

Using disk intersection graphs as a model for such communication networks also leads to

Figure 1.6: Map of the Austrian states. Note that Salzburg is in fact adjacent to Tirol along two distinct and not connected pieces of border. Used unchanged under CC BY 2.5 license.

the following problem. Can we realize a given communication graph with communication nodes, such that the topology of the network is exactly the unit disk intersection representation of the nodes? This obviously boils down to the NP-hard recognition question. In recent years unit disk intersection graph recognition has even be proven to be $\exists\mathbb{R}$-complete (a complexity class located between NP and PSPACE) [68, 82] and even when we already know that the graph is planar, it remains NP-hard [27] (in the case of unit disk contact representation this extends down to outerplanar graphs [72]). On the other hand, for some very simple graph classes this can be decided in linear time [72]. In Chapter 6 we continue this line of research and investigate the hardness of the unit disk intersection graph recognition question for outerplanar graphs, embedded trees and caterpillars.

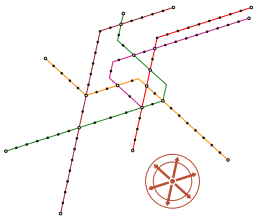
For the last section we return to the already introduced concept of a dual graph. Assume we have a map which is represented as the intersection pattern of a set of intersecting curves in the plane. The dual graph of the map could possibly contain multiples of certain edges if a single region is adjacent to a neighbor over more than one distinct border (see Figure 1.6 for an example). One application where we want to avoid such edge multiples in the dual graph are picture puzzles, where the solution picture is represented as such an intersection pattern of curves. The puzzles in question, which are the focus of the last section, are so called curved nonograms (introduced by van de Kerkhof et al. [70]), which use curves to subdivide the plane. Solving procedures for nonograms have been considered in previous work [12, 18, 71], but our focus is on the creation of such schematic subdivisions of the plane, i.e., the creation of the puzzles. Recently a Dagstuhl seminar report [29] discussed challenges in the creation of these puzzles. In particular the curved nature of the region outlines might lead to some undesired properties in the created

subdivisions. Since the actual geometry of the boundaries is important, Chapter 7 tackles the question how hard it is to remove these undesired properties from a given curve arrangement by only adding a single new curve. Interestingly while the specific geometry of the arrangement is important for the puzzle it is actually irrelevant to answer this question, for which only the topology of the dual graph needs to be considered.

The rest of this thesis is structured as follows. Chapter 2 will introduce some basic concepts, which are relevant for the thesis, although specific definitions, which are relevant to only a single chapter are defined within the chapter in question. Chapters 3 and 4 present results about the computation of schematic transit maps and Chapter 5 investigates the underlying geometric problem of including tariff zones in a schematic map. Chapter 6 pivots and focuses on the recognition question of unit disk graphs for restricted graph classes and Chapter 7 represents the last content chapter and investigates the creation of schematic puzzles. Finally we close the thesis with a summary and conclusion in Chapter 8.

Before we continue with the next chapter, we give a short overview of the five main chapters, as well as their relevant keywords. As every chapter is based on a publication, we also list the relevant publications, as well as possible alternative versions (preliminary or under review). The relevant publications are also given at the start of each chapter.

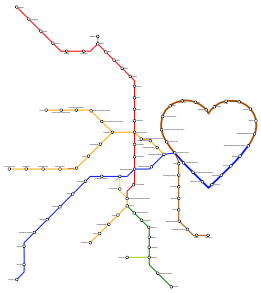
Chapter 3: Computing Data-Driven k-linear Irregular Transit Maps



Despite growing interest in more general *multilinear* or (k-linear) metro maps that deviate from the well-established octolinear de-facto design standard for transit maps, generic algorithms to draw metro maps based on a system of $k \geq 2$ not necessarily equidistant slopes have not been investigated thoroughly. In this chapter we present and implement an adaptation of the octolinear mixed-integer linear programming approach of Nöllenburg and Wolff [97] that can draw metro maps schematized to any set $\mathcal{C}$ of arbitrary orientations. We further present a data-driven approach to determine a suitable set $\mathcal{C}$ by either detecting the best rotation of an equidistant orientation system or by clustering the input edge orientations using a k-means algorithm. We demonstrate the new possibilities of our method using six real-world transit systems, specifically Montreal, Vienna, Washington, Sydney, Berlin and London.

Keywords: [metro map layout | $\mathcal{C}$-oriented schematization | mixed integer linear programming]
Related Publications: Workshop [88], Conference [89], Journal [121]

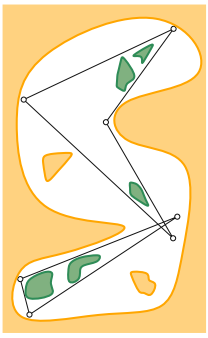
Chapter 4: Including User Defined Geometric Motifs in Transit Maps



Transit maps are often advertised on a web page or pamphlet highlighting routes from source to destination stations. To visually support such route-finding, designers often distort the layout by embedding symbolic shapes (e.g., circular routes) in order to guide readers' attention (e.g., Moscow map in Figure 1.3a and Japan railway map in Figure 1.3c). However, manually producing such maps is labor-intensive and the effect of shapes remains unclear. In this chapter, we propose an approach to generalize such mixed metro maps that take user-defined shapes as an input. In this mixed design, lines used to approximate the shapes are arranged symbolically, while the remaining lines follow classical layout convention. A three-step algorithm, including (1) detecting and selecting routes for shape approximation, (2) shape and layout deformation, and (3) aligning lines on a grid, is integrated to guarantee good visual quality. Our contribution is the definition of the mixed metro map problem and the formulation of design criteria so that the problem can be resolved systematically using the optimization paradigm. Finally, we evaluate the performance of our approach and perform a user study to test if the embedded shapes are recognizable or reduce the map quality.

Keywords: [shape matching | grid-based | multistyle schematization]
Related Publications: Poster [14], Conference/Journal [15]

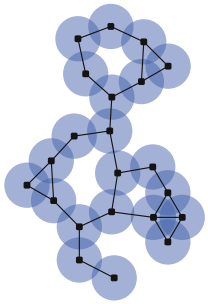
Chapter 5: Finding Minimum-Link Containment Fences



The placement of tariff zones can be thought of as computing minimum complexity containing areas for a given set of vertices of a graph. While stations are usually thought of as points in the plane, in an actual render of a map, they are depicted as circles or pill-shaped objects (and therefore are 2D shapes similar to polygons). We study a variant of this problem inspired by the geometric multicut problem, where we are given a set $\mathcal{P}$ of colored and pairwise interior-disjoint polygons in the plane. The objective is to compute a set of simple closed polygon boundaries (*fences*) that separate the polygons in such a way that any two polygons that are enclosed by the same fence have the same color, and the total number of links of all fences is minimized. We call this the *minimum link fencing* (MLF) problem and consider the natural case of *bounded minimum link fencing* (BMLF), where $\mathcal{P}$ contains a polygon Q that is unbounded in all directions and can be seen as an outer polygon. We show BMLF is NP-hard in general and it is XP-time solvable when each fence contains at most two polygons and the number of segments per fence is the parameter (i.e. there is a polynomial time algorithm for every fixed value of maximum segments per fence). We also present an $O(n \log n)$-time algorithm if the convex hull of $\mathcal{P} \setminus \{Q\}$ does not intersect Q.

Keywords: [XP-algorithm | minimum-link metric | polygon nesting | polygon separation]
Related Publications: Workshop [20], Conference [21]

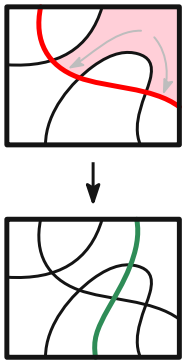
Chapter 6: Representing Graph Connections with Intersecting Unit Disks



A unit disk intersection representation (UDR) of a graph G represents each vertex of G as a unit disk in the plane, such that two disks intersect if and only if their vertices are adjacent in G. A UDR with interior-disjoint disks is called a unit disk contact representation (UDC). Previous work has investigated the hardness of deciding if a given graph can be represented as a UDR or a UDC, with a central result from Breu and Kirkpatrick [27] showing that both questions are NP-hard even for planar graphs. These results have been strengthened for UDCs. Klemz et al. [72] show that it remains NP-hard for outerplanar graphs and, if the embedding of the graph is given and should be reflected in the UDC it is NP-hard even for trees [25]. We provide analogous results for UDRs and show that it is NP-hard to decide if an outerplanar graph or an embedded tree admits a UDR. We further provide a linear-time decidable characterization of caterpillar graphs that admit a UDR. Finally we show that it can be decided in linear time if a lobster graph admits an x-*monotone* UDR, which is a UDR in which the disks of the spine of the caterpillar are placed in an x-monotone chain. It remains an open question if every caterpillar, which has a UDR also has an x-monotone UDR.

Keywords: [geometric intersection graph | unit disk graph | graph recognition | NP-hardness]
Related Publications: Workshop [23], Conference [22]

Chapter 7: Guaranteeing Properties of Curved Schematic Area Partitions



Schematized images are used to indicate the correct solution of nonogram puzzles. While a classical nonogram puzzle is represented as a grid, with the result resembling a pixelated image, there have been variants proposed [70], which result in a visually more pleasing picture by using curves rather than straight lines. However these curved nonograms using curve arrangements present new challenges. A face in a curve arrangement is called *popular* if it is bounded by the same curve multiple times. Motivated by the automatic generation of curved nonogram puzzles, we investigate possibilities to eliminate the popular faces in an arrangement. This can be done by breaking up and reconnecting curves in the nonogram, a methods which has been investigated recently [29]. However, we aim for a method, which considers the given arrangement as fixed in order to not change the outline of the represented image. To do so we consider the problem of removing popular faces by inserting a single additional curve. This turns out to be NP-hard; however, it becomes tractable when the number of popular faces is small: We present a probabilistic FPT-approach in the number of popular faces.

Keywords: [puzzle generation | curve arrangements | fixed-parameter tractable (FPT)]
Related Publications: Workshop [100], Conference [101]

CHAPTER 2

Preliminaries

Specific notations and concepts crucial for the singular main chapters are explained within those chapters. However there are some basic concepts, which are relevant to most of this thesis. Any experienced reader might skip this chapter.

All chapters in this thesis use graphs in some capacity. We will start by introducing the concept of a graph and some basic graph theory in Section 2.1. And finally there are multiple reductions in this thesis proving a lower bound to the hardness of solving some problem. The basic knowledge for this is given in Section 2.3, which reviews necessary concepts of complexity theory.

2.1 Graphs

A *graph* $G = (V, E)$ is (if not otherwise defined) a tuple of two sets, i.e., the *vertices* V and the *edges* $E \subseteq V^2$. Every edge $e \in E$ connects exactly two vertices $u \in V$ and $v \in V$. We will often use the notation $e = (u, v)$ and we will call u and v *endpoints* of e. We will say that u and v are *adjacent* or *connected* by e (which can also be stated as "u and v are *neighbors*") and that e is *incident* to u and v. The *degree* of a vertex v is the number of unique edges, which are incident to v. We will call a vertex with a degree of k a *degree-k vertex*. Degree-1 vertices are also often called *leafs*.

Edges can be either *directed* or *undirected*. An undirected edge e does not differentiate between the two vertices, i.e., $e = (u, v) = (v, u)$. A directed edge e' will have a designated *source* and *target*, i.e., $e' = (u, v) \neq (v, u)$. If the edges of a graph are directed, we will refer to this graph as a *directed graph*, otherwise we will call it an undirected graph. If directionality is not specified, we assume that the graph is undirected. In a directed graph, we can differentiate the *in-degree* and the *out-degree* of a vertex v, which are the number of unique edges, which have v as their target and their source, respectively. Sometimes we assign a number to an edge of a graph. Such a number is referred to as

the *weight* of the edge. If a graph contains edge weights it is called a *weighted* graph, otherwise it is an *unweighted* or *unit-weight* graph.

An edge, which has the same vertex as source and target is referred to as a *loop*. Two edges, which have the same source and target as each other are called *multi-edges*. If a graph contains multi edges, we call it a *multi-graph*. If a graph contains neither multi-edges nor loops, we call this graph *simple*[1]. If not otherwise specified, we will assume all graphs to be simple, unweighted and undirected graphs.

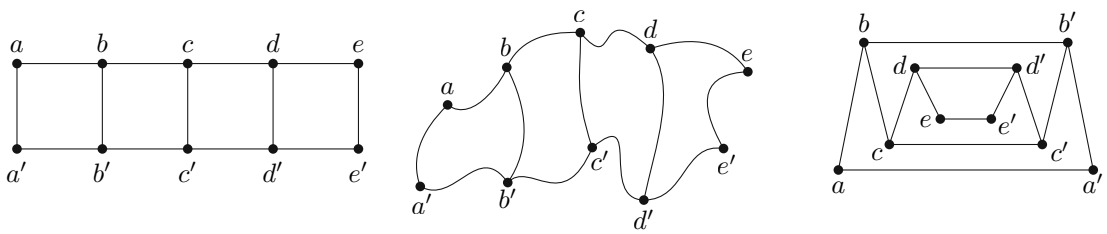
We can traverse a graph G by specifying a list of k vertices in order, such that, there is always an edge between two consecutive vertices. If no vertex repeats in the list, we call this list a *path* of *length* k in G. If vertices are allowed to repeat, which also allows edges to be traversed twice, we call this list a *walk* of *length* k in G. If there is an edge between the first and the last vertex, this traversal is *closed*, i.e., a closed path or a closed walk. If a graph contains a closed path of length more than 1, we say that G contains a *cycle*. Note that in a simple graph, any cycle is at least of length 3, since edges can not be used twice.

If for all pairs of vertices $u, v \in V$, there is a path from u to v we call G *connected*, otherwise it is disconnected. If a connected graph G contains a single vertex v, whose removal would turn G into a disconnected graph (this is also called *disconnecting* G), then v is a *cut-vertex*. If G does not contain a cut-vertex, it is bi-connected, i.e., one would need to remove at least two vertices to disconnect G.

Subgraphs A *subgraph* $H = (V', E')$ of a larger graph $G = (V, E)$ is a graph that is contained in G. This means that $V' \subseteq V$ and $E' \subseteq E$. If for every vertex $v \in V'$, every edge that has v as an endpoint is contained in E', we call $H = (V', E')$ an *induced subgraph* of E. Moreover we say that H is *induced* by V'. For a disconencted graph G, we will call a maximally connected subgraphs, i.e., a connected subgraph of G, which itself is not a subgraph of a larger connected subgraph of G, a *connected component* of G and we will say that G *consists* or *is made up of* its connected components.

Drawings A graph G is an abstract object, which represents connectivity information between the different objects represented by the vertices. However, we often use graphs to describe the connectivity of objects in some space (e.g., between geo-spatial objects or points embedded into the two dimensional plane $\mathbb{R}^2$). In this case it is natural to assign a *position* to the vertices, which is usually given as coordinates in the plane. This is sometimes written as a mathematical function $f : V \to \mathbb{R}^2$, where $f(v)$ is the coordinate of v. Similarly, the edges are also "drawn" in the plane as curves. We can again write this as a function $g : E \to C$, where C is a set of curves, i.e., for every $e = (u, v)$ there is a function $g(e) = c \in C$, such that $c : [0, 1] \to \mathbb{R}^2$, $c(0) = u$ and $c(1) = v$. The tuple of

[1] We would like to point out here that some terms appear in different definitions, e.g., "simple" will also describe properties of a polygon later on. However it will be clear from context, which definition is being used in the main chapters.



(a) Straight-line drawing (b) Curved drawing (c) Different embedding

Figure 2.1: Examples of drawings of the same graph, with different node and edge positions, but the same embedding (a-b) and a drawing of the same graph with a different embedding.

functions f and g is called a *drawing* $\Gamma = (f, g)$ of G. If two curves $g(e)$ and $g(e')$ cross, then the point they have in common is called their *crossing point*. We do not consider shared starting and endpoints of curves crossing points. If no two curves have a crossing point in a drawing, we will call such a drawing *plane*. In a plane drawing, we can define the concept of a *face*, which is an area bounded by curves of edges, that does not contain a curve. The vertices of the edges, whose curves bound a face form a cycle, such that, no subset of these vertices forms a cycle. We say that all of these vertices are *incident* to the face. A drawing usually has a number of bounded *inner* faces and one unbounded *outer* face.

Embeddings In a planar drawing, we sometimes use the concept of the *radial order* of neighbors of a vertex v around v. This refers to the order in which we encounter edges, if we start at one edge incident to v and proceed to rotate around v clockwise. If two drawings have the same rotational order around a vertex v, then they are *equivalent at* v. If two drawings have the same rotational order around every vertex and have the same outer face, we will say that they have the same *embedding* (See Figure 2.1).

2.2 Graph Classes

Graphs can be grouped into different classes with similar properties. We will introduce some relevant graph classes and their relative inclusions here.

Trees. If a graph does not contain a cycle, we will call it a *tree*. A tree with n vertices contains exactly $n - 1$ edges, and in fact any connected graph on n vertices with $n - 1$ edges is a tree. A graph whose connected components are all trees is called a *forest* If a tree contains only one vertex v of degree larger than 1, i.e., all other vertices are leafs, then this tree is called a *star* and v is called the *center*.

Paths and Beyond. Analogue to paths and cycles contained within graphs, if an entire graph forms a path, then we call this graph also a *path* and similarly if the entire

graph forms a cycle, we call the graph a *cycle*. If the subgraph of a tree G induced by all vertices of degree larger than 1 is a path (which is also called the *spine*), then G is called a *caterpillar*. If the subgraph of a tree G induced by all vertices of degree larger than 1 is a caterpillar, G is called a *lobster*. This definition can be continued, with each step increasing the distance of vertices to the spine. In trees, this distance is unbounded.

Planar Graphs. If a graph has a drawing without crossings in the plane, we call it *planar*. Simple planar graphs have at most $3|V| - 6$ edges. If a planar graph has a plane drawing, such that, all vertices are incident to the outer face, this graph is called *outerplanar*. All previously mentioned graph classes of this section are *subclasses* of outerplanar graphs, i.e., every graph that is in one of the previously mentioned graph classes is also outerplanar.

Bi- and k-partite Graphs. If all vertices of a graph G can be partitioned into two sets A and B, such that, there exists no edge of G, which has both endpoints in the same set, then G is *bipartite*. If a similar partition can be done with k sets, the we call G *k-partite*. If any pair of vertices, which are in different sets are connected with an edge, such a graph is also called a *complete bipartite* or *complete k-partite* graph.

2.3 Complexity Theory

Complexity theory is a vast field of research and we can not hope to give a comprehensive introduction here. Instead we simply aim to explain the concepts relevant to this thesis, which includes the complexity classes P, NP, FPT and XP. We would like to point the interested reader to the works of Arora and Barak [5] for a thorough introduction to complexity theory and to Downey and Fellows work [47] for details about parameterized complexity. We assume familiarity with the concepts of a deterministic and non-deterministic Turing machine as well as with the Landau's Symbols (also called big-O notation).

An *alphabet* Σ is a finite set of symbols. A *word* over Σ is a string of characters of Σ. We denote the set of all words of length k using the symbols of Σ by Σ^k. In general the set of all words of any length over the alphabet is Σ^*. A *language* L is a set of such strings, i.e., $L \subseteq \Sigma^*$. We can equally call L a decision problem, since the natural question to be answered for a given word w is to decide wether $w \in L$. A Turing machine *decides* L if it computes a function $f_L : \Sigma^* \to \{0, 1\}$ that maps every word in L to 1 and every other word to 0. We can now define a class of decision problems based on the number of steps needed by a Turing machine to decide the languages in the class. One such class will be called a *complexity class*. Let w be a word of length n. All decision problems, for which the question $w \stackrel{?}{\in} L$, can be decided in $c \cdot T(n)$ steps by a deterministic Turing machine (for a constant c and a computable function $T : \mathbb{N} \to \mathbb{N}$) are in the complexity $\mathsf{DTIME}(T(n))$ and $\mathsf{NTIME}(T(n))$ is similarly defined for non-deterministic Turing machines.

P and NP. Two of the most relevant classes are directly related to these definitions. The classes are P and NP. The class P is defined as $\mathsf{P} = \bigcup_{c>0} \mathsf{DTIME}(n^c)$. This is all decision problems, which can be decided polynomially many steps by a deterministic algorithm relative to the length of the input word. The class NP is defined as $\mathsf{NP} = \bigcup_{c>0} \mathsf{NTIME}(n^c)$. These are all decision problems, which can be decided in polynomially many steps by a non-deterministic Turing machine. However more intuitively, one can also state that these are all problems, for which we can verify in polynomial time if a given solution is correct. The relationship between these two complexity classes is the topic of a famous and long standing open problem in computer science, as it is still unknown if there is a decision problem contained in NP, which is not in P and this question, often stylized as $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$, is one of the Millennium Prize Problems[2].

Reductions, Hardness and Completeness. We say that one decision problem L_1 (over an alphabet Σ_1) can be *reduced in polynomial time* to a second decision problem L_2 (over an alphabet Σ_2) if there is a polynomial-time computable function $g : \Sigma_1^* \to \Sigma_2^*$, which maps any word $w_1 \in \Sigma_1^*$ to a word $g(w_1) \in \Sigma_2^*$, such that, $w_1 \in L_1$ if and only if $g(w_1) \in L_2$. We also call L_1 *poly-time reducible* to L_2 and write $L_1 \leq_{poly} L_2$. We call a problem L NP-*hard*, if every other problem in NP is poly-time reducible to L. This can be conceptualized as a problem, which is at least as hard to solve as any problem contained in NP (and possibly even harder). Note that it is sufficient to show that one already proven NP-hard problem is poly-time reducible to L, since poly-time reducability is a transitive relation. For any complexity class C, if a problem L is C-hard and $L \in C$, we say that L is C-complete. Therefore any problem $L \in \mathsf{NP}$, which is NP-hard is therefore also NP-complete. Most problems, that are discussed in the main chapters of this thesis, are NP-hard, or even NP-complete.

Parameterized Complexity It is clear that $\mathsf{P} \subset \mathsf{NP}$ and, as mentioned above, it is an open question if this is a proper inclusion. These complexity classes group problems based on the time (number of steps) required to solve them, relative to the size of the input. Problems, which are NP-hard are often described as being "difficult to solve" or even "infeasible", as it is generally assumed, that P is not equal to NP. However the difficulty of a problem instances, might be due to a specific property of the instance rather than simply its size. In order to analyze what properties of a problem instance are contributing to the complexity of finding a solution, *Parameterized Complexity* introduces the concept of a *parameter*. Such a parameter is supposed to capture a specific aspect of a problem instance under the assumption that the size of the parameter does not increase according to the instance size. Then we can allow the number of steps to depend in a larger (possibly an arbitrarily large) way on the size of this parameter, while the dependency on the rest of the instance is kept in check.

To give an example, a common parameter is the size of the solution. Consider the classical problem of Vertex Cover, which, given a graph, asks for a subset V^c of vertices, such

[2] https://www.claymath.org/millennium-problems/

that every edge of the graph is incident to at least one of the vertices in V^c. This is a famously NP-hard problem [69], however when considering the parameter $k = |V^c|$, the question "Is there a vertex cover of cardinality at most k?" can be solved in time $\mathcal{O}\left(1.2738^k + (k \cdot n)\right)$.

We can now extend how a problem represented. Recall that so far a problem was a subset $L \subseteq \Sigma^*$. We now define a *parameterized problem* L_p as a subset of $\Sigma^* \times \mathbb{N}$ and a singular instance of a parameterized problem is a word $w \in \Sigma^*$ together with a a natural number $k \in \mathbb{N}$. Now if we can decide if an instance $(w, k) \in L_p$, where w is of length n in at most $\mathcal{O}(f(k) \cdot p(n))$ steps, where f is a computable function and $p(n)$ is a polynomial that depends exclusively on the the size of w and not on the size of the parameter, then L_p is *fixed parameter tractable* (in the class FPT), i.e., for every fixed value of the parameter k, the runtime is the same polynomial. A larger class, which allows a lot more dependency on the parameter is the class XP. The problem L_p is in XP if the decision can be made in $\mathcal{O}(n^{f(k)})$, where f is again a computable function. This intuitively means, that for every fixed parameter value we only need polynomially many steps, however the polynomial can be dependent not only on n but also on k. We can state that $\mathsf{P} \subseteq \mathsf{FPT} \subseteq \mathsf{XP} \subseteq \mathsf{NP}$, where the inclusions are believed to be strict. In fact there is an entire hierarchy of complexity classes between P and NP, however no knowledge beyond FPT and XP will be required for this thesis.

CHAPTER 3

Computing Data-Driven k-linear Irregular Transit Maps

This chapter is (partially) based on the following publications:
[121]: Nöllenburg & Terziadis – Towards Data-Driven Multilinear Metro Maps (Cartographic Journal, to appear)
[89]: Nickel & Nöllenburg – Towards Data-Driven Multilinear Metro Maps (Diagrams'20)
[88]: Nickel & Nöllenburg – Drawing k-linear Metro Maps (Schematic Mapping Workshop'19)

Metro maps are ubiquitous schematic network diagrams that aid public transit passengers in orientation and route planning in almost all types of urban public transit systems worldwide. Since Henry Beck's classic schematic London Tube Map of 1933, metro maps have developed a common visual language and adopted similar design principles [133]. Designing professional metro maps is still mostly a manual task today, even if cartographers and graphic designers are supported by digital drawing tools.

3.1 Related Work

Algorithms for automated layout of metro maps have received substantial interest in the graph drawing and network visualization communities as well as in cartography and geovisualization over the last 20 years [95, 133, 131, 133]. The vast majority of metro map layout algorithms focus on so-called *octolinear* (sometimes also called *octilinear*) metro maps, which are limited to Henry Beck's classical and since then widely adopted 45°-angular grid of line orientations [58]. However, not all metro maps found in practice are strictly octolinear. In fact there is does not seem to be a one-size-fits-all layout for every network.

There is empirical evidence from usability studies that the best set of line orientations for drawing a metro map depends on different aspects of the respective transit network, and it may not always be an octolinear one [112, 111]. A chain of recent publications [93, 92, 9] developed a Topology-Shape-Metrics Framework for orthoradial graph drawing. In this style all edges are either radial segments of concentric circles or straight-line segments of lines through the center of the circles. Orthoradial drawings of metro maps have received come attention outside of scientific research, for example through the change in the official metro map of Cologne, Germany to an orthoradial map [76]. Another framework, which is capable of computing ortho-radial maps is the octi-framework of Bast et al. [11, 10], which uses repeated (or simultaneous) shortest path computation on a predefined grid. This discretization enables alternative layouts, however it is restricted to those that are easily expressible as a regular grid.

In this chapter we present an algorithmic approach using global optimization for computing (unlabeled) metro maps in the more flexible *k-linearity* setting, where each edge in the drawing must be parallel to one of $k \geq 2$ equidistant orientations whose pairwise angles are multiples of $360°/2k$. In this sense, a k-linear map for $k = 4$ corresponds to the traditional octolinear setting. In fact, most octolinear maps use a horizontally aligned orientation system, i.e., a system that includes a horizontal orientation. It is possible though, for some transit networks and city geometries, that a rotation of the orientation system by an angular offset yields a more topographically accurate metro map layout. Hence we also consider such *rotated* k-linear maps. In addition to equiangular k-linear orientation systems, we further study irregular *multilinear* (or *$\mathcal{C}$-oriented*) maps [111], in which the edges are parallel to any given, not necessarily equiangular set $\mathcal{C}$ of orientations. There exist a number of metro map layout algorithms (see [95, 132, 133] for comprehensive surveys) that would technically permit an adaptation to a different underlying angular grid, yet most previous papers optimize layouts in the well-known octolinear setting only and do not discuss extensions to different linearities. A few algorithms for generic multilinear or k-linear layouts exist [87, 83, 43, 30], but they are aimed at paths or polygons rather than entire metro maps. In the field of graph drawing many algorithms for planar orthogonal network layouts with $k = 2$ as well as for polyline drawings with completely unrestricted slopes are known [48], but they do not generalize to k-linearity and multilinearity.

Contributions. We first present two efficient approaches for deriving suitable, data-dependent linearity systems (rotated k-linear and irregular multilinear) by minimizing the angular distortion of the input edge slopes (Section 3.3). We then adapt the octolinear mixed-integer linear programming (MIP) model of Nöllenburg and Wolff [98] by generalizing their mathematical layout constraints to k-linearity and multilinearity (Section 3.4). The main benefit of this model in comparison to other approaches is that it defines sets of hard and soft constraints and guarantees that the computed layout satisfies all the hard constraints, while the soft constraints are globally optimized. The trade-off for providing such quality guarantees from a global optimization technique is that computation time is typically higher compared to other methods [132]. By modeling

fundamental metro map properties such as strict adherence to the given linearity system and topological correctness as hard constraints, we obtain layouts that satisfy these layout requirements strictly. The soft constraints optimize for line straightness, compactness, and topographicity [110], i.e., low topographical distortion. Our modifications yield a flexible MIP model, whose complexity measured by the number of variables and constraints grows linearly with the number of orientations k. We finally demonstrate the effect of horizontally aligned and rotated k-linear and multilinear orientation systems by providing sample layouts of six metro networks and evaluating the resulting number of bends and angular distortions for typical small values of $k = 3, 4$ and 5 (Section 3.5).

3.2 Preliminaries

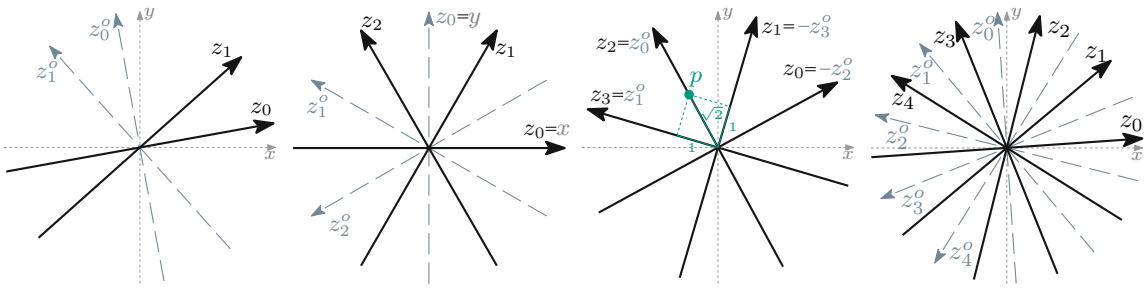
We reuse the notation of Nöllenburg and Wolff [98]. The input is represented as an embedded planar[1] metro graph $G = (V, E)$ with n vertices and m edges. Each vertex $v \in V$ represents a metro station with x- and y-coordinates and each edge $e = (u, v) \in E$ is a segment linking vertices u and v that represents a physical rail connection between them. Let $\mathcal{L}$ be a *line cover* of G, i.e., a set of paths in G such that each edge $e \in E$ belongs to at least one path $L \in \mathcal{L}$. An element $L \in \mathcal{L}$ is called a *line* and corresponds to a metro line in the underlying transit network. Note that multiple lines can pass through the same edge. Finally, $k \geq 2$ is an input parameter that defines the number of available edge orientations in the orientation system $\mathcal{C}$. The set $\mathcal{C}$ and the parameter k can be part of the input or they can be derived automatically from the input geometry, see Section 3.3. Figure 3.1 shows some examples of orientation systems. Since every orientation can be used in two directions this yields $2k$ available drawing directions. Let $\mathcal{K}$ be this set of $2k$ directions. We note that every edge is assigned exclusively to an outgoing direction of its incident vertices, which implies that the maximum degree of G can be at most $2k$. Thus the maximum degree in G provides a lower bound on the required number of orientations.

The general algorithmic metro map layout problem studied in this chapter is to find a *$\mathcal{C}$-oriented schematic layout* of $(G, \mathcal{L})$, i.e., a graph layout that preserves the input topology, uses only edge directions parallel to an orientation from $\mathcal{C}$, and optimizes a weighted layout quality function (here composed of line straightness, topographicity, and compactness). If $\mathcal{C}$ corresponds to a k-linear orientation system, we also call the layout k-linear instead of $\mathcal{C}$-oriented; otherwise it can alternatively be called multilinear.

3.3 Orientation systems

A set of edge orientations (or an *orientation system*) $\mathcal{C} = \{c_1, \ldots, c_k\}$ is a set of k angles (expressed in radian), where $0 \leq c_1 < \cdots < c_k < \pi$. We distinguish three different kinds of possible edge orientation sets. An edge orientation set $\mathcal{C}$ is called *regular* (or *equiangular*)

[1] For non-planar metro graphs we temporarily introduce a dummy vertex for each edge crossing, which preserves the crossing in the output layout.



(a) $k = 2$ irregular (b) $k = 3$ regular aligned (c) $k = 4$ regular rotated (d) $k = 5$ regular rotated

Figure 3.1: Coordinate axes for different orientation systems. (c) illustrates a point p with the redundant coordinates $p = (0, 1, \sqrt{2}, 1)$.

if the angles $\{c_1, \dots c_k\}$ divide the range $[c_1, c_1 + \pi)$ into k parts of equal size π/k, i.e., $c_i - c_{i-1} = \pi/k$ for all $i \in \{2, \dots, k\}$. Otherwise we call $\mathcal{C}$ *irregular*. The special case of a regular orientation system $\mathcal{C}$, in which $c_1 = 0$ is called *aligned*. Note that a classical octolinear layout is based on the aligned orientation system $\mathcal{C}_o = \{0, \pi/4, \pi/2, 3\pi/4\}$.

Opposed to an aligned orientation system, which is fully specified by defining the number k of orientations, we have more degree of freedom in a regular (non-aligned) and an irregular system. The next two sections describe how to derive a suitable system $\mathcal{C}$ from the geometric properties of the input data. The idea behind this approach is to better minimize the topographic distortion of the schematized edges compared to their input direction.

We measure the distortion $\text{dist}_G(\mathcal{C})$ of a system $\mathcal{C}$ with respect to a metro graph G by summing up the difference in slope between each edge $e \in E$ (with slope $\gamma_e \pmod \pi$) and the angle $c \in \mathcal{C}$ which is closest to γ_e:

$$\text{dist}_G(\mathcal{C}) = \sum_{e \in E} \left(\min_{c \in \mathcal{C}} |c - \gamma_e| \right).$$

Note here already, that the closest direction γ_e is chosen independently for all edges. In an actual schematization, the direction of one edge can influence the direction of another and therefore not every edge can necessarily be drawn in the direction that would minimize $|c - \gamma_e|$. Let $\text{dist}_\Gamma(\mathcal{C})$ be the distortion of a computed schematic map Γ of G, in which every edge has an assigned direction c_e, then

$$\text{dist}_\Gamma(\mathcal{C}) = \sum_{e \in E} |c_e - \gamma_e|$$

and $\text{dist}_G(\mathcal{C}) \leq \text{dist}_\Gamma(\mathcal{C})$. Moreover, dist_G is in general not a tight lower bound for the actually minimum obtainable distortion of a schematization of G. More precisely, let G'

be a metro graph and let $\mathfrak{G}$ be the set of all possible k-linear schematizations of G'. Then for the distortion-minimal schematization $\Delta' = \arg\min_{\Delta\in\mathfrak{G}} \mathrm{dist}_\Delta(\mathcal{C})$ it might be that $\mathrm{dist}_G(\mathcal{C}) < \mathrm{dist}_{\Delta'}(\mathcal{C})$. We ask the reader to keep this difference in optimizing $\mathrm{dist}_G(\mathcal{C})$ and $\mathrm{dist}_\Delta(\mathcal{C})$ in mind.

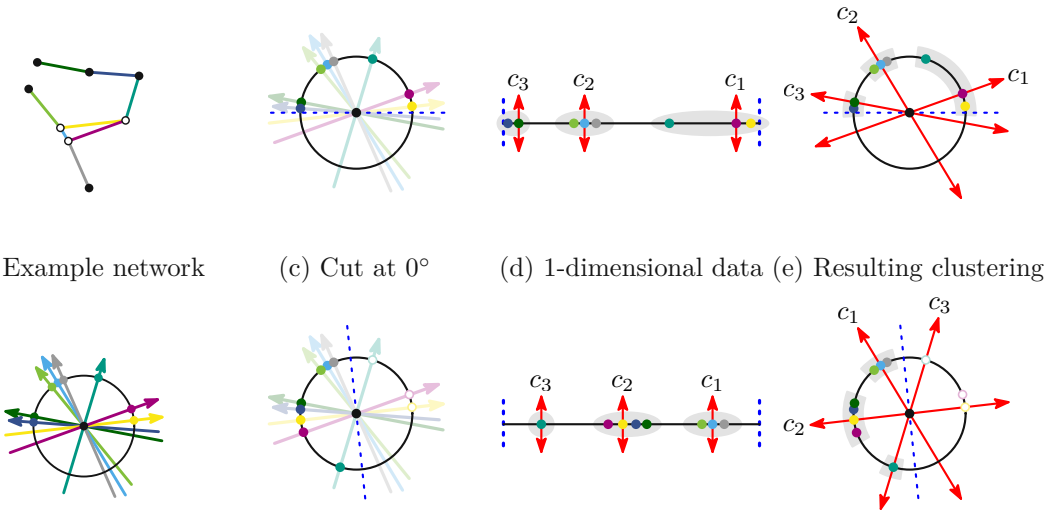
3.3.1 Regular orientation systems

Fixing a single angle in a regular orientation system $\mathcal{C}$ fixes all other orientations. It is therefore sufficient to specify the first orientation $c_1 \in \mathcal{C}$. We denote by $\mathcal{C}_{\mathrm{opt}}$ a regular orientation system minimizing the distortion, i.e., $\mathrm{dist}_G(\mathcal{C}_{\mathrm{opt}}) \leq \mathrm{dist}_G(\mathcal{C})$ for any k-regular orientation system $\mathcal{C}$. The next lemma will help us to find such a system efficiently.

Lemma 3.1. *For any integer k and metro graph G there is an optimal regular orientation system $\mathcal{C}$ with $\mathrm{dist}_G(\mathcal{C}) = \mathrm{dist}_G(\mathcal{C}_{\mathrm{opt}})$, in which at least one orientation $c \in \mathcal{C}$ is equal to the slope of an input edge.*

Proof. Let $\mathcal{C}_{\mathrm{opt}}$ be a minimum-distortion regular orientation system. For each edge $e \in E$ we define $c_{\mathrm{opt}}(e)$ to be the orientation $c \in \mathcal{C}_{\mathrm{opt}}$ that minimizes the distortion $|c - \gamma_e|$ of e. Let ε be a sufficiently small angle such that a rotation of $\mathcal{C}_{\mathrm{opt}}$ by ε in clockwise (resp., counterclockwise) direction results in an orientation system $\mathcal{C}^{cw}$ (resp., $\mathcal{C}^{ccw}$) with $c^{cw}(e) = c_{\mathrm{opt}}(e) - \varepsilon$ and $c^{ccw}(e) = c_{\mathrm{opt}}(e) + \varepsilon$. This implies that in these rotations, every edge e moves (in slope) either strictly closer to of strictly farther away from $c_{\mathrm{opt}}(e)$. Let E_+^{cw} and E_-^{cw} be the sets of edges that increase and decrease, respectively, their distance to $c_{\mathrm{opt}}(e)$ during the clockwise rotation of the orientation system by ε. Analogously, we define E_+^{ccw} and E_-^{ccw} for a counterclockwise rotation by ε. Note that if there is an edge that is in E_+^{cw} and E_+^{ccw} simultaneously, its slope coincides with a direction in the orientation system and we are done. So assume that every edge is either in E_-^{cw} or in E_-^{ccw} and therefore $|E_-^{cw}| + |E_-^{ccw}| \geq |E_+^{cw}| + |E_+^{ccw}|$. If $|E_+^{cw}| < |E_-^{cw}|$, then $\mathrm{dist}_G(\mathcal{C}^{cw}) < \mathrm{dist}_G(\mathcal{C}_{\mathrm{opt}})$ which contradicts the minimality of $\mathcal{C}_{\mathrm{opt}}$. If $|E_+^{cw}| > |E_-^{cw}|$ then $|E_+^{ccw}| < |E_-^{ccw}|$ and $\mathrm{dist}_G(\mathcal{C}^{ccw}) < \mathrm{dist}_G(\mathcal{C}_{\mathrm{opt}})$, which again contradicts the minimality of $\mathcal{C}_{\mathrm{opt}}$. So finally, we must have $|E_+^{cw}| = |E_-^{cw}|$ and then $\mathrm{dist}_G(\mathcal{C}^{cw}) = \mathrm{dist}_G(\mathcal{C}_{\mathrm{opt}})$. We can thus continue the clockwise rotation until one of two things will happen. Either the slope of an edge will coincide with a direction in the rotated orientation system, in which case we are done, or the bisector between two of the rotated orientations coincides with the slope of an edge. If we continue the rotation, this edge will change from E_+^{cw} to E_-^{cw} and hence $|E_+^{cw}| < |E_-^{cw}|$. A further minimal rotation will thus result in $\mathrm{dist}_G(\mathcal{C}^{cw}) < \mathrm{dist}_G(\mathcal{C}_{\mathrm{opt}})$, which contradicts the minimality of $\mathcal{C}_{\mathrm{opt}}$. □

By this lemma we can restrict our search to regular orientation systems in $\mathfrak{C}(E) = \{\mathcal{C} \mid \exists e \in E : \gamma_e \in \mathcal{C}\}$, i.e., to orientation systems, where at least one orientation coincides with the slope of an edge in E. The set $\mathfrak{C}(E)$ contains $O(|E|)$ elements and we select $\mathcal{C}_{\mathrm{opt}}$ as the one yielding the minimum $\mathrm{dist}_G(\mathcal{C})$ for all $\mathcal{C} \in \mathfrak{C}(E)$. Since the graph of a metro network is assumed to be planar, this procedure runs in $O(|V|)$ time if the input graph



(a) Example network (c) Cut at 0° (d) 1-dimensional data (e) Resulting clustering

(b) Edge slopes (f) Cut at 94° (g) 1-dimensional data (h) Resulting clustering

Figure 3.2: Illustration how the slopes of the edges of a network (a) are interpreted as a set of circular data (b). 2 distinct cases of clustering are shown in (c-e) and (f-h). We can cut the data (c & f) to obtain a 1-dimensional list of data points (d & g). After using a k-median clustering algorithm, we obtain an orientation system (e & h) from the median of every cluster. It is clear to see that the quality of the clustering is dependent on the choice of cut, e.g., 0° in (c) or 94° in (f). Note that our data range is only 0° to 180° and three points have been mirrored in (f) and (h) for illustrative purposes.

is planar[2]. In order to compare different values $k = 1, \ldots, t$ we can compute optimal k-linear regular orientation systems for each value of k in a total time of $O(t|E|^2)$.

3.3.2 Irregular orientation systems

In an irregular orientation system $\mathcal{C}$ with k orientations, each orientation can be selected independently. By sorting all slopes of edges (which can be expressed as their angle with the x-axis) present in the input we obtain a list of cyclical one dimensional values. We measure the distance between two slopes by the smaller angle between them. Recall that γ_e is the slope of an edge e in the input. We can interpret an orientation system as a clustering of the set $\Gamma = \{\gamma_e \mid e \in E\}$ of all input edge slopes, where each cluster is formed around the closest orientation in $\mathcal{C}$. By a similar argument as in the proof of Lemma 3.1, we can state that in a dist_G-optimal irregular orientation system all chosen directions can be assumed to be the slope of an edge in the input. Our goal is to find a

[2]Note that we are working with non-planar graphs, whose edges can be in $\Theta(|V|^2)$, however actual real world metro networks tend to have few or no crossings, which makes the assumption of planarity reasonable.

set $\mathcal{C}$ of k orientations that minimizes $\text{dist}_G(\mathcal{C})$. To this end we can use a 1-dimensional k-medians clustering of the set Γ.

The dynamic programming algorithm [94] solves the 1-dimensional k-medians clustering problem of n points exactly with a running time of $O(n^2k)$. Their approach uses a dynamic program and a precomputed auxiliary matrix as a look-up table, which contains the distances between all pairs of elements. For more details we refer the reader to the original publication [94].

The difference to Nielsen and Nock's setting is that our data, while one-dimensional, are cyclic. For example, if we look at an edge which forms a $\pi/2 + \delta$ angle with the x-axis, for some $\delta > 0$, then the 1-dimensional cluster algorithm would claim it has a distance of $\pi/2 + \delta$ to the representative direction, while in fact we would measure the smaller opposite angle, i.e., $\pi/2 - \delta$. We have to account for this cyclic nature. To do so, we can make use of the fact that (a) if we cut the cyclic data and flatten it into a list (as shown in Figure 3.2) exactly at the boundary between two clusters of an optimal cyclical clustering, then the 1-dimensional clustering results in the same set of clusters and (b) clusters of slopes will never overlap and therefore there are exactly n such candidate gaps (see Figures 3.2c and 3.2f) at which we can cut the data and flatten it. This of course creates in n calls to the dynamic program, resulting in a total runtime of $\mathcal{O}(n^3k)$. There exist bespoke algorithms for optimal circular clustering like the FOCC algorithm [41], which achieves an optimal clustering in $\mathcal{O}(kn \log^2 n)$ time, however we decided on the dynamic program of Nielsen and Nock since it is (a) easy to implement and (b) not the bottleneck of our approach since any clustering is followed by a the construction and subsequent solving of a mixed integer program.

However, during the experiments this led to a number of instances, for which, with the orientations resulting from a dist_G-optimal clustering, the mixed integer program admitted no solution. Possible reasons for this are discuss at the end of Section 3.5.3. Note that since Nielsen and Nock's dynamic programming approach [94] is optimal, this is not an issue of their particular approach, but instead of the k-medians clustering itself and would persist, even if we would use the FOCC algorithm. In contrast, simply sorting all edges by slope and cutting the cyclical order at 0, lead to clustered directions, for which a solution existed in all cases. All further tables, results and figures, which refer to irregular clustering were computed using the cut at 0 unless explicitly stated otherwise.

3.4 MIP Model

Next we sketch how the constraints of the MIP model of Nöllenburg and Wolff [98] must be modified in order to compute more general $\mathcal{C}$-oriented metro maps for a given arbitrary set $\mathcal{C}$ of k orientations. In our description we focus on the differences to the octolinear MIP model [98].

3.4.1 Hard Constraints

The hard constraints comprise four aspects: $\mathcal{C}$-oriented coordinate system, assignment of edge directions, combinatorial embedding, and planarity.

Coordinate System. Every vertex u of G has two Cartesian coordinates in the plane $\mathbb{R}^2$, specified as $x(u)$ and $y(u)$. In order to address vertex coordinates for a flexible number k of orientations, we define a redundant system of k coordinates $z_0, \ldots, z_{k-1}$, which are all real-valued variables in the MIP model and can all be obtained by rotating the x-axis counterclockwise by one of the angles in the orientation system $\mathcal{C} = \{\theta_0, \ldots, \theta_{k-1}\} \subset [0, \pi)$. We define the coordinate $z_i(u)$ using $x(u)$ and $y(u)$ as $z_i(u) = \cos(\theta_i) \cdot x(u) + \sin(\theta_i) \cdot y(u)$.

In order to be able to express later that two vertices u, v are collinear on a line with slope in $\mathcal{C}$, we need the orthogonal orientation z_i^o for each coordinate z_i. Note that while z_i^o can coincide with other coordinates, this is guaranteed only in a regular orientation system with an even number of orientations. In general, this is not the case and we need to define a second set of redundant coordinates, see Figures 3.1a, 3.1b and 3.1d. Using a rotation by $\pi/2$ we obtain $z_i^o(u) = -\sin(\theta_i) \cdot x(u) + \cos(\theta_i) \cdot y(u)$.

Edge Directions and Minimum Length. Every edge $(u, v) \in E$ has an original direction in the input layout of G, defined as the direction from u to v. Our $\mathcal{C}$-oriented coordinate system splits the plane into exactly $2k$ sectors numbered from 0 to $2k-1$ for each vertex $u \in V$, see Figure 3.3a. We store the sector in which an edge (u, v) lies in the input drawing as a constant $\sec_u(v)$ that we call the *original sector* of (u, v).

Next we define an integer variable $\mathrm{dir}(u, v)$ to encode the selected direction of the edge (u, v) in a $\mathcal{C}$-oriented solution. The range for $\mathrm{dir}(u, v)$ includes the original sector $\sec_u(v)$ and $s \geq 1$ admissible neighboring sectors in both directions. In their original MIP model Nöllenburg and Wolff [98] uses $s = 1$, which results in a range of three admissible edge directions for each edge.

For each edge (u, v) we define the set $\mathcal{S}(u, v)$ of admissible directions[3] as $\mathcal{S}(u, v) = \{i \mid \sec_u(v) - s \leq i \leq \sec_u(v) + s\}$. For each $i \in \mathcal{S}(u, v)$ we define its corresponding direction number as $\sec_u^i(v)$ and define a binary variable $\alpha_i(u, v)$ of which only one can be true at any given time (3.1). These are then used to assign the correct value of $\mathrm{dir}(u, v)$ (3.2).

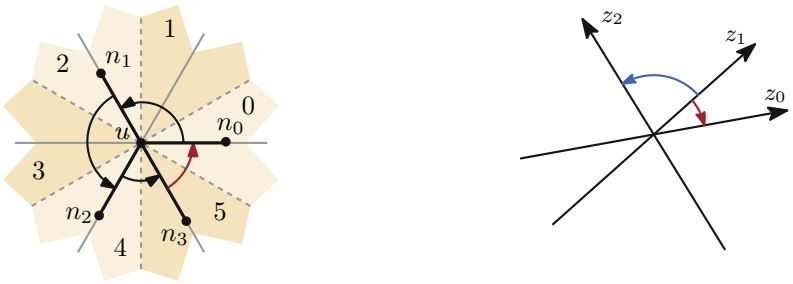
$$\sum_{i \in \mathcal{S}(u,v)} \alpha_i(u, v) = 1 \tag{3.1}$$

$$\mathrm{dir}(u, v) = \sum_{i \in \mathcal{S}(u,v)} \sec_u^i(v) \cdot \alpha_i(u, v) \tag{3.2}$$

We further define $\mathrm{dir}(v, u) = \mathrm{dir}(u, v) + k$ for the opposite edge (v, u).

To guarantee that the output layout draws the edge (u, v) in the selected direction $\mathrm{dir}(u, v)$ we need to ensure that the variables of u and v for the orthogonal coordinate axis z_i^o

[3] All index calculations are modulo $2k$.



(a) For edges $(u, n_3), (u, n_0)$ the direction value decreases from 5 to 0.

(b) $k = 3$, irregular

Figure 3.3: (a) For edges $(u, n_3), (u, n_0)$ the direction value decreases from 5 to 0. (b) The difference in angle between the orientations z_0 and z_1 (red arrow) is significantly smaller than between z_1 and z_2 (blue arrow).

are equal, i.e., $z^o_{\mathrm{dir}(u,v)}(u) = z^o_{\mathrm{dir}(u,v)}(v)$ (3.3a) and that the coordinates $z_{\mathrm{dir}(u,v)}(u)$ and $z_{\mathrm{dir}(u,v)}(v)$ differ by at least the minimum edge length L_{min}, i.e., $z_{\mathrm{dir}(u,v)}(v) - z_{\mathrm{dir}(u,v)}(u) \geq L_{min}$ (3.3b).

$$\begin{aligned} z^o_{i'}(u) - z^o_{i'}(v) &\leq M(1 - \alpha_i(u,v)) \\ -z^o_{i'}(u) + z^o_{i'}(v) &\leq M(1 - \alpha_i(u,v)) \end{aligned} \tag{3.3a}$$

$$\begin{aligned} z_{i'}(v) - z_{i'}(u) &\geq -M(1 - \alpha_i(u,v)) + L_{\min} \quad \text{if } i < k \\ z_{i'}(u) - z_{i'}(v) &\geq -M(1 - \alpha_i(u,v)) + L_{\min} \quad \text{if } i \geq k \end{aligned} \tag{3.3b}$$

Note four things. First, the constraints are created for every $i \in \mathcal{S}(u,v)$. Second, we use $i' = i \bmod k$, since we only have k coordinates, but $2k$ possible directions. Third, we need to distinguish whether the direction i is smaller than the number k of orientations, in which case u must have a smaller value than v in coordinate z_i, or otherwise if $i \geq k$ then v must have the smaller coordinate and we need to invert the difference in (3.3b). And fourth, every triple of constraints for which $\alpha_i(u,v) = 0$ is trivially satisfied by using a sufficiently big constant M in the constraints. Due to (3.1), $\alpha_i(u,v) = 1$ for exactly one index i and only for that index i the constraints have the desired effect on the coordinates.

Combinatorial embedding. We want to keep the combinatorial embedding, i.e., the topology of the input layout, which translates into preserving the cyclic order of the neighbors of each vertex. This can be expressed by requiring that the edge direction values strictly increase when visiting the incident edges in counterclockwise input order. There is exactly one exception, namely when going from the last used sector to the first one. Figure 3.3a illustrates this situation, where the crossover point lies between the neighbors n_3 and n_0, marked in red. Here we can add an offset of $2k$ instead to make the condition hold. Since this must happen exactly once, we can use binary variables

$\beta_1(v), \beta_2(v), \ldots, \beta_{\deg(v)}(v)$ to select the respective edge pair in equations (3.4) and (3.5).

$$\sum_{i=1}^{\deg(v)} \beta_i(v) = 1 \tag{3.4}$$

$$\begin{aligned} \operatorname{dir}(v, u_1) + 1 \leq \quad & \operatorname{dir}(v, u_2) + 2k \cdot \beta_1(v) \\ \operatorname{dir}(v, u_2) + 1 \leq \quad & \operatorname{dir}(v, u_3) + 2k \cdot \beta_2(v) \\ \vdots \\ \operatorname{dir}(v, u_{\deg(v)}) + 1 \leq \quad & \operatorname{dir}(v, u_1) + 2k \cdot \beta_{\deg(v)}(v) \end{aligned} \tag{3.5}$$

Planarity. For every pair of non-adjacent edges $e = (u, v)$ and $e' = (u', v')$ we need to find (at least) one separation line between e and e' in a direction of $\mathcal{K}$ to guarantee that e, e' do not intersect. We define a set of $2k$ binary variables $\gamma_i(e, e')$ for which we require that at least one of them is set to true.

$$\sum_{i=0}^{2k-1} \gamma_i(e, e') \geq 1 \tag{3.6}$$

Now we ensure that every pair of edges e, e' has a minimum distance $d_{\min}$ in the selected directions, i.e., both endpoints of e have a distance of at least $d_{\min}$ to both endpoints of e'.

$$\begin{aligned} z_{i'}(u') - z_{i'}(u) \geq \quad & -M(1 - \gamma_i(e, e')) + d_{\min} \\ z_{i'}(u') - z_{i'}(v) \geq \quad & -M(1 - \gamma_i(e, e')) + d_{\min} \\ z_{i'}(v') - z_{i'}(u) \geq \quad & -M(1 - \gamma_i(e, e')) + d_{\min} \\ z_{i'}(v') - z_{i'}(v) \geq \quad & -M(1 - \gamma_i(e, e')) + d_{\min} \end{aligned} \tag{3.7}$$

Note that the constraints are created for every $0 \leq i < 2k$, that we use $i' = i \mod k$ and that the first k sets of these equation look like (3.7), while the rest needs to invert the differences, e.g., $-z_{i'}(u') + z_{i'}(u)$, since they change sides with respect to direction $z_{i'}$.

3.4.2 Soft constraints

Soft constraints model the aesthetic quality criteria to be optimized in the layout. We adapt the three criteria of the octolinear MIP [98] to arbitrary orientation systems: line straightness, topographicity, and compactness. Each requires a set of linear constraints and a corresponding linear term in the objective function.

Line straightness. We optimize for line straightness by minimizing the number and angles of bends along the metro lines in $\mathcal{L}$. First we create a variable $\theta(u_1, u_2, u_3)$ for all pairs of consecutive edges $e_1 = (u_1, u_2), e_2 = (u_2, u_3)$ along some path $L \in \mathcal{L}$ that represents the cost of a potential bend between e_1 and e_2 on the metro line L. To assign $\theta(u_1, u_2, u_3)$ we subtract the direction of e_2 from the direction of e_1. If the edges do not have the same direction, the difference $\operatorname{dir}(u_1, u_2) - \operatorname{dir}(u_2, u_3)$, which we will

call $\Delta\,\mathrm{dir}_{u_1,u_2,u_3}$, will either be positive or negative and $\Delta\,\mathrm{dir}_{u_1,u_2,u_3} \in [-2k+1, 2k-1]$. From [98] we know that $\theta(u_1,u_2,u_3) = \min\{|\Delta\,\mathrm{dir}_{u_1,u_2,u_3}|, 2k - |\Delta\,\mathrm{dir}_{u_1,u_2,u_3}|\}$, i.e., $\theta \in [-k+1, k-1]$. Using two binary correction variables δ_1 and δ_2 we can ensure that θ takes the desired minimal value (3.8), which then lets us define the bend cost function (3.9).

$$\begin{aligned} -\theta(u_1,u_2,u_3) &\leq \Delta\,\mathrm{dir}_{u_1,u_2,u_3} - 2k\cdot\delta_1 + 2k\cdot\delta_2 \\ \theta(u_1,u_2,u_3) &\geq \Delta\,\mathrm{dir}_{u_1,u_2,u_3} - 2k\cdot\delta_1 + 2k\cdot\delta_2 \end{aligned} \tag{3.8}$$

$$\mathrm{cost}_{\mathrm{bends}} = \sum_{L\in\mathcal{L}} \sum_{(u_1,u_2),(u_2,u_3)\in L} \theta(u_1,u_2,u_3) \tag{3.9}$$

Topographicity. In order to support the mental map [84] of the user, we want the shape of the output drawing to resemble the input drawing as closely as possible. For this we try to preserve the input directions of the edges. Formally we want to minimize the difference between the input direction and the output direction. However we have to consider the type of orientation system that is in use. Specifically, for aligned and regular orientation systems, we simply minimize $\sum_{(u,v)\in E} |\,\mathrm{dir}(u,v) - \mathrm{sec}_u(v)|$. In order to minimize the absolute value we define a new variable $\xi(u,v) = |\,\mathrm{dir}(u,v) - \mathrm{sec}_u(v)|$ by imposing (3.10) and minimizing $\xi(u,v)$ in the cost function (3.11). The topographicity cost function that is minimized is simply the sum over all ξ-variables (3.11).

$$\begin{aligned} \mathrm{dir}(u,v) - \mathrm{sec}_u(v) &\leq \xi(u,v) \\ -\,\mathrm{dir}(u,v) + \mathrm{sec}_u(v) &\leq \xi(u,v) \end{aligned} \tag{3.10}$$

$$\mathrm{cost}_{\mathrm{topo}} = \sum_{(u,v)\in E} \xi(u,v) \tag{3.11}$$

Here every edge (u,v), which is drawn in a sector that deviates from its input sector $\mathrm{sec}_u(v)$ incurs a penalty of exactly the number of deviated sectors (which in our experiments has been capped to at most one and therefore every edge either produces a penalty of 1 or no penalty).

The attentive reader might already have spotted that this lines up for aligned and regular orientation systems, since they use equiangular orientations. In an irregular system on the other hand, a deviation of one sector to the left could be a very small actual angle difference in the produced drawing, while a similar deviation of one sector to the right could, at the same time, mean a large angle difference (see Figure 3.3b). To account for this, we can define an individual sector deviation penalty for every edge and every possible deviation of that edge.

Let $e = (u,v)$ be an edge with an input sector $\mathrm{sec}_u(v)$. Then we adapt the constraints (3.10) as follows.

$$\begin{aligned} \mathrm{dir}(u,v) - \mathrm{sec}_u(v) &\leq \xi_1(u,v) \\ -\,\mathrm{dir}(u,v) + \mathrm{sec}_u(v) &\leq \xi_2(u,v) \end{aligned} \tag{3.12}$$

Now $\xi_1 > 0$ corresponds to a counterclockwise deviation of an edge, while $\xi_2 > 0$ corresponds to a clockwise deviation. We can multiply each variable with an individual constant weight to account for the actual angle difference of the corresponding deviation in an irregular orientation system.

$$\text{cost}_{\text{topo}} = \sum_{(u,v)\in E} (w_1^e \cdot \xi_1(u,v)) + \sum_{(u,v)\in E} (w_2^e \cdot \xi_2(u,v)) \tag{3.13}$$

While the penalty weights w_1^e and w_2^e can be chosen freely, we aim to replicate the scale of the aligned and regular systems, where a deviation of $\alpha = 2\pi/2k$, i.e., the only available deviation by exactly one sector, equals a penalty of 1. Let β be the angle between two orientations of an irregular orientation system. Then we choose the penalty for an edge, which deviates from one orientation to the other as $w = \beta/\alpha = k\beta/\pi$.

Compactness. To ensure a compact layout we minimize the total edge length of the output drawing. Here we use that the Euclidean length of an edge $e = (u, v)$ in a $\mathcal{C}$-oriented layout is defined by the maximum absolute value $|z_i(u) - z_i(v)|$ in all k coordinates (the projections in all other directions are shorter), which we model by a variable $\lambda(u, v)$. The compactness cost function is the sum of all edge lengths.

$$\begin{aligned} z_i(u) - z_i(v) &\leq \lambda(u,v) \\ -z_i(u) + z_i(v) &\leq \lambda(u,v) \end{aligned} \tag{3.14}$$

$$\text{cost}_{\text{length}} = \sum_{(u,v)\in E} \lambda(u,v) \tag{3.15}$$

Objective Function. The objective function to be minimized is composed of the three different terms $\text{cost}_{\text{bends}}, \text{cost}_{\text{topo}}$ and $\text{cost}_{\text{length}}$ defined above. Each term can be weighted with factors f_1, f_2, f_3 depending on their relative importance.

$$\text{minimize } f_1 \cdot \text{cost}_{\text{bends}} + f_2 \cdot \text{cost}_{\text{topo}} + f_3 \cdot \text{cost}_{\text{length}} \tag{3.16}$$

3.4.3 Improvements

Further, Nöllenburg and Wolff [98] devised several practical improvements to accelerate their method.

Planarity on Demand. The number of planarity constraints (Sect. 3.4.1) grows quadratically with the number of edges, but most of them are never critical as any reasonable layout satisfies them trivially. So they suggested ways of reducing the number of necessary constraints and to add them only on demand, which immediately carries over to our generalized implementation. In practice, whenever we find a new best solution during the iterative solving process of the MIP, we check if any edges cross. If this is

not the case, we have found a valid intermediate solution, otherwise we add planarity constraints forcing each of the two edges to be non-overlapping with all other edges and continue the search.

Contraction of low degree vertices. Another speed-up method is to make use of the assumption that sequences of vertices with degree 2 (chains of edges between interchange or terminal stations) are usually drawn in the same direction. We can therefore remove these edges and replace them with a dummy edge, whose minimal length is set to a large enough value, such that, all stations can be re-inserted on this edge in a final solution even at its minimal length. To allow further flexibility, we in fact replace each chain of such edges not by one, but by a chain of three edges, such that, there is enough space for the reinsertion of the stations. Finally, after re-inserting the station on this edge we distribute the stations equally along the chain of three edges to achieve a more well-distributed look in the final map. While the application of this edge contraction is a commonly used method, it stands somewhat in contrast to the goal of accurately representing the original edge direction of each individual edge. It should therefore not be seen as a strict improvement, but rather as a speed-up tool with a trade-off.

3.5 Experiments

We performed experiments on real-world metro networks to compare the computational performance and visual quality of the computed metro maps with different linearity systems.

3.5.1 Setup

We generated schematic layouts of the metro networks of Montreal ($n = 65$, $m = 66$), Vienna ($n = 90$, $m = 96$), Washington DC ($n = 97$, $m = 101$), Sydney ($n = 173$, $m = 181$), Berlin ($n = 242$, $m = 293$) and London ($n = 267$, $m = 320$) using aligned, regular and irregular orientation systems with $k \in \{3, 4, 5\}$ orientations (Berlin and London were restricted to $k \in \{4, 5\}$ due to the existence of vertices with degree 7 or 8). All layouts were created with two different weight vectors for the objective function, a more balanced setting $(f_1, f_2, f_3) = (3, 2, 1)$ and one emphasizing bend minimization $(f_1, f_2, f_3) = (10, 5, 1)$, resulting in 96 instances in total. For all layouts we added planarity constraints on demand and used $s = 1$ admissible neighboring sectors for each original edge direction (recall Section 3.4.1). Finally we contracted chains of degree two vertices as described in Section 3.4.3.

The experiments were run on a computing cluster with 3 nodes, each with 1024GB RAM and two 24-core AMD EPYC 7402, 2.80GHz. All experiments were restricted to a single thread within IBM ILOG CPLEX. The operating system runs a Linux kernel version 4.15.0-208. Our implementation uses IBM ILOG CPLEX 12.8 to solve the integer linear programs as single threads. Each experiment had an allocated available memory space of 32GB for Vienna, Montreal, Sydney, and Washington and 60GB for London and Berlin.

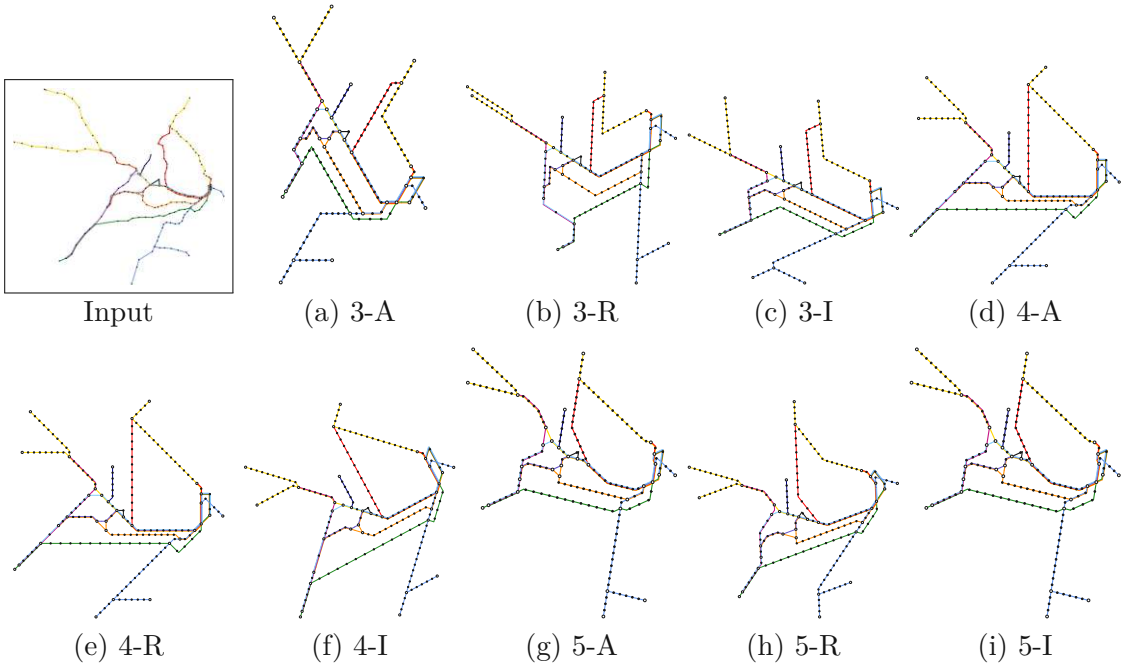
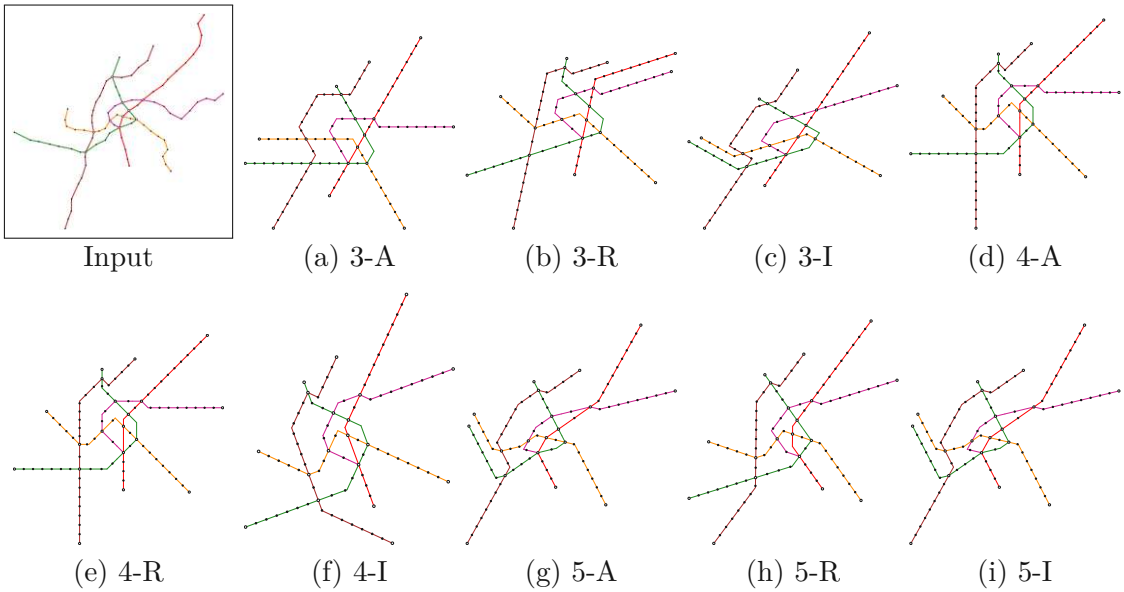


Figure 3.4: Sydney generated with objective function weights $(f_1, f_2, f_3) = (3, 2, 1)$ for $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems. All shown instances reached the time-out limit of 1 hour.

To reflect a reasonable real world application and to showcase the ability of the MIP approach to compute reasonable results before reaching optimality, we set a time limit of 1 hour, chosen as a compromise between an attempt of finding as many optimal solutions to the instances as possible and modeling a reasonable use case, which would not require the user to let the solver run over night. To judge the quality and performance of a layout, we use several measurements. Line straightness was measured by the bend cost in the MIP (see Section 3.4.2). The sector deviation is a coarse measure of topographicity, counting how many edges are not drawn in their preferred direction (see Section 3.4.2). Sector deviation is measured in total and on average per edge. Another measure of topographicity is the angular distortion, i.e., the actual angular difference between input edges and schematized output edges, which is measured on average per edge. Finally we measure the runtime in seconds (instances which reached the computation time limit of 1 hour are marked with a clock symbol 🕓) and the optimality gap of the best found solution after the time limit as given by the solver in percent.

3.5.2 Results

Here we show a representative set of nine layouts of the Sydney metro network in Figure 3.4, a similar set of the Vienna metro network in Figure 3.5 and the performance and quality measurements in Tables 3.1 and 3.2, respectively. Additionally a comparison



Input (a) 3-A (b) 3-R (c) 3-I (d) 4-A

(e) 4-R (f) 4-I (g) 5-A (h) 5-R (i) 5-I

Figure 3.5: Vienna generated with objective function weights $(f_1, f_2, f_3) = (3, 2, 1)$ for $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems. All shown instances have been solved to optimality within the time-out limit of 1 hour.

between the three methods of determining the set $\mathcal{C}$ of drawing directions is depicted in Figure 3.6 for the weight vector $(f_1, f_2, f_3) = (3, 2, 1)$. Similar plots and tables for the five other networks and the weights $(f_1, f_2, f_3) = (10, 5, 1)$ can be found in the supplementary material. The 1-hour time limit for CPLEX was reached by all Sydney instances. Note, however, that during the process of solving the MIP, we have access to intermediate, but possibly suboptimal solutions. Even in instances, which run out of time (like Sydney), we tend to find intermediate solutions, which are visually already quite close to the final solutions (at the 1 hour time limit) after only a few minutes of computation. As shown in Table 3.1, the optimality gap of all solutions is at most 20%. Since solving these instances as close to optimality as possible increases comparability between instances via the taken measurements, we set the rather long time limit.

Specific instances in this section will be referred to by name followed by their number of orientations k and the weights f_1, f_2, f_3 in parentheses. Our first observation from generalizing the model of Nöllenburg and Wolff [98] is that the MIP model size, i.e., the numbers of constraints and variables scale linearly with the number k of orientations. So as long as k is a (small) constant the asymptotics with respect to the graph size parameters n and m remain the same. Yet, in practice, doubling the size of the model may yield a significant slow-down in the actual solution time.

An initial comparison between the metrics for the two different weight vectors $(3, 2, 1)$ and $(10, 5, 1)$ makes clear that, while the specific values are different, the overall trends are very similar, as was expected.

Table 3.1: Metric results for the Sydney network displaying the results for the different parameters, i.e., the number of available directions (k) and the orientation system (**A**ligned, **R**egular, **I**rregular). Metrics are the number of bends, sector deviation (total and per edge), distortion per edge, the runtime in seconds (instances which reached the computation time limit of 1 hour are marked with a clock symbol 🕓) and the optimality gap in percent (listed as number between 0 and 1). For one set of objective function weights and linearity k combination the best value across all orientation systems of every metric is marked in bold.

weights	instance	Sydney								
	linearity	$k=3$			$k=4$			$k=5$		
	orientation system	A	R	I	A	R	I	A	R	I
$(3,2,1)$	bends	42.0	43.0	**35.0**	**49.0**	**49.0**	**49.0**	**63.0**	64.0	72.0
	sector deviation	41.0	32.0	**29.0**	11.0	**8.0**	18.0	12.0	**10.0**	14.0
	↳ per edge	0.42	0.33	**0.3**	0.11	**0.08**	0.19	0.12	**0.1**	0.14
	distortion per edge	1.0	**0.78**	1.03	**0.31**	0.38	0.62	0.63	0.71	**0.43**
	time in seconds	🕓	🕓	🕓	🕓	🕓	🕓	🕓	🕓	🕓
	optimality gap	0.2	0.2	**0.16**	0.13	0.12	**0.1**	0.11	0.08	**0.07**
$(10,5,1)$	bends	**31.0**	32.0	35.0	48.0	**47.0**	**47.0**	**59.0**	**59.0**	**59.0**
	sector deviation	36.0	32.0	**24.0**	**8.0**	10.0	17.0	**12.0**	**12.0**	18.0
	↳ per edge	0.37	0.33	**0.25**	**0.08**	0.1	0.18	**0.12**	**0.12**	0.19
	distortion per edge	0.88	**0.84**	1.02	**0.29**	0.33	0.59	0.65	0.68	**0.49**
	time in seconds	🕓	🕓	🕓	🕓	🕓	🕓	🕓	🕓	2216.0
	optimality gap	0.09	0.07	**0.05**	**0.01**	0.02	**0.01**	0.01	0.01	**0.0**

Comparing the number of bends, we can see in Table 3.1 that increasing k leads to a greater number of bends. This can be explained in part by an increase in forced bends, where the probability that consecutive edges in a metro line cannot possibly be drawn in the same direction increases with k. This could be counteracted by increasing the parameter s, i.e., allowing more than two admissible neighboring sectors for each edge (an action which in turn will have an effect on the time needed to solve the MIP). The difference in bends between the aligned, the regular and the irregular orientation systems is small and when emphasizing the bend cost more (by increasing f_1 from 3 to 10), we obtain almost identical values across the board with the exception of the two larger instances Berlin and London, where the irregular system leads to a larger number of bends with an increase of around 11% to 15% (Berlin) and 24% to 68% (London) when comparing the best aligned or regular map to the best irregular map. We observe here already that the irregular system for $k=5$ for London obtains very poor metrics overall, indicating that the heuristic determination of suitable directions for the map was not successful for this instance. Unsurprisingly we have almost always fewer bends, when emphasizing the objective function that minimizes bends with the sole exception of the

regular Vienna-5 instance, where a reduction in sector deviation offsets the larger amount of bends.

Table 3.2: Metric results for the Vienna network displaying the results for the different parameters, i.e., the number of available directions (k) and the orientation system (**A**ligned, **R**egular, **I**rregular). Metrics are the number of bends, sector deviation (total and per edge), distortion per edge, the runtime in seconds (instances which reached the computation time limit of 1 hour are marked with a clock symbol 🕑) and the optimality gap in percent (listed as number between 0 and 1). For one set of objective function weights and linearity k combination the best value across all orientation systems of every metric is marked in bold.

weights	instance	Vienna								
	linearity	$k=3$			$k=4$			$k=5$		
	orientation system	A	R	I	A	R	I	A	R	I
(3,2,1)	bends	**11.0**	13.0	**11.0**	**19.0**	20.0	23.0	26.0	**22.0**	29.0
	sector deviation	23.0	**14.0**	35.0	6.0	8.0	**4.0**	8.0	14.0	**6.0**
	↳ per edge	0.43	**0.26**	0.65	**0.11**	0.15	0.07	0.15	0.26	**0.11**
	distortion per edge	**0.69**	0.84	0.7	**0.32**	0.39	0.42	**0.57**	0.58	**0.57**
	time in seconds	687.0	**262.0**	2661.0	**116.0**	1403.0	209.0	**804.0**	3181.0	2739.0
	optimality gap	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
(10,5,1)	bends	**9.0**	10.0	10.0	**19.0**	**19.0**	**19.0**	25.0	**24.0**	26.0
	sector deviation	22.0	**20.0**	36.0	**6.0**	9.0	8.0	**7.0**	11.0	**7.0**
	↳ per edge	0.41	**0.37**	0.67	**0.11**	0.17	0.15	**0.13**	0.2	**0.13**
	distortion per edge	**0.7**	0.79	0.74	**0.32**	0.41	0.41	0.59	0.6	**0.54**
	time in seconds	84.0	**74.0**	319.0	**45.0**	60.0	47.0	**109.0**	382.0	168.0
	optimality gap	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Overall sector deviation decreases or remains stagnant with increasing k with a more pronounced drop from $k=3$ to $k=4$ than from $k=4$ to $k=5$. In contrast, the actual distortion per edge of the computed map is smallest for $k=4$. This is surprising, since with a higher k and the same allowed discrepancy of one sector to the left or right, one would expect that edges are forced to be closer to their original edge direction. A possible explanation would be that the available directions for a single edge are too restrictive leading to a large number of conflicts and in particular a larger number of edges which have to "escape" to one of the neighboring sectors. This seems unlikely since as explained above the actual sector deviation per edge is mostly smaller for $k=5$ than for $k=4$. This leads us to conjecture that for our chosen test instances, the selected angles for the $k=5$ orientation systems happened to be less suited for the instances when compared to the angles of the $k=4$ orientation systems.

Comparing the aligned to the regular orientation system, we can see that, while they

are often comparable, in some instances like Montreal-3-(10, 5, 1) there is a significant difference in sector deviation of up to 0.17 per edge. In turn, Washington-4-(3, 2, 1) shows a 0.11 increase in sector deviation per edge for regular systems, indicating that here the heuristically chosen directions are in fact less suitable for a schematic map than the fixed aligned directions. The comparison with the irregular system looks very similar, although slightly more promising. For some instances (e.g., Montreal, Vienna, and Washington for $k = 4, 5$), the irregular orientation system is significantly better than the other two. Overall the deviation is comparable and for Berlin-4 it (somewhat surprisingly) tends to be worse than in the aligned setting, with differences of up to 0.3 (average sector deviation per edge).

The general picture of the actual angle distortion is again comparable for most instances between the regular and the aligned setting, with no clear trend visible. The irregular setting is also often close to the other two for this metric, with some notable exceptions. For Sydney-4 and Berlin-4 it is significantly worse, while for Washington-5, Sydney-5, and Berlin-5 the opposite is true. It might be the case that with increasing k, the irregular system is increasing its ability to approximate the slopes in the input in a more fine-grained way. Again, rather unsurprisingly, we increase the angle distortion, when we emphasize line straightness with weights (10, 5, 1). A surprising observation is that angle distortion is generally smaller for $k = 4$ than for $k = 5$.

Due to the time limit, runtime results are only comparable between the three smallest instances Montreal, Vienna and Washington. Here too, there is no clear trend. Montreal is solved quickest for $k = 3$, for Vienna, we see that $k = 4$ has the lowest runtime, while Washington acutally preforms best for $k = 5$.

3.5.3 Discussion

Our approach of increasing topographicity in metro maps through data-driven orientation systems seems to be working in a reasonable number of computed instances. Choosing an irregular orientation system is a valid option to increase topographicity, even if the irregular set of slopes is unfamiliar. Moreover we can see that in isolated instances distortion can be lower when restricted to a smaller set of directions, which might be an indication that some input maps lend themselves more naturally to a specific k, which is not always the octolinear $k = 4$ or the highest possible number.

The actual metro maps produced by our system, we can see one major caveat of our approach to minimize distortion by deciding the directions based on the input. While for most edges we have a very suitable representative direction in the orientation system, the constraints of the MIP might still force an edge to be drawn in a different sector. In Figure 3.4(c) the top left end of the yellow line is drawn to the top, despite there being a direction available which is closer to its general direction in the input. However due to the other yellow line occupying this direction already it is drawn in a direction with more topographic distortion.

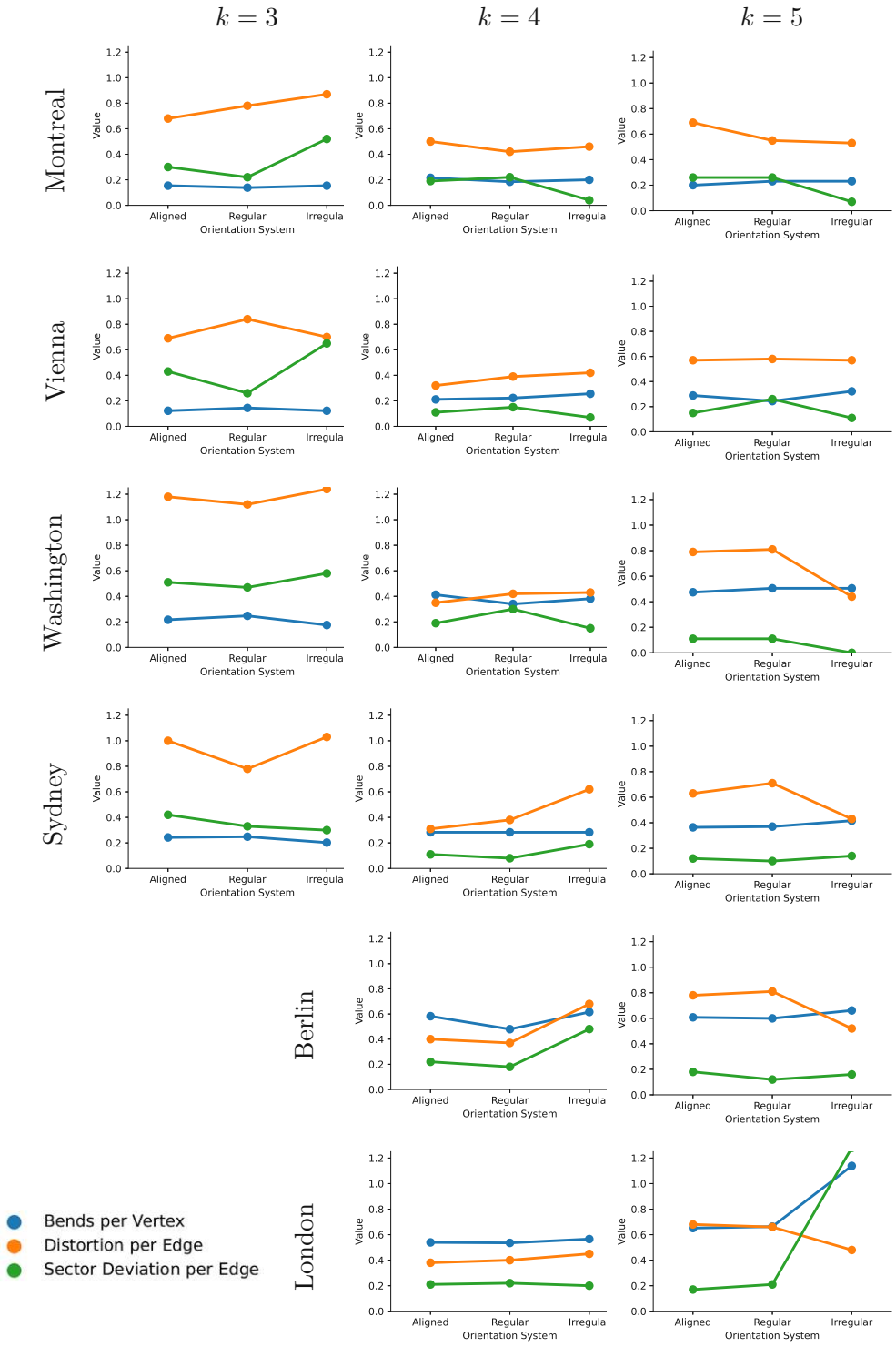


Figure 3.6: Plots of the results of the experiments for the objective function weights $(f_1, f_2, f_3) = (3, 2, 1)$. Every metric is shown as a colored line, the columns are the three orientation systems.

On a positive note we can see that irregular orientation systems can create metro maps that resemble the input more closely than typical aligned systems. This can be seen when comparing Figures 3.4 (a) and (c), (d) and (f) respectively. We can also see that most of the layouts, which are not using an aligned orientation system do not include the horizontal direction. This might be helpful in labeling these metro maps, since it is difficult to place the visually preferred horizontal labels along a horizontal line with clear association to a station.

Concerning the contraction method used, it should be pointed out that while this is a commonly used approach and decreases the number of edges significantly resulting in a beneficial effect on the runtime, metro maps very commonly employ bends within such a sequence of degree two stations. While it would be possible to encode such an additional bend in the MIP, it would require us to a priori fix where such a bend can be. This is a limitation of the MIP, since a number of possible and relevant solutions are cut off with this.

It remains to discuss the clustering method for irregular orientation systems. A stated in Section 3.3.2, we aimed to find a suitable irregular orientation system based on the slopes of edges present in the input map. To this end we employed a 1-dimensional k-medians clustering algorithm, partitioning these slopes into appropriate clusters, which are in turn represented by their median direction. We had to adjust for the circular nature of a set of slopes. This can be done by turning it into normal 1-dimensional data, via cutting between two elements and flattening the circular list of slopes into a regular 1-dimensional one. If we cut at all possible $|E|$ intervals between two consecutive elements, we can choose the clustering, which yields the lowest sum of intra-cluster costs and therefore the lowest overall error.

However this resulted in some instances whose constructed MIP did not admit a feasible solution. A reason for this could be that the resulting irregular orientation systems contain an orientation c with a large angular difference to its neighboring orientations. As a result the interval for slopes whose closest orientation is c becomes comparatively large. If four or more edges with a shared endpoint all consider c to be their representative orientation, any MIP which allows for a sector deviation of at most 1 (see Section 3.4.2) is infeasible. We would therefore advise to exercise caution in the use of irregular orientation systems, in particular when the differences between one orientation and its neighbors become large.

In contrast, by choosing the (arbitrary) cutting point at 0 degrees resulted in feasible instances for all tested cases. This underlines that the optimality of the clustering of slopes in the input does not necessarily correlate with the actual optimality of an irregular orientation system, but is instead a coarse approximation. Section 3.6 also points out how further research might aim to improve this.

3.6 Conclusion

We presented and implemented an adaptation of an existing MIP model for octolinear metro maps [98] that can draw metro maps schematized to any set $\mathcal{C}$ of arbitrary

orientations. This is supplemented by a data-driven approach to optimize the set $\mathcal{C}$ based on k-means clustering of the input edge orientations or by finding the best rotation of a regular orientation system. Finally we performed, presented, and discussed experiments of our system and its results for different real-world metro maps.

An approach to choose a suitable k for a given input might be to use the smallest k which generates visually appealing layouts in order to find a middle ground between the schematic appearance of the metro map and geographic accuracy. This leads to the general idea, that it is still important for a metro map designer to consider a number of possible layouts in different linearities for a given input network in order to find the most suitable metro map style. Hence our system should not be understood as a stand-alone method to metro map generation, but rather as an automated tool to help pre-select possible candidates for layouts and increase the number of layout settings a designer can explore at low time cost. This pre-selection might be refined in the future if a more global metric to judge the quality of a metro maps is devised.

As future work, we want to include station labeling [91]. Similarly, there is additional functionality which can be encoded in the MIP (like preventing bends in stations). One motivation for k-linearity is based on the idea that certain linearities are more suited to a metro network than others. It would be interesting to compare user performance between maps of different linearities with a user study. Further there is a clear path for additional investigations, when it comes to deriving the drawing directions based on the input. Currently the clustering is executed after contracting degree two stations. However, since the goal is to represent all edges as close as possible, there is a possibility to base the directions on the complete input map instead of the contracted one. And finally, the idea of basing the design decisions of the algorithm on the input data can be extended to, for example, the decision how exactly we should contract the chains of degree two edges. In particular in large transit maps, there are sometimes such chains, which form quite generous loops around other edges or geographic obstacles in their input geography, a feature which we might want to preserve to increase topographicity, but also to avoid representing such a chain with a direct connection between its start and end point, which might pose a significant obstacle.

As mentioned in the previous section, the connection between the optimality of the clustering of the edge-slopes and its suitability for a schematic map is unclear. A possibility to improve on this would be to consider a constrained clustering variant, which aims to capture the goal more directly. Since we suspect that the infeasibility of the MIP models stems from neighboring edges all being clustered in the same cluster and therefore all attempting to be drawn in the same direction, a clustering which constrains neighboring edges to be grouped in different clusters might improve this relationship.

CHAPTER 4

Including User Defined Geometric Motifs in Transit Maps

This chapter is (partially) based on the following publications:
[15]: Batik et al. – Shape-Guided Mixed Metro Map Layout
(Computer Graphics Forum)
[14]: Batik et al. – Mixed Metro Maps with User-Specified Motifs (GD'21, poster)

A metro or transit map used to show transportation line services is an intuitive, schematic representation of a transit network. Here, a schematic representation is a simplified network geometry (e.g., straightened lines, uniform spacing of stations, etc.), to facilitate effective way-finding activities [110]. Such nicely arranged representations make transit maps popular visual metaphors for network visualization in physics, biology, engineering, etc. [132]. However, manually creating schematic maps is not straightforward and requires intensive iterative processes. Automatic approaches have been thus investigated to solve this high-complexity problem [133], however, most existing approaches aim for one single style. For example, an *octolinear* (also *octilinear*) layout limits all edge orientations to horizontal, vertical, or diagonal at 45° [98, 116, 126]. A *curvilinear layout* constrains metro lines as continuous and smooth curves [53]. Other styles such as *concentric circles* [54] that align metro lines along concentric circles, or *multilinear* [89] designs that relax octolinear layout by allowing multiple edge directions were also proposed and their algorithmic complexity and perceptual effectiveness were investigated.

However, in reality, if we take a close look at the handcrafted metro maps by professional designers [32, 139], multiple styles are often incorporated in a single diagram. Designers often distinguish primary and secondary information to emphasize the difference between specific lines in the system. To achieve this, some lines in the map are arranged using a simple geometry (e.g., circle, etc.) or an iconic shape (e.g., heart, etc.), while the

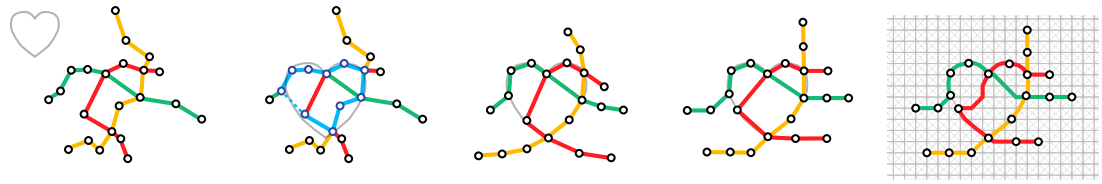
(a) Schematic Mapping Workshop logo (b) Embedded heart

Figure 4.1: Examples of shape-embedded layouts, including (a) the logo of Schematic Mapping Workshop 2019 (courtesy of Dr. Maxwell J. Roberts), and (b) an example illustration from One Metro World [32] (courtesy of Mr. Jug Cerović).

remaining lines follow the layout convention. The official Moscow map with a circular route highlighted is a typical example [86], while maps with more complicated shapes [139] are more often used for advertising purposes or special events [77, 128]. Figure 4.1a shows more examples, where a designer embedded the Vienna metro map to create a workshop logo [122]. Cerović explains the influence of shapes in mental map development [32]. Nevertheless, as summarized by Wu et al. [133], creating transit maps is an iterative process, and an automatic approach provides opportunities for scientists to effectively investigate advantages and disadvantages of various metro maps.

There exist automatic approaches for synthesizing metro maps with more than one style; however, supported geometry (i.e., shape) is limited to either polylines [136] or circles [134] to the best of our knowledge. This chapter presents a more general approach to the *mixed layout problem*, which allows the integration of specific shapes and classical metro map layout. This is achieved by introducing a user-defined *guide shape*, represented as single or multiple polylines, as input. Figure 4.1b shows an example, where we embed a heart shape into the Taipei metro system. This is done with a three-step approach by firstly matching routes for shape approximation, then deforming the network layout, and finally, aligning lines on a grid. Note that each individual step here has been categorized as a challenging problem [55, 133, 80]. To guarantee the visual quality of the generated maps, the guide shape should be recognizable in the final layout, while the remaining part of the layout should still fulfill the classical octolinear design criteria [98]. Our contributions in the chapter are summarized:

- A general definition and solution for the mixed metro map problem beyond the state-to-the-art.
- An algorithmic approach to perform shape matching, shape approximation, and metro line alignment as a whole.



(a) Input (b) Matching (c) Smooth (d) Octolinear (e) Grid alignment

Figure 4.2: Algorithmic pipeline of our layout approach. (a) The input metro network and a user-specified shape. (b) Shape and route matching, shown in blue. (c) Shape fitting using a smooth layout. (d) Shape fitting using an octolinear layout. (e) Final layout after aligning stations and lines on a modified grid, which includes the shape.

- Quantitative and qualitative evaluation to test shape recognizability and the corresponding influence on route planning tasks.

The chapter is structured as follows: In Section 4.1, we summarize the state-of-the-art literature relevant to our approach. We then begin with a definition of the proposed problem, followed by an explanation of the design criteria and a high-level description of our algorithm (Section 4.2). The method used to solve the proposed problem is detailed in Sections 4.3-4.5. We collect several results (Section 4.6), examine the approach performance and quality (Section 4.7), and conclude this chapter and propose open topics in Section 4.8.

4.1 Related Work

Metro maps are designed to help passengers navigate transit lines when taking trains in a rail system. Since train routes are fixed and passengers mainly need to know at which station to get on or off a train, some sort of distortions are allowed in metro maps for improving readability. Since exact geographic information is no longer needed, cartographers usually enlarge downtown areas to label station names and prevent visual clutters [102, 114]. The stations on metro lines are re-positioned to fulfill several criteria, such as even spacing of stations [42, 79], minimum direction changes in routes [108], and the relative positions of stations [111]. Generally, the layout of a metro map can be curvilinear, multilinear [111], k-linear (including octolinear, hexalinear, and tetralinear) [89], and even based on concentric circles [107, 93]. In addition to the network map, text or image labels are placed around stations for passengers to connect the map and the real world [103]. The labels are expected to be overlap-free, close to their corresponding stations, and have consistent orientations if they represent neighboring stations.

Schematic Layout Approaches. So far the most commonly used metro map layout is octolinear, which was introduced by Henry Beck for the London Underground in 1933. It is known that computing an optimal octolinear layout is NP-hard [96]. Several categories of methods were then introduced to pursue high-quality metro map layouts

within a reasonable computation time. The first category is line simplification, which attempts to find a set of linked line segments to represent each path on the map [87, 83, 49, 31, 78]. During the simplification process, the deviation between each station and the closest line segment is bounded, and the number of line segments is minimized. The second is local optimization. Methods in this category iteratively move stations to nearby positions to align edges in octolinear directions [8, 6]. Several strategies, such as simulated annealing [130], ant colony optimization [129], magnetic force field [37], and least-squares optimization [45] were applied to prevent the algorithm from falling into local minimums. The third category is to formulate a metro map layout problem into a mixed-integer linear program [98, 134], which distinguishes hard constraints that must be fulfilled and soft constraints that are globally optimized. Typically, the octolinear layout is mandatory since edge orientation is formulated using a set of binary variables in the model, whereas other aesthetic requirements can be relaxed if they conflict with others. Finally a recent result computes shortest paths between candidate positions of stations on a predefined grid [11, 10]. This was formulated as an ILP, but heuristic solutions through repeated applications of shortest path algorithms can be obtained very quickly. In addition to methods belonging to the above-mentioned categories, the works of [126] and [127] applied a two-step deformation technique to achieve the goal. Their methods first compute curvilinear layouts of metro maps and then rotate each edge to the closest octolinear direction. Each step solves a least-squares optimization problem. Compared to previous works, our approach allows embedding a guide shape with arbitrary edge directions into a schematic metro map, which is challenging to integrate with their frameworks.

Station Names Labeling. Station names are essential in metro maps, yet the labeling problem itself is also NP-hard [59, 91]. Several previous methods solve the layout and the labeling problems in two steps to reduce the computation cost [126, 135]. Niedermann and Haunert in particular present a general framework for labeling network maps [91]. Since occlusions could be inevitable when the layout does not provide enough space for labeling, the methods of [91, 138, 118] systematically scale certain edges on the map to solve the problem. There are also methods considering both layout and labeling problems simultaneously when computing metro maps [117, 116, 98]. Please refer to [133] for details as our work does not focus on the labeling problem.

4.2 Overview

Our approach was developed by firstly investigating several real-world examples [139, 86, 32] and existing design criteria [110, 133], and then summarizing common design principles for our mixed metro map layout. Secondly, we transform these design principles to aesthetic constraints. Finally, these aesthetic constraints are formulated into equations for a mathematical model that can be solved systematically. We will demonstrate these design principles, followed by an introduction of our approach pipeline achieving these goals.

4.2.1 The Mixed Metro Map Problem

We define the mixed metro map problem more generally by relaxing the user-specified shape constraints introduced by Wu et al. [136, 134]. Note that to retain consistency, we follow the terminology and symbols used in the recent transit map survey [133]. A transit network is defined as $N = (S, C)$, where $S = \{v_1, v_2, ..., v_n\}$ represents a vertex set of stations and $C = \{e_1, e_2, \ldots, e_m\}$ is an edge set describing station connectivity. A *path cover* T of N is used to describe a set of paths, in N, indicating the different service lines in the transportation system. Note that each connection $e \in C$ belongs to one or more paths in T. Moreover, a *guide shape* P, which can consist of one or more open or closed polylines $P = \{p_1, p_2, ..., p_k\}$, is introduced to express the user's design goal.

4.2.2 Design Principles

Based on criteria collected for layout approaches [133] and open criteria raised by the design studio [139, 32], we summarize the design principles for our mixed metro maps as follows:

D1 Constrained Layouts: To increase legibility [133, 110] line configurations are often simplified to an octolinear design [98]. In such a layout, the preservation of the input topology is often assumed as a constraint [133].

D2 Topographic Accuracy and Relative Positions of Stations: The topographic accuracy has been discussed as an important factor in a schematic representation [110] to preserve users' mental map of a city. Relative positions between pairs of stations should be maintained.

D3 Spatially-separated Stations: To avoid overlaps between stations, a minimum distance between stations is incorporated. This distance is preferably uniform [58, 98].

D4 Simplification of Trajectories: Paths are expected to be as smooth as possible [58, 98].

The aforementioned design rules have been defined previously [133] and are applied to real-world metro maps in general. In this chapter, we further propose design criteria, which allow us to achieve the desired affect when embedding the guide shape in the final layout.

D5 Translation and Scale Invariant Shape Embedding: The shape can be translated and scaled when embedding it, while we exclude rotation to avoid adding another layer of recognition complexity [120].

D6 Shape Representation: The edges in the transit network should be appropriately selected to approximate the shapes. Perceptually, the embedded shape in the layout should be recognizable.

D7 Hybrid Edge Orientation: Edges that are used to approximate the guide shape should follow the shape structure, while the remaining edges follow the aforementioned classical design.

4.2.3 Mixed Metro Map Pipeline

Figure 4.2 gives a conceptual overview of the proposed algorithmic pipeline. Initially, the user needs to select a transit network and provide a target shape as input to our approach. Figure 4.2(a) shows an example, where the transit network is drawn with colored lines and a user-specified shape (heart) is marked in gray. In the second step, as shown in Figure 4.2(b), we scale and translate the guide shape in order to find an appropriate sub-network to approximate this shape (Section 4.3). Once the guide shape is uniformly scaled and translated, we adapt a two-step deformation procedure inspired by Wang et al. [126, 127] in order to create a relatively well-representative network embedding. Here, we first generate a smooth layout (Figure 4.2(c)) to balance mutual distance between stations and straighten lines with degree 2 stations, followed by synthesizing a nearly octolinear layout (Figure 4.2(d)). This optimization process is detailed in Section 4.4. In the last step, to guarantee that the metro lines are fully aligned on a grid (Figure 4.2(e)), we introduce a grid alignment approach while retaining the mixed layout structure (see Section 4.5).

4.3 Route Matching and Shape

We compute a metro map layout that exhibits the given guide shape. Limited by the metro network's route structure, determining the optimal size and position of the guide shape is essential. However, we do not rotate either the guide shape or the metro map when computing the map layout because their orientations are restricted. For example, the upright direction of a *"heart"* shape is meaningful to humans; and the relative positions of stations help passengers navigate themselves in an urban area. An upside down metro map would conflict with users' mental map and decrease the map's usability. To determine the position of a guide shape, we first determine a path W consisting of metro connections $e \in C$. Then, we scale and translate the guide shape P to align with the path W. Note that the guide shape's size and position will not be updated during the deformation process.

We focus on an automatic approach for aligning guide shapes and metro maps, which determines a path inside the transit network visually similar to the guide shape P. However, as outlined at the start of this chapter, there are use cases (artistic or advertising purposes), where specific lines must lie on iconic shapes. We optionally let designers define W manually.

4.3.1 Automatic Route Matching

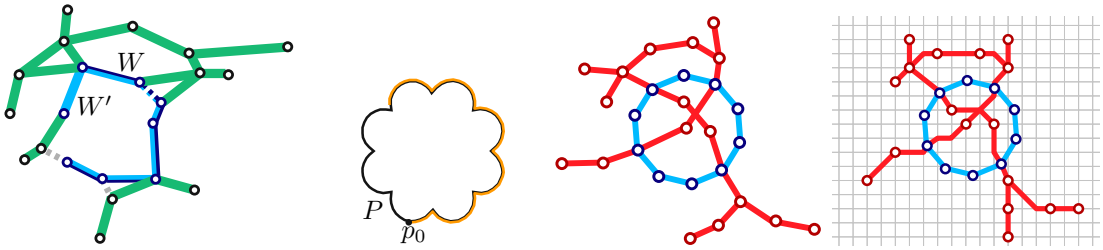
We search for a path $W = (e_1, e_2, ..., e_k)$ on the transport network that is visually similar to the user-defined guide shape. Since the guide shape P can be arbitrary and the path W similar to P may not exist, we insert dummy edges into the network to enhance visual quality. Specifically, if the geographic distance between two stations v_i and $v_j \in S$ is smaller than a predefined threshold and no edge connecting v_i and v_j exists, we insert a dummy edge to connect the two stations (Figure 4.3). We prefer to use as few dummy edges as possible to represent the guide shape.

Next we adapt the shape-preserving Dijkstra algorithm [56] to find a path W that can approximate P. The cost of a path in the shape-preserving Dijkstra algorithm, in contrast to the normal shortest path algorithm, is given by similarity between the path and P. Specifically, we quantify the difference between the path W and the P using the direction-based integral Fréchet distance [19] between W and P, which we write as $\delta(W, P)$. We also use the concept of the *partial similarity* [19] $\delta_{partial}(W, P)$, which is defined by a subpath P' of P, which minimizes $\delta(W, P')$ (see Figure 4.3).

The cost of a connection is given by the difference between the slope of the metro connections and the slope of the corresponding section of the guide shape. So in case the difference between those slopes is small, the additional cost of the connection for the path is small, too. Rather than considering the Euclidean distance between points in the original shape-preserving Dijkstra algorithm [56], the direction-based integral Fréchet distance is beneficial for our use case because it is scale and translation invariant (i.e., P is given with an arbitrary size and position). It provides a robust mapping between P and W, and small variations between two lines are not disproportionately penalized.

In our implementation, we grow the path W from a station on the metro map until none of the inserted stations can reduce the Fréchet distance. Let W_0 be the initial empty path and W_k be the path composed of $k + 1$ stations. Namely, the growing process can be expressed as $W_0 \to W_1 \to W_2 \to ... \to W$. At each step k, we examine all neighboring stations of W_k and insert the station $\mathbf{v}$ into a queue if it can reduce the direction-based integral Fréchet distance. In other words, $\delta(W_k + \mathbf{v}, P) < \delta(W_k, P)$. Then, for each station $\mathbf{u}$ in the queue, we compute the partial matching $\delta_{partial}(W_k + \mathbf{u}, P)$ to obtain a subsection of the guide shape P that is the most similar to $W_k + \mathbf{u}$. The station $\mathbf{u}$ that has the shortest distance $\delta_{partial}$ will be inserted to extend the path from W_k to W_{k+1}. Note that we only consider subsections of the guide shape P with the intent, that W_{k+1} approximates P better than W_k in each iteration. The algorithm repeats until the queue is empty.

Using paths with a few dummy edges to represent the guide shape is preferable. To achieve this goal, we give dummy edges a high cost and real metro edges a small one. Since the starting vertex of path W is unknown, we run the Dijkstra method from each potential starting station and select the path W that has the least cost $\delta(W, P)$. If the guide shape consists of multiple polylines, for example the eye-formed shape shown in Figure 4.10, we perform the route matching only for the the polyline containing the top



(a) Path W visually similar to the guide (b) (Partial) guide shape (c) Smooth deformation (d) Grid alignment

Figure 4.3: (a) A path W, which is visually similar to the guide shape and a possible next iteration W'. Dummy edges are shown with dotted lines. (b) The complete guide shape P is shown in black and a subsection of the guide shape used for partial similarity matching is drawn with an orange stroke. (c) Smooth deformation and (d) grid alignment steps, where shape stations $v'_i \in S'_{\text{shape}}$ and the corresponding edges that approximate a circular guide shape are visualized in blue, while stations and connections that do not approximate the guide shape are shown in red.

left vertex, and all other polylines are placed at the correct scaling and offset to the matched one.

Once the path W is obtained, we transform the metro network to align W with the guide shape P. As mentioned previously, we assume that the guide shape and the metro map are in the right orientation, and thus the transformation contains only translation and scaling. For simplicity, the alignment is achieved based on the bounding boxes of P and W.

4.4 Deformation

After aligning the guide shape P with the metro network, we deform the metro network to fulfill the design principles outlined in Section 4.2.2. This process is inspired by the two-step approach of Wang et al. [126]. First, we create a smooth layout (Figure 4.2.c) that aims to space stations evenly, avoid sharp bends, and maximize angular resolution. Additionally, the guide shape P is approximated by aligning metro stations with segments of P (Figure 4.4). The smooth, as well as the mixed layout, are created using the least squares optimization by minimizing constraints iteratively.

In the following sections, we denote the geographic position of a station by $v_i \in S$, the transformed position in the smooth optimization stage by $v'_i \in S'$, and in the mixed stage by $\tilde{v}_i \in \tilde{S}$. We assume that the input metro network is planar. Otherwise, we planarize the network by inserting dummy stations if two connections intersect.

To approximate the guide shape, a subset of stations $v'_i \in S'_{\text{shape}}$ are pushed toward the guide shape segments that are located closest to them. As a result, the edges $e'_i \in C'_{\text{shape}}$ connecting those stations can represent the guide shape P. Figure 4.4 illustrates the connections and stations that should and should not approximate the guide shape. It

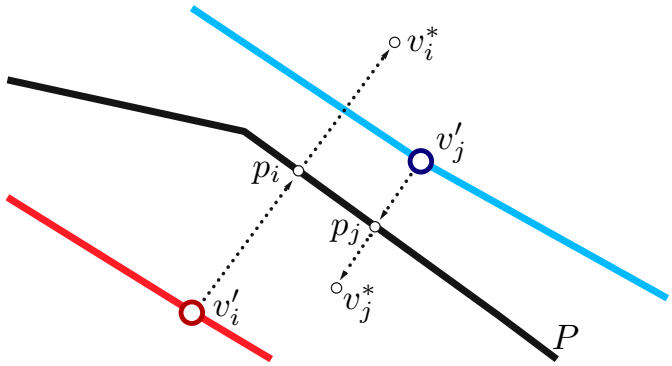


Figure 4.4: The station v_j', rather than v_i', is used to approximate the guide shape. This guarantees that no other metro line is closer to the guide shape. Therefore, v_j' is added to S'_{shape}, but not v_i'.

deserves noting that we use W only to align the guide shape with the network, rather than to define C'_{shape}. The reason is that the automatically computed path W represents only the overall shape of the guide shape and may miss details. Therefore, we update S'_{shape} at each step of the smooth deformation. This is done as follows: let v_i^* be a reflection of v_i' over the closest point p_i on the guide shape. We assign v_i' to S'_{shape} if the line segment between v_i' and v_i^* does not intersect any of the edges in C', as illustrated in Figure 4.4. Otherwise, such an intersection implies that another station and metro line must be closer to the guide shape.

4.4.1 Smooth Layout

We optimize four constraints to compute the smooth layout,

$$\Omega_{smooth} = w_c\Omega_c + w_l\Omega_l + w_a\Omega_a + w_p\Omega_p. \tag{4.1}$$

Specifically, Ω_c forces the layout to approximate the guide shape. Ω_l causes uniform edge lengths, Ω_a maximizes angular resolutions, and Ω_p minimizes the distance of the position of a metro station $v_i \in S$ to its geographical location. The corresponding weight to balance the potentially contrary constraints is denoted by w. Let $D(v_i')$ be the degree of v_i'. The constraint Ω_c penalizes the distance of smooth stations and the polyline and is given as

$$\Omega_c = \sum_{v_i' \in S'_{\text{shape}}} D(v_i')|p_i - v_i'|^2. \tag{4.2}$$

To implement the Design principles D1 to D4 (outline in Section 4.2.2), three energy terms are applied to all stations $v_i \in C$. The constraint Ω_l forces stations to be evenly spaced. This eliminates the information about the geographic distance between stations,

but creates a more uniform and balanced layout. Ω_l is given by

$$\Omega_l = \sum_{\{i,j\} \in C} |(v'_i - v'_j) - s_{ij} R_{ij} (v_i - v_j)|^2 \ ,$$
$$\text{where} \quad s_{ij} = \frac{L}{|v_i - v_j|} \quad \text{and} \quad R_{ij} = \begin{bmatrix} \cos\theta_{ij} & -\sin\theta_{ij} \\ \sin\theta_{ij} & \cos\theta_{ij} \end{bmatrix}, \tag{4.3}$$

L denotes the target length of the edges. This length is equal for all edges where stations are not dummy nodes, and equals the average length of the metro connections in the initial layout. In case v_i or v_j of $\{i,j\} \in C$ is a dummy node and not a regular station, the target length for $\{i,j\} \in C$ is $L/2$. Besides, θ describes the unknown angle of a connection $\{i,j\} \in C'$ and R is a rotation matrix, ensuring that the rotation of the connection is not penalized.

The constraint Ω_a aims to maximize the angle between connections sharing a station. This separates the different metro lines in case of station with a degree > 2 (e.g. an interchange), making it easier for the user to distinguish two metro lines. In case of a station with degree $= 2$, the metro lines are straightened naturally. Ω_a is formulated as

$$\Omega_a = \sum_{v'_i \in S'} \sum_{\{i,j\},\{i,k\} \in C'_{\text{shape}}} |v'_i - (v'_j + c'_{ij} + \tan(\frac{\pi - \theta_i}{2}) c'_{jk})|^2, \tag{4.4}$$

where $c'_{ij} = \frac{1}{2}(v'_k - v'_j)$. θ_i depends on the number of outgoing connections of v'_i, namely, $\theta_i = 2\pi / D(v'_i)$.

To preserve the overall geographic structure of the metro system, we add the energy term Ω_p. This term can avoid large deformations of the network and avoid a conflict with the mental map of users. For example, to avoid a layout where stations located in the north of a city are moved to the south. Ω_p is given as:

$$\Omega_p = \sum_{v_i \in S} |v'_i - v_i|^2. \tag{4.5}$$

4.4.2 Mixed Layout

Following the smooth optimization stage, we create a mixed layout. Similar to the previous step, we differentiate between connections $\tilde{e}_j \in \tilde{C}_{\text{shape}}$ that aim to approximate the guide shape and octolinear connections $\tilde{e}_i \in \tilde{C}_{\text{octo}}$ that do not contribute to the recognisability of the shape. Octolinear edges should have an octolinear slope (e.g. a multiple of $\frac{\pi}{4}$), whereas shape connections approximate P and should align to a section of the guide shape P. The differentiation between octolinear and shape connections is based on the assignment used for the smooth stage. Therefore, if a connection was in C'_{shape} in the smooth layout, then this edge is treated as a shape edge $\tilde{e}_i \in \tilde{C}_{\text{shape}}$. Otherwise, the edge is an octolinear one.

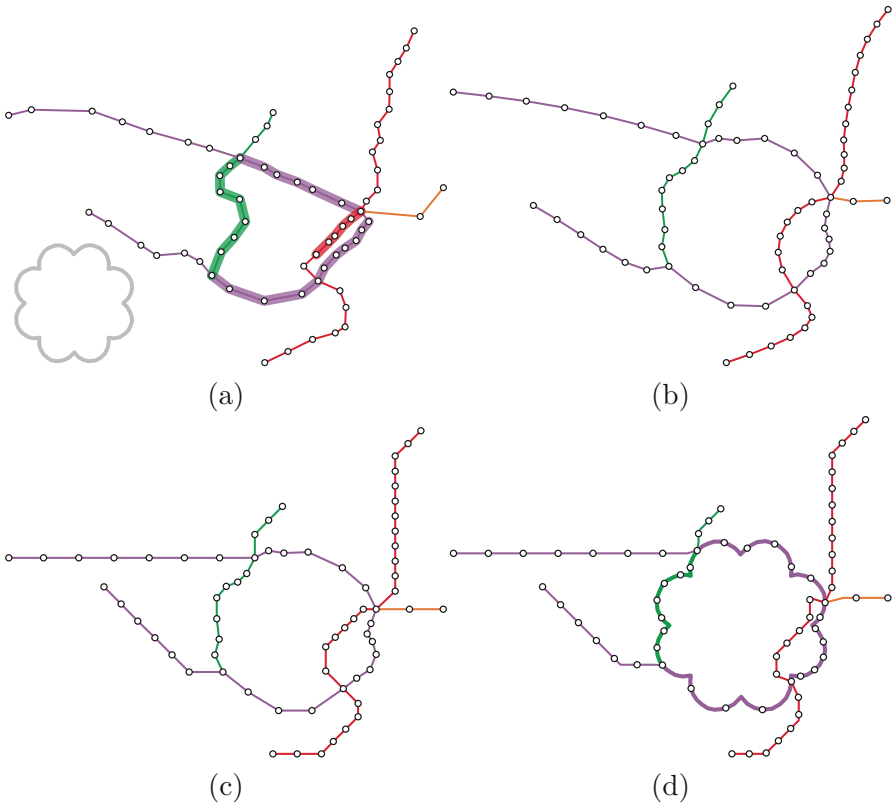


Figure 4.5: Montreal metro map created with a cloverleaf-shaped guide inspired by the city's logo. (a) Geographically accurate layout with the automatically computed path P highlighted. The input guide shape is shown on the bottom left. (b) The smooth layout, (c) the mixed layout and (d) the final grid-aligned layout.

To determine the mixed layout, each connection $\tilde{e_i} \in \tilde{C}_{\text{octo}}$ is assigned to its closest octolinear slope. In case of a conflict, the Hungarian algorithm is applied to reassign the octolinear slope of each edge by minimizing the sum of rotations. We minimize the term

$$\Omega_o = \sum_{\{i,j\} \in \tilde{C}_{\text{octo}}} |(\theta'_{ij} + \Delta\theta_{ij}) - \tilde{\theta}_{ij}|^2, \tag{4.6}$$

to rotate octolinear edges $\tilde{e_i} \in \tilde{C}_{\text{octo}}$, where θ'_{ij} denotes the angle of the connection after the smooth optimization, $\Delta\theta_{ij}$ is the difference between θ'_{ij} and the target angle, and $\tilde{\theta}_{ij}$ is the current slope of the connection $\{i,j\} \in \tilde{C}_{\text{octo}}$. Further we apply the constraints Ω_p and Ω_c equally as for the smooth layout. The mixed layout is than computed by minimizing

$$\Omega_{mixed} = w_o\Omega_o + w_p\Omega_p + w_c\Omega_c. \tag{4.7}$$

During the smooth and the mixed deformation stages, the planarity is checked after each optimization step. In case of an intersection the corresponding stations are moved back

to the position at the previous iteration. An energy term is applied to avoid the same intersection in the following optimization step. If the distance between a station and a connection is below a pre-defined threshold, we apply an energy term to move them apart.

4.5 Grid Alignment

While the previous mixed layout step tries to align the shape edges with the shape and the remaining edges with an octolinear direction, the resulting layout may still contain some edges that are not fully octolinear yet. Our goal is to compute a final layout, which is as similar to the mixed layout as possible, while all edges of C_{octo} are exactly aligned with one of the octolinear directions and all edges of C_{shape} are tracing the guide shape.

This issue can not easily be resolved by simply snapping every edge to the closest octolinear direction. However we can adapt an existing layout algorithm developed by Bast et al. [11], which computes octolinear layouts of metro maps given as geographical inputs on predefined grids, and renders edges as piece-wise octolinear curves.

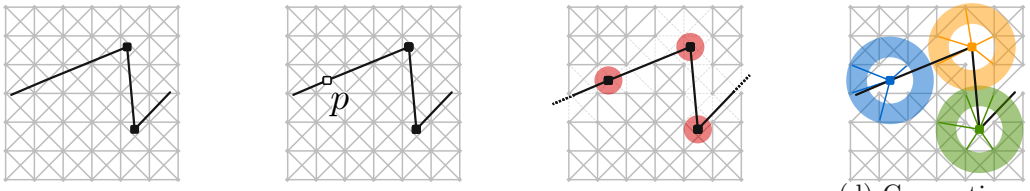
We will proceed by giving a description of the original approach first and then we will describe all adaptions we made to accommodate for our setting.

4.5.1 The Octi framework

In their Octi framework for metro map layout, Bast et al. [11, 10] consider a geographic network N as input. Let x_{min}, y_{min}, x_{max} and y_{max} be the minimal maximal x and y coordinate over all stations in N respectively. A grid graph $G = (V_G, E_G)$ (with cell diagonals) is created, with a cell size d equal to the average distance between two stations in N multiplied by a factor f_d with $\lceil (x_{max} - x_{min})/d \rceil$ columns and $\lceil (y_{max} - y_{min})/d \rceil$ rows. The dimensions are chosen, such that, N can be placed on top of G and the axis-aligned bounding box of N is completely contained in that of G. Now every edge $\{a, b\}$ of the grid is replaced with a path a, a_b, b_a, b of length three. The two nodes a_b and b_a are called the port of a in the direction of b and the port of b in the direction of a, respectively, while a and b are called the sink nodes of their respective ports. Note that every port node has as many ports as its original degree. Finally for every sink node, edges are introduced to create the complete graph on its port nodes. Every edge between two ports of different sink nodes has weight c_{cop}, every connection of a port to its own sink node has a (sufficiently large) weight c_{sink} and every connection $\{a_b, a_c\}$ in the complete graph between port nodes of a node a have one of four possible values, inversely proportional to the angle $\measuredangle bac$.

Let $x(s)$ and $y(s)$ be the x and y coordinates of a station s in N. Now the candidate set $V^c(s)$ of a station s is defined as all sink nodes of G, which have a distance of r or smaller to $(x(s), y(s))$.

By iterating over all edges of N in a computed order Σ, the edges are routed as paths through the grid. For the first edge $\{s, t\}$, this is done – conceptually – by temporarily



(a) P on G (b) Subsampling (c) Removal (d) Connections

Figure 4.6: Adaption of the underlying grid, based on the guide shape P. (a) P is overlayed on G. (b) P is subsampled to ensure similar distances on P as on G. (c) Edges of G, which intersect P and vertices of G, which are too close to a vertex of P are removed. (d) Vertices of P are reconnected with vertices of G, which are within a certain distance range.

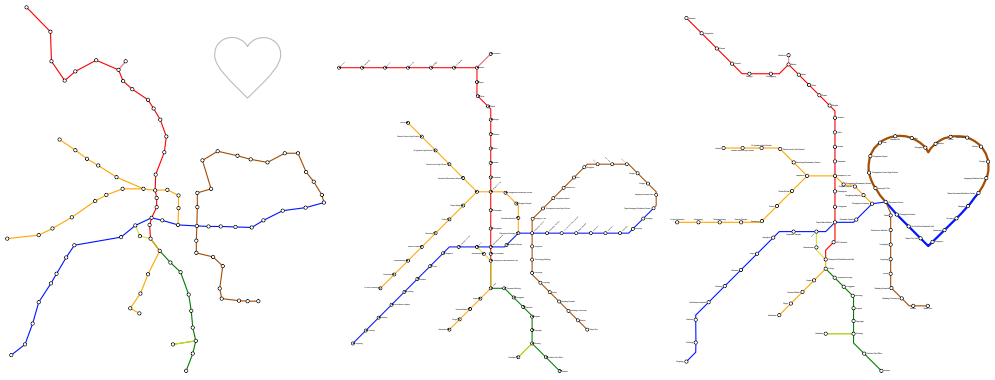
adding *virtual* vertices a_s and a_t to G and adding the edges (a_x, k) for $k \in V^c(x)$ and $x \in \{s, t\}$. Every such edge is relative to the distance of x and k. Now a shortest path $\Pi_{s,t}$ between a_s and a_t is computed, a_s and a_t are removed from G, the positions for s and t are fixed to the first and last sink node of $\Pi_{s,t}$ and the piece-wise octolinear path between s and t is obtained by concatenating all sink nodes of $\Pi_{s,t}$ or the sink nodes of port nodes in $\Pi_{s,t}$. This is now done for all edges of Σ. Should an endpoint of an edge already be fixed to a grid node, no virtual vertex is added and the shortest path is instead computed to or from the already fixed position.

Two main factors require adaption of this process, namely the additional design principle D7, i.e., the presence of the guide shape and that we do not use an input map with geographical station positions, but the mixed layout, which lets us assume that our input is already reasonably close to the final layout we want to compute.

4.5.2 Adaptions

We reduce the size of the map by removing stations and reinserting them afterwards, similar to Section 4.4.2. While the original algorithm [11] uses a similar approach and retains any vertex with degree three or higher, we also retain every vertex of S_{octo} of degree two, if its two adjacent edges are not assigned to the same (or opposing) octolinear angles, since the mixed layout already aims – in accordance with design principle D4 – to simplify line trajectories. Moreover, the mixed layout also, by design principle D6, approximates P and we therefore also keep all stations in S_{shape}.

Next, all computed paths are paths in G. Recall that P is given as a polygonal shape. To ensure that the distances along the shape are comparable to the average distances in the grid, P is subsampled, however, no vertices are removed towards this goal to ensure that no essential features of P are lost. We need a suitable representation of P in G. This is done by overlaying P onto G, removing all edges of G, which cross an edge of P and all vertices, which are closer than a small threshold value d_{min} to a vertex of P. Then we reconnect a node $p_i \in P$ with a node $a \in G$ if their distance d' is $d_{low} \leq d' \leq d_{up}$, where



(a) Input and guide (b) Octolinear map (c) Map with guide

Figure 4.7: Taipei Metro map (a) created without (b) and with (c) a heart-shaped guide, shown in the top right of (a). The final maps (b & c) have been labeled as a post-processing step.

$d_{low} < d < d_{up}$. We choose d_{low} and d_{up}, such that the connecting edges are roughly the same size as the cell size of the grid. Since the nodes on the guide shape are not perfectly aligned with nodes in G the connecting edges are not necessarily octolinear. We could approximate the shape using nodes of G, however we emphasize design principle D6 over D7 and allow some edges to deviate from the octolinear or shape aligned directions. This process is illustrated in Figure 4.6

The newly introduced edges also need to be assigned weights. Port-to-sink connections for shape nodes are weighted the same as for normal nodes of G. Edges between port nodes of different sinks are weighted with $c_{hop}/2$ or $c_{hop}/20$ if one or both sinks belong to P respectively. These cost reductions are intended to encourage the routing of edges in N using edges of P. Connections between port nodes of the same sink node are again weighted inversely proportional to the angle of their connected edges, however, since the shape is not necessarily octolinear, we need to be able to derive these weights based on any arbitrary angle θ and we therefore use $2\pi - \theta$. Since $\theta \leq \pi$, we retain the desired property that no path of length two between two ports of the same sink is ever cheaper than their direct connecting edge.

Since the mixed layout already assigns and aligns stations and edges with the guide shape, we first and foremost want to ensure that these edges are still routed along edges of P. Therefore we ensure that all edges of C_{shape} appear in Σ before any edge of C_{octo}.

When computing $V^c(s)$ for a station $s \in S_{shape}$, we drastically reduce its size. In particular, we only keep the two nodes of G which, based on our experiments, are closest to s. After routing all edges, all previously removed stations are reinserted and finally all maximal sequences of stations of degree two are equally distributed.

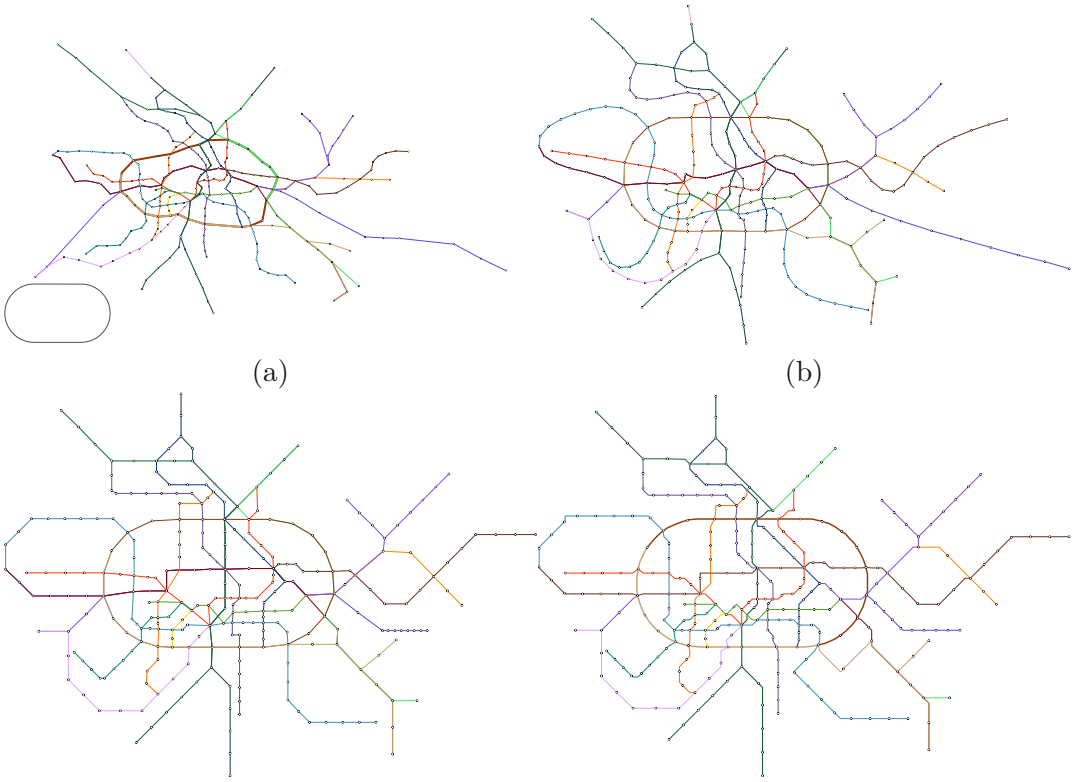


Figure 4.8: Berlin metro and S-Bahn map, consisting of 272 stations. The layout is created with a stadium-shaped guide and a path W for the stadium is provided. (a) Geographically accurate layout with the path W highlighted. The input guide shape is shown on the bottom left. (b) The smooth layout, (c) the mixed layout and (d) the final grid-aligned layout.

4.6 Experimental Results

The route matching and the computation of the smooth and mixed layout was performed on a a MacBook Pro (2017) with a 2.9 GHz Quad-Core Intel processor. The grid alignment was implemented in Python 3.8 using the NetworkX 2.6.3 module for shortest path and the bentley-ottman module 7.2.0 for segment crossing computation. It was computed on a standard laptop running Ubuntu 21.20 with an Intel Core i5 processor (8 $\times$ 2.6 GHz) and 16GB of memory. The grid size factor f_d was 0.2 for Moscow, Paris and Berlin and 0.3 for all other instances. The edge weights in G were set to $c_{hop} = 20$ and d_{min}, d_{low} and d_{up} were set to $\frac{d}{5}$, $\frac{d}{2}$ and $\frac{3d}{2}$, respectively.

Our test cases can be grouped into two scenarios/potential use-cases for our approach. In the first cases the path W was computed automatically as outlined in Section 4.3.1. Figure 4.7 shows the metro system of Taipei with a heart shape, and Figure 4.5 shows the

Network	Figure	Guide	Route(s)	Smooth(s)	Mixed(s)	Total(s)
Lisbon (60)	Figure 4.11	Coast	–	1.47	0.39	1.86
Montreal (68)	Figure 4.5	Clover	3.82	1.67	0.20	5.69
Taipei (96)	Figure 4.7	Heart	4.94	4.77	0.53	10.24
Moscow (204)	Figure 4.12	Circle	–	39.44	5.98	45.42
Berlin (272)	Figure 4.8	Stadium	–	133.21	21.12	154.33
Berlin (272)	Figure 4.9	Bear	80.15	167.47	25.21	272.84
Paris (304)	Figure 4.10	Eye	223.52	189.45	25.46	438.43

Table 4.1: Test instances and guide shapes with number of vertices (after planarization and before subsampling, respectively) in parenthesis with their corresponding runtimes for the route matching and deformation processes. All times are given in seconds. Shapes marked with * were manually defined, and therefore have no route matching time.

one of Montreal with a cloverleaf shape, taken from the city's logo. Figures 4.9 and 4.10 use more complex guide shapes embedded in larger metro networks, demonstrating the capabilities and limitations of our approach.

For the second type of test case, we provide a manually defined path W. In Figure 4.8 a layout of the Berlin metro and train network is created where the ring line is emphasized in an stadium shape. The provided path W is highlighted. For the Moscow system (Figure 4.12) a similar concept is applied. For the Lisbon metro network (Figure 4.11) a schematic representation of the coastline guide the train connection reaching from top right to bottom left. Table 4.1 lists all the presented test-cases, the number of stations of the networks, the guide shape, and if W was provided or automatically computed.

4.7 Evaluation and Discussion

In this section, we examine the performance of our approach, and the visual quality of the maps generated using our system.

4.7.1 Performance Evaluation

The measured computation times for the figures of the chapter are summarized in Table 4.1. We observe that the running time increases based on the network sizes and route sizes. As the grid alignment is built based on the shortest path algorithm with complexity $N(|V|^2)$, the runtimes are not yet competitive for real-time applications (a minute for smaller instances like Montreal and Lisbon, multiple minutes for medium instances like Singapore and Moscow and up to 15 minutes for the large instances Berlin, Paris, since we did not parallelize our code).

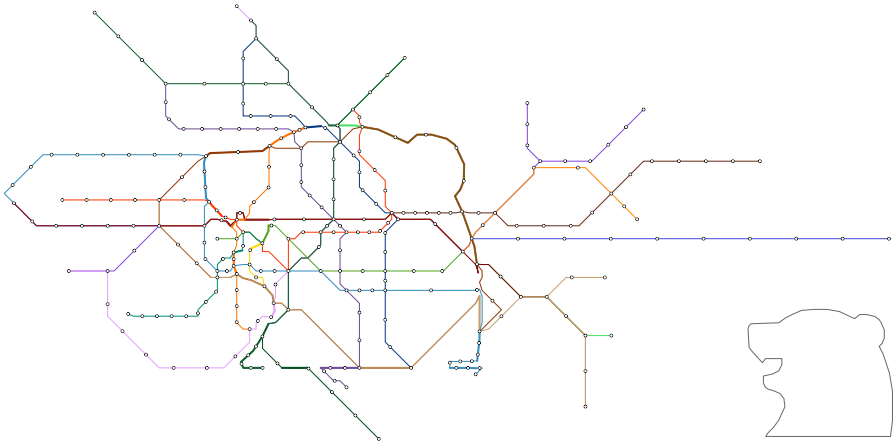


Figure 4.9: Berlin metro and S-Bahn map created with a bear-shaped guide, shown in the bottom right. The guide shape is inspired by the logo of Berlin.

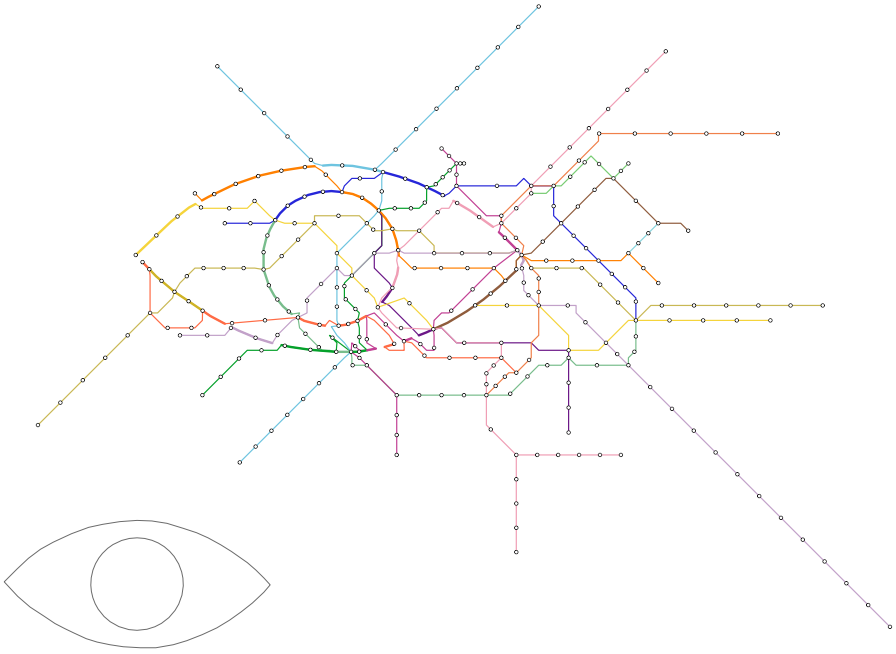


Figure 4.10: Paris metro map created with an eye-shaped guide consists of multiple polylines, shown in the bottom left.

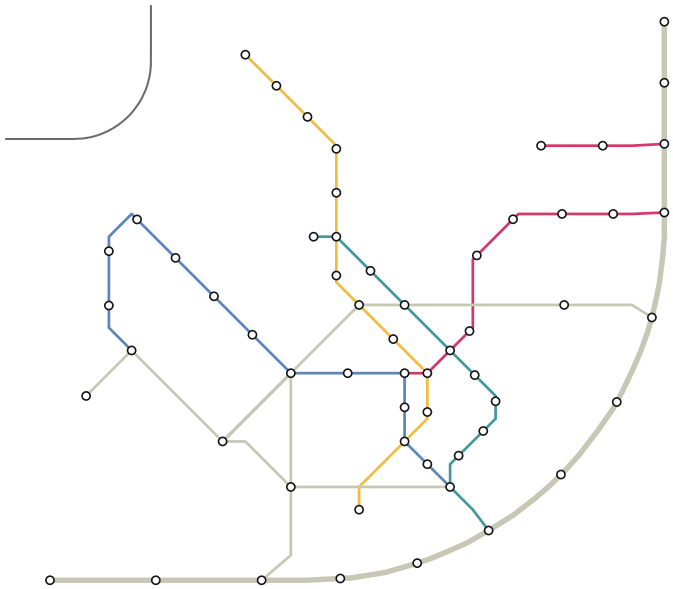


Figure 4.11: Lisbon metro and train map. Created with an curved guide shape, aiming to schematize the right and bottom train line that follows the coast. The guide shape is shown at the top left.

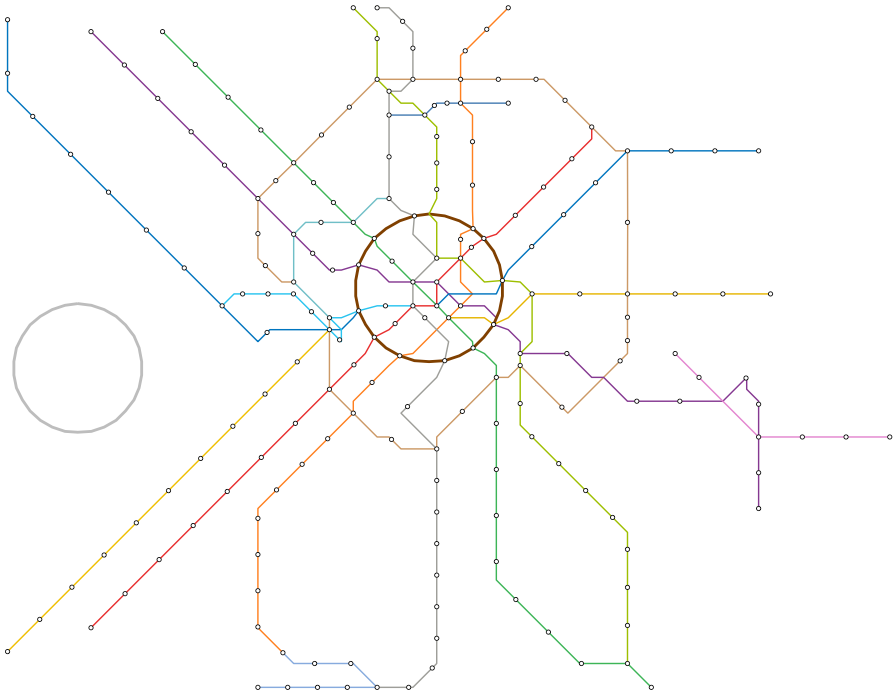


Figure 4.12: Moscow metro map. Created with a circular guide.

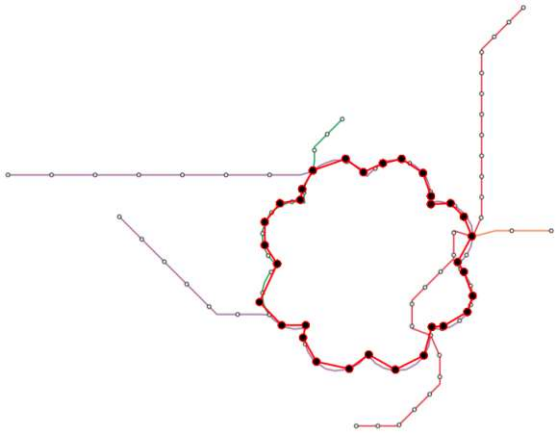


Figure 4.13: An example of the shape recognition task.

4.7.2 Visual Quality Evaluation

To examine the visual quality of our layouts, we conducted an online user study. As previously summarized in Section 4.2.2, one goal of our approach is to make the embedded shapes recognizable on maps (D6) so that they become helpful for general route-finding tasks that can be done on the maps. Therefore, we are interested in two research questions, including (1) *Are shapes in maps significant enough to be identified?* and (2) *Are embedded shapes helpful or harmful for route planning tasks in comparison to classical design?* For the *shape recognizability* (E1), we asked the participants to mark a shape if they saw one using a polyline drawing tool via mouse clicking. To evaluate the *map usability* (E2), we explicitly asked participants to trace and count the number of stations between a start station and a destination station. The tasks are done on classical octolinear maps and our mixed metro maps. To analyze the result of (E1), we overlay the polyline coordinates with the map and annotate each result manually. As for (E2), we analyze task accuracy and time performance on both map styles. The study begins with example training, followed by formal questions on small (approx. 100 stations), medium (approx. 200 stations), and large (approx. 300 stations) networks.

We received in total 65 results, while we removed one incomplete answer and one which did not pass our training exercises. We analyzed 63 results from the participants (16 females, 40 males, and 7 not disclosed) with ages ranging from 20 to 59. Five are professional transit map users (e.g., cartographers, designers, transport company employees, etc). 15 are everyday users, 25 are frequent users, and 15 are occasional users. One participant rarely uses a transit map and two participants never used it. Figure 4.13 gives an example of a route marking result from a participant, and Figure 4.14a and 4.14b shows a summary of the route finding task accuracy. For (E1), we prepared maps from cities in several countries and with different guide shapes, including Montreal (flower), Singapore (heart), Paris (circle), Berlin (bear), and Paris (eye). More than 95% of the participants marked the shapes correctly in the the Montreal (flower), Singapore (heart), and Paris

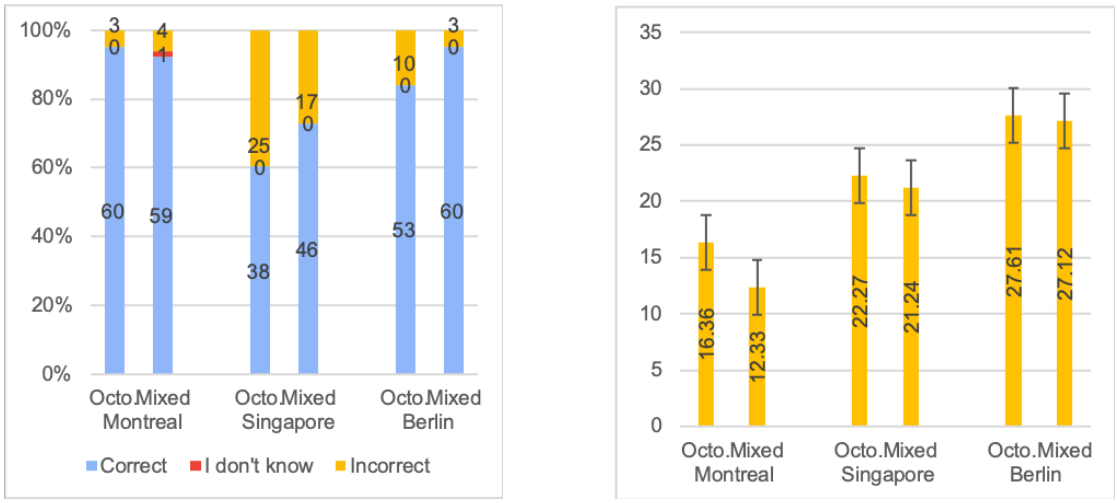
(circle) maps. The eye in the Paris (eye) map was recognized by 89% of the participants. The coarse shape of the bear was recognized by 60% of the participants, while none of them identified the teeth of the bear. We assume this could be because the teeth are a small portion of the entire shape and the stations are not sufficient enough to represent it. Except for the Berlin (bear) map (58%), more than 97% participants considered the shapes in the study significant. We also asked participants if the symbolic shapes helped them recognize the maps. This is done trough a 5 Point Likert Scale Analysis (yes, partially yes, neutral, partially no, and no). When the shapes are simple (flower (3.75), heart (3.51), and circle (3.71)), the answers lean toward positive, while if the shapes are complicated (bear (2.85) and eye (3.31)), the preference is close to neutral or negative.

For (E2), we compare task performance on a classical octolinear layout and a layout with embedded shapes on Montreal (flower), Singapore (circle), and Berlin (stadium). With simple shapes (e.g., circle, stadium, etc.), we see a slight improvement in task accuracy on Singapore and Berlin mixed maps (Figure 4.14a), and the time used to accomplish the route planning tasks is reduced in general (Figure 4.14b). The Berlin (stadium) has better accuracy because the stadium is a nice shape for representing a circular route of this network. We also received some explicit feedback from the participants. Ten participants mentioned in the feedback form that they found the shape design interesting and potentially helpful, especially when the shapes are simple, while two participants disliked embedding artificial shapes due to the inaccuracy introduced in comparison to more topographically correct maps. Two participants mentioned that corners on the shapes were important features that were expected to be mapped to stations in the final result. It is an interesting observation, while we did not come up with an intuitive extension to achieve this. One participant suggested that embedding a shape covering a larger area would increase the map's memorability. Another two participants mentioned that the embedded shapes might influence their planning decisions since they introduced additional bends in the map.

4.7.3 Limitations and Discussion

Although the route matching generally found a good alignment between the guide shape and the metro network in our test cases, we cannot guarantee their global optimality. In addition, formalizing and predicting the recognizability of shapes is a hard task, especially on shapes with multiple polylines. Likely in such a scenario not all features and subsections of a shape contribute equally to the recognizability. Our approach does not differentiate sub-regions of the guide shape and treats the entire guide shape equally.

Our deformation process creates the layout by minimizing sets of local constraints, which may cause unexpected results. For example, when deforming the transit network, the guide shape can prevent areas with a higher station density to expand, like in the Moscow map (Figure 4.12), where a larger circular route would be beneficial. Note that we also do not claim that any shape will be appropriate for a transit map. However, we see other potentials in the marketing, such as logo (Figure 4.1) and poster design [139]. Another limitation is inherited from the octolinear transit map design, where we do not accept



(a) (b)

Figure 4.14: (a) Accuracy and (b) time of the route finding task. The time is recorded in seconds and the error bars represent the standard errors.

a station degree that is larger than eight. In our implementation, we split high-degree stations into multiple stations with degrees less than eight in a pre-processing step.

The grid alignment requires that the vertices of P are connected with a suitable set of vertices of the grid, which results in non-octolinear edges at the interface between the grid and P. The iterative computation might not always find a routing for all edges, although even for large networks, we found layouts where only few number of edges (< 5) fail. An example is the layout of Tokyo, shown in Figure 4.15. The running time bottleneck of the grid alignment is the grid size and the shortest path algorithm. However, the underlying method of using repeated shortest path computations has been shown to be highly efficient on both octolinear [11] and flexible grids [10] and has recently been made openly available as part of a toolchain for transit map generation [28]. Finally, our current approach applies a post-processing labeling technique [91], which can potentially produce rather small text labels. The implementation is available at https://github.com/TobiasBat/Shape-Guided-Mixed-Metro-Map-Layout.

4.7.4 Contrast to Chapter 3

Both this and the previous chapter are concerned in some way with the generation of transit maps, a task for which they use quite different approaches. Here we would like to shortly discuss some of the differences and if and how the two approaches could reasonably be combined.

The smooth and the octolinear layout generated in this chapter (before the grid alignment step) are not necessarily strictly octolinear as they are produced by a force directed method, which aims to provide but can not guarantee octolinearity. This problem is

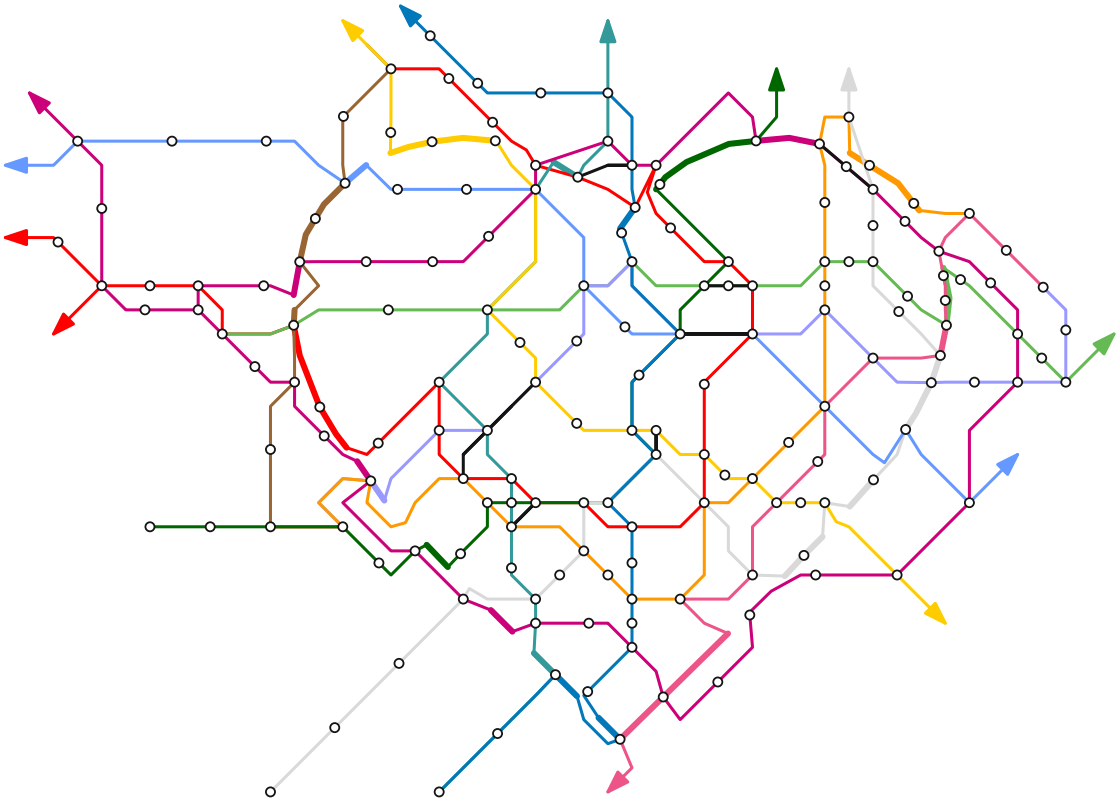


Figure 4.15: Tokyo metro map with a heart shaped guide guide. Five edges could not be routed. The grid alignment was computed based on the smooth layout and only the dense center of the map is shown.

fixed with the grid alignment phase, which makes the resulting layout more similar to the layouts generated in Chapter 3. However there are still edges to and from the guide shape, which do not adhere to the linearity and the grid alignment is based on a method which can insert edge bends at computation time, while the MIP of the previous chapter has to predefine bend points if they should appear in a final layout. The resulting maps are therefore visually quite different.

Concerning different linearities, all steps of the pipeline in this chapter can deal with different linearities to an extend. Wile the force directed methods can simply increase the number of directions to which we snap the edges, the case is slightly more tricky for the grid alignment. There has been recent work [10] which has shown the high flexibility of using the method of Bast et al. [11] in particular when it comes to different underlying grids. The main difference is the conceptualization of the edges in contrast to the MIP of Nöllenburg and Wolff [98].

In the first system, edges follow predefined edges of a grid, which can be at various inclines. In the second, an edge is parallel to an angle, which is not fixed in the plane.

To illustrate this difference, the reader can recall that while there are nice regular tetra-, hexa- and octolinear grids that tile the plane and can therefore be used as an underlying grid for the system of Bast et al. [10], there is no such nice grid for, e.g., a dekalinear system (i.e. a system which includes 5 different edge directions), a use case which can easily be formulated in the MIP.

In the other direction it would be exceptionally tricky to define adherence to a guide shape within the MIP if the edges are not assigned to the shape and already fixed in the plane, since the choice of one edge following the incline of an edge of the shape also carries an implication of where it should be placed in the plane and what the inclines of surrounding edges should be.

We do not see an easy method of combining these two results as they have been presented within this thesis and therefore leave it as an open question.

4.8 Conclusion

In this chapter, we introduced a new layout approach for synthesizing more engaging, mixed-style metro maps. This is achieved by embedding recognizable shapes into a classical metro map. The presented algorithm handles automatic and interactive route matching, shape-aware deformation, and finally, grid alignment sub-problems. With our results and evaluation, we show that the synthesized maps are of good quality and that the embedded shapes are intuitively recognizable. As a primary future research direction, we aim to investigate embedding more complex shapes, but also in combination with non-octolinear grids. One initial idea here is to decompose the network together with the shape hierarchically. We also plan to investigate more scalable methods to enable a real-time workflow for larger networks with proper labeling as well as to develop systematic quality metrics on such representations that are heavily linked to human shape recognition. Finally, we aim to conduct a usability test that covers the aforementioned spectrum to examine the function of shapes in a layout design.

CHAPTER 5

Finding Schematic Minimum-Link Containment Fences

This chapter is (partially) based on the following publications:
[21]: Bhore et al. – Minimum Link Fencing (ISAAC'22)
[20]: Bhore et al. – Minimum Link Fencing (EuroCG'21)

In the *geometric multicut* problem [2], we are given κ disjoint sets of polygons in the plane, each with a different *color*, and are asked for a subdivision of the plane such that no cell of the subdivision contains multiple colors. The goal is to minimize the total length of the subdivision edges.

A different kind of separation is achieved in the *polygon nesting* problem [124], where for two polygons P and Q with $P \subset Q$ one asks for a polygon P' with the smallest number of links, such that $P \subset P' \subset Q$. There exists a series of work that addressed the

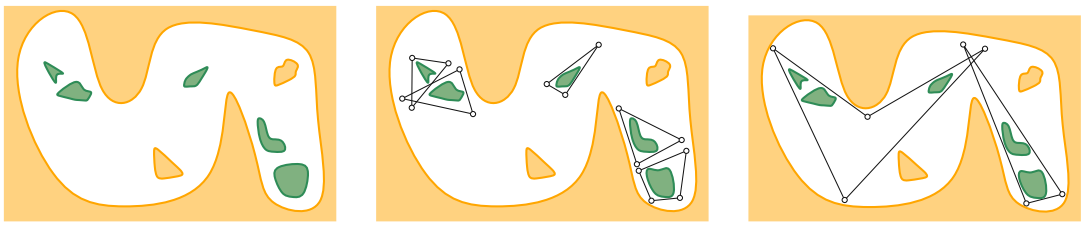


Figure 5.1: Two sets of polygons in the plane (left) with different colors (green and yellow). The yellow set effectively acts as an outer polygon with holes. Separating the two sets with, possibly intersecting, individual fences (middle) can lead to significantly more links in the fences (here 16) than grouping same-colored polygons (right), which achieves this with just seven links.

algorithmic complexity of nesting problems for various polygon families [3, 60, 124, 39, 85]. See Section 5.1 for more detail.

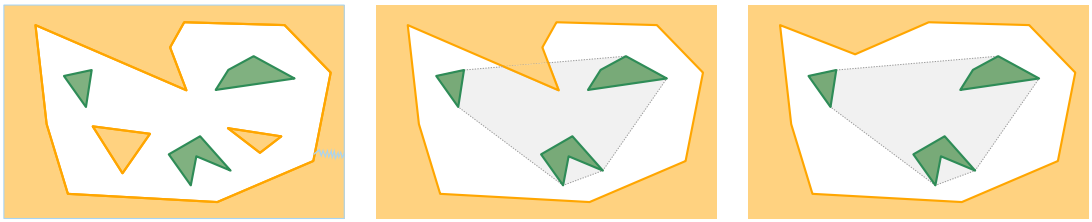
In this chapter, we consider a variant of geometric multicut inspired by polygon nesting, where we separate the sets from each other with a set of closed polygon boundaries called *fences*, which enclose only polygons of one color and have the smallest possible number of links. If one or more sets are not connected, we need to solve the combinatorial problem of choosing which polygons should be grouped in each fence. Figure 7.1 illustrates the problem. Some variants of the fencing problem already become NP-hard for point objects with two colors, e.g., if we require the fence to be a single closed curve [50].

In this chapter, we assume the input sets are collections of polygons, one color covers the plane minus a single polygonal hole (the outer polygon, a parallel to polygon nesting), and we will focus on the case $\kappa = 2$ of two colors. We use n to denote the total number of corners of the input polygons. Even in this simple setting the problem turns out to be non-trivial. If both sets are connected, then the problem is equivalent to finding a minimal *nested* polygon, which can be solved in $O(n \log n)$ time [3]. If both sets are not connected we show this problem to be NP-hard in Section 5.2. Note the contrast to the geometric multicut problem, which is polynomially solvable for $\kappa = 2$ [1] but becomes NP-hard when $\kappa = 3$ [2]. In Section 5.3 we show that, when restricting every fence to contain at most two polygons, the problem admits an XP-algorithm when parameterized by the maximal number of segments per fence, a result which holds for any κ. Finally, in Section 5.4, we show that the problem is polynomial-time solvable if the convex hull of the second color (the *inner polygons*) is contained in the outer polygon and the first color is connected. For a set $\mathcal{P}$ of polygons $P_1, \ldots, P_n$, each with corner points $v_1^1, \ldots, v_{|P_1|}^1$, $\ldots$, $v_1^n, \ldots, v_{|P_n|}^n$, respectively, we define the convex hull convex hull $C(\mathcal{P})$ as the smallest convex region R_c, s.t., $\forall 1 \leq i \leq n \;\; \forall 1 \leq j \leq |P_i| : v_j^i \in R_c$.

Problem Definition. Throughout this chapter we consider polygons in $\mathbb{R}^2$ without self-intersections but potentially with holes. Moreover, we consider a polygon as the boundary together with its interior, unless stated otherwise. We consider the following problem.

Definition 5.1 (Minimum Link Fencing (MLF))**.** *We are given n pairwise interior-disjoint polygons $\mathcal{P} = \{P_1, \ldots, P_{|\mathcal{P}|}\}$ in the plane, with a coloring function $f : \mathcal{P} \to \{1, \ldots, \kappa\}$, which assigns a color to every input polygon. We write $\mathcal{P}_i = \{P \mid f(P) = i\}$. We want to find a set of simple closed polygon boundaries $\mathfrak{F} = \{F_1, \ldots, F_m\}$ such that the total number of links $|F|$ on the boundary of $F = \bigcup_{i=1}^{k} F_i$ is minimized and if two polygons P_a and P_b are enclosed by the same fence or are both in $\mathbb{R}^2 \setminus \bigcup_{i=1}^{k} \overline{F_i}$, where $\overline{F_i}$ is the polygon bounded by F_i, then $f(P_a) = f(P_b)$. We call F_i a fence and $\mathfrak{F}$ a* minimum link fencing *of $\mathcal{P}$.*

Note the important difference in Definition 5.1 between $\mathfrak{F}$, which is the set of all fences of a solution, and F, which is the union over all fences, i.e., one (possibly disconnected) polygon. Thus $|\mathfrak{F}|$ is the number of fences and $|F|$ is the number of all segments in these fences.



(a) BMLF (b) SMLF (c) CMLF

Figure 5.2: Different problem inputs corresponding to (a) BMLF, (b) SMLF and (c) CMLF. In (b) and (c) the convex hull of all input polygons indicated in gray.

Throughout the chapter we refer to $\mathbb{R}^2 \setminus \bigcup_{i=1}^{|\mathcal{P}|} P_i$ as the *free space* (between polygons). We refer to $\mathcal{P}$, which contains polygons of κ different colors, as κ-colored and to the problem setting as the κ-colored problem. We consider several problem variations.

If there exists a polygon $Q \in \mathcal{P}$ which is unbounded in every direction, i.e. $\mathbb{R}^2 \setminus Q$ is finite, this polygon Q effectively acts as an outer boundary. In this case we call the problem Bounded Minimum Link Fencing (BMLF). We denote the polygon Q as the *outer polygon*. As a consequence, the size of the outer polygon automatically bounds the length of any link in a fence. Else, in general, one fence could conain a very long link, while retaining small complexity when counting the number of links only. Note that Q can be emulated in an instance of Minimum Link Fencing, by adding a large rectangular polygon $P_c \setminus (\mathbb{R}^2 \setminus Q)$, i.e., a large rectangle, of which the area, which did not belong to Q is cut out. Then we ensure that all polygons remain simple by connecting the unbounded region outside P_c with the hole of P_c via a highly complex narrow channel (light blue channel in Figure 5.2a). If Q is the only polygon of its color $f(Q)$ we call this setting Simply Bounded Minimum Link Fencing (SMLF). Moreover, if in an instance of SMLF we have $CH(\bigcup_{i=1}^{\kappa} \mathcal{P}_i \setminus Q) \subset \mathbb{R}^2 \setminus Q$, i.e., the convex hull of all input polygons except Q does not intersect Q, we speak of Convex Bounded Minimum Link Fencing (CMLF). The differences are illustrated in Figure 5.2.

5.1 Related Work

Despite the fact that the problem is natural and fundamental, little previous work exists. The problem of *enclosing* a set of objects by a shortest system of fences has recently been considered with a single set B_1 [1]. The task is to "enclose" the components of B_1 by a shortest system of fences. This can be formulated as a special case of our problem with $\kappa = 2$ colors: We add an additional set B_2, far away from B_1 and large enough so that it is never optimal to surround B_2. Thus, we have to enclose all components of B_1 and separate them from the unbounded region. In this setting, there will be no nested fences. Abrahamsen et al. [1] gave an $O(n \text{ polylog } n)$-time algorithm for inputs that consist of n unit disks.

Some variations with additional constraints on the fence become NP-hard already for point objects with two colors. For example, if we require the fence to be a single closed

curve, it has been observed by Eades and Rappaport [50] already in 1993 that one can model the Euclidean Traveling Salesman Problem of computing the shortest tour through a given set of sites by placing two tiny objects of opposite color next to each site. If we require the fence to be connected, the same construction will lead to the Euclidean Steiner Tree Problem, which was shown to be NP-hard by Garey et al. in 1977 [57].

Polygon Nesting & Separation. Polygon nesting is considered to be a fundamental problem in computational geometry, and has been extensively studied since its inception. Aggarwal et al. [3] considered the problem of finding a polygon nested between two given convex polygons that has a minimal number of vertices. They gave an $O(n \log k)$ time algorithm for solving the problem, where n is the total number of vertices of the given polygons, and k is the number of vertices of a minimal nested polygon. Das [39] considered a variant of MLF in his thesis, which restricts every fence to enclose exactly one polygon, and showed that the problem is NP-hard. Given a polygon Q of m vertices inside another polygon P of n vertices, Ghosh [60] gave an $O((n+m)\log k)$ time algorithm for constructing a minimum nested convex polygon, where k is the number of vertices of the output polygon, improving upon the $O((n+m)\log(n+m))$ time algorithm of Wang and Chan [125]. However, on the other hand, given a family of disjoint polygons $P_1, P_2, \ldots, P_k$ in the plane, and an integer parameter m, it is NP-complete to decide if the P_i's can be pairwise separated by a polygonal family with at most m edges. Mitchell and Suri [85] presented efficient approximation algorithms for constructing separating families of near-optimal size.

5.2 Two-colored BMLF is NP-hard

In this section we will call polygons of color 1 *boundary polygons* and polygons of color 2 *inner polygons*. An instance of planar 3, 4-SAT consists of a Boolean CNF-formula ϕ with a set of variables $\mathcal{V} = \{v_1, \ldots, v_n\}$ and a set of clauses $\mathcal{C} \subset 2^{\mathcal{V}}$, such that every clause is a disjunction of three literals and every variable occurs at most four times as a literal in a clause. Additionally, we are given the embedded plane incidence graph $G_\phi = (\mathcal{V} \cup \mathcal{C}, E)$, where $E = \{vc \mid v \in \mathcal{V}, c \in \mathcal{C}, v \text{ occurs as a literal in } c\}$. It is known that deciding if a 3, 4-SAT-formula has a satisfying assignment is NP-complete [66].

Given an instance of planar 3, 4-SAT we create an instance of 2-colored BMLF $\mathcal{P}$, emulating the shape of G_ϕ with one unbounded outer polygon Q and multiple boundary polygons of the same color $f(Q) = 1$ (Figure 5.3), such that ϕ is satisfiable if and only if there exists a minimum link fencing for $\mathcal{P}$ with at most a certain fixed number of total segments.

Note that each gadget is described as a basic construction of gray polygons, in which inner polygons are placed. This is possible, because we will invert all gray polygons at the end of the reduction, such that the area of their union makes up exactly the actual free space of our entire construction, see Figure 5.3. Stating that fences are computed inside the gray polygons should be understood as fences being placed in the free space between polygons. Throughout this reduction we distinguish fences based on the inner

polygons they include. We call two fences F and F' *congruent*, if and only if they enclose the same set of inner polygons. We call two fencings $\mathfrak{F}$ and $\mathfrak{F}'$ *congruent* if there is a bijective mapping $f : \mathfrak{F} \to \mathfrak{F}'$, such that, every $F \in \mathfrak{F}$ is congruent to $f(F) \in \mathfrak{F}'$.

Let $\mathcal{P}$ be an instance of BMLF and S_1, S_2, and S_3 disjoint connected subsets of $\mathbb{R}^2 \setminus \cup_{P\in\mathcal{P}} P$. We call the ordered set $\mathcal{S} = \{S_1, S_2, S_3\}$ a *non-collinear triple* if there are no three points $p_1 \in S_1$, $p_2 \in S_2$, and $p_3 \in S_3$ such that the straight-line segment s from p_1 to p_3 contains p_2 and s lies completely inside $\mathbb{R}^2 \setminus \cup_{P\in\mathcal{P}} P$. The choice of S_2 only matters if there exists a straight-line segment in $\mathbb{R}^2 \setminus \cup_{P\in\mathcal{P}} P$ connecting points in S_1 and S_3. Therefore we can often omit S_2 from the description of the triple or assume it as arbitrarily chosen. We call S_2 the *bend-set* of $\mathcal{S}$. Let $\mathfrak{F}$ be a fencing of $\mathcal{P}$, and S_1, S_2, and S_3 a non-collinear triple. We say a fence $F \in \mathfrak{F}$ *crosses* the triple $\{S_1, S_2, S_3\}$ if the boundary of F contains at least one point p_i from each set S_i for $i = 1, 2, 3$ and there is a cyclic traversal of the boundary of F in which we see first p_1, then p_2 and finally p_3. We write $]S_1, S_3[$ to denote the part of the boundary of F that lies in-between p_1 and p_3 and contains p_2.

Observation 5.1. *Any fence in a fencing for an instance of BMLF crossing a non-collinear triple $\{S_1, S_2, S_3\}$ contains at least one bend in the interval $]S_1, S_3[$.*

For $t > 0$ let $\mathcal{S}_1, \ldots, \mathcal{S}_t$ be non-collinear triples that are crossed by a fence F of a fencing $\mathfrak{F}$ for some instance of BMLF. Let $\mathcal{S}_i = \{S_j, S_{j+1}, S_{j+2}\}$ for $i = 1, \ldots, t$ and $j = 3(i-1)+1$. We say that the triples are crossed by F *in-order* if there exist points p_i on F such that p_i is in the bend-set of $\mathcal{S}_i$ and there exists a cyclic traversal of F in which we see the points p_i in order of their indices. Without loss of generality we will assume throughout that when F crosses $\mathcal{S}_1, \ldots, \mathcal{S}_t$ in-order it always crosses for some $\mathcal{S}_i$ first the set S_j, then the bend-set S_{j+1}, and finally S_{j+2}. We write $]S_a, S_b[$ with $a < b$ and $a = 1, \ldots, 3t-1$ for the part of the boundary of F that lies between a point $p_a \in S_a$ and $p_b \in S_b$ such that there exist points $p_a, \ldots, p_b$ with $p_i \in S_i$ that we see in this order in a cyclic traversal of F. For a segment s of F we say it is *completely* contained in $]S_a, S_b[$ if the start- and endpoint of s are contained in $]S_a, S_b[$ for any choice of points p_a and p_b.

We say two non-collinear triples $\mathcal{S} = \{S_1, S_2, S_3\}$ and $\mathcal{S}' = \{S'_1, S'_2, S'_3\}$ are *non-overlapping* if there exist no two segments that intersect all six elements of $\mathcal{S} \cup \mathcal{S}'$ in order S_1, S_2, S_3, S'_1, S'_2, S'_3. In other words we require at least three different straight-line segments to connect a point $p_1 \in S_1$ with a point $p_6 \in S'_3$ and containing points $p_2 \in S_2$, $p_3 \in S_3$, $p_4 \in S'_1$, and $p_5 \in S'_2$ in order of their indices. Observe that by this definition the non-collinear triples $\{S_1, S_2, S_3\}$ and its reverse $\{S_3, S_2, S_1\}$ are non-overlapping. For a sequence of non-collinear triples $\mathcal{S}_1, \ldots, \mathcal{S}_t$ we say that the triples are non-overlapping if S_i is non-overlapping with S_{i+1} for $i = 1, \ldots, t+1 \mod t$.

Observation 5.1 together with the definition of non-overlapping gives the following.

Observation 5.2. *Any fence in a fencing for an instance of BMLF crossing $t > 0$ non-overlapping non-collinear triples $\mathcal{S}_i = \{S_j, S_{j+1}, S_{j+2}\}$ for $i = 1, \ldots, t$ and $j = 3(i-1)+1$ in-order contains at least t bends and therefore at least $t-1$ complete straight-line segments in the interval $]S_1, S_{3t}[$.*

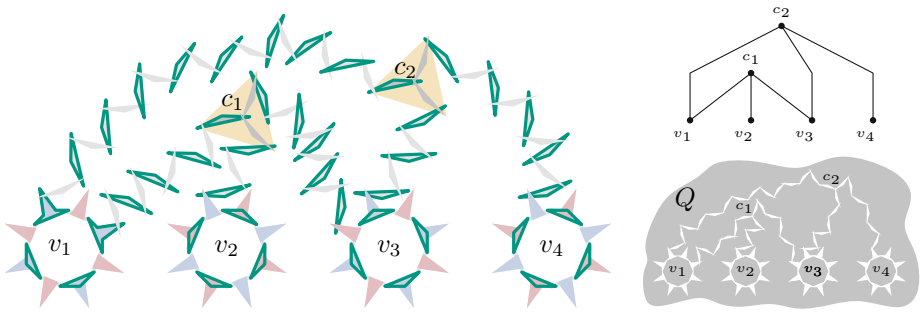


Figure 5.3: A (schematized) complete construction for a small instance $(v_1 \vee v_2 \vee \neg v_3) \wedge (v_1 \vee v_3 \vee v_4)$. The incidence graph is shown in the top right. Fences are highlighted in green. Also note that the boundary polygons make up most of the available area including an unbounded outer polygon Q as shown in the bottom right corner. For better readability, we will invert these colors in all subsequent figures.

We can now show a lower bound for the number of links a minimum-link fence uses in any solution of a BMLF instance. The lower bound essentially follows from Observation 5.2 after observing that the segment closing the fence can never reuse one of the $t-1$ segments that lie completely inside the sequence of non-collinear triples.

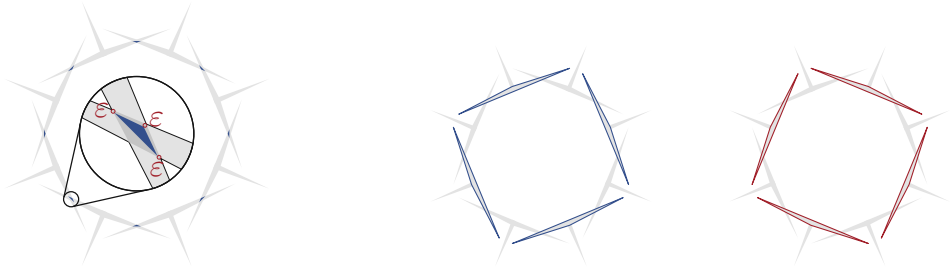
Lemma 5.1. *Let $\mathcal{P}$ be an instance of BMLF and $\mathfrak{F}$ a minimum-link fencing for $\mathcal{P}$, then any fence $F \in \mathfrak{F}$ that crosses $t > 0$ non-overlapping non-collinear triples in-order consists of at least t straight-line segments.*

Proof. By Observation 5.2 the fence F has at least $t-1$ straight-line segments Assume F has also exactly $t-1$ segments. Let $\mathcal{S}_1, \ldots, \mathcal{S}_t$ be the non-overlapping non-collinear triples and $S_1, \ldots, S_{3t}$ be the sets of the non-collinear triple in the order in which F crosses them as above. Since F is a simple polygon there exists a polygonal chain C starting at some point in S_{3t} and ending at some point in S_1. By Observation 5.2 there are $t-1$ straight-line segments which are completely contained in $]S_1, S_{3t}[$ which implies that C must consist of segments that are also completely contained in $]S_1, S_{3t}[$.

Now, we charge every of the $t-1$ segments to the piece of F that connects for each $\mathcal{S}_i$ with $i = 1, \ldots, t-1$ some point in S_{i+1} with some point in S_{i+2}. Since the triples are non-overlapping each such connection also requires a distinct segment. Then, for the connection of S_{3t-1} and S_{3t} we require at least one more segment. Let s be the segment charged to the connection of S_{3t-4} to S_{3t-3}. If we could extend this segment to also intersect S_{3t} we would violate that $\mathcal{S}_{t-1}$ and $\mathcal{S}_t$ are non-overlapping. Symmetrically for the segment charged to S_2 and S_3. Consequently, we require at least one more segment. □

5.2.1 Variable gadget

Every variable gadget consists of eight *T-polygons* (two per clause in which the variable can occur). Figure 5.4a illustrates the construction; T-polygons are marked in gray.



(a) Variable gadget construction with ε-gaps. (b) Fencing for *true*-state. (c) Fencing for *false*-state.

Figure 5.4: The variable gadget and its two possible fencings.

Every T-polygon has an isosceles triangle as the *arm* of the T (the horizontal part of the T shape) and a *spike* (alternatively called a *true spike* and a *false spike*) protruding from the arm and two consecutive polygons overlap at the end of their arms. For every variable $v \in \mathcal{V}$, we construct a variable gadget $\mathcal{G}(v)$ as a circular arrangement of eight overlapping T-polygons.

For every pair of overlapping T-polygons A and B, we place an inner polygon P, such that $P \subset A \cap B$. Let us fix some A, B, and P as above, then we place P such that its three corner points have only a very small distance $\varepsilon > 0$ to some corner point of $A \cap B$. All three ε-length segments between a corner point of P and the closest corner point of $A \cap B$ have to be crossed by every fence enclosing P.

Crucially, the variable gadget has only two minimum link fencings. These two states are shown in Figure 5.4. We associate the one shown in Figure 5.4b to the variable gadget encoding the value *true* and the one shown in Figure 5.4c to encoding *false*.

Lemma 5.2. *There are exactly two minimum link fencings $\mathfrak{F}_t$ and $\mathfrak{F}_f$ of the variable gadget, both of which will enclose only triangles in the same T-polygon with each fence, resulting in a fencing with 12 links for the whole variable gadget, such that every other minimum link fencing is congruent to either $\mathfrak{F}_t$ or $\mathfrak{F}_f$.*

Proof. To prove this lemma, we will first number the eight inner polygons of the variable gadget $A_1, \ldots, A_8$. Any fence inside the variable gadget can include any combination of k inner polygons. We will prove this lemma, by enumerating for every $k \in \{1, 8\}$ all possible (non-symmetric) combinations of including these inner polygons; Note that technically a fence is allowed to include non-consecutive inner polygons. Note that any fence has to consist of at least 3 segments and clearly every inner polygon of $\mathcal{G}(v)$ can be fenced alone with 3 segments. For any possible fence including multiple inner polygons, we show the existence of a certain number of non-overlapping non-collinear triples, which provide a lower bound on the number of segments for such a fence. All triples and the resulting lower bounds on the number of segments in fencings including all possible combinations of 2 to 8 inner polygons are shown in Subsection 5.2.2. Enumeration shows that a fence F including k inner polygons, has the minimum amount of segments, if and

only if the indices of the contained inner polygons are consecutive (assuming A_8 and A_1 to be consecutive). In particular, such a fence including 2, 3, 4, 5, 6, 7 or 8 inner polygons, requires (and can be realized with) 3, 6, 8, 11, 13, 16 and 18 segments, respectively.

First observe that there are exactly two minimum link fencings, which include exactly two neighboring inner polygons in one fence, both consisting of 12 segments total, both are shown in Figures 5.4b and 5.4c. We can therefore exclude any fence including six or more inner polygons, since they clearly require more segments by themselves, which immediately eliminates the possibility of including 6 or more inner polygons in one fence. Any fence including five inner polygons requires at least 11 segments and at least one additional fence is needed, which increases the segment count to at least 14. Any fence including four inner polygons requires at least 8 segments. If the remaining four segments are fenced together, we require at least 16 segments. If they include a group of three polygons, we require at least 14 segments. If two remaining polygons are grouped, we require 11 segments, but at least one more fence (14 segments in total) are needed and if all remaining polygons are fenced alone, we need 20 segments in total.

Next if three segments are fenced together, we require 6 segments and have five inner polygons left. We already know that no fence including four or more segments can be part of a minimum link fencing. If three of the remaining five polygons are grouped we arrive at a total of at least 12 segments, with at least one more fence needed. If at most two polygons are grouped, we need at least three more fences and arrive at a total of at least 15 segments.

Finally, polygons could be fenced individually. Clearly there must be an even number of individually fenced polygons. If all eight polygons are fenced alone, we reach at least 24 segments, for six inner polygons fenced alone we get 18 segments, four individually fenced polygons lead to 12 with at least one more fence needed. Two inner polygons being fenced alone require at least 6 segments, with at least three more fences needed for the six left over polygons, which require at least 9 segments leading to a total of 15 segments.

Therefore only two minimum link fencings exist, which require exactly 12 segments, and they group two neighboring inner polygons pairwise. We call the fencing, which groups inner polygons, which are both contained in a gray triangle with a true spike $\mathfrak{F}_t$ and the other $\mathfrak{F}_f$. □

5.2.2 Complete enumeration of possible fences in a variable gadget

This section contains the complete enumeration of all possible cases, which are considered in the proof of Lemma 5.2. In particular, we enumerate all possibilities of which k inner polygons could be included in a single fence, but compensate for rotational and axial symmetry, i.e., two groupings are considered rotationally symmetric if we can construct one from the other, by a combination of shifting all indices of included polygons by the same constant (recall that all computations are considered modulo 8, and we write the index 0 and 8 interchangeably) and relabeling all indices i as $8-i$ (mirroring the instance at a straight line).

Clearly there is exactly one possibility for fencing 0, 1, 7 or 8 inner polygons. Moreover, the number of ways to choose k polygons to fence is also characterizing how to chose $8-k$ polygons (which are not fenced) and therefore the number of cases is symmetric for 2 and 6, and 3 and 5. It remains to compute the correct number of cases for 2, 3 and 4.

Exclusively accounting for rotational symmetry, we can compute the number $f^r(k)$ with the formula

$$f^r(k) = \frac{1}{8} \cdot \sum_{d|\gcd(k,8-k)} \phi(d) \binom{8/d}{k/d}$$

where ϕ is Eulers ϕ-function, i.e., the number of co-prime integers smaller than k including 1. Using this formula we obtain four cases for two polygons (non of which are symmetric to each other), seven cases for three polygons (two of which can be eliminated due to the additional axial symmetry) and ten cases for four polygons (three of which are symmetric).

All cases are shown below in two individual figures, one illustrating the lower bound (the black numbers) on the left, using non-overlapping, non-colinear triples and one showing that this bound is in fact tight, by providing a fence achieving this exact number of segments (shown in blue).

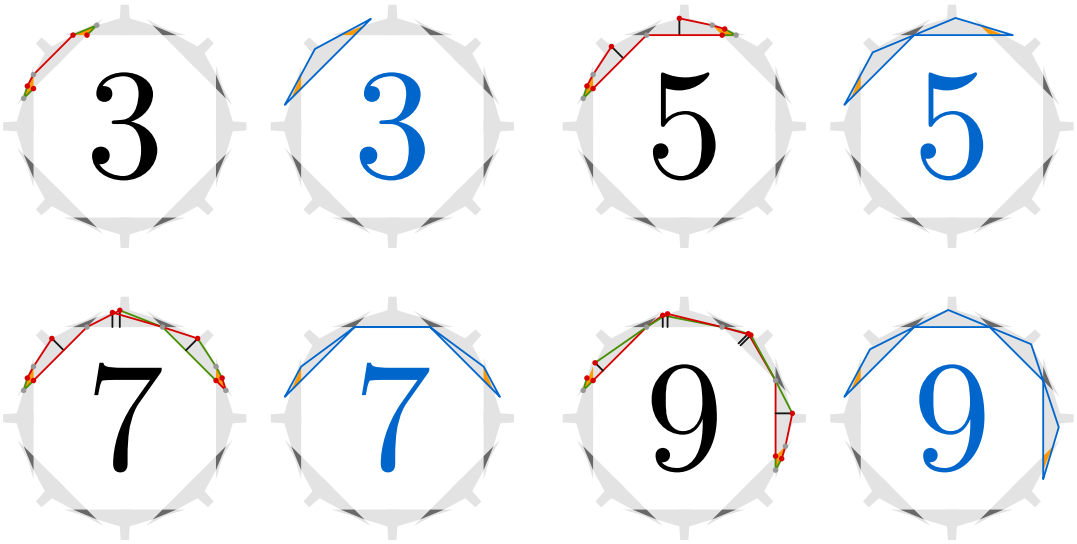


Figure 5.5: Fences including 2 polygons

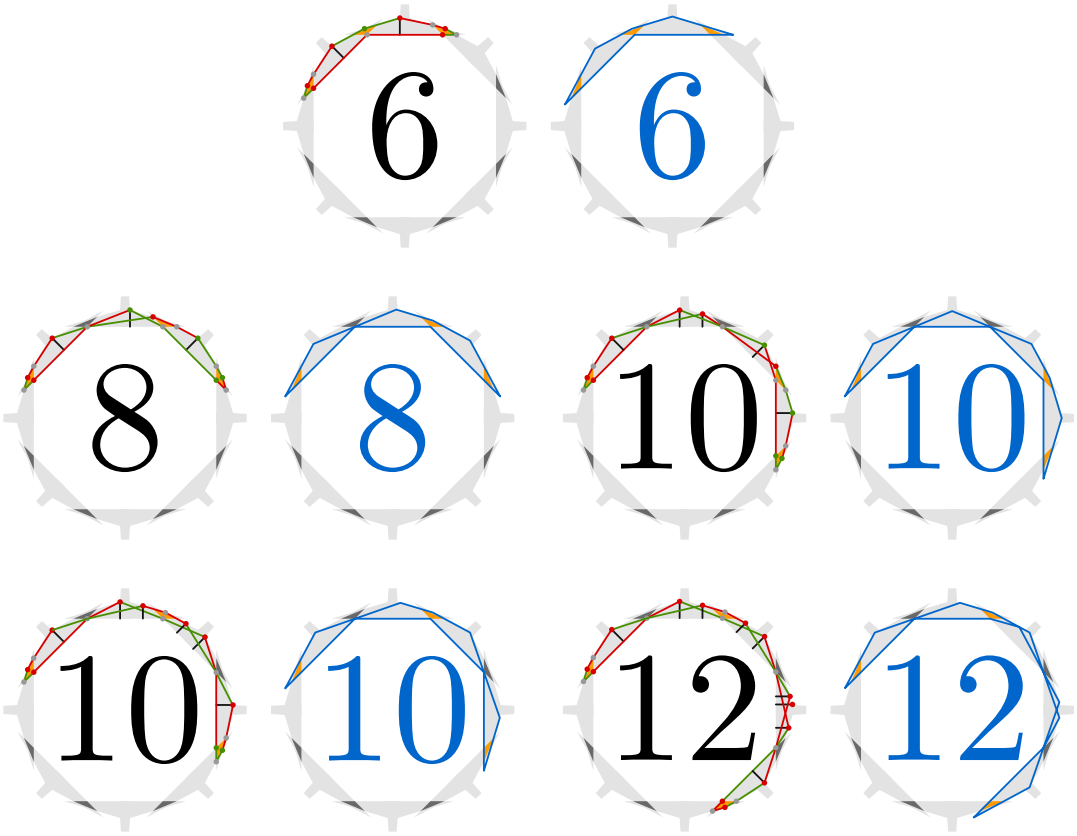


Figure 5.6: Fences including 3 polygons

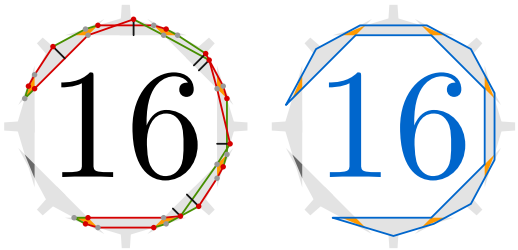


Figure 5.7: Fences including 7 polygons

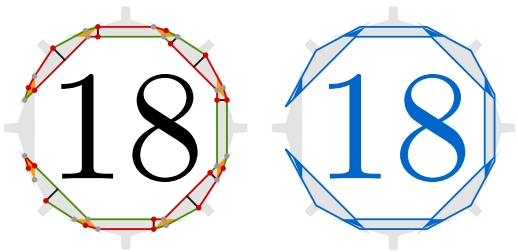


Figure 5.8: Fences including 8 polygons

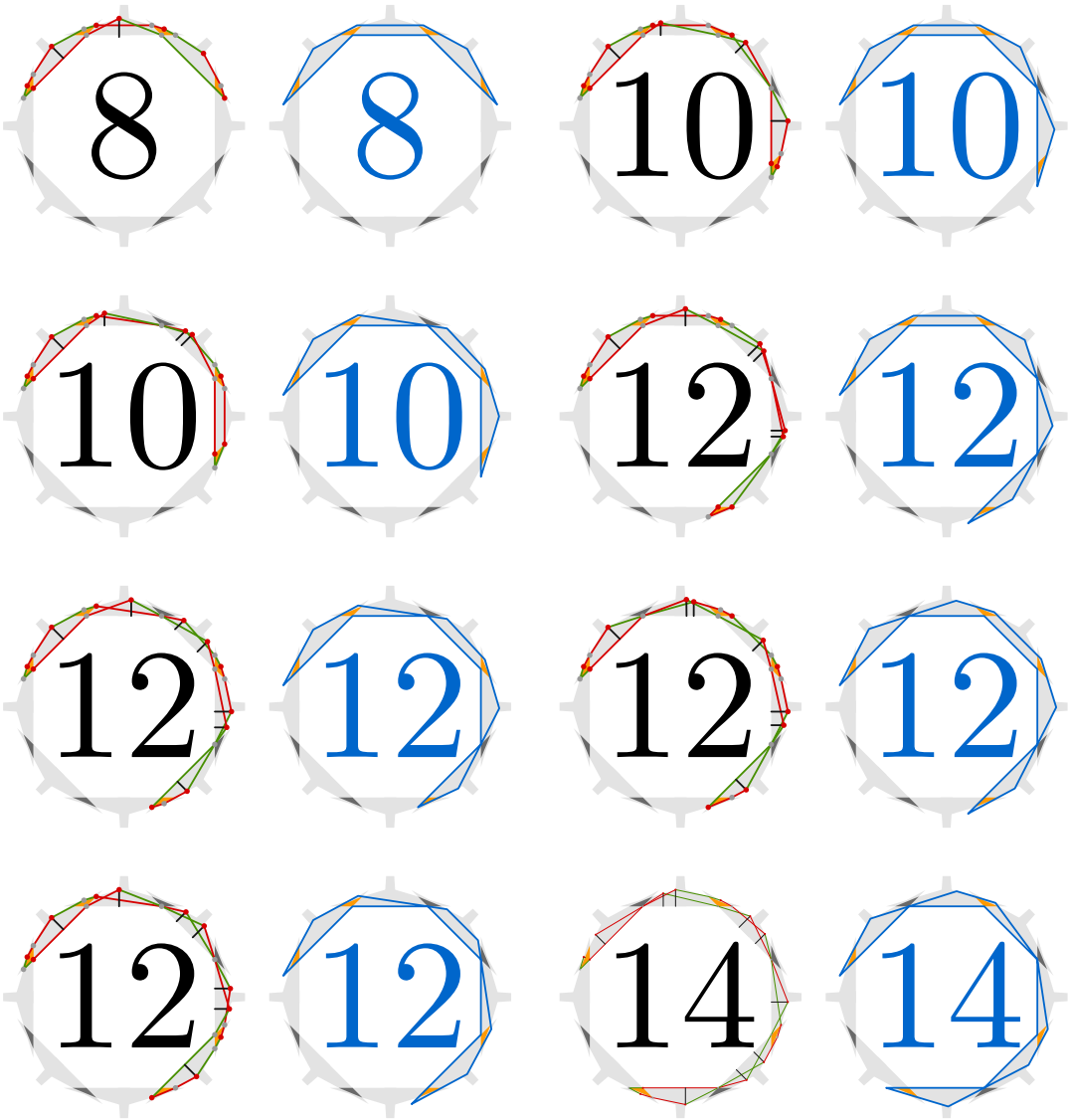


Figure 5.9: Fences including 4 polygons

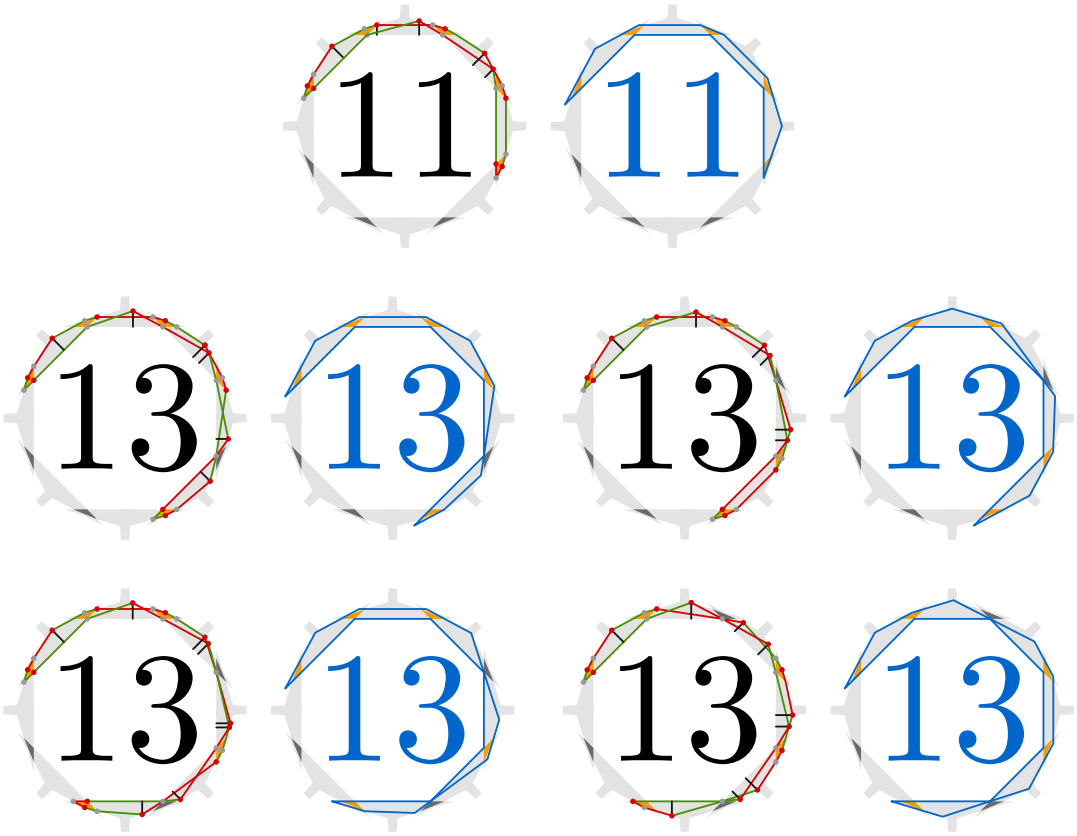


Figure 5.10: Fences including 5 polygons

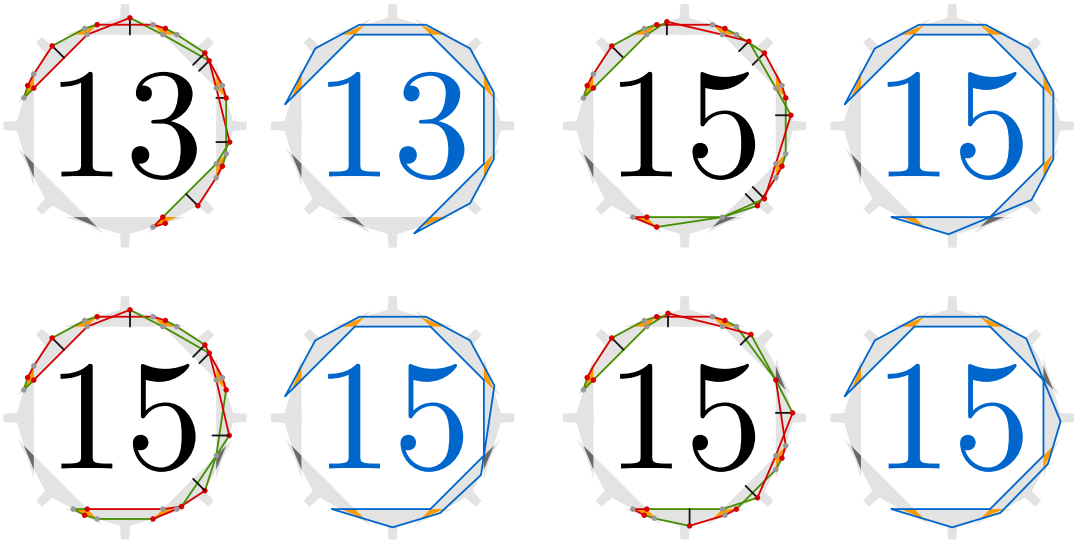


Figure 5.11: Fences including 6 polygons

(a) (b)

Figure 5.12: Wires are constructed from consecutive gray triangles places such that two consecutive triangles always contain a non-collinear triple, which are pairwise non-overlapping.

5.2.3 Clause gadget

For every clause $c \in \mathcal{C}$ in which three variables v_1, v_2, v_3 occur either as a positive or a negative literal, we create a clause gadget $\mathcal{G}(c)$. A clause gadget consists of three chains of an even number of gray triangles. These triangles are placed such that their hypotenuses intersect at an angle of at most π as shown in Figure 5.12a. The triangles are sufficiently long and thin, such that, we can define two sets in every gray triangle (one to either side of the central line), such that, the second set a' of the i-th triangle and the first set b of the $(i+1)$-th triangle form a non-collinear triple. By construction the non-collinear triple between the $(i-1)$-th and the i-th triangle and the one between the i-th and the $(i+1)$-th triangle are non-overlapping.

We place the three chains such that the three first triangles of the chains have a common intersection. Moreover, they intersect in such a way that their hypotenuses pairwise form $\frac{2\pi}{3}$ angles (Figure 5.12b). The last gray triangle of the first, second and third chain intersect a spike of $\mathcal{G}(v_1)$, $\mathcal{G}(v_2)$ and $\mathcal{G}(v_3)$, respectively. They intersect a true or false spike if the variable occurs as a positive or negative literal, respectively. We refer to each chain of gray polygons as a *wire*. The *length* of a wire is the number of gray triangles in its corresponding chain.

Let W_1, W_2, and W_3 be the wires of a clause gadget $\mathcal{G}(c)$ for clause c, where W_i intersects the spike of $\mathcal{G}(v_i)$ for $i \in \{1,2,3\}$. We place an inner triangle, denoted the *clause triangle* B_c of $\mathcal{G}(c)$, in the overlap of W_1, W_2, and W_3. Moreover, for wire W_i with gray triangles T_j^i we place inner triangles B_j^i in the overlap of the j-th and $(j+1)$-th gray triangle of the respective wire and a final triangle in the intersection with the spike of $\mathcal{G}(v_i)$. In the following we write $T_1, \ldots, T_k$ for the gray triangles and $B_1, \ldots, B_k$ for the inner polygons of one wire W_i, if i is clear from the context. Hence, inner triangle B_i is contained in the gray triangles T_i and T_{i+1} and gray triangle T_i for $i > 1$ contains the inner triangles B_i and B_{i+1}.

Let $B_1, \ldots, B_k$ be the inner polygons of a wire and F a fence containing B_i and B_j for some $i < j-1$ and $i = 1, \ldots, k-2$ but not B_z for $i < z < j$, then we say F *bypasses* B_z. For indices $1 \leq i_1 < i_2 < j_1 < j_2 \leq k$, we say two fences F_1 and F_2 containing some polygons of the wire *interleave* if B_{i_1} and B_{j_1} are in F_1 and F_1 bypasses B_{i_2} as well as

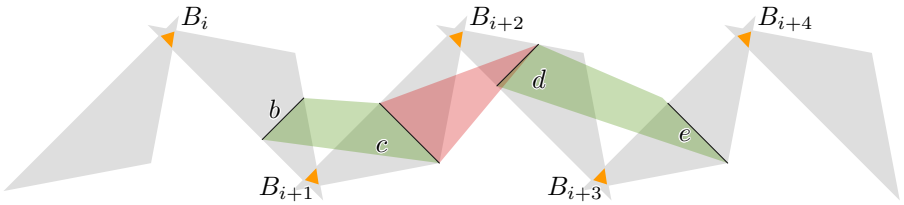


Figure 5.13: The highlighted areas show that b and c, c and d, and d and e form non-collinear triples.

B_{i_2} and B_{j_2} are in F_2 and F_1 bypasses B_{j_1}.

Let F be a fence of a minimum link fencing $\mathfrak{F}$ for a clause gadget $\mathcal{G}(c)$. Let s be a segment contained in the union of the gray triangles that form $\mathcal{G}(c)$ such that F crosses s in two points p and q. Then *splitting* F at s means the following. Delete F in an ε-region around p and q this creates two polygonal-chains, say F' and F'' with endpoints p' and q' on one side of s and p'' and q'' on the other. Connect p' with q' and p'' with q'' to form the two new fences F' and F''. Clearly, $|F'| + |F''| = |F| + 2$.

One isolated wire For the following we fix an arbitrary clause c. Let $\mathcal{G}(c)$ be the clause gadget of c and W one of the wires of $\mathcal{G}(c)$ with inner polygons $B_1, \ldots, B_k$. We denote as *isolated wire* the gray triangles of the chain of W that do not contain the clause triangle.

We are interested in how a minimum link fencing of an isolated wire looks like. Crucially, we first show that a fence of a minimum link fencing of an isolated wire cannot bypass any inner polygon of a wire. To cover all possible ways this could happen, we establish two lemmata, which investigate (a) the possibility that a fence could bypass more than one inner polygon (Lemma 5.3) and then (b) that there are two fences, which both only bypass a single inner polygon at a time and therefore form an interleaving pattern (Lemma 5.4).

Lemma 5.3. *No fence in a minimum link fencing $\mathfrak{F}$ of an isolated wire W of $\mathcal{G}(c)$ bypasses two or more consecutive inner polygons of W.*

Proof. Let $B_1, \ldots, B_k$ be the inner polygons of an isolated wire W, $T_1, \ldots, T_k$ the corresponding gray triangles, and $F \in \mathfrak{F}$ a fence that bypasses two or more consecutive inner polygons of W. Also, throughout the proof, let B_i and B_j with $i < j$ be two inner polygons that are contained in F such that $B_{i+1}, \ldots, B_{j-1}$ are bypassed by F.

Assume that $i + 1 < j - 1$, i.e., there are at least three consecutive inner polygons bypassed by F. Let B_{i+1}, B_{i+2} and B_{i+3} be three such consecutive polygons. Now, we find one sequence of at least three non-overlapping non-collinear triples by construction. Let b, c, d, and e be four sets such that b and c, c and d, and d and e form such triples. Compare Figure 5.13 for an illustration.

By Observation 5.2 F contains at least two complete segments in $]b, e[$. Moreover, F has to cross the same triples in reverse order since it is a simple polygons and contains

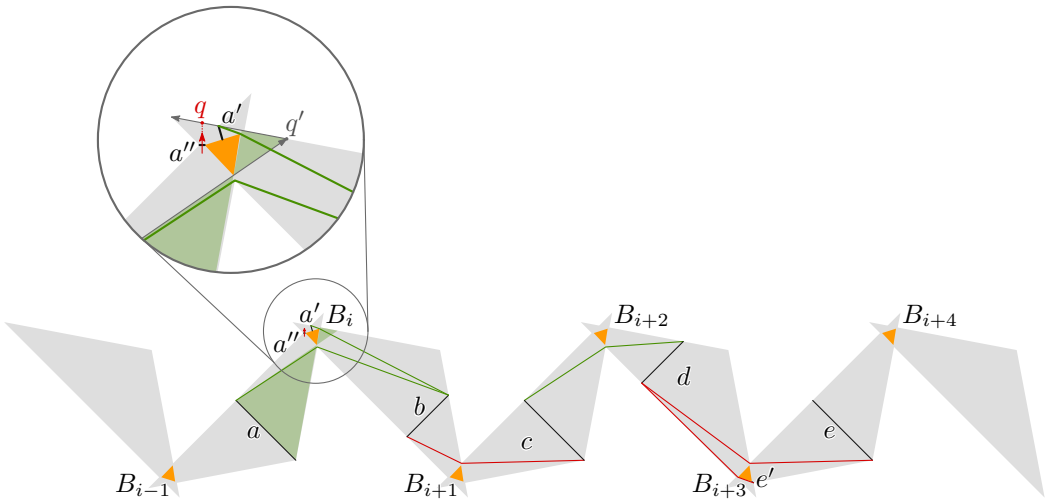


Figure 5.14: We can locally shortcut any fence F passing a and b in both directions by connecting the two intersection points of F with a (b)

both B_i and B_j for $j > i + 3$. Consequently, we find another two complete segments in $]e, b[$. We split F at b and e. This creates three fences F_1, F_2, and F_3 for which it holds that $|F_1| + |F_2| + |F_3| = |F| + 4$. One fence does not contain any inner polygons and can be deleted, let F_2 be this fence. Since F_2 was created by splitting along b and e it contained at least four complete segments plus the two segments introduced in the splitting operation. Hence, with $|F| + 4 = |F_1| + |F_2| + |F_3| \geq |F_1| + |F_3| + 6$ it follows that $|F| \geq |F_1| + |F_3| + 2 > |F_1| + |F_3|$. Since F was part of a minimum link fencing we may conclude that F at most bypasses two consecutive inner polygons.

In the following let B_{i+1} and B_{i+2} be the two consecutive polygons bypassed by F. Consequently, we know that F contains B_i and B_{i+3}. Figure 5.14 illustrates the following cases.

Assume F does not contain any other inner polygons, i.e., F contains only two inner polygons. In this case we find a sequence of four non-overlapping non-collinear triples a' and b, b and c, c and d, d and e' which have to be crossed by F. Moreover, F has to cross each such triple in the reverse direction. Consequently, F has crossed eight non-overlapping non-collinear triples and consists by Lemma 5.1 of at least eight segments. Replacing F by two fences, both consisting only of triangles, creates a fencing with less segments than $\mathfrak{F}$. It remains to argue the case that F contains more than two inner polygons.

Let F be such that there does not exist an inner polygon B_z contained in F with $z > i+3$. Then there exists an inner polygon $B_m \in F$ and $m < i$. Observe that there is a sequence of four non-overlapping non-collinear triples a' and b, b and c, c and d, d and e', which is as in the case before and a sequence of four non-overlapping non-collinear triples e' and d, d and c, c and b, and b and a which is almost as before, with the difference that we use a set a which lies in T_{i-1} and can be chosen as a segment which F intersects twice. By Observation 5.2 we find that $]a', a[$ completely contains at least seven segments of F.

We split F at a, let F_m and F_i be the resulting fences such that F_m contains B_m and F_i contains B_i and B_{i+3}. It holds that $|F_m| + |F_i| = |F| + 2$ and $|F_i| \geq 8$ since F_i contains only B_i and B_{i+3} and bypasses B_{i+1} and B_{i+2}. Hence, $|F| \geq |F_m| + 6$ since $|F| + 2 = |F_m| + |F_i| \geq |F_m| + 8$. Consequently, replacing F_i by two triangles only yields a fencing with the same number of fences as $\mathfrak{F}$. In the following assume that we delete F_i and introduce one triangular fence F_Δ that includes only B_{i+3}. The goal is then to include B_i into F_m using at most three segments.

Consider F_m and remove the segment introduced when splitting F at a. Let p and p' be the two intersection points of F with a and assume p is closer to the base side of T_{i-1}. Without loss of generality we assume that neither p nor p' are vertices of F. Let c_i be the corner of T_{i-1} that would be disconnected from the component containing a when removing T_i from the plane. If there exists a point c in the component containing c_i such that the straight-line segments pc and $p'c$ both do not intersect B_i, are completely contained in T_{i-1}, and their supporting lines leave B_i in different half-planes, we just add these two segments to F_m which now contains also B_i.

Now assume such a point does not exist. By construction there exists a straight-line segment starting at p, p' respectively, ends at the boundary of T_i, and also intersects the boundary of T_i. Let s be such a segment for p and s' one for p' and let q and q' be their endpoints on the boundary of T_i. Sine p is closer to the base of T_{i-1} we may assume that the supporting line of s leaves B_i to the right and the one of s' leaves B_i to the left, and s and s' do not intersect. Connect the two endpoints of s and s' with one segment along the boundary of T_i. This uses at most three segments as required.

In preparation for the next case we are going to remove one additional segment. Consider a set a'' that is intersected by s, such a set is indicated in Figure 5.14. Since F contained B_i it had to contain at least one point of a'' as well. Moreover, since F was a minimum link fence containing B_i[1] we may assume that there is a vertex v of F that is now a vertex of F_m and there is a straight-line segment starting at v, intersecting a'', intersecting the boundary of T_i twice, and it lies in T_{i-1} and does not intersect B_i. Let q'' be the endpoint of this segment. Replace the segments vp, pq, and qq' by vq'' and qq''. That is only two additional segments.

Finally, assume that there exist inner polygons $B_m, B_l \in F$ such that $m < i$ and $l > i+3$. Split F as above at a and also at some set e in T_{i+3}. This leaves three fences F_m, F_l, and F_i. Similarly to above $|F| + 4 = |F_m| + |F_l| + |F_i| \geq |F_m| + |F_l| + 8$ from which we derive $|F| \geq |F_m| + |F_l| + 4$. By the above argumentation we can hence replace F by two fences F_m and F_l such that the new fencing has the same number of segments. But now, observe that there is a sequence of non-overlapping non-collinear triples which F_i all has to cross, namely a' and b, b and c, c and d, d and e, e' and d, d and c, c and b, and b and a. By Observation 5.2 intervals $]a', e[$ and $]e', a[$ completely contain three segments each. Moreover, these sets can be chosen in such a way that the segments introduced when

[1] We are only aiming to contradict minimality with respect to inclusion of polygons in the fence, for a given set of polygons we may still assume that the initial fence was as short as possible.

splitting F do only contain points of a and e. See also Figure 5.14 for an illustration. Hence, $|F_i| \geq 10$ and consequently $|F| \geq |F_m| + |F_l| + 6$. □

We now know that no fence of a minimum link fencing of an isolated wire bypasses two or more consecutive inner polygons of that wire. However, it might still bypass an unbounded number of polygons in total. Since there is a sequence of four non-overlapping non-collinear triples for a fence that includes B_i and B_{i+2} we obtain the following observation for a fence bypassing at least one inner polygon.

Observation 5.3. *Let $\mathfrak{F}$ be a minimum link fencing of an isolated wire W of $\mathcal{G}(c)$, let $B_1, \ldots, B_k$ be the inner polygons of W, and $F \in \mathfrak{F}$ a fence that bypasses $z > 0$ inner polygons, then F contains at least $4z + 2$ segments.*

While it is not possible anymore by Lemma 5.3 to bypass multiple consecutive polygons there could still be multiple fences in a minimum link fencing that interleave and bypass many individual inner polygons.

Lemma 5.4. *A minimum link fencing $\mathfrak{F}$ of an isolated wire W of $\mathcal{G}(c)$ does not contain two distinct fences that interleave.*

Proof. Let $B_1, \ldots, B_k$ be the inner polygons of W and assume that there are fences $F, F' \in \mathfrak{F}$ of W that interleave. By Lemma 5.3 we know neither F nor F' can bypass two or more consecutive inner polygons. Consequently, there exist inner polygons B_i, B_{i+1}, B_{i+2}, and B_{i+3} such that without loss of generality $B_i, B_{i+2} \in F$ and $B_{i+1}, B_{i+3} \in F'$.

First, assume that F and F' only contain these four polygons. By Observation 5.3 each fence bypassing one polygon contains at least six segments. Hence, $|F| + |F'| \geq 12$. By replacing F and F' by two new fences, consisting of a triangle each, one containing B_i and B_{i+1} and the other containing B_{i+2} and B_{i+3}, we create a new fencing with six segments less. A contradiction to $\mathfrak{F}$ being a minimum link fencing.

Now consider the case that F and F' might contain more inner polygons. First assume all further inner polygons of F are before B_i and all further inner polygons of F' are after B_{i+3}. Observe that then $|F| + |F'| \geq 14$ since we need to add at least one segment to each fence. We split the fences F and F' in the gray triangles T_{i-1}, T_{i+3} respectively. We first consider just F. After splitting we now have two fences $F_{<i}$ and F_i. Let $F_{<i}$ be the fence containing all inner polygons before B_i and F_i the fence containing B_i and B_{i+2}. By Observation 5.3 we have $|F_i| \geq 6$. Consequently $|F| = |F_{<i}| + |F_i| - 2 \geq |F_{<i}| + 6 - 2 = |F_{<i}| + 4$ and hence $|F_{<i}| \leq |F| - 4$. Doing the same for F' we obtain $F'_{<i}$ and F'_i. As before we can replace F_i and F'_i by two triangles F_Δ and F'_Δ. In total the new fencing created like this has $|F_{<i}| + |F_\Delta| + |F'_{<i}| + |F'_\Delta| \leq |F| + |F'| - 8 + 6 = |F| + |F'| - 2$ segments.

Finally, F and F' could contain polygons both before B_i and after B_{i+3}. By Lemma 5.3 we know that then the fences repeatedly interleave until at some inner polygon one fence stops. Let $m < l$ and without loss of generality let B_m be the last polygon included

Figure 5.15: All six non-collinear triples of a fence bypassing exactly one inner polygon.

by F and B_l the first polygon included by F' and assume that between B_l and B_m the two fences interleave, i.e., F includes B_m, B_{m-2}, etc. and F' contains B_{m-1}, B_{m-3} , etc.. Moreover, F contains B_{l-1} and F' contains B_{m+1} We can split F and F' as before in T_{l-2} and T_{m+2} using just four segments. Let $F_{<m}$ and F_m be the fences created from splitting F such that F_m contains $B_m, \ldots, B_{l-1}$ and $F_{<m}$ the remaining polygons contained in F. Analogously for F' and $F_{<l}$ and F_l. Let M be the number of bypassed polygons for F and L the number of bypassed polygons for F'. We get that $|F| + |F'| = |F_{<m}| + |F_m| + |F_{<l}| + |F_l| - 4$. We also know from Observation 5.3 that $|F_m| \geq 4M + 2$ and $|F_l| \geq 4L + 2$. Hence, $|F| + |F'| \geq |F_{<m}| + |F_{<l}| + 4M + 4L$ and hence $|F_{<m}| + |F_{<l}| \leq |F| + |F'| - 4M - 4L$. Finally, we can replace F_m and F_l by a series of triangular fences that in sum have $3 \cdot (M + L + 2)/2$ which results in

$$\begin{aligned} |F_{<m}| + |F_{<l}| + 3/2(M + L + 2) - 4 &\leq |F| + |F'| - 4M - 4L + 3/2(M + L + 2) - 4 \\ &= |F| + |F'| - (5/2(M + L) + 1). \end{aligned}$$

This concludes the proof as $\mathfrak{F}$ cannot have been a minimum link fencing for W. □

With both of these Lemmata established, we can state the generalized result in the following Lemma.

Lemma 5.5. *A minimum link fence $\mathfrak{F}$ of an isolated wire W of $\mathcal{G}(c)$ does not contain a fence $F \in \mathfrak{F}$ such that F bypasses an inner polygon B_i with $i \in \{2, \ldots, k-1\}$ of W.*

Proof. Let $B_1, \ldots, B_k$ be the inner polygons of W. Assume that F bypasses an inner polygon B_j for $2 \leq j \leq k-1$. By Lemma 5.3 we know that F never bypasses more than one consecutive inner polygon at a time. Hence, we know that B_{j-1} and B_{j+1} are both contained in F. We distinguish if F bypasses B_j above or below (both cases are shown in Figure 5.16).

Assume F bypasses B_j above as shown in Figure 5.16 (a). Then we construct a new fence including also B_j as follows. Let s_1 and s_2 be two segments such that s_1 lies inside the gray triangle t_{i-1} of W that contains B_{j-1} and B_j and s_2 such that it lies inside the gray triangle t_i of W containing B_j and B_{j+1}. More specifically, we choose s_1 such that its supporting line leaves B_{j-1} and B_j in one and B_{j-2} in the other half-plane. Similarly, we chose s_2 such that its supporting line leaves B_j and B_{j+1} in one and B_{j+2} in the other half-plane. The segments s_1 and s_2 can then be extended such that they meet in a point that is inside the overlap of t_{i-1} and t_i and below B_j. Moreover, they can be extended such that they intersect any fence that contains B_{j-1} and B_{j+1} at least twice below B_{j-1} and B_{j+1} respectively. See Figure 5.16(a) for an illustration.

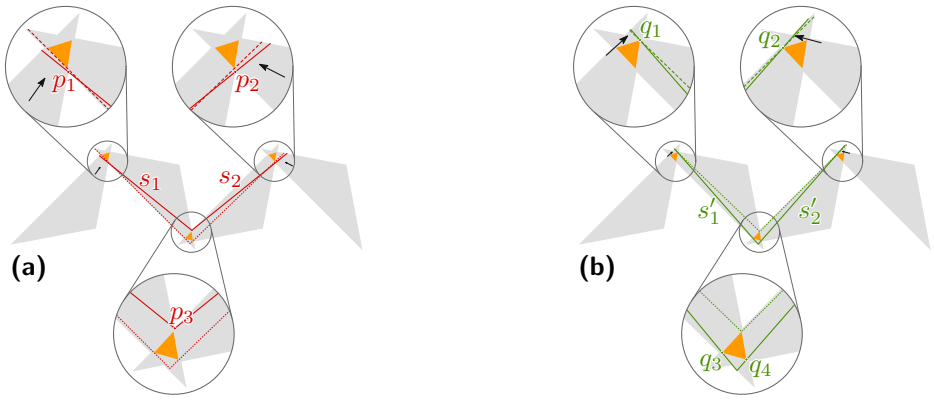


Figure 5.16: Rerouting of segments to include a bypassed inner polygon.

Symmetrically we find two segments s'_1 and s'_2 whose intersection point is above B_j and whose supporting lines leave B_{j-1} and B_j in the same half-plane and B_{j+1} in the other, B_{j+1} and B_j in the same and B_{j-1} in the other respectively. Again, these segments can be extended to intersect F twice, this time above B_{j-1} and B_{j+1}. See Figure 5.16(b) for an illustration.

As F contains at least four segments inside t_{i-1} and t_i we can replace those by s_1, s_2, s'_1, and s'_2 which yields a fence with at most equal number of links. Now, deleting the single fence that fenced B_j removes at least three links, a contradiction to $\mathfrak{F}$ being a minimum link fencing.

We can apply this procedure to all bypassed polygons. Since no fence can be interleaving with F, we can simply remove all fences, which included any polygon B_i completely and still obtain a valid fencing, contradicting that $\mathfrak{F}$ is a minimum link fencing. □

In the following we are going to bound the number of consecutive polygons that are contained in one minimum link fence of an isolated wire. We compare this then to a fence containing all inner polygons of an isolated wire. Such a fence, by construction, contains $2z$ non-collinear triples and hence requires $2z$ segments by Lemma 5.1. Figure 5.17 shows these triples. Constructing such a fence is straight-forward by following these non-collinear triples. The following lemma summarizes this statement.

Lemma 5.6. *Let $\mathfrak{F}$ be a minimum link fencing of an isolated wire W of $\mathcal{G}(c)$, any fence $F \in \mathfrak{F}$ that contains $z > 2$ consecutive inner polygons of W has at least $2z$ segments and such a fence exists.*

Lemma 5.7. *Let $\mathfrak{F}$ be a minimum link fencing of an isolated wire W of $\mathcal{G}(c)$, then every fence of $\mathfrak{F}$ contains at most three consecutive inner polygons.*

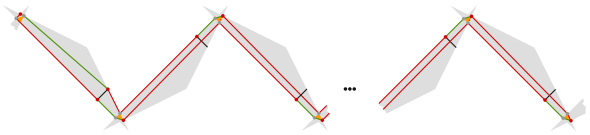


Figure 5.17: Non-collinear triples in a fence including a series of consecutive inner polygons.

Proof. Let $B_1, \ldots, B_k$ be the inner polygons of W. By Lemma 5.5 we can assume that the inner polygons of W contained in F are consecutive in the sequence of inner polygons. Let $F \in \mathfrak{F}$ be a fence containing $z > 3$ inner polygons of W.

By Lemma 5.6 we know that F consists of $2z$ segments. We replace F with a fence F_1 including the two first polygons included in F and a fence F_2 including all $z-2$ following inner polygons. Again by Lemma 5.6 it follows that $|F_2| = 2z - 4$ and it holds that $|F_1| = 3$. In sum, we get that $|F_1| + |F_2| = 2z - 4 + 3 = 2z - 1 \leq |F|$, a contradiction. □

Lemmata 5.7 and 5.6 now lead to a characterization of minimum link fences of isolated wires.

Lemma 5.8. *Let $\mathfrak{F}$ be a minimum link fencing of an isolated wire W of $\mathcal{G}(c)$ with k inner polygons, then $\mathfrak{F}$ has in total $3k/2$ segments and $F \in \mathfrak{F}$ contains exactly two consecutive inner polygons B_i and B_{i+1} for i odd.*

Proof. Let $B_1, \ldots, B_k$ be the inner polygons of W. By Lemma 5.7 every fence in $\mathfrak{F}$ contains either one, two, or three consecutive inner polygons. Let f_i be the number of fences in $\mathfrak{F}$ containing i inner polygons. Then the number of segments of the fencing $\mathfrak{F}$ can be computed as $|\mathfrak{F}| \geq 3f_1 + 3f_2 + 6f_3$ since for one or two inner polygons we always can use one triangle and for three inner polygons we have six segments by Lemma 5.6. Now, let k_i be the number of segments in a fence containing i inner polygons. From this we get that $f_i = k_i/i$. Substituting in the previous calculation we get $|\mathfrak{F}| \geq 3k_1 + 3/2k_2 + 2k_3$. Hence, maximizing the number of inner polygons in fences containing also two polygons minimizes the number of segments in the fencing. Since k is always even the minimum is attained at $3/2k$, i.e., when all inner polygons are part of a fence containing only two inner polygons as claimed. □

Integrating the clause triangle So far we only considered one arbitrary isolated wire of $\mathcal{G}(c)$. To put things together we need to consider the interaction of the three wires of $\mathcal{G}(c)$. Specifically, we need to show that no fence in a minimum link fencing of $\mathcal{G}(c)$ contains inner polygons from two different wires.

We extend the definition of bypassing an inner polygon of a wire to a whole clause gadget. Let F be a fence for $\mathcal{G}(c)$, then F *bypasses* an inner polygon B_j^i of wire W_i of $\mathcal{G}(c)$ if F contains the clause triangle B_c or some inner polygon of a wire $W_{i'}$ with $i' \neq i$ and F contains B_l^i for wire W_i with $l > j$. We say F *bypasses* the clause triangle of $\mathcal{G}(c)$ if F

contains inner polygons of at least two different wires of $\mathcal{G}(c)$ but not the clause triangle B_c of $\mathcal{G}(c)$.

As for an isolated wire we can show that no inner polygon for a whole clause gadget can be bypassed. This can be seen after observing that no fence can bypass the inner polygons of an isolated wire without violating Lemma 5.5. The remainder of the proof is then a careful case enumeration, see Section 5.2.4.

Lemma 5.9. *Let $\mathfrak{F}$ be a minimum link fencing of $\mathcal{G}(c)$ and $B_1, \ldots, B_k$ the inner polygons of one of the wires of $\mathcal{G}(c)$. Then there is no fence $F \in \mathfrak{F}$ that bypasses an inner polygon B_i with $i \in \{1, \ldots, k-1\}$.*

Proof. Assume there exists a fence F that bypasses some polygons of $\mathcal{G}(c)$. If F only bypasses and contains polygons of one isolated wire W and is contained in the gray triangles of W we can apply Lemma 5.5 contradicting the existence of F.

Next, assume that F lies not only in the gray triangles of W but still only contains and bypasses inner polygons of W. This implies that F contains B_1 of W as else we could by construction find a fence that is restricted to only the gray triangles of W. Let T_c be the triangle intersecting T_1 of W and containing B_c. If F was not restricted to the $T = T_c \cup T_1 \cup \ldots \cup T_k$ we could replace F by a congruent minimum link fence that is only contained T by either splitting F if a whole segment is outside of T or moving the one corner that lies outside of T which is always possible by construction. After that we can again apply Lemma 5.5 by considering T_c and B_c part of the isolated wire.

This means that F has to include either B_c or two inner polygons of different wires if F is to bypass any polygon of $\mathcal{G}(c)$. Now assume F was bypassing any inner polygon B_i of some wire W and $i > 1$. Let F_T be F's restriction to $T = T_c \cup T_1 \cup \ldots \cup T_k$. Add one temporary segment to F_T to close the fence, this can always be done in T_c. Now we can apply Lemma 5.5 to F_T. Each proof of one of the lemmata leading to Lemma 5.5 implies a procedure of how to split F_T. This results in two or more fences. Among those exists one that contains B_j with smallest j. Let F_T^j be this fence. Either F_T^j is such that it still contains the segment added initially to F_T in which case we just remove this segment and obtained a fencing with less segments or F_T^j does not contain this segment anymore. Recall, that every procedure we implicitly defined saves at least two segments. Hence, adding the segment we had added to F_T to the remainder of $F - F_T$ still creates a fencing with one segment less.

Consequently, we may assume that F bypasses any combination of B_1^1, B_1^2, B_1^3, or B_c. Without loss of generality we assume a fence F bypasses the first inner polygon B_1^1 of wire W_1. We can again show that the amount of non-collinear triples between the lines a and b is four, inducing at least four bends in this part of F.

We define twelve sets $a, a^\star, a', a'', b, b^\star, b', b'', c, c^\star, c'$ and c'', as shown in Figure 5.18. Any fence crossing the following pairs of sets contains a non-collinear triple starting at one set and ending at the other:

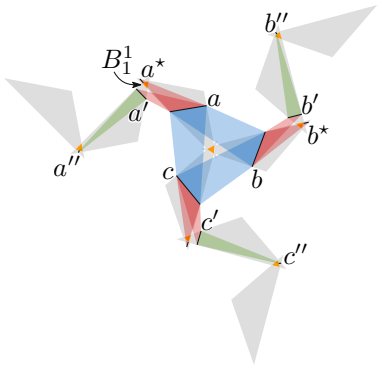


Figure 5.18: The twelve sets.

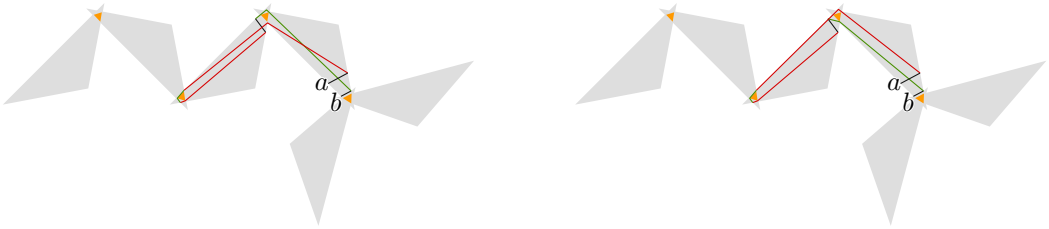


Figure 5.19: Non-collinear triples for any fence bypassing B_1

- a and b
- a and $a^\star$
- b and $b^\star$
- c and $c^\star$
- a and c
- a and a'
- b and b'
- c and c'
- b and c
- a' and a''
- b' and b''
- c' and c''

Recall that F is a fence bypassing B_1^1 hence F has to include some B_j^1 for $j > 1$. More precisely, we may assume that F contains B_2^1 as else we could find a shorter fence by the above discussion. Let F' be the fence that contains B_1^1. We make a case distinction over which subset of $\{B_c, B_1^2, B_2^1, B_1^3, B_2^3\}$ the fence F' contains as well.

Note that regardless of inclusion or exclusion of the clause triangle, any fence crossing a and b, a and c or b and c has one non-collinear triple starting at one and ending at the other line segment. We therefore only analyze the cases, which do not contain the clause triangle.

We analyze these cases one by one. In all cases we will turn F and F' into shorter fences $\hat{F}$, $\hat{F}_1'$, and $\hat{F}_2'$ where $\hat{F}$ is going to be the fence including at least B_1^1 and B_1^2 and is formed by just including B_1^1 instead of bypassing it. The fences $\hat{F}_1'$ and $\hat{F}_2'$ containing the remaining triangles are going to be formed by splitting F' at some point and removing the remaining empty part. We call this *shortcutting* F' at the segment where we split it. Note that we can include B_1^1 into $\hat{F}$ at no additional cost. This reduces the number of segments in $\mathfrak{F}$ contradicting that it is a minimum link fencing. To shorten the notation we write $[a, b, c]$ for a sequence of non-collinear triple that F has to cross.

$\emptyset$: If B_1^1 is contained in its own fence, we could instead reroute F (Figure 5.19), and omit F' entirely, reducing the number of segments.

$\{B_1^2\}$ and F' has to cross the sets $[a^\star, a, b, b^\star, b, a, a^\star]$: We can shortcut F along b for an additional cost of one segment, which omits three segments, which are necessarily contained in the sequence $[b, a, a^\star, a, b]$.

$\{B_2^2\}$ and F' has to cross the sets $[a^\star, a, b, b', b'', b', b, a, a^\star]$: This fence can be shortcut along b' for an additional cost of one segment, which omits five segments, which are necessarily contained in the sequence $b', b, a, a^\star, a, b, b'$.

$\{B_1^2, B_2^2\}$ and F' has to cross the sets $[a^\star, a, b, b', b'', b', b, a, a^\star]$: This case can be resolved in the same way as $\{B_1^2\}$.

$\{B_2^2, B_2^3\}$ and F' has to cross the sets $[a^\star, a, b, b', b'', b', b, c, c', c'', c', c, a, a^\star]$: This fence can be shortcut along b' and along c', splitting it into two fences containing only inner polygons of the same wire. This can be done at an additional cost of two segments saving five segments, which are necessarily contained in the sequence $b', b, a, a^\star, a, c, c'$.

$\{B_1^2, B_2^2, B_2^3\}$ and F' has to cross the sets $[a^\star, a, b, b', b'', b', b, c, c', c'', c', c, a, a^\star]$: This fence can be shortcut along b and along c', splitting it into two fences containing only inner polygons of the same wire. This can be done at an additional cost of two segments saving four segments, which are necessarily contained in the sequence $b, a, a^\star, a, c, c'$ and one segment, necessarily contained in the sequence c', c, b.

$\{B_1^2, B_2^2, B_1^3, B_2^3\}$ and F' has to cross the sets $[a^\star, a, b, b', b'', b', b, c, c', c'', c', c, a, a^\star]$: This fence can be shortcut along b and along c, splitting it into two fences containing only inner polygons of the same wire. This can be done at an additional cost of two segments saving three segments, which are necessarily contained in the sequence $b, a, a^\star, a, c$.

$\{B_1^2, B_1^3\}$ and F' has to cross the sets $[a^\star, a, b, b^\star, b, c, c^\star, c, a, a^\star]$: This case can be resolved in the same way as $\{B_1^2, B_2^2, B_1^3, B_2^3\}$.

$\{B_1^2, B_2^3\}$ and F' has to cross the sets $[a^\star, a, b, b^\star, b, c, c', c'', c', c, a, a^\star]$: This case can be resolved in the same way as $\{B_1^2, B_2^2, B_2^3\}$.

$\{B_1^2, B_2^2, B_1^3\}$ and F' has to cross the sets $[a^\star, a, b, b', b'', b', b, c, c^\star, c, a, a^\star]$: This case can be resolved in the same way as $\{B_1^2, B_2^2, B_1^3, B_2^3\}$. □

Finally, we show that no minimum link fence of a clause gadget can ever fence two polygons that are in different wires. Again, this is shown essentially via a case enumeration that considers how a minimum link fence includes the first two to three polygons of each wire

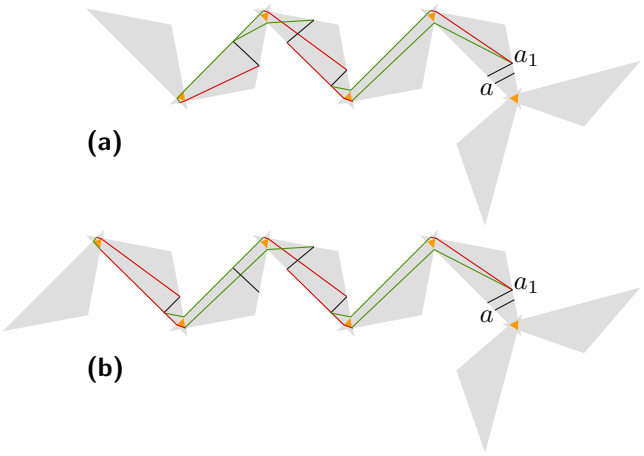


Figure 5.20: Non-collinear triples in a fence including at least (a) four or (b) five inner polygons of a wire.

together with the clause triangle. In each case we can conclude that there exists a fence with fewer segments that in fact does not use the inner polygons of two distinct wires.

Lemma 5.10. *Let $\mathfrak{F}$ be an optimal fencing of a clause gadget, then there exists no fence $F \in \mathfrak{F}$, which includes inner polygons belonging to two different wires.*

Proof. Assume there is a fence F, which includes inner polygons of two wires. By Lemma 5.9, we know that all inner polygons of a wire included in F are consecutive in that wire. First also assume that F contains $m > 3$ inner polygons of a single wire W_1. We split such a fence at a set a in T_1^1 (again using the sets marked in Figure 5.18) into two fences $\hat{F}$ containing all m inner polygons of the wire and F_o and we have $|\hat{F}| + |F_o| = |F| + 2$. By Lemma 5.6, we know that $\hat{F}$ contains at least $2m \geq 8$ segments. Assume m to be odd, then we can replace $\hat{F}$ with fences containing two consecutive polygons each (exactly $\frac{m-3}{2}$), plus one containing three inner polygons, resulting in $\frac{3m-9}{2} + 6$ segments, which is smaller than $2m - 2$ for any odd $m > 5$, yielding a better solution and contradicting $\mathfrak{F}$ being a minimum link fence. Assume m to be even. Then by Lemma 5.8, we obtain $\frac{3m}{2}$ segments in total, which is smaller than $2m - 3$ for any even $m > 4$, again yielding a better solution and contradicting $\mathfrak{F}$ being a minimum link fence.

It remains to analyze the cases $m = 4$ and $m = 5$. Assume $m = 4$. We obtained F_o by shortcutting F at a and therefore $|F_o| = |F| + 1$. Since there are seven segments completely contained in the part of F (the part of F starting and ending at a, and being completely contained in W_1) which is now omitted (using Observation 5.2), we can cover the four inner polygons of W_1 with two triangles, using only six segments, which contradicts $\mathfrak{F}$ being a minimum link fence.

Assume now $m = 5$. Since there are seven segments completely contained in the part of F which is now omitted (using Observation 5.2), we can cover the five inner polygons of W_1

with two fences including three and two inner polygons, respectively, i.e., $|\hat{F}_1| + |\hat{F}_2| = 9$. Therefore we can transform F into a fence, which includes less than 4 inner polygons of W_1 (in this case we turned it into F_o, which contains none), and it suffices to analyze all cases, in which we contain at most the first one, two or three inner polygons of any wire.

We again enumerate all cases, which are not symmetric to each other. We label these cases with the included polygon of each wire with the highest index (since no polygon of a wire is bypassed, this completely characterizes the included inner polygons). Also, since we are only investigating non-symmetric cases, we assume that the largest index of a polygon included in the first wire is greater or equal to the one in the second one, which in turn is greater or equal to the one in the third.

The possible cases are given below. For every case, we can give a tight lower bound on the number of segments, which are at least needed, for any fence including these inner polygons. This bound is achieved by analyzing non-collinear triples of such a fence and by providing a fence which achieves this bound. The complete enumeration is shown in Section 5.2.4.

- $\{B_1^3, B_2^3, B_3^3\}$
- $\{B_1^3, B_2^3, B_3^2\}$
- $\{B_1^3, B_2^3, B_3^1\}$
- $\{B_1^3, B_2^3\}$
- $\{B_1^3, B_2^2, B_3^2\}$
- $\{B_1^3, B_2^2, B_3^1\}$
- $\{B_1^3, B_2^2\}$
- $\{B_1^3, B_2^1, B_3^1\}$
- $\{B_1^3, B_2^1\}$
- $\{B_1^2, B_2^2, B_3^2\}$
- $\{B_1^2, B_2^2, B_3^1\}$
- $\{B_1^2, B_2^2\}$
- $\{B_1^2, B_2^1, B_3^1\}$
- $\{B_1^2, B_2^1\}$
- $\{B_1^1, B_2^1, B_3^1\}$
- $\{B_1^1, B_2^1\}$

For the first twelve of these fourteen cases, we show how F can be replaced with a set of fences, which in sum contain less segments than F contradicting that $\mathfrak{F}$ is a minimum link fence; for details, we refer again to Section 5.2.4). All replacement fences (including and excluding the clause triangle) are shown in Section 5.2.4. The only exceptions are the two cases $\{B_1^1, B_2^1, B_3^1\}$ and $\{B_1^1, B_2^1\}$. In both cases we can still replace F as shown in Section 5.2.4, however this yields a fencing, which uses the same amount of segments as F. Note that both fencings only contain fences, which are completely contained in $T = T_c \cup T_1 \cup \ldots \cup T_k$ (reusing the notation of the proof of Lemma 5.9). The replacements for both cases contain at least one fence, which includes only the first polygon of a wire. We will show that this implies that we can replace all fences in that wire with a different set of fences, which uses a smaller number of segments and thereby again contradicting that $\mathfrak{F}$ is a minimum link fencing.

Let F_1 be a fence in a wire including only the first polygon of that wire. Since every wire contains an even number of inner polygons and no fence includes four or more inner polygons by Lemma 5.7, we know that this wire has to contain at least one other fence of size one or three. Let $F_2 \neq F_1$ be a fence in the wire, such that, B_j is the inner polygon with the smallest index contained in F_2 and no other polygon B_k with $1 < k < j$ is contained in a fence of size one or three. Therefore there are an even number of inner polygons between F_1 and F_2. In particular if F_2 contains three inner polygons, we can create new fences, such that, B_2, B_3, B_4 are contained in one fence of

(a) $\mathcal{G}(v)$ in correct state (b) $\mathcal{G}(v)$ in incorrect state

Figure 5.21: If $\mathcal{G}(v)$ is in the correct truth state (a) inclusion of the first inner polygon of $\mathcal{G}(v, c)$ induces an additional cost of two links, otherwise (b) the additional cost is at least three.

size three and $B_5, \ldots, B_{k+2}$ are contained in $\frac{k-2}{2}$ fences of size two. F_1 and F_2 require nine segments in total, while we can replace them with two fences including B_1, B_2 and B_3, B_4, respectively, requiring only six segments. If F_2 contains one inner polygon, we can create new fences, such that, B_2 is contained in one fence of size one and $B_3, \ldots, B_k$ are contained in $\frac{k-2}{2}$ fences of size two. F_1 and F_2 require six segments in total, while we can replace them with one fence including B_1, B_2, requiring only three segments. □

We can now use Lemma 5.10 to argue that the clause triangle is only included in a fence together with inner polygons of at most one wire. We say that such a wire is in a *satisfying* state. The other two wires should therefore, by Lemma 5.8, only use fences including two inner polygons; leading to $\frac{3(k_a+k_b+k_c)}{2} + 3$ segments in total (k_a, k_b and k_c being the number of inner polygons in the wires). If we include the clause triangle in a fence of a wire, we get the same amount of segments, however, we can choose fences, such that, the last inner polygon of the wire which fences the clause triangle, is fenced alone. This will be crucial in the argument of how the wires and therefore the clause gadget interacts with the variable gadget.

Interaction with the variable gadgets It remains to describe the interaction between the variable and clause gadgets. Depending on the state of the variable gadget we can fence the last inner polygon of a wire in the fence of a variable gadget. We provide a fence with 5 segments (i.e., two additional ones) for the case, where the variable gadget is in the correct state and the existence of six non-collinear triples for the other case, see Figure 5.21.

Lemma 5.11. *The last inner polygon of a wire can be included in a fence of the variable gadget, whose spike it is connected to for the cost of two additional segments if the variable gadget is in the correct state and at least three additional segments otherwise.*

Concluding the interaction between clause and variable gadget we show that given a variable gadget is in the correct state w.r.t. a clause gadget we can fence the inner polygons of the wires of a clause gadget using 3/2 segments per polygon and adding only two segments to the fence of the variable gadget.

Lemma 5.12. *If and only if at least one of the connected variable gadgets is in the correct state, the clause gadget can be fenced with a total of* $\frac{3(k_a+k_b+k_c)}{2}$ *segments plus two additional segments to a fence of the variable gadget, which is connected to the wire in the satisfying state.*

Proof. Note that inclusion of the last polygon of a wire into a fence of a variable gadget always incurs an additional cost (of two or three, depending on the state of the gadget). The only reason a minimum link fencing would choose to do so is the fact that, such an inclusion reduces the number of inner polygons, which have to be fenced in a wire from the even number k down to the uneven number $k-1$, which in turn allows the wire to treat the clause triangle as its first inner polygon, and we save the three segments of the clause triangle, leading to a cost of $\frac{3(k_a+k_b+k_c)}{2}$ and the variable gadget segment number increases from 12 to 14.

Note further that a minimum link fencing would never choose to do so for two or three wires, since the benefit of including the clause triangle in a fence of a wire can only be achieved once. Assume that there is a second wire in the satisfying state, which add a further charge of (at least) two additional segments at its variable gadget. By Lemma 5.10, we know that the clause triangle is only included in a fence of one of the two wires. The second wire has to fence an odd number of inner polygons and therefore has to include either a fence of size one or three. It therefore still requires $\frac{3k}{2}$ segments. Therefore only one wire will ever be put in the satisfying state, even if two or all three variables would allow their connected wires to be put in the satisfying state for an additional charge of two segments each.

□

5.2.4 Complete enumeration of possible fences at the clause triangle

This section illustrates the enumeration argument of the proof of Lemma 5.10. Every case is shown four times in a column. The first row shows the lower bound on the number of segments for any fence including all polygons of this case, indicated by the three numbers in the upper left corner. These labels should be read as XYZ corresponding to the case $\{B_X^1, B_Y^2, B_Z^3\}$. The second row shows that all given bounds are in fact tight, as they can be achieved with the shown blue fences. The third and fourth row show (except for cases 111 and 110), that such a fence can be replaced with a collection fences, which in total achieve a lower number of segments, while either including (3rd row in green) or excluding (4th row in red) the clause triangle, i.e., in both cases, the original fence was not minimal.

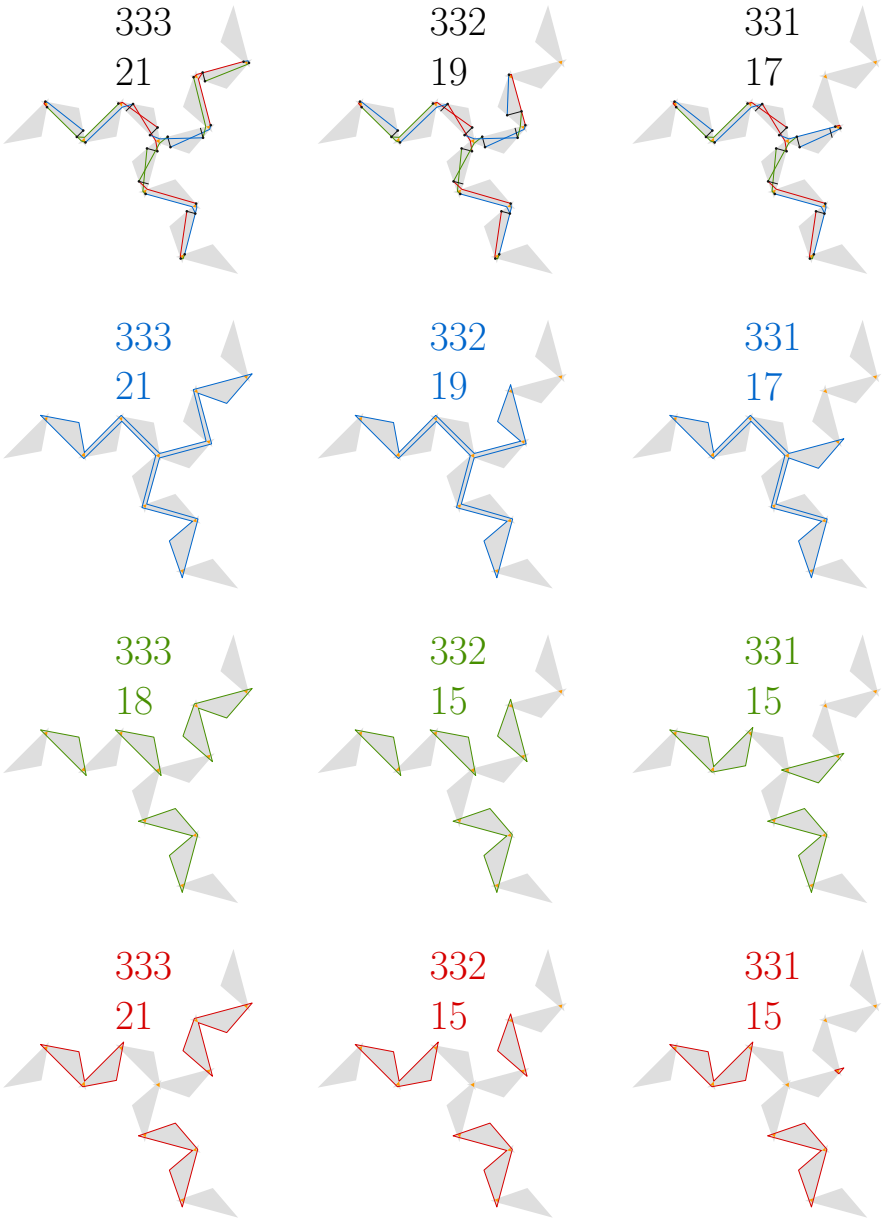


Figure 5.22: Fences including exactly 3 inner polygons of one and at most 3 of any other wire.

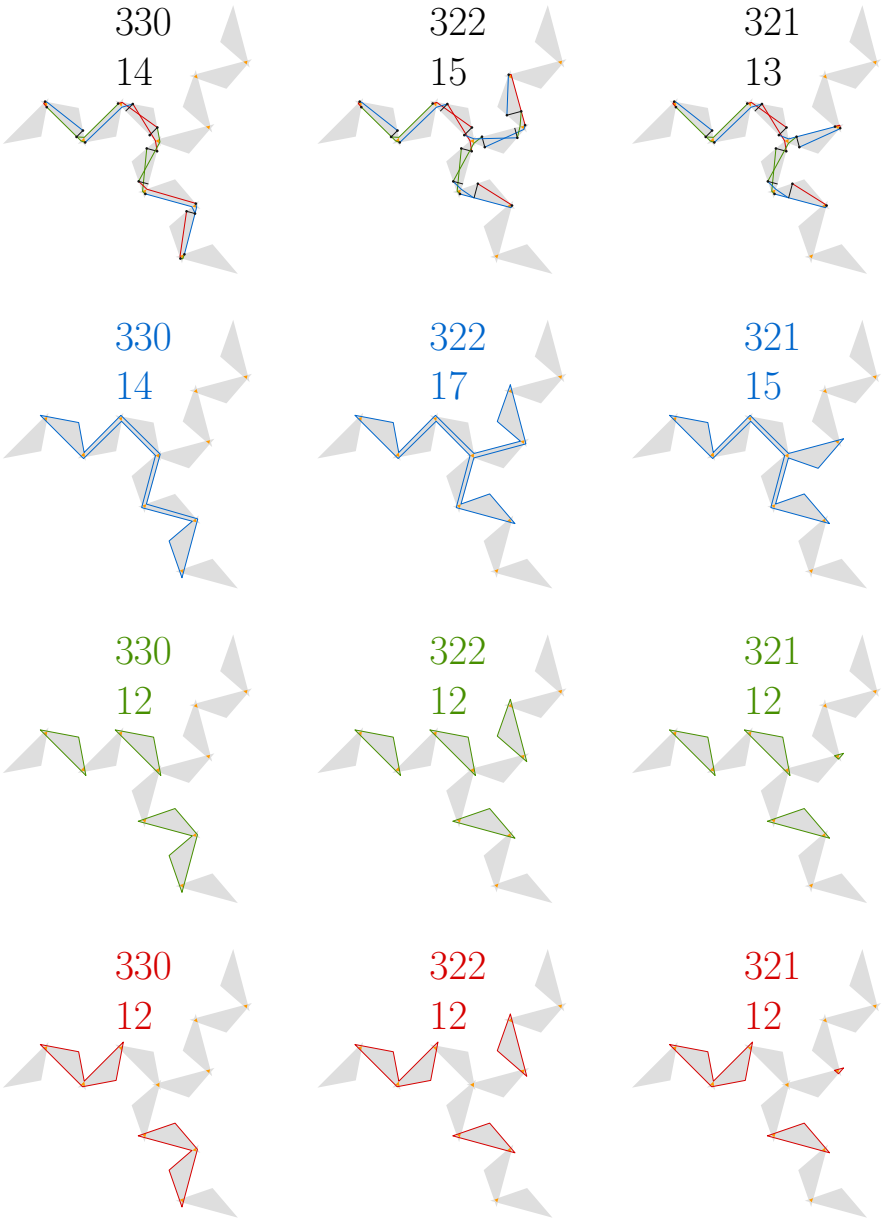


Figure 5.23: Fences including exactly 3 inner polygons of one and at most 3 of any other wire.

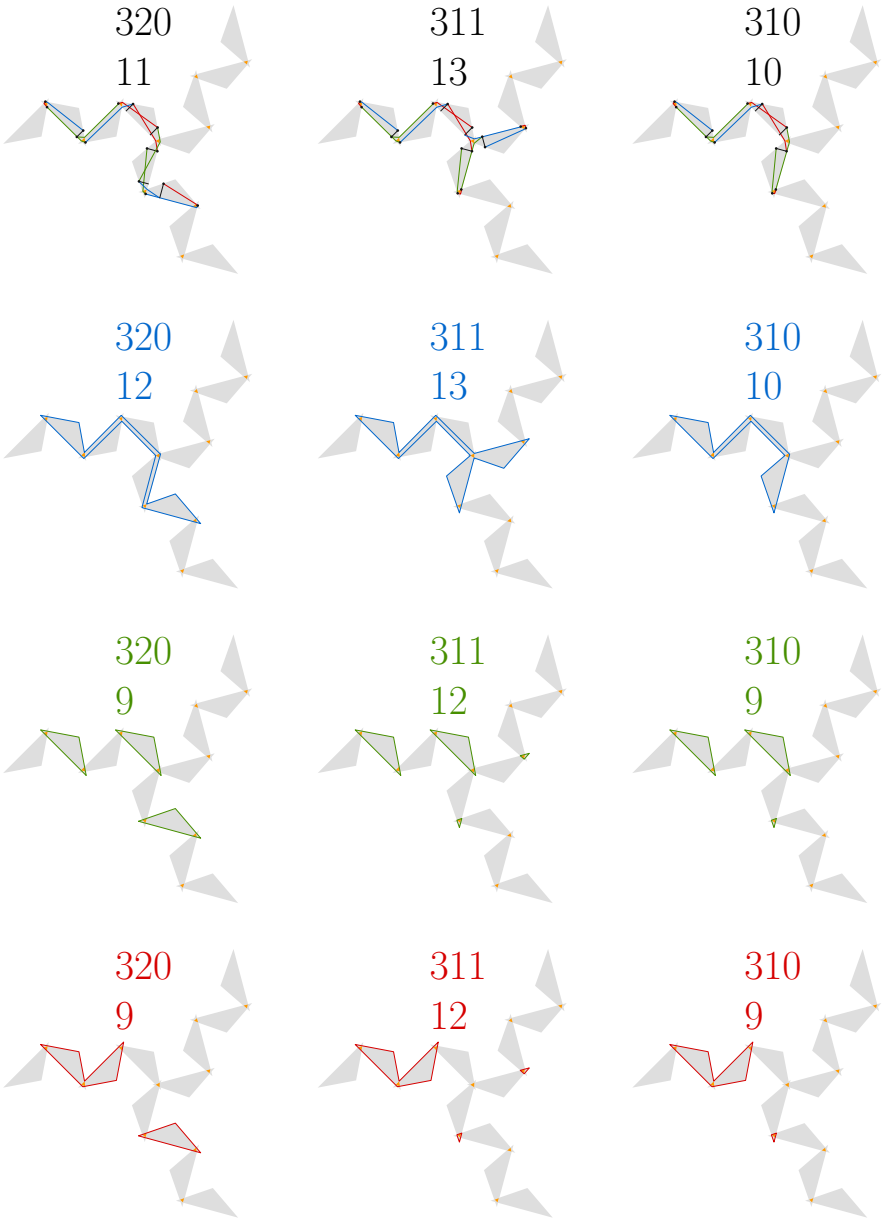


Figure 5.24: Fences including exactly 3 inner polygons of one and at most 3 of any other wire.

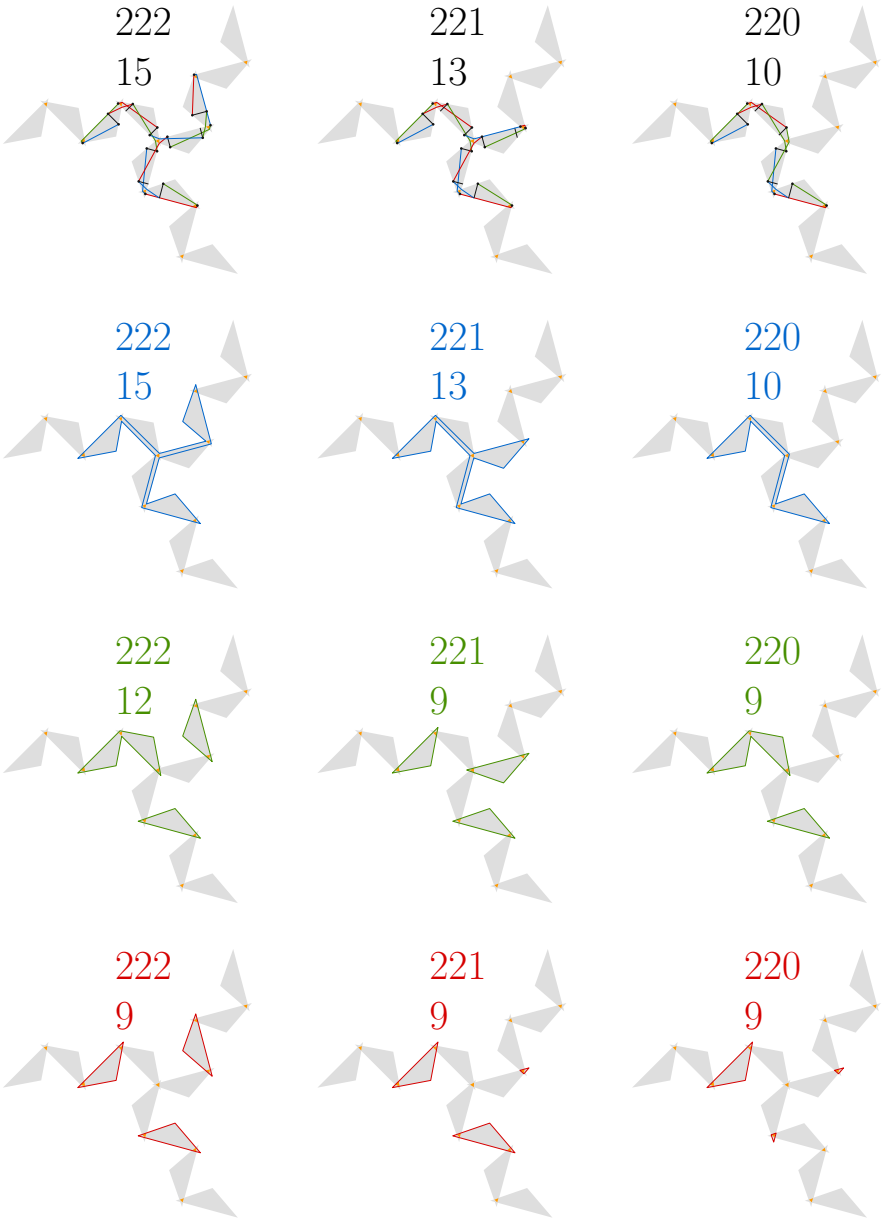


Figure 5.25: Fences including exactly 2 inner polygons of one and at most 2 of any other wire.

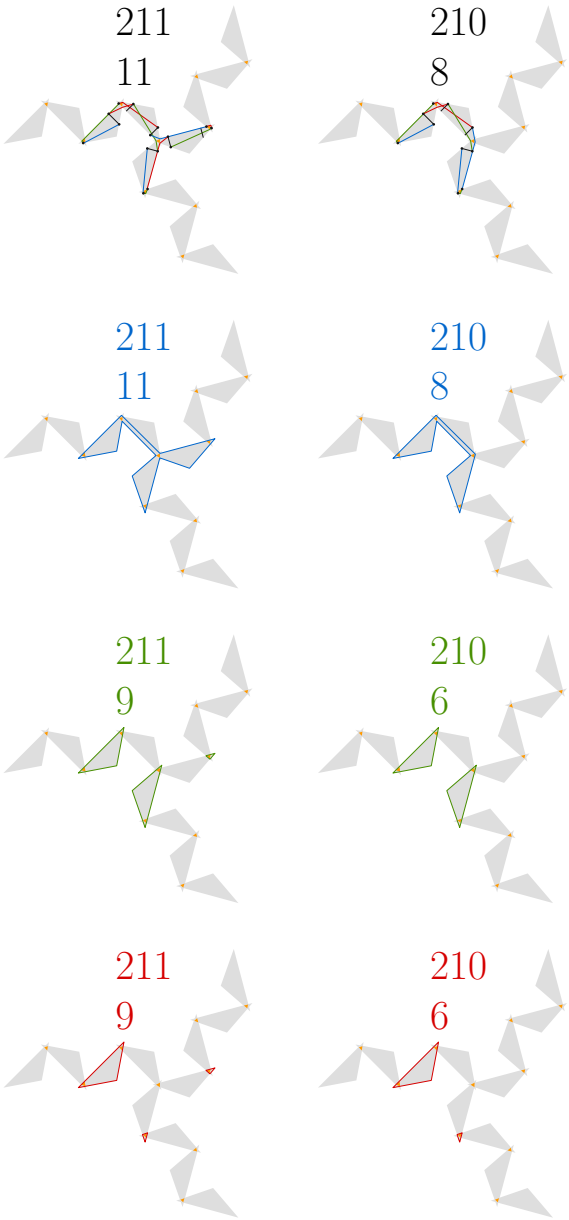


Figure 5.26: Fences including exactly 2 inner polygons of one and at most 2 of any other wire.

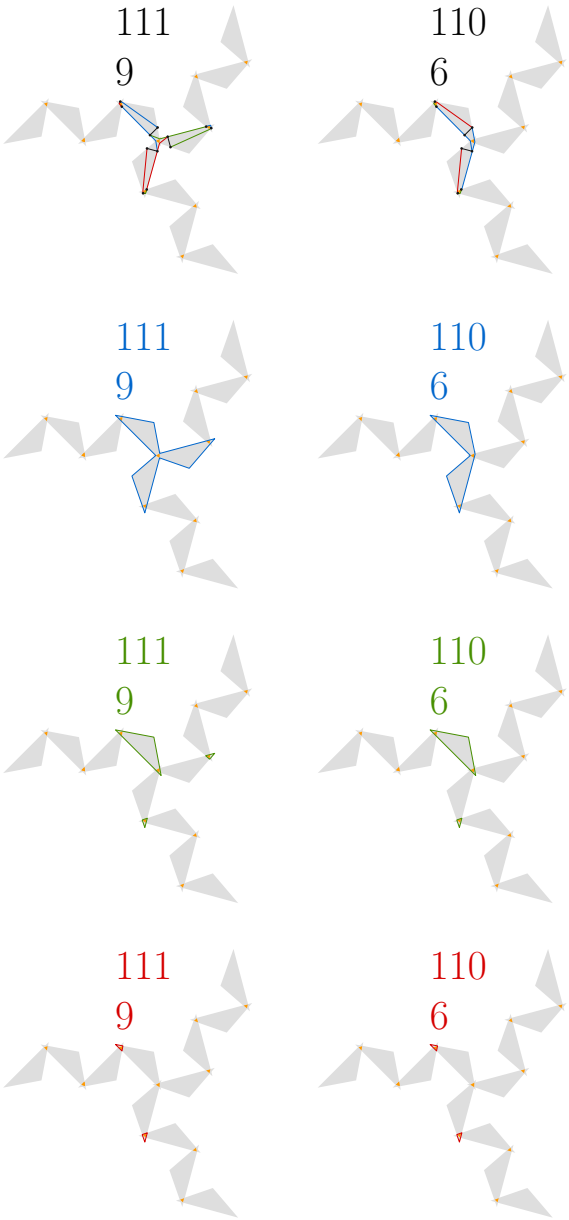


Figure 5.27: Fences including exactly 1 inner polygons of one and at most 1 of any other wire.

5.2.5 Correctness

It remains to argue the correctness of our reduction which then implies our main theorem.

Theorem 5.1. *Two-colored BMLF is* NP*-hard even when restricting all fences to include at most three polygons.*

Proof. Given an instance of planar 3, 4-SAT, we create and place a variable gadget $\mathcal{G}(v)$ for every variable $v \in \mathcal{V}$ and a clause gadget $\mathcal{G}(c)$ for every clause c, as described above. The wires of $\mathcal{G}(c)$ connect to a true spike of $\mathcal{G}(v)$ if v appears as a positive literal and to a negative spike if v appears as a negative literal in c.

Assume we are given a satisfying variable assignment for the instance of planar 3, 4-SAT. By Lemma 5.12 in order to be able to find a fencing with $\frac{3(k_a+k_b+k_c)}{2}$ segments for every clause gadget, at least one wire must be in the satisfying state. If one wire is in the satisfying state we can include the last inner polygon of the wire for an additional two segments if and only if the corresponding variable gadget is in the proper state (Lemma 5.11). Since every clause has one variable, which satisfies this clause, we choose this variable gadget to be in this state, and fence the variable gadget accordingly, leading to twelve segments plus two segments, per clause, which is connected via a wire in the satisfying state to it. Since in the variable assignment, every clause has such a literal and every variable is either true or false, we can do this for every clause and are never required to put a variable gadget both in its true and its false state. The final cost is $|\mathcal{V}| \cdot 12 + \sum_{c \in \mathcal{C}} (\frac{3(k^c)}{2} + 2)$, where k^c is the number of inner polygons of all three wires of $\mathcal{G}(c)$ summed up.

Now assume that we are given a fencing of the created BMLF instance with exactly $|\mathcal{V}| \cdot 12 + \sum_{c \in \mathcal{C}} (\frac{3(k^c)}{2} + 2)$ segments. This implies that for every clause gadget, there is one wire gadget in the satisfying state. We follow this wire up to the variable gadget, which has to be in the true state if the wire is connected to a true spike and in a false state otherwise. We set the corresponding variable of the variable gadget to true in the former and to false in the latter case. Therefore every clause has one variable, which satisfies the clause. Since no fencing of a variable gadget exists, in which both a wire connected to a true spike and a wire connected to a false spike can be put into the satisfying state (Lemma 5.11), the implied variable assignment is consistent. Finally, this might not necessarily assign all variables to a fixed truth assignment, since even if a clause might be satisfiable with two or even all three literals, we will never set more than one wire into the satisfying state. All variables, which do not have a truth value assigned yet, can safely be assigned a random value (e.g., true or false if their variable gadget is true or false).

We conclude that the instance of planar 3, 4-SAT is satisfiable if and only if the constructed two-colored BMLF instance admits a fencing with exactly $|\mathcal{V}| \cdot 12 + \sum_{c \in \mathcal{C}} (\frac{3(k^c)}{2} + 2)$ segments.

□

5.3 An XP-algorithm for BMLF with at most two polygons in each fence

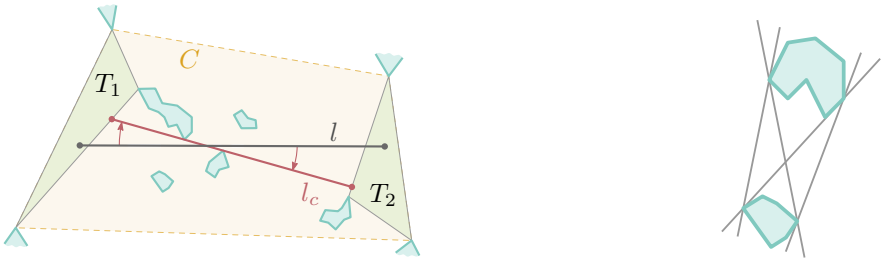
In Section 5.2 we showed that BMLF is NP-hard when there are only two colors, each fence contains at most three polygons, and each fence consists of at most five links. In contrast, we are going to show in this section that BMLF can be solved in XP-time when parameterizing the problem by the maximum number of links in any fence and allowing at most two polygons per fence, i.e., the problem can be solved in polynomial-time when fixing the maximum number of links in any fence and restricting each fence to contain at most two polygons.

For our algorithm we make use of the following result derived from the work of Hershberger and Snoeyink [63]. It allows us to compute for a given *loop*, i.e., a closed polygonal curve, inside a polygon with holes, a minimum-link loop of the same homotopy in time $O(nk)$, where n is the complexity of the polygon and k is the size of the resulting fence.

Theorem 5.2 (Derived from Section 5.2 [63])**.** *Given a polygon P without self-intersections but potentially with holes of complexity n, an integer k, and a loop α lying in the interior of P with $O(nk)$ corners, we can decide in time $O(nk)$ if there exists a loop α' of the same homotopy-class as α with at most k links.*

Remark 1. *It is worth noting that in the paper by Hershberger and Snoeyink [63] Theorem 5.2 is only stated in text. The runtime is given as $O(C_\alpha + \Delta_\alpha + \Delta_{\alpha'})$, where C_α is the complexity of α, the free space between polygons is assumed to be triangulated and Δ_α and $\Delta_{\alpha'}$ are the number of triangulation edges intersected by α and the fence α', respectively. However an example of an instance with multiple obstacles is given, in which $\Delta_{\alpha'} \in \Omega(nk)$, where n is the number of corners over all polygons. Since in our scenario we can find a path α such that $C_\alpha \in O(nk)$ and $\Delta_\alpha \in O(nk)$, we can make the assumption that α''s complexity is in $O(nk)$.*

Let P be a polygon without self-intersections. We denote with $\mathcal{T}_P = \{T_1, \ldots, T_z\}$ a triangulation of P with triangles $T_1, \ldots, T_z$. Note that we do not require any further properties of $\mathcal{T}_P$. If P is clear from the context we omit it and set $\mathcal{T} = \mathcal{T}_P$. Let $T_1, T_2 \in \mathcal{T}$ be two triangles and let l be a line segment with endpoints p and q such that $p \in T_1$ and $q \in T_2$. We call l a *splitting segment*. Consider Figure 5.28a for an example for T_1 and T_2 if l contains no points of $\mathbb{R}^2 \setminus P$. Intuitively, a splitting segment separates the holes that intersect the convex hull of $T_1 \cup T_2$ into two sets. Let $\mathcal{H}$ be all the holes of P that intersect or are fully contained in the interior of the convex hull of $T_1 \cup T_2$. We say that a hole $H \in \mathcal{H}$ is to the *left* (*right*) of l if the from p to q oriented supporting line of l leaves H in the left (right) half-plane. We call two splitting segments of T_1 and T_2 *equivalent* if the same holes of $\mathcal{H}$ are to their respective left and right. Segments which intersect holes are not splitting segments.



(a) A splitting segment l_c equivalent to l and bitangent to two holes. (b) Bitangents

Figure 5.28: (a) We can obtain the splitting segment l_c from a splitting segment l by rotating l clockwise until it is a bitangent to two holes. (b) Any pair of two polygons admits at most four bitangents, only one of which can not be rotated clockwise without intersecting one of the polygons.

Lemma 5.13. *Let P be a polygon without self-intersection, $\mathcal{H}$ a set of holes and $\mathcal{T}$ a triangulation of P. Then for every pair of triangles $T_1, T_2 \in \mathcal{T}$ with $T_1 \neq T_2$ there are at most $4|\mathcal{H}|^2$ different equivalence classes of splitting segments.*

Proof. Let $\mathcal{H}^\cap \subseteq \mathcal{H}$ be the set of holes, which intersect the joint convex hull C of T_1 and T_2 or are fully contained in it. Let l be a splitting segment with endpoints $p \in T_1$ and $q \in T_2$. Observe, that since l is completely contained in $C - \mathcal{H}$ we, by definition of equivalent splitting segments, can disregard any hole $H \notin \mathcal{H}^\cap$.

Let $l \in C$ be a splitting segment of T_1 and T_2, which has a set $L \subseteq \mathcal{H}^\cap$ of holes to its left and a set $R \subseteq \mathcal{H}^\cap$ to its right. There exists another splitting segment l_c whose supporting line is a bitangent on two holes, say $H_L \in L$ and $H_R \in R$, such that l_c has also L to its left and R to its right. We obtain the splitting segment l_c from l by rotating l first around its center point until it touches either H_L or H_R in a point r and then continue to rotate l around r until it touches the second hole, see Figure 5.28a. Since any pair of holes has at most four bitangents (Figure 5.28b) there are at most $4|\mathcal{H}^\cap|^2$ equivalence classes. □

Theorem 5.3. *Given an instance $\mathcal{P}$ of BMLF with outer polygon $Q \in \mathcal{P}$, we can decide in time $O(kn^{2k+4})$ if a minimum link fencing $\mathfrak{F}$ of $\mathcal{P}$ exists, in which every fence contains at most two polygons, each fence in the fencing has at most k segments, and n is the number of corners in $\mathcal{P}$.*

Proof. Throughout, we consider the triangulation $\mathcal{T}$ of the free space of $\mathcal{P}$. Let $t+1$ be the number of colors in the given instance and $c_1, \ldots, c_t$ be the colors of polygons in $\mathcal{P}$ with $c_i \neq f(Q)$ for all $i = 1, \ldots, t$.

Observe that the homotopy of a fence including exactly one polygon $P \in \mathcal{P}$ is unique. We find a path α with this homotopy in the triangulated free space between the polygons in $\mathcal{P}$ by traversing the boundary of P clockwise and at every corner of the polygon adding

every incident triangle of $\mathcal{T}$ in clockwise order to a list. This yields a series of triangles from which we can construct a loop α such that P is contained in α. This can be done, for example, by connecting all midpoints of triangulation segments of a triangle and its successor in the loop. Then, we use Theorem 5.2 to obtain a minimum-link fence for P from α or determine that no fence with at most k links exists. Computing all individual fences requires $O(|\mathcal{P}|kn) = O(kn^2)$ time.

Next, we consider every pair $P_1, P_2 \in \mathcal{P}$ with $P_1 \neq P_2$, $c_i = f(P_1) = f(P_2)$ and $i \in \{1, \ldots, t\}$. Note that in contrast to a fence containing only one polygon, a fence containing exactly two polygons can belong to several different homotopy-classes w.r.t. the remaining polygons.

Now, we describe how to compute a minimum-link fence F for two polygons P_1 and P_2 of $\mathcal{P}$. Recall that all polygons in $\mathcal{P}$ together have n corners. Since we have only $O(n)$ triangles in $\mathcal{T}$ we can iterate over all possible $O(\binom{n}{k}) \in O(n^k)$ ordered tuples of k triangles. Fix in the following such an ordered k-tuple $(T_1, \ldots, T_k)$ of triangles in $\mathcal{T}$. There are only $O(|\mathcal{P}|^2)$ many non-equivalent splitting segments connecting points in triangles T_i and T_{i+1} by Lemma 5.13. Consequently, we can iterate over the $O((|\mathcal{P}|^2)^k) = O(n^{2k})$ many different combinations of splitting segments between consecutive triangles. In case there are two consecutive triangles between which no possible splitting line exists we reject this tuple of triangles. Assume in the following that we fix for every pair T_i and T_{i+1} a splitting segment l_i.

It remains to construct a plane loop α as input for the algorithm of Hershberger and Snoeyink [63] or decide that no such loop exists for the fixed choices of triangles and splitting segments. From the triangles $T_1, \ldots, T_k$ and the splitting segments $l_1, \ldots, l_k$ we derive a sequence of triangles $\tau_1, \ldots, \tau_z$ of $\mathcal{T}$ that α has to visit. Since the triangulation $\mathcal{T}$ is defined by the corners of polygons in $\mathcal{P}$ each splitting segment l_i gives rise to a unique sequence of triangles. We concatenate all these sequences starting with the sequence induced by l_1 to obtain the sequence $\Theta = (\tau_1, \ldots, \tau_z)$. Observe, that triangles along this sequence may repeat and that $\tau_1 = \tau_z$.

It remains to decide if there exists a plane loop α visiting each triangle of Θ in order. To make the following description simpler let s_i be the shared boundary of τ_i and τ_{i+1}. If for no i with $i \in \{1, \ldots, z-1\}$, we find that $s_i = s_{i+1}$ we create a loop α by connecting the centerpoint of s_i with the one of s_{i+1}. Since no boundary repeats, this is always possible without any centerpoint and hence triangle being used twice. Finally, we add the segment from the centerpoint of s_z to the centerpoint of s_1 which is also always possible as $\tau_1 = \tau_z$.

Now assume there exist at least two indices i and j with $i \neq j$ and $i, j \in \{1, \ldots, z\}$ such that $s_i = s_j$. Build the loop as before and let $\alpha_1, \ldots, \alpha_z$ be the segments in the constructed loop. Since we allow repeated boundaries there exist subsequences among the $\alpha_1, \ldots, \alpha_z$ that are repeated. In the following we assume that we only consider inclusion maximal repeated subsequences. Let $A = \{\alpha'_1, \ldots, \alpha'_a\}$ be one occurrence of such a subsequence of the α_i's that is repeated at least once and let $\hat{A}$ be a different

occurrence. Let $s'_1, \dots, s'_a$ be the subsequence of the s_i's that correspond to the triangle boundaries passed by the segments in A and $\hat{A} = \{\hat{\alpha_1}, \dots, \hat{\alpha_a}\}$. Now observe that in a plane loop the vertices of A and $\hat{A}$ have to always appear in the same order along $s'_1, \dots, s'_a$. If they would not, let s'_i and s'_{i+1} for $i \in \{1, \dots, a-1\}$ be two segments such that the vertices of A and $\hat{A}$ on s'_i and s'_{i+1} are not in the same order. Without loss of generality assume the vertex of A on s'_i is above the one of $\hat{A}$ and the opposite is true for s'_{i+1}, then we find that α_i and $\hat{\alpha_i}$ cross. Hence, the only decision to make is to decide, for each pair of repeated subsequences, in which order their vertices appear along the corresponding triangle-boundaries. Since every repeated subsequence implies an intersection between two segments l_i and l_j with $i \neq j$ and $i, j \in \{1, \dots, k\}$ we find that there are at most k^2 such repeated sequences. Consequently, there are at most $2^{O(k)}$ possible ways to distribute the center points in each shared part along the boundaries.

To sum up, for one pair of polygons P_1, P_2 we have to consider $O(n^{2k})$ possible non-homotopy equivalent fences and for each homotopy we can check in $O(n^2 \cdot 2^k)$ if there exists a plane loop α realizing it, leading to a total runtime of $O(2^k \cdot n^{2k+2})$ to enumerate every potential homotopy of a minimum link fence. For each of the $O(n^{2k})$ different homotopies, we can use Theorem 5.2 to compute a minimum link fence in $O(kn)$, hence we can compute a minimum link fences for all pairs of polygons in $O(n^2 \cdot kn^{2k+2}) = O(kn^{2k+4})$.

If for any polygon no fence, alone or in a pair with another polygon, with k or fewer links is found, we return that no solution exists. Otherwise, let λ_{uv} be the number of links for a minimum link fence containing $P_u, P_v \in \mathcal{P}$ and λ_u the number of links for a minimum link fence containing only $P_u \in \mathcal{P}$. Consider a complete graph G containing one vertex u for each polygon $P_u \in \mathcal{P}$ and one more vertex x if $|\mathcal{P}|$ is odd. Set the edge-weights $w(u, v) = \min\{\lambda_{uv}, \lambda_u + \lambda_v\}$ and $w(x, u) = \lambda_u$ for $P_u, P_v \in \mathcal{P}$. If for some $P_u \in \mathcal{P}$ or pair $P_u, P_v \in \mathcal{P}$ no fence with $\leq k$ segments existed we remove that edge.

To find a minimum-link fencing of $\mathcal{P}$ it now suffices to compute a minimum weight perfect matching in G. Let M be such a matching. Then, a minimum link fencing $\mathfrak{F}$ of $\mathcal{P}$ can be constructed from M in the following way. If $uv \in M$, we add the (pre-computed) minimum link fences containing only P_u and P_v to $\mathfrak{F}$ if the weight $w(u, v) = \lambda_u + \lambda_v$ or the fence containing P_u and P_v if $w(u, v) = \lambda_{uv}$. If $|\mathcal{P}|$ was odd we also find an edge $xu \in M$ and we add the fence containing only P_u to the fencing.

Finding a minimum weight perfect matching in a general graph with V vertices and E edges can be done for example in $O(V^2E)$ time via finding a maximum weight perfect matching (e.g. [51]) in the same graph with edge weights set to maximum edge-weight plus one minus the original edge-weight. Since G has $O(|\mathcal{P}|)$ vertices and $O(|\mathcal{P}|^2)$ edges we can compute this matching in $O(|\mathcal{P}|^4) = O(n^4)$, which is dominated by the initial computation of the minimum link fences. □

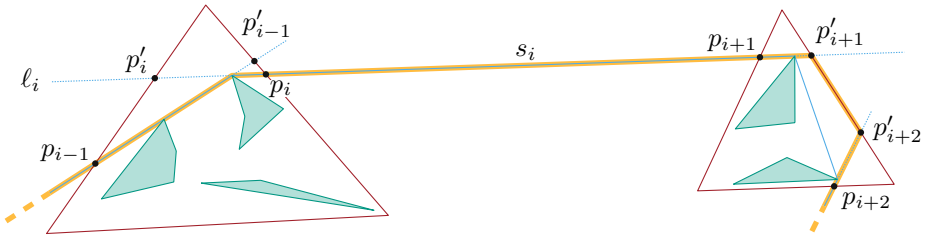


Figure 5.29: Computing a new fence (orange) from the old fences (purple) and the convex hull (blue).

5.4 An algorithm for two-colored CMLF

In this section we present an algorithm for solving two-colored CMLF. Computing a minimum-link fence in this setting can be done by computing a fence for the convex hull of the contained polygons with the algorithm by Wang [124] which runs in time $O(n \log n)$ with n being the number of corners of the contained polygons. Throughout this section an instance of CMLF is given as $(\mathcal{P}, Q)$ where Q is the outer polygon and $\mathcal{P}$ is the set of polygons contained in Q.

Lemma 5.14. *Given an instance $(\mathcal{P}, Q)$ of two-colored CMLF, let $\mathfrak{F}$ be a solution for $(\mathcal{P}, Q)$. There exists a solution $\mathfrak{F}'$ for the two-colored CMLF instance $(CH(\mathcal{P}), Q)$ with $|F| = |F'|$.*

Proof. We construct a fence F° for $CH(P, Q)$ from F where the number of segments in F° is at most the total number of segments in F. Let $(p_1, \ldots, p_z)$ be the intersection points between F and $CH(\mathcal{P})$ ordered as they appear in a clockwise traversal of the convex hull, and observe that z is even. Let p_i, p_{i+1} be pairs of intersection points between F and $CH(\mathcal{P})$ such that the straight-line segment s_i connecting p_i and p_{i+1} lies on $CH(\mathcal{P})$ and completely outside of F (see Figure 5.29). Consider the supporting line ℓ_i of s_i. If the fence lies completely in one of the closed half-planes bounded by ℓ_i we add s_i to F°. Assume this is not the case. As s_i is on $CH(\mathcal{P})$ we get that ℓ_i does not intersect any polygon in $\mathcal{P}$. Moreover, as $\mathfrak{F}$ consists of closed simple polygons we find two intersection points p'_i and p'_{i+1} that lie on ℓ_i, such that, the parts of F appearing in a clockwise traversal from p'_i to p_i, as well as the ones in a clockwise traversal from p_{i+1} to p'_{i+1} lie outside of $CH(\mathcal{P})$. We add the segment s'_i between p'_i and p'_{i+1} to F°. Doing this for every pair of intersections we obtain a set of segments F°, where all segments are on the convex-hull of $\mathcal{P}$. Note that it is possible for these segments to intersect; if that is the case we only keep the parts until their intersection point. Finally, the start and end-points of connected chains of segments in F° lie on segments of fences in $\mathfrak{F}$. We can convert F° into a fence of $CH(\mathcal{P})$ by connecting these endpoints along the fences in $\mathfrak{F}$ and that fence will be disjoint from $\mathcal{P}$ (except possibly touching $\mathcal{P}$ in corner points).

It remains to argue that indeed $|F^\circ| \leq |F|$. We partition F° into two categories, segments that coincide with segments in F and segments that do not. Each of them is either a

full segment of F or originates from the intersection of at most two different s_i''s and a segment of F. Furthermore, we add $z/2$ segments s_i' that are not sub-segments of segments in F. For each such s_i' we find at least one segment of F for which we did not add any sub-segment to F°. These are the segments of F on which p_i and p_{i+1} lie or that are fully outside of F°. □

Theorem 5.4. *Two-colored CMLF can be solved in time $O(n \log n)$ where n is the number of corners of polygons in $\mathcal{P}$.*

5.5 Conclusion

We have shown BMLF to be NP-hard even if every fence contains at most three polygons, each fence has at most five links, and only two different colors of polygons are present. Our reduction holds regardless of requiring disjoint fences or not. Note, that our reduction can be adapted to not require the outer bounding polygon Q. Instead, we can replace Q by one polygon with a narrow and very complex channel, connecting the "inside" with the "outside". On the algorithmic side, we gave an XP-algorithm for BMLF parameterized by the maximum number of links in a fence and allowing at most two polygons per fence. We also showed that two-colored CMLF can be solved in polynomial time.

It is open if one can eliminate the exponential dependency on the number of links in our algorithm for BMLF. Furthermore, while our reduction holds when replacing the outer bounding polygon, our algorithm does not since we cannot immediately apply Theorem 5.2. Similarly, requiring the fences to be disjoint for BMLF is an interesting open direction.

A generalization of the algorithmic result to the case with more than two colors is interesting. It is reasonable to assume that the XPalgorithm simply generalizes to more than one color, by treating all other colors as being part of the color of the outer polygon.

And finally the complexity of **SMLF**, where the convex hull of all inner polygons is not entirely contained within the outer polygon is still open. Since our reduction is heavily dependent on a large number of highly complex holes in the outer polygon, we do not expect that it generalizes to SMLF.

CHAPTER 6

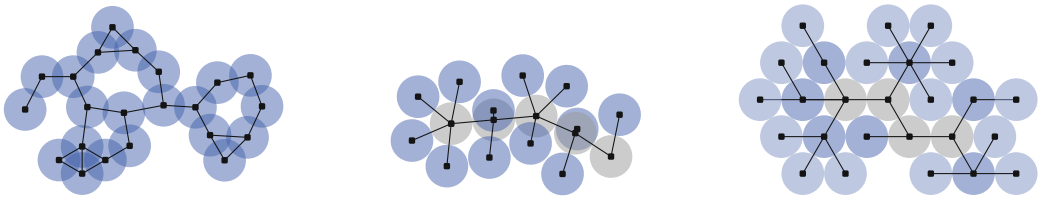
Representing Network Topology with Intersecting Unit Disks

This chapter is (partially) based on the following publications:
[22]: Bhore et al. – Unit Disk Representations of Embedded Trees, Outerplanar and Multi-legged Graphs (GD'21)
[88]: Bhore et al. – Recognition of Unit Disk Graphs for Caterpillars, Embedded Trees, and Outerplanar Graphs (EuroCG'21)

The representation of graphs as contacts or intersections of disks has been a major topic of investigation in geometric graph theory and graph drawing. The famous circle packing theorem states that every planar graph has a representation by touching disks (of not-necessarily the same size) and vice versa [74]. Since then, a large body of research has been devoted to the representation of planar graphs as contacts or intersections of various kinds of geometric objects [33, 34, 52, 62]. In this chapter, we are interested in unit-radius disks. A set of unit disks in $\mathbb{R}^2$ is a *unit disk intersection representation* (UDR) of a graph $G = (V, E)$, if there is a bijection between V and the set of unit disks such that two disks intersect if and only if they are adjacent in G. *Unit disk graphs* are graphs that admit a UDR. *Unit disk contact graphs* (also known as *penny graphs*) are the subfamily of unit disk graphs that have a UDR with interior-disjoint disks, which is thus called a *unit disk contact representation* (UDC). This can be relaxed to *weak* UDCs, which permit contact between non-adjacent disks; see Figure 6.1 for examples.

6.1 Related Work

The objective of the recognition problem is to decide whether a given graph admits a UDR. This problem has a rich history [25, 26, 64, 65]. Breu and Kirkpatrick [27] proved that it is NP-hard to decide whether a graph G admits a UDR or a UDC, even for planar



(a) Outerplanar graph (b) Caterpillar graph (c) Lobster graph

Figure 6.1: We investigate specific contact and intersection graphs of unit disks. In a UDR **(a-b)** disks are allowed to overlap, and contact of two disks implies an edge between their vertices. In a weak UDC the disks are interior disjoint, but contact between non-adjacent disks is allowed. The disks of spine vertices are colored grey **(b-c)**. Note that the grey vertices in **(c)** are connected with an x-monotone polyline.

graphs. Klemz et al. [72] showed that recognizing outerplanar unit disk contact graphs is already NP-hard[1]. Moreover recognition of unit disk graphs is $\exists\mathbb{R}$-hard for general graphs [82].

Recognition with a fixed embedding is an important variant of the recognition problem. Given a plane graph G, the objective in this problem is to decide whether G admits a UDC in the plane that preserves the cyclic order of the neighbors at each vertex. Some recent works investigated the recognition problem of UDCs with/without fixed embedding, and narrowed down the precise boundary between hardness and tractability; see [25, 36, 38, 72]. A remaining open question is to settle the complexity of recognizing non-embedded trees that admit a UDC. Some of these works focused on *weak* UDCs, where disks of non-adjacent vertices may touch. In this model, the recognition of non-embedded trees that admit a weak UDC is NP-hard [38]. All results are summarized in Table 6.1.

While several results of the past years have shed light into the recognition complexity gap for UDCs, not much is known in this regard for the more general class of UDRs since the NP-hardness for planar graphs from 1998 [27].

Our Contribution. We investigate the unit disk graph recognition problem for subclasses of planar graphs, in particular trees, caterpillars and lobsters (which we summarize under the name *multi-legged graphs*). Section 6.3 is a step-by-step explanation, how a specific graph exhibits certain properties in every possible UDR.We show that recognizing unit disk graphs remains NP-hard for non-embedded outerplanar graphs (see Figure 6.1a) – strengthening the previous hardness result for planar graphs [27] – and for embedded trees (Section 6.4). This line of research aims to extend earlier investigations showing hardness for UDCs of outerplanar graphs [73] and embedded trees [25] to the UDR model and builds in particular on the work of Bowen et al. [25].

[1]The authors originally presented an algorithm to recognize caterpillars that admit a UDC in linear time [73]. However, due to a flaw in the proof the claim was weakened to a conjecture.

	graph class	weak UDC	UDC	UDR
non-embedded	general	↑ NP-hard	↑ NP-hard	∃ℝ-complete [82]
	planar	↑ NP-hard	NP-hard [27]	NP-hard [27]
	outerplanar	↑ NP-hard	NP-hard [72]	NP-hard (Thm.6.2)
	trees	NP-hard [38]	open	open
	lobsters	$O(n)$ (monotone, Thm.6.4)	open	open
	caterpillars	$O(n)$ [38]	open	$O(n)$ (Thm.6.3)
embedded	general	↑ NP-hard	↑ NP-hard	↑ NP-hard
	planar	↑ NP-hard	↑ NP-hard	↑ NP-hard
	outerplanar	↑ NP-hard	↑ NP-hard	↑ NP-hard
	trees	↑ NP-hard	NP-hard [25]	NP-hard (Thm.6.1)
	lobsters	↑ NP-hard	open	open
	caterpillars	NP-hard [36]	open	open

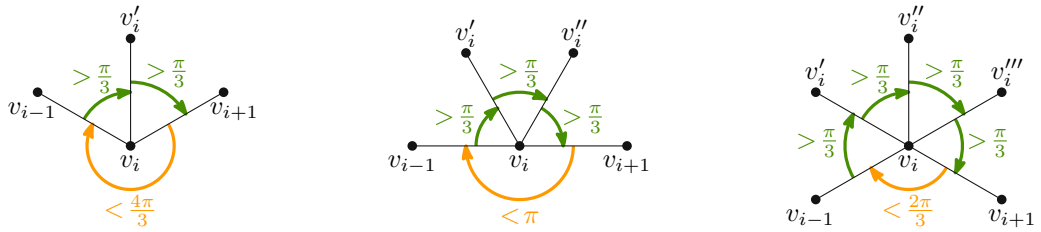
Table 6.1: State of the art, our results, and open problems on unit disk graph recognition. Upward arrows indicate that a result follows from the one below.

On the positive side, we provide a linear-time algorithm to recognize caterpillar graphs (see Figure 6.1b) that admit a UDR (Section 6.5). In Section 6.6, we return to the problem of recognizing unit disk contact graphs and extend the tractability boundary for non-embedded graphs. While it was known that a weak UDC for caterpillar graphs can be constructed in linear time (if one exists), and that the same recognition problem is NP-hard for trees [38], we prove that we can decide in linear time if a lobster graph admits a *monotone* weak UDC on the triangular grid, where a *lobster* is a tree whose internal vertices form a caterpillar (see Figure 6.1c) and a weak UDC is monotone if the disks of the path obtained by removing all leaves of the graph twice can be connected with a monotone polyline through their centers. Table 6.1 summarizes our results and remaining open problems.

6.2 Preliminaries

A graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ is a unit disk graph if there exists a set of closed unit disks $\mathcal{D} = \{d_1, \ldots, d_n\}$ in the plane and a bijective mapping $d\colon V \to \mathcal{D}$ such that $d(v_i) = d_i$ and $v_iv_j \in E$ if and only if d_i and d_j intersect. We call $\mathcal{D}$ a *unit disk intersection representation* (UDR) of G. If all disks in $\mathcal{D}$ are pairwise interior disjoint we also call $\mathcal{D}$ a *unit disk contact representation* (UDC) of G. A graph is a *unit disk contact graph* if it admits a UDC. A *weak* UDC permits contact between two disks d_i and d_j, even if $v_iv_j \notin E$.[2]

[2]Note that weak UDRs, in contrast, are not interesting, since a complete graph K_n can easily be represented as a UDR and therefore every graph admits a weak UDR.



(a) Observation 6.1 (b) Observation 6.2 (c) Observation 6.3

Figure 6.2: Illustration of three simple observations based on Lemma 6.1.

A *caterpillar* graph G is a tree which yields a path, when all leaves are removed. We call this path, the spine B_G of G. Similarly a *lobster* graph G' is a tree which yields a caterpillar graph G'', when all leaves are removed. The spine of G' is the spine of G''. For each vertex v of the spine, we denote the set of vertices that are reachable from v on a path that does not include any other spine vertex, as the *descendants* of v.

The UDR $\mathcal{D}$ of a graph G induces an embedding $\mathcal{E}_\mathcal{D}(G)$ of the graph G by placing every vertex v at the center of $d(v)$ and linking neighboring vertices by straight-line edges. The UDR $\mathcal{D}$ also induces a polygon $P_\mathcal{D}$, which is defined as the convex hull of the center points of all disks in $\mathcal{D}$. We will therefore also use v as the center of $d(v)$. Let a, b, c be three points in $\mathbb{R}^2$. We use $\measuredangle abc$ to denote the clockwise angle defined between the segments ab and bc. We use $\measuredangle d_i d_j d_k$ as the clockwise angle $\measuredangle v_i v_j v_k$ in $\mathcal{E}_\mathcal{D}(G)$.

6.3 Basic Lemmata

In this section we state the construction and properties of a fundamental graph, the augmented path, which will be central in the following sections. We start by stating some structural restrictions on the angles between centers of disks in a UDR. Let G' be an induced subgraph of a graph G, such that G' has three vertices v_1, v_2, and v_3 and two edges v_1v_2 and v_2v_3, i.e., G' is a 3-chain.

Lemma 6.1. *The angle $\measuredangle v_1v_2v_3$ of a UDR of G' is minimal if $|v_1v_2| = |v_2v_3| = 2$. Moreover a minimal angle is $> \pi/3$.*

Proof. Without loss of generality, assume that $|v_1v_2| \geq |v_2v_3|$. Then v_1 can be moved along the ray $\overrightarrow{v_2v_1}$ until $|v_1v_2| = 2$. Next it can be rotated around v_2 until $|v_1v_3| = 2$. Now $\measuredangle v_1v_2v_3 = \cos^{-1}(|v_2v_3|/4)$, which is monotonically decreasing in the interval $0 < |v_2v_3| \leq 2$. Since $\cos^{-1}(2/4) = \pi/3$ and we know that $|v_1v_3|$ is strictly larger than 2, we obtain the observation. □

The following three simple observations showcase how the previous lemma is applied.

Observation 6.1. *Let G be a graph as shown in Figure 6.2a. Then in any UDR of G it holds that $\measuredangle v_{i+1}v_iv_{i-1} < 4\pi/3$.*

The statement follows from two applications of Lemma 6.1 on the angles $\measuredangle v_{i-1}v_iv_i'$ and $\measuredangle v_i'v_iv_{i+1}$.

Observation 6.2. *Let G be a graph as shown in Figure 6.2b. Then in any UDR of G it holds that $\measuredangle v_{i+1}v_iv_{i-1} < \pi$.*

The statement follows from three applications of Lemma 6.1 on the angles $\measuredangle v_{i-1}v_iv_i'$, $\measuredangle v_i'v_iv_i''$ and $\measuredangle v_i''v_iv_{i+1}$.

Observation 6.3. *Let G be a graph as shown in Figure 6.2c. Then in any UDR of G it holds that $\measuredangle v_{i+1}v_iv_{i-1} < 2\pi/3$.*

The statement follows from three applications of Lemma 6.1 on the angles $\measuredangle v_{i-1}v_iv_i'$, $\measuredangle v_i'v_iv_i''$ and $\measuredangle v_i''v_iv_{i+1}$.

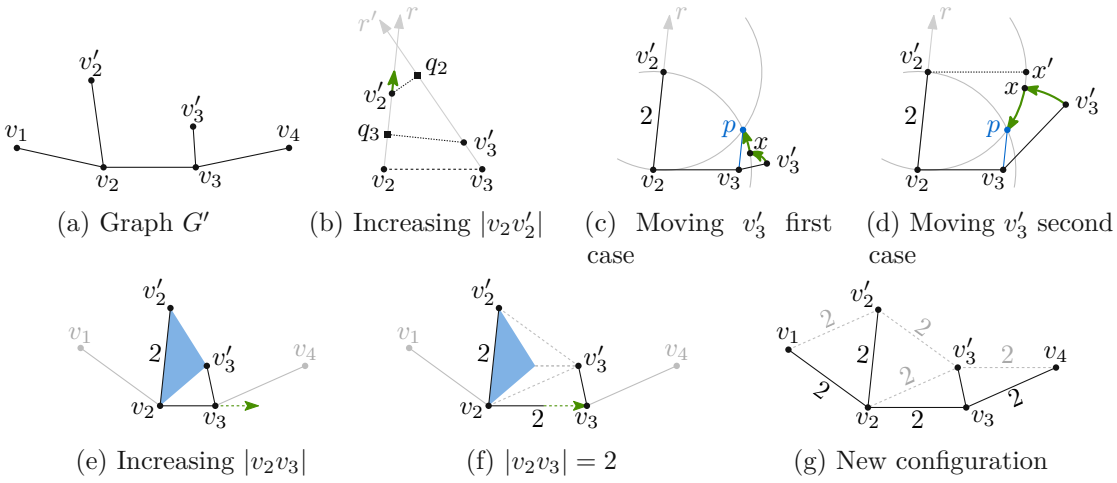
Now let G' be an induced subgraph of G, such that G' has six vertices $v_1, v_2, v_2', v_3, v_3', v_4$ with edges v_iv_{i+1} for $i \in \{1,2,3\}$ and v_iv_i' for $i \in \{2,3\}$. Let the clockwise rotational order of the neighbors around v_2 be fixed to v_1, v_2', v_3 and around v_3 to v_2, v_3', v_4, see Fig. 6.3a.

Lemma 6.2. *In a UDR of G with the above defined induced subgraph G' it holds that $\measuredangle v_1v_2v_2' + \measuredangle v_2'v_2v_3 + \measuredangle v_2v_3v_3' + \measuredangle v_3'v_3v_4 > \frac{5\pi}{3}$.*

Proof. Without loss of generality, we assume that v_2v_3 is horizontal. Since we are looking for a lower bound, due to Lemma 6.1 know that $\measuredangle v_1v_2v_2' > \pi/3$ and $\measuredangle v_3'v_3v_4 > \pi/3$. If $\measuredangle v_2'v_2v_3 + \measuredangle v_2v_3v_3' \geq \pi$, then the lemma holds.

Now assume $\measuredangle v_2'v_2v_3 + \measuredangle v_2v_3v_3' < \pi$ and observe that this means that the two rays $r = \overrightarrow{v_2v_2'}$ and $r' = \overrightarrow{v_3v_3'}$ intersect. We orthogonally project v_2' onto r' creating $q_2 \in r'$ and v_3' onto r creating $q_3 \in r$. Observe that either $|v_2v_2'| > |v_2q_3|$ or $|v_3v_3'| > |v_3q_2|$, since r and r' cross. We assume without loss of generality that $|v_2v_2'| > |v_2q_2|$ (see Figure 6.3b). Note that $\measuredangle v_2'q_2v_3' = \frac{\pi}{2}$. Therefore moving v_2' upwards on r will only increase its distance to v_3 and v_3' and therefore we can increase the distance $|v_2v_2'|$ to 2 without creating unwanted overlap of disks in the UDR of G'. Let c, c' be circles of radius two around v_2 and v_2' respectively and let p be their intersection point to the right of r. Now we can rotate v_3' around v_3 such that $\measuredangle v_2v_3v_3'$ decreases until we hit either c or c' at a point x. Since v_3' does not enter c or c', this still keeps v_3' at sufficient distance from v_2 and v_2'.

In the first case (Figure 6.3c), note that a continuous movement of v_3' along the boundary of c towards p strictly decreases $\measuredangle v_2v_3v_3'$ and we can place v_3' at p. In the second case (Figure 6.3d), note that $|xv_3| \leq 2$. Let x' be the intersection point of c' and a horizontal line through v_2'. The tangent of c' through x' is vertical. A direct consequence is that



(a) Graph G' (b) Increasing $|v_2v_2'|$ (c) Moving v_3' first case (d) Moving v_3' second case

(e) Increasing $|v_2v_3|$ (f) $|v_2v_3| = 2$ (g) New configuration

Figure 6.3: Transformations of the placement of the disks in the proof of Lemma 6.2. The movements described in the proof are indicated with green arrows. **(a)** v_2' can be moved to have a distance of 2 to v_2. Vertex v_3' can be rotated around v_3 and then either **(b)** v_2 or **(c)** v_2', continuously decreasing $\measuredangle v_2v_3v_3'$ until v_3' is placed at p. **(d)** The distance between v_2 and v_3 can be stretched to a distance of 2 **(e)**, while not changing any angle and only increasing distances. These transformation lead to a final configuration **(f)** with some fixed distances.

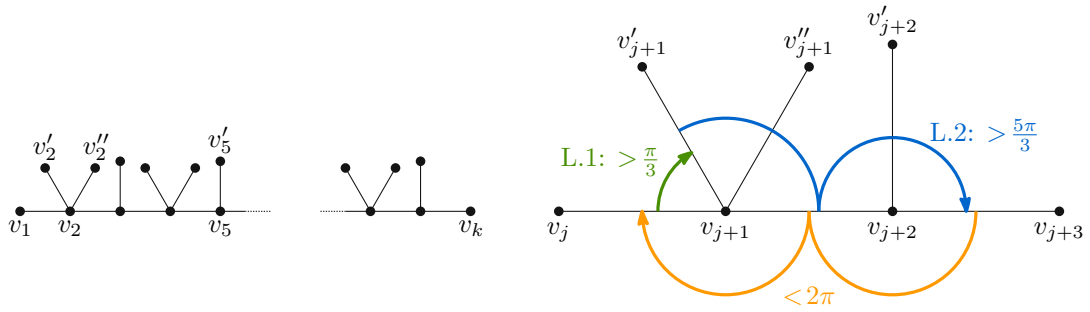
x lies exactly at x' or lower on the boundary of c'. Therefore we obtain again that a continuous movement of v_3' along the boundary of c' towards p strictly decreases $\measuredangle v_2v_3v_3'$.

Next we observe that we can now simply increase v_2v_3 (as shown in Figures 6.3e and 6.3f), since there exists a vertical line that intersects only v_2v_3. Finally we can apply the previous procedure of rotating v_3' again. The following distances are now by construction equal to 2: $|v_1, v_2|, |v_2, v_2'|, |v_2, v_3|, |v_3, v_4|, |v_2, v_3'|, |v_2', v_3'|$. Further in any UDR of G', where $|v_1v_2'| > 2$ we can rotate v_1 around v_2 until $|v_1v_2'| = 2$, a procedure which technically puts $d(v_1)$ and $d(v_2')$ into contact, but since this only decreases $\measuredangle v_1v_2v_2'$ it is sufficient to prove the lower bound for this unreachable edge case. Similarly in any UDR of G', where $|v_3'v_4| > 2$ we can transform it such that $|v_3'v_4| = 2$, yielding the final configuration shown in Figure 6.3g. It is now sufficient to analyze this case, where the only variable distance is $|v_3v_3'|$.

We can replace $\measuredangle v_2'v_2v_3$ by $\measuredangle v_2'v_2v_3' + \measuredangle v_3'v_2v_3$ and observe the following angle values:

$$\measuredangle v_1v_2v_2' = \measuredangle v_2'v_2v_3' = \pi/3$$

Using the Law of Cosines, we obtain the following angles: $\measuredangle v_2v_3v_3' = \measuredangle v_3'v_3v_4 = \cos^{-1}(|v_3v_3'|/4)$. Since the triangle $\triangle v_2v_3v_3'$ is isosceles, we finally observe $\measuredangle v_3'v_2v_3 = \pi - 2 \cdot \measuredangle v_2v_3v_3' = \pi - 2\cos^{-1}(|v_3v_3'|/4)$. Now we can plug everything into the sum and considering that all angles can only come infinitesimally close to their lower bounds, we



(a) Augmented path (b) Angle (sum) restrictions

Figure 6.4: The induced subgraphs (a) G' and (b) G'_j. Bounds on the angles are indicated by which lemma enforces them. Note that the orange bound of 2π is the result of Lemma 6.3.

obtain:

$$\measuredangle v_1v_2v'_2+\measuredangle v'_2v_2v_3+\measuredangle v_2v_3v'_3+\measuredangle v'_3v_3v_4 > 2\pi/3+\pi-2\cos^{-1}\left(|v_3v'_3|/4\right)+2\cos^{-1}\left(|v_3v'_3|/4\right) = 5\pi/3$$

□

We can see that the minimal angle turns out to be independent of the distance $|v_3v'_3|$. With the previous lemma established, we will now use it to show how angles and sums of angles in a larger subgraph can be analyzed.

The augmented path. Let G' be the the following graph. It consists of a chain of vertices v_i for $i \in \{1, \ldots, k\}$, for any odd k (In the remainder of this section we will assume that k is odd). The first and last vertex v_1 and v_k have no degree one neighbors. All other vertices have either one degree one neighbor v'_i if i is even or two such neighbors v'_i, v''_i if i is odd (see Figure 6.4a). Let G' be an induced subgraph of a larger graph G. We will call G' an *augmented path* of *length* k and we will refer to $v_1, \ldots, v_k$ as *the vertices of the augmented path*, i.e., this term excludes the leaf neighbors (note that v_1 and v_k are not considered leaf neighbors, despite being of degree 1). From here on out we will refer to the angle $\measuredangle v_{i+1}v_iv_{i-1}$ also as the *inner angle* at v_i. Conversely we refer to the counter angle also as the *outer angle*, which is the sum $\measuredangle v_{i-1}v_iv'_i + \measuredangle v'_iv_iv_{i+1}$ if i is odd and $\measuredangle v_{i-1}v_iv'_i + \measuredangle v'_iv_iv''_i + \measuredangle v''_iv_iv_{i+1}$ if i is even.

Lemma 6.3. *Let G be a graph with an induced subgraph G', such that, G' is an augmented path of length k. In a UDR of G it holds that $\sum_{i=1}^{k-2} \measuredangle v_{i+2}v_{i+1}v_i < (k-2)\pi$.*

Proof. Consider an induced subgraph G'_j of G' on vertices $v_j, v_{j+1}, v'_{j+1}, v''_{j+1}, v_{j+2}, v'_{j+2}, v_{j+3}$ for an odd $j < k$ (see Figure 6.4b). Due to Lemma 6.1, we know that $\measuredangle v_jv_{j+1}v'_{j+1} > \pi/3$. Due to Lemma 6.2, we know that

$$\measuredangle v'_{j+1}v_{j+1}v''_{j+1} + \measuredangle v''_{j+1}v_{j+1}v_{j+2} + \measuredangle v_{j+1}v_{j+2}v'_{j+2} + \measuredangle v'_{j+2}v_{j+2}v_{j+3} > 5\pi/3.$$

Now let $W = \measuredangle v_j v_{j+1} v'_{j+1} + \measuredangle v'_{j+1} v_{j+1} v''_{j+1} + \measuredangle v''_{j+1} v_{j+1} v_{j+2} + \measuredangle v_{j+1} v_{j+2} v'_{j+2} + \measuredangle v'_{j+2} v_{j+2} v_{j+3}$ and we can state that $W > 2\pi$. For the sum of angles on the inner side we have

$$\measuredangle v_{j+3} v_{j+2} v_{j+1} + \measuredangle v_{j+2} v_{j+1} v_j < 4\pi - W = 2\pi.$$

By adding up the sum of angles on the inner side of G'_j for all $j \in \{2, 4, \ldots, k-3\}$ (i.e. $(k-3)/2$ many subgraphs) we obtain exactly $\sum_{i=1}^{k-3} \measuredangle v_{i+2} v_{i+1} v_i$. The bound from the previous paragraph immediately yields

$$\sum_{i=1}^{k-3} \measuredangle v_{i+2} v_{i+1} v_i < \left(\frac{k-3}{2}\right) \cdot 2\pi = (k-3)\pi$$

We additionally have $\measuredangle v_{k-2} v_{k-1} v'_{k-1} + \measuredangle v'_{k-1} v_{k-1} v''_{k-1} + \measuredangle v''_{k-1} v_{k-1} v_k > \pi$. This also means that $\measuredangle v_k v_{k-1} v_{k-2} < \pi$. By adding up the sum of angles on the inner side of G'_j for all $j \in \{2, 4, \ldots, k-3\}$ plus $\measuredangle v_k v_{k-1} v_{k-2}$ we again obtain $\sum_{i=1}^{k-2} \measuredangle v_{i+2} v_{i+1} v_i$ and in total $\sum_{i=1}^{k-2} \measuredangle v_{i+2} v_{i+1} v_i < (k-2)\pi$. □

The previous lemma can intuitively be stated as follows. Assuming that the degree one neighbors are placed to the left of the chain in a UDR of G', the chain itself cannot bend to the left.

Clearly at every v_i, where i is even, the inner angle is smaller than π since there are two leaf neighbors on the outside. In other words assuming that $v_{i-1}v_i$ is horizontal, $v_i v_{i+1}$ points downward. Corollary 6.1 intuitively states that if $v_{i-1}v_i$ is horizontal and v_i is to the right of v_{i-1}, while we can bend upward again at v_{i+1}, $v_{i+1}v_{i+2}$ will still point at least slightly downwards and cannot return to a horizontal state or even point upwards.

We next state a result, which again uses angle bounds to restrict the actual location of disks in a UDR. We assume the same subgraph G' as for the previous Lemma 6.3.

Lemma 6.4. *If $\sum_{i=1}^{k-2} \measuredangle v_{i+2} v_{i+1} v_i \geq (k-2.5)\pi$, then in any UDR of G, v_j for $j \in \{3, 4, \ldots, k\}$ lie to the right of the ray through v_1 and v_2.*

Proof. We can assume that v_1, v_2 lie on a horizontal line ℓ. First observe that the lower bound of $(k-2.5)\pi$ is only π below the upper bound stated in Lemma 6.3. This means in particular that the UDR cannot place any disk with its center to the left of v_1. Now consider v_3. Due to Observation 6.2, we know that $\measuredangle v_3 v_2 v_1 < \pi$. Therefore v_3 is below ℓ. Now consider v_4. Due to Lemma 6.3, we know $\measuredangle v_3 v_2 v_1 + \measuredangle v_4 v_3 v_2 < 4\pi - 2\pi = 2\pi$. Therefore v_4 has to lie below v_3. From here we can repeat the argument for every v_{i+2} and v_{i+3} using the line through v_i and v_{i+1}. □

Note that the lower bounds in the previous lemma will later be enforced by placing a similar subgraph to G' opposite of G' and mirrored so that in a UDR the chains can only bend towards each other and additionally, the copy will be forced to be placed close

enough, such that, there is not sufficient space for one of the chains to "turn around" and be placed in the space between. The impossibility to turn around can also be formalized as the sum of the inside angles being within a value of π of its upper bound. Additionally we obtain a useful corollary of the observations used in the previous two proofs.

Corollary 6.1. *The sum of the inner angles of two consecutive vertices v_i, v_{i+1} is less than 2π and as a direct consequence, if $\sum_{i=1}^{k-2} \measuredangle v_{i+2}v_{i+1}v_i \geq (k-3)\pi$ the ray starting at v_{i-1} through v_i and the ray starting at v_{i+1} through v_{i+2} do not intersect.*

Now we establish a minimum length of a UDR of G' (defined exactly as in the previous two lemmata), when it is restricted to a certain bounding box.

Lemma 6.5. *Let $L = \sum_{i=1}^{k-1} |v_iv_{i+1}|$ and assume $\sum_{i=1}^{k-2} \measuredangle v_{i+2}v_{i+1}v_i \geq (k-2.5)\pi$. Then in any UDR og G', we have $2k-5 < L \leq 2k-2$.*

Proof. This proof uses the assumed lower bound $(k-2.5)\pi$ on the sum of the inner angles to establish a constant lower bound on the sum of distances between consecutive vertices. After arguing the upper bound of L, we will start to proof the lower bound of L by defining the sum of all outer angles first (which inversely has an upper bound). We will call the bound on the outer angles, the *angle bound* in order to avoid confusion with the lower and upper bound of L, we want to proof. Then we will state some transformations of G', which do not change the sum of distances, and only decrease the sum of outer angles, so the same angle bound applies. Next we split this sum into its component angles and for each of them identify a minimum value. Some of these are trivial, while one particular angle component needs to be expressed as a function of the sum of distances. Analysis of this function reveals that, in order to obtain the minimum overall angle value for this component, we can redistribute lengths, while maintaining the total value of the sum of outer angles. Finally we show that the sum over all minimum values is still lower bounded by a constant, which implies the lower bound of L we want to establish.

It is easy to see that $L \leq 2k-2$, since $v_1, \ldots, v_k$ is a sequence of $k-1$ edges and the centers of disks of consecutive vertices have a maximum distance of 2.

We can assume that v_1v_2 is horizontal with v_1 on the left. We use the following shorthands for angles (see also Figure 6.5):

- For all even i let $\alpha_i = \measuredangle v_i'v_iv_i''$.
- For all even i let $\beta_i = \measuredangle v_i''v_iv_{i+1} + \measuredangle v_iv_{i+1}v_{i+1}'$.
- For all odd i let $\beta_i = \measuredangle v_i'v_iv_{i+1} + \measuredangle v_iv_{i+1}v_{i+1}'$.
- For all even i let $\theta_i = \measuredangle v_{i-1}v_iv_i' + \measuredangle v_i'v_iv_i'' + \measuredangle v_i''v_iv_{i+1}$
- For all odd i let $\theta_i = \measuredangle v_{i-1}v_iv_i' + \measuredangle v_i'v_iv_{i+1}$

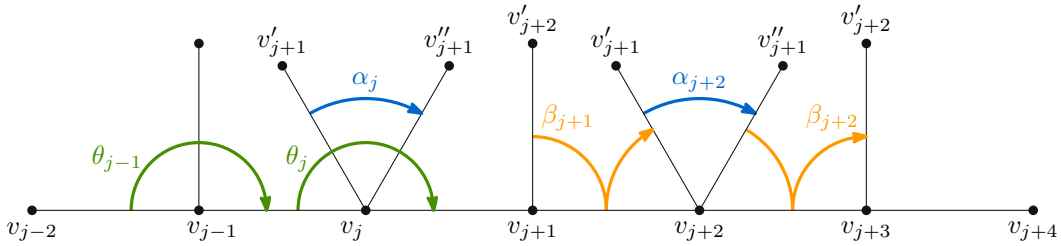


Figure 6.5: Shorthands used for angles in the proof of Lemma 6.5

Initially we observe that for all $i \in \{2, \ldots, k-1\}$ we have $\measuredangle v_{i+1}v_iv_{i-1} = 2\pi - \theta_i$ as well as (by Lemma 6.1) $\alpha_i > \pi/3$, $\measuredangle v_1v_2v'_2 > \pi/3$ and $\measuredangle v''_{k-1}v_{k-1}v_k > \pi/3$. Next we can express the sum of inner angles in terms of the outer angles

$$\sum_{i=1}^{k-2} \measuredangle v_{i+2}v_{i+1}v_i = \sum_{i=2}^{k-1}(2\pi - \theta_i).$$

Using the angle bound assumed by the lemma, we can obtain a bound on the sum of outer angles

$$\sum_{i=2}^{k-1}(2\pi - \theta_i) \geq (k - 2.5)\pi$$

$$(k - 1.5)\pi \geq \sum_{i=2}^{k-1} \theta_i.$$

The sum of all outer angles can be expressed with the α and β variables

$$\sum_{i=2}^{k-1} \theta_i = \sum_{j=2}^{k-2} \beta_j + \sum_{i=1}^{(k-3)/2} \alpha_{2i} + \measuredangle v_1v_2v'_2 + \measuredangle v''_{k-1}v_{k-1}v_k.$$

In particular this means that we can use the bounds from Lemma 6.1 to obtain

$$\sum_{i=2}^{k-1} \theta_i > \sum_{j=2}^{k-2} \beta_j + \frac{k-3}{2}\frac{\pi}{3} + \frac{2\pi}{3} = \sum_{j=2}^{k-2} \beta_j + \frac{(k+1)\pi}{6}.$$

Combining the above we arrive at

$$(k - 1.5)\pi > \sum_{j=2}^{k-2} \beta_j + \frac{(k+1)\pi}{6}$$

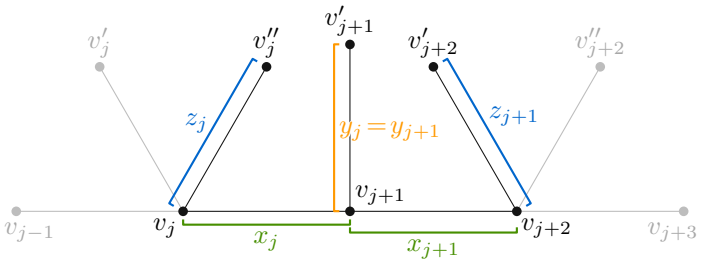


Figure 6.6: Shorthands used for distances in the proof of Lemma 6.5

$$\frac{(5k-8)\pi}{6} > \sum_{j=2}^{k-2} \beta_j.$$

Next we consider how we can compute the values β_j. We will call these variables collectively β *variables*. The reader might recall that we already computed the value of one β variable in the proof of Lemma 6.1, however, there we used the assumption that $|v_j v_{j+1}| = 2$. Since we here want to investigate the possibility of a shorter distance between v_j and v_{j+1}, we will instead treat it as a variable. The list of variables (depicted in Figure 6.6) we will be using is

- $x_j = |v_j v_{j+1}|$,
- $y_j = |v_{j+1} v'_{j+1}|$ and $z_j = |v_j v''_j|$ if j is even and
- $y_j = |v_j v'_j|$ and $z_j = |v_{j+1} v'_{j+1}|$ if j is odd.

The value of β_j is dependent on three variables x_j, y_j and z_j. Since $y_i = y_{i+1}$, consecutive β variables are not independent, however by treating them as independent we can obtain a lower bound on their values.

Now we will again show that some assumptions can be made about these variables. First assume a fixed value of β_j in a valid UDR of G. Without loss of generality, assume that j is even. We can assume that $z \geq y$, since if $y_j > z_j$, we can simply mirror v''_j and v'_{j+1} on the bisector of $v_j v_{j+1}$. Then we can assume that $z_j = 2$ by the same argument as at the start of the proof of Lemma 6.2 (as illustrated in Figure 6.3b). These transformations have either not changed $|v''_j v'_{j+1}|$ and $|v_j v'_{j+1}|$ or have increased them. Therefore we know that $|v''_j v'_{j+1}| > 2$ and we can also apply the second transformation, which rotates v'_{j+1} around v_{j+1} to decrease $\measuredangle v_j v_{j+1} v'_{j+1}$ until $|v''_j v'_{j+1}| = |v_j v'_{j+1}| = 2$. In total the new value β'_j after these transformations is smaller than the original β_j.

Note that, while these transformations will not create incorrect overlap between the disks of v_j, v''_j, v_{j+1} and v'_{j+1} in the UDR of G, they might create unwanted overlap with other disks. In other words, the value of β'_j we have obtained now, might not be achievable in a valid UDR. However since we only use this to obtain a lower bound for β_j and therefore the fact that this angle might not be achievable is not a problem.

We will now use the remaining variables x_j, y_j and express β'_j as a function of these two variables.

Since after the above transformations the triangle $\triangle v_i v''_i v'_{i+1}$ is equilateral we state

$$\beta'_j = \measuredangle v''_j v_j v'_{j+1} + \measuredangle v'_{j+1} v_j v_{j+1} + \measuredangle v_j v_{j+1} v'_{j+1} = \frac{\pi}{3} + \measuredangle v'_{j+1} v_j v_{j+1} + \measuredangle v_j v_{j+1} v'_{j+1}.$$

We use $\gamma_j = \measuredangle v'_{j+1} v_j v_{j+1} + \measuredangle v_j v_{j+1} v'_{j+1}$ to obtain

$$\beta'_j - \frac{\pi}{3} = \gamma_j$$

and we can express γ_j as a function $f(x, y)$

$$\gamma_j = \measuredangle v'_{j+1} v_j v_{j+1} + \measuredangle v_j v_{j+1} v'_{j+1} = \pi - \measuredangle v_{j+1} v'_{j+1} v_j = \pi - \cos^{-1}\left(\frac{4 + y_j^2 - x_j^2}{4y_j}\right) = f(x_j, y_j).$$

Every fixed value of $x_j = C$ yields a uni-variate function $f(C, y_j)$, which can be interpreted as a function describing the minimal angle value possible for $x_j = C$ and a variable y_j.

Now consider the partial derivative f_y in y of $f(x, y)$

$$f_y(x, y) = \frac{x^2 + y^2 - 4}{y^2 \sqrt{-\frac{x^4 - 2x^2(y^2+4) + (y^2-4)^2}{y^2}}},$$

which describes the change in value of $f(x, y)$, when y changes for a fixed x. By setting $f_y(x, y) = 0$, we find the extremal points of $f_y(x, y)$ at $y = \pm\sqrt{4 - x^2}$ (of which obviously only the positive value is interesting).

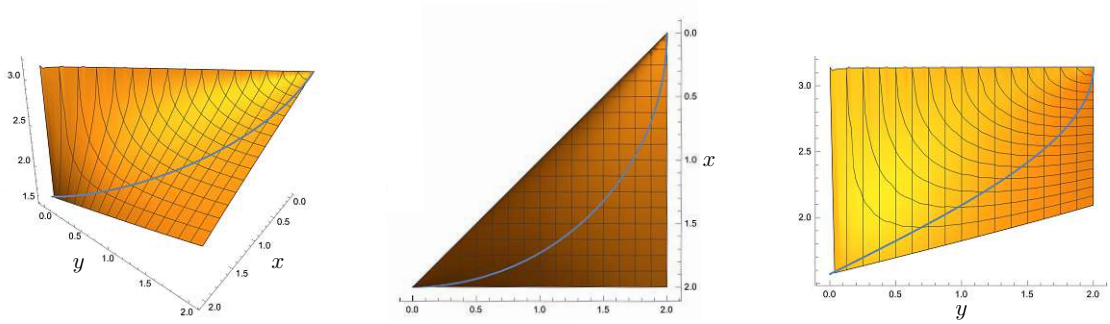
Since $\forall y' \neq \sqrt{4 - x^2} : f(x, y) < f(x, y')$, we have a local minimum at $y = \sqrt{4 - x^2}$ (note that this is a local minimum of the uni-variate function $f(C, y)$, but not necessarily of the bi-variate function $f(x, y)$). We now substitute $y = \sqrt{4 - x^2}$ to obtain

$$g(x) = f(x, \sqrt{4 - x^2}) = \pi - \cos^{-1}\left(\frac{\sqrt{4 - x^2}}{2}\right)$$

The functions $f(x, y)$ and $g(x) = f(x, \sqrt{4 - x^2})$ are illustrated and put into relation in Figure 6.7.

For a given UDR of G, with fixed values for all x variables, we can state that

$$\sum_{j=2}^{k-2} \gamma_j \geq \sum_{j=2}^{k-2} g(x_j)$$



(a) Perspective view (b) Top view (c) Front view

Figure 6.7: A (a) perspective, (b) top and (c) front view of the two functions $f(x, y)$ (the orange surface) and $g(x)$ (the blue line traced along the surface). At every point on the blue line, it holds that $y = \sqrt{4 - x^2}$. At every fixed x value (which can be though of as horizontal cuts in (b)), the point on the blue line indicates the minimal value.

We now investigate the first derivative $g_x(x)$ and second derivative $g_{xx}(x)$ of this function[3]

$$g_x(x) = -\frac{1}{\sqrt{4 - x^2}}$$
$$g_{xx}(x) = -\frac{x}{\sqrt{4 - x^2}^3}$$

It is easy to see that both derivatives are negative within the entire domain $0 < x \leq 2$. Since $g_x(x)$ is entirely negative within the domain, we know that any increase (decrease) of x will decrease (increase) the minimum value of $g(x)$. Since $g_{xx}(x)$ is entirely negative within the domain, we know that for two values x, x', such that, $x < x'$ we have $g(x) - g(x + \delta) < g(x') - g(x' + \delta)$ for any $\delta \in \mathbb{R}, 0 < \delta < 2$.In other words, the effect of any increase (decrease) δ of x on the value of $g(x)$ increases if x increases.

We use this to show that we can redistribute lengths between the different x variables, to lower the minimal angle overall, while keeping the sum over all x variables equal. Assume we are given a UDR of G, where there are at least two indices i, j, such that, $x_i < 2$, $x_j < 2$ and $x_i < x_j$. Further let $\delta = \min(2 - x_j, x_i)$, let $\xi_k = x_k$ for all $k \neq i, j$, let $\xi_i = x_i - \delta$ and let $\xi_j = x_j + \delta$. Clearly $\sum_{i=2}^{k-2} x_i = \sum_{i=2}^{k-2} \xi_i$ and for all $2 \leq i \leq k - 2 : 0 \leq \xi_i \leq 2$. Then the following holds

$$\sum_{j=2}^{k-2} g(x_j) > \sum_{j=2}^{k-2} g(\xi_j).$$

Through repeated application of this redistribution, we can arrive at a set of x variables, where all variables are either 2 or 0 and there is exactly one variable, which has a (possibly)

[3]Both derivatives have been simplified using the assumptions $x \in \mathbb{R}$ and $0 < x \leq 2$.

different value in the range $[0, 2]$ (if there would be at least two such variables, we could continue redistributing until there is only one of these variables). Let Ξ_i $i = 1, \ldots, k-1$ be these variables – we refer to them again collectively as Ξ variables – and let Ξ_p be the variable, which is possibly not equal to 0 or 2. Now we will combine the bounds stated so far:

$$\frac{k\pi}{2} + \frac{\pi}{3} = (k-1)\frac{5\pi}{6} - \sum_{j=2}^{k-2}\frac{\pi}{3} > \sum_{j=2}^{k-2}\beta_j - \sum_{j=2}^{k-2}\frac{\pi}{3} > \sum_{j=2}^{k-2}\beta'_j - \sum_{j=2}^{k-2}\frac{\pi}{3} = \sum_{j=2}^{k-2}\gamma_j > \sum_{j=2}^{k-2} g(\Xi_j).$$

Now let a be the number of Ξ variables equal to 0. Since $g(x)$ is minimal for $x = 2$ within the range $[0, 2]$, we underestimate $g(\Xi_p)$ with $g(2)$ and obtain

$$\frac{k\pi}{2} + \frac{\pi}{3} > \sum_{j=2}^{k-2} g(\Xi_j) = (k-2-a)\cdot g(2) + g(\Xi_p) + a \cdot g(0) \geq (k-1-a)\cdot g(2) + a \cdot g(0)$$

$$\frac{k\pi}{2} + \frac{\pi}{3} > \frac{(k-1-a)\pi}{2} + a\pi$$

$$\frac{1}{3} > a$$

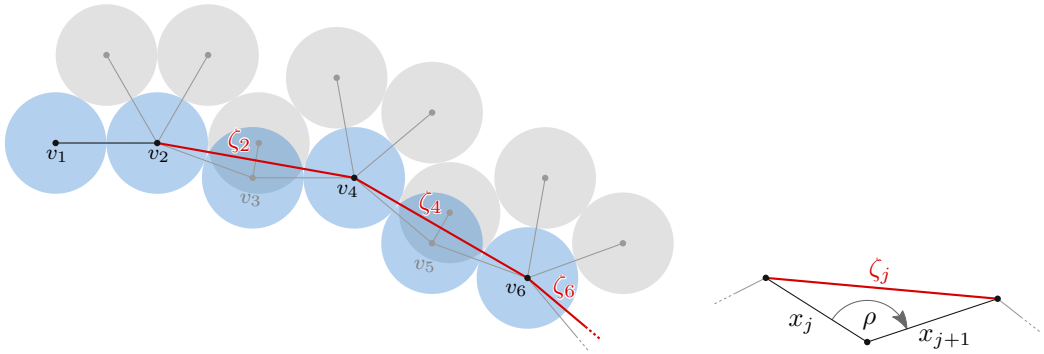
Recall that there are $k-2-a$ variables x_i, which are exactly of length 2, and we can underestimate Ξ_p, by simply setting it to 0. Then the lemma follows from this final inequality

$$L = \sum_{i=1}^{k-1} x_i > \sum_{i=2}^{k-2} x_i = \sum_{i=2}^{k-2} \Xi_i = 2\,(k-2-a) + \Xi_p > 2\left(k-2-\frac{1}{3}\right) = 2k - \frac{14}{3} > 2k-5$$

□

With the previous lemma we have established that the sum of distances is only at most a constant value smaller than its maximum if the realization of G' cannot fold in on itself or spiral (which is the intuitive interpretation of the lower bound on the sum of inner angles). This shows that we cannot use overlap between consecutive disks to reduce the total width of a UDR of G' beyond a constant offset of the maximal length if the angle bound is enforced. The inner angles are not actually equal to π and in fact the path through all v_i's has to follow a zig-zag pattern in order to properly represent G' without wrong overlaps.

We now define an auxiliary path starting at v_1, ending at v_k and containing all even indexed vertices in-between. We call this path Z and we will refer to the distance $|v_i v_{i+2}|$



(a) Definition (b) Detail view

Figure 6.8: The definition (a) of the ζ variables between vertices of distance two and the x variables and the angle ρ between them and (b) which are relevant for the definition of a single ζ variable.

between two consecutive inner vertices v_i, v_{i+2} of this path as ζ_i (see Figure 6.8a). Since the inner angle at every vertex that is part of Z is smaller than π (which can easily be shown via three applications of Lemma 6.1), Z only bends to the right and forms a convex chain, i.e., for every consecutive triple of vertices v_i, v_{i+1}, v_{i+2}, the vertex v_{i+2} lies locally to the right of the ray starting at v_i through v_{i+1}.

The next lemma states that similar to L the length of Z is also at most a constant value less than $2k$.

Lemma 6.6. *If* $\sum_{i=1}^{k-2} \measuredangle v_{i+2}v_{i+1}v_i \geq (k-2.5)\pi$ *then* $\sum_{i=2}^{(k-3)/2} \zeta_{2i} > 2k - 14$.

Proof. A single ζ_i variable is determined by the variables x_i, x_{i+1} and the angle $\rho = \measuredangle x_i x_{i+1} x_{i+2}$ as shown in Figure 6.8b. Using the law of cosines, we arrive at the following value (which we again interpret as a function h, here of three variables).

$$\zeta_i = \sqrt{x_i^2 + x_{i+1}^2 - 2x_i x_{i+1}\cos\rho} = h(x_i, x_{i+1}, \rho)$$

By Lemma 6.1, we know that $2\pi/3 < \rho$. Now we will use a similar redistribution argument as in the proof of Lemma 6.5. We start again by stating the first and second derivatives $h_\rho(x,y,\rho)$ and $h_{\rho\rho}(x,y,\rho)$ of $h(x,y,\rho)$ in ρ.

$$h_\rho(x,y,\rho) = \frac{xy\sin(\rho)}{2\sqrt{x^2+y^2-xy\cos(\rho)}}$$

$$h_{\rho\rho}(x,y,\rho) = \frac{xy\cos(\rho)}{2\sqrt{x^2+y^2-xy\cos(\rho)}} + \frac{x^2y^2\sin^2(\rho)}{4\sqrt{(x^2+y^2-xy\cos(\rho))^3}}$$

Neither h_ρ nor $h_{\rho\rho}$ are equal to 0 anywhere within the bounds $0 < x_1, x_2 \leq 2$ and $2\pi/3 < \rho < \pi$. Moreover, h_ρ is entirely positive, meaning that, for any fixed x and y any

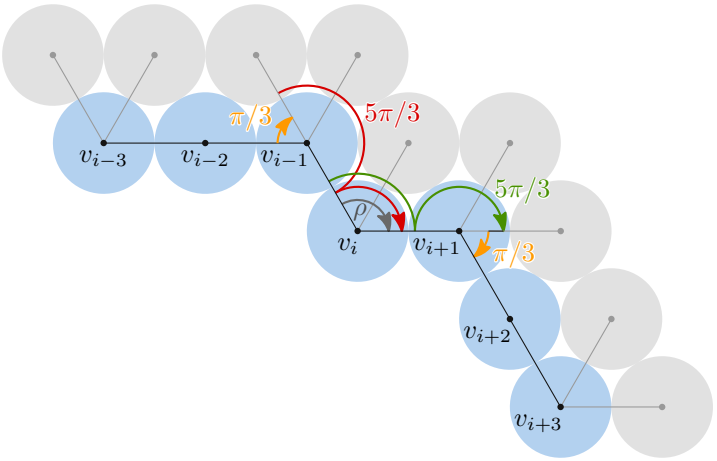


Figure 6.9: Illustration of how an angle $\rho < \pi$ causes bend towards the inside, which is then propagated. The individual parts of the sum S are highlighted in different colors and labeled with their individual lower bounds. Note how ρ is double counted in the red and green sum.

increase of ρ will increase $h(x, y, \rho)$. Conversely $h_{\rho\rho}$ is entirely negative and therefore we know that for two values ρ, ρ', such that, $\rho < \rho'$ we have $h(x, y, \rho) - h(x, y, \rho + \delta) > h(x, y, \rho') - h(x, y, \rho' + \delta)$ for any $\delta \in \mathbb{R}, 0 < \delta < \pi/3$.. In other words, the effect of any increase (decrease) δ of ρ on the value of $h(x, y, \rho)$ increases if ρ decreases. Recall that this is an identical argument to the one in the proof of Lemma 6.5.

We can now again repeat this distribution – taking from smaller angles and redistributing to larger angles, while keeping the sum over all outer angles equal and at the same time decreasing the sum over all ζ variables – until all angles are either π or $2\pi/3$ and exactly one angle has possibly different value within the interval $[2\pi/3, \pi]$.

Let ρ_i be one of the angles equal to $2\pi/3$. By Lemma 6.2, we know that the following two sums are both larger than $5\pi/3$ (see Figure 6.9 for an illustration).

$$A = \measuredangle v''_{i-1} v_{i-1} v'_{i-1} + \measuredangle v'_{i-1} v_{i-1} v_i + \rho_i$$
$$B = \rho_i + \measuredangle v_i v_{i+1} v'_{i+1} + \measuredangle v'_{i+1} v_{i+1} v''_{i+1}$$

Therefore we can express the sum S of outer angles at the vertices v_{i-1}, v_i and v_{i+1} as

$$S = \measuredangle v_{i-2} v_{i-1} v'_{i-1} + A + B - \rho + \measuredangle v''_{i+1} v_{i+1} v_{i+2} > \frac{(2 + 10 - 2)\pi}{3} = 3\pi + \frac{\pi}{3}$$

We conclude that every angle $\rho_j = 2\pi/3$ adds an additional $\pi/3$ to the minimum value of the sum of outer angles along the chain. Since the maximal value of the sum of outer angles is $(k - 2.5)\pi$ (and therefore only $\pi/2$ above the general lower bound of $(2 - k)\pi$

established in Lemma 6.3), we know that at most one such angle can exist. Let ρ_j be this angle and let ρ_k be the angle with a value in the interval $]2\pi/3, \pi]$.

Now consider that $h(x, y, \pi) = x + y$ and therefore for any other angle $\rho_m = \pi$, we have $h(x_m, x_{m+1}, \rho_m) = x_m + x_{m+1}$. Further consider that $h(x, y, 2\pi/3) = \sqrt{3}$. We use this value for ζ_j and ζ_k (whose value can be overestimated as $(x_j + x_{j+1})$ and $(x_k + x_{k+1})$, respectively) to obtain an underestimation for the sum over all ζ variables. Then

$$\sum_{i=1}^{(k-3)/2} \zeta_{2i} > \sum_{i=1}^{(k-3)/2} \zeta_{2i} - \zeta_j - \zeta_k + 2\sqrt{3} > \sum_{i=1}^{(k-1)/2} (x_{2i} + x_{2i+1}) - (x_j + x_{j+1}) - (x_k + x_{k+1}) + 2\sqrt{3}$$

$$\sum_{i=1}^{(k-1)/2} (x_{2i} + x_{2i+1}) - (x_j + x_{j+1}) - (x_k + x_{k+1}) + 2\sqrt{3} = \sum_{i=2}^{k-2} x_i - 8 + 2\sqrt{3} > \sum_{i=2}^{k-2} x_i - 5$$

Using the fact that L is the sum over the x variables and the fact that x_1 and x_{k-1} are both at most 2 we obtain

$$\sum_{i=2}^{k-2} x_i - 5 = L - x_1 - x_{k-1} - 5 > L - 9$$

Now we use the bound proven in Lemma 6.5 which yields the lemma statement.

$$\sum_{i=1}^{(k-3)/2} \zeta_{2i} > L - 9 > 2k - 5 - 9 = 2k - 14$$

□

Observe that the lower bound on the sum of inner angles in the previous lemma can be understood as the chain through all v_i's being x-monotone (recall the assumption of v_1v_2 being horizontal). Note that this implies that Z is also x-monotone.

Now that we know that we can identify a convex x-monotone chain of length $2k - \mathcal{O}(1)$ within every UDR of G' we can finally provide a lower bound on the width of the UDR of G'. The following lemma assumes that the UDR is heavily constrained in its height (in particular it is bounded by a constant). It also still assumes that v_1v_2 is horizontal.

Lemma 6.7. *In every UDR of G' if $\sum_{i=1}^{k-2} \measuredangle v_{i+2}v_{i+1}v_i \geq (k - 2.5)\pi$ and the axis aligned rectangular bounding box of all disks of the UDR is upper bounded in height by a constant C, then the width of the bounding box is lower bounded by $2k - 14 - C$.*

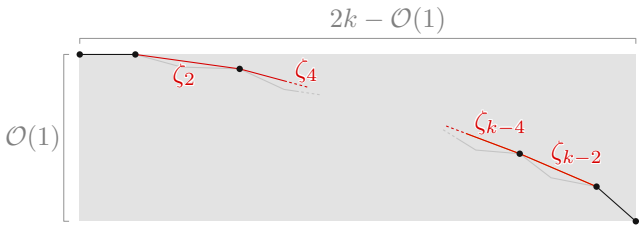


Figure 6.10: The x-monotone convex chain Z within its bounding box of height C. The longest possible chain within the bounding box is indicated with the red line.

Proof. This lemma is straight forward. We identify the path Z. Due to the assumed lower bound on the sum of inner angles, we know that this path is x-monotone and convex. The longest possible x-monotone convex path that can be inscribed in a bounding box of width W and height H is of length $W + H$ (the path follows the box horizontally for the entire width performs a 90 degree turn and follow is vertically for the entire height). By Lemma 6.6 we know the length of Z is at least $2k - 14$. Therefore the width of the bounding box is at least $2k - 14 - C$. The proof is illustrated in Figure 6.10. □

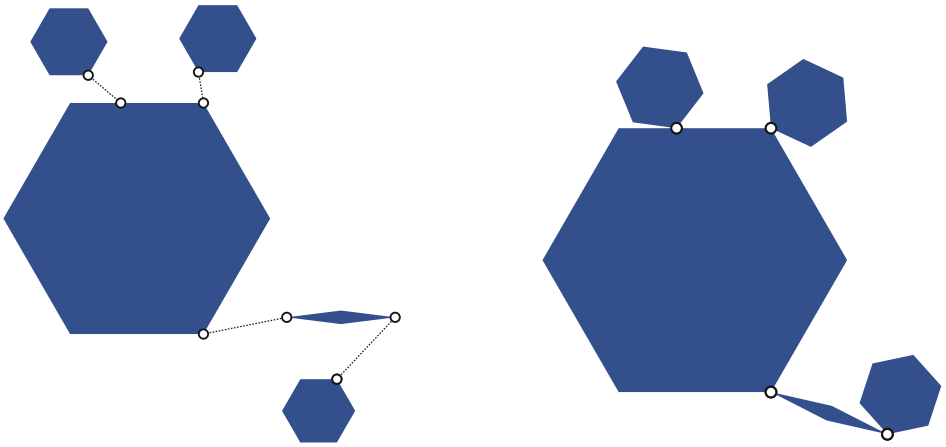
In this section we have provided a step-by-step reasoning that the graph structure shown in Figure 6.4a, i.e., an augmented path of length k, forces various properties in every UDR, specifically if all leafs are forced to one side, the sum of angles on the other side has an upper bound and if this sum is also lower bounded to be within a value of π of its upper bound, a bounding box of the UDR with a constant height $\mathcal{O}(1)$ has a width of $2k - \mathcal{O}(1)$.

In the following chapter we will present graph constructions that force the necessary features for these properties to hold and we will use multiple copies of this graph to build gadgets whose UDRs have very restricted geometry.

6.4 NP-Hardness Results

In this section, we prove that recognizing unit disk graphs, which is known to be NP-hard for planar graphs [27], remains NP-hard for non-embedded outerplanar graphs and for embedded trees.

Our proofs apply the generic machinery of Bowen et al. [25] to decide realizability of polygonal linkages, which requires to construct gadgets that can model hexagons and rhombi in a stable way. We give a high-level overview of their approach in Section 6.4.1. Then we describe our constructions of the required stable structures with embedded tree (Section 6.4.2) and outerplanar gadgets (Section 6.4.3).



(a) A set $\mathcal{P}$ of polygons with hinges (b) A realization of (a)

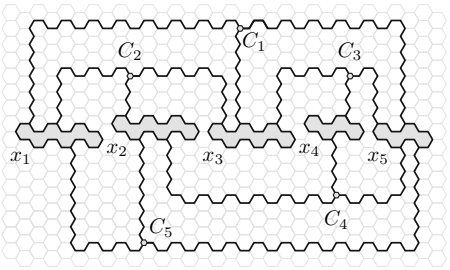
Figure 6.11: An instance of PLR **(a),** where points belonging to the same hinge are indicated with a dotted line connection. A placement of (rotated and/or translated) copies of the polygons in $\mathcal{P}$ in the plane is a realization **(b)** if the points of the same hinge are identified.

6.4.1 Summary of Foundational Hardness Results

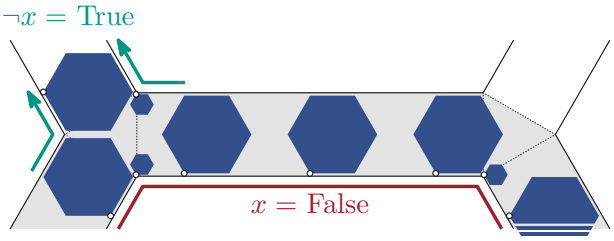
Bowen et al. [25] proved that recognizing unit disk contact graphs is NP-hard for embedded trees, via a reduction from planar 3-SAT, which uses an auxiliary construction formulated as a realization of a polygonal linkage. Polygonal linkages are explained in Section 6.4.1. The details of this auxiliary structure are explained in Section 6.4.1. Then a tree, whose UDC is an approximation of the auxilliary structure and which mimics the shape and mechanics. This construction is summarized in Section 6.4.1.

Polygonal Linkages

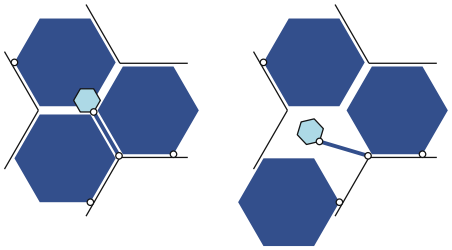
Bowen et al. [25] considered multiple problems in their work, one of which is the *polygonal linkage realizability* (PLR) problem. A polygonal linkage is a set $\mathcal{P}$ of convex polygons and a set H of hinges. One hinge is a set of two or more points on the boundaries of distinct polygons. A polygonal linkage is realizable, if every $p \in \mathcal{P}$ can be placed in the plane such that all polygons $p \in \mathcal{P}$ are interior disjoint and for every hinge $h \in H$, all points of the hinge coincide. The problem variant *PLR with fixed orientation* additionally requires that a predefined cyclic order of adjacent polygons around every hinge is kept. The set H implicitly defines a graph $G_H = (\mathcal{P}, E)$ (called a *hinge graph*), where $(P_1, P_2) \in E$ if there is a hinge $h \in H$ containing two points p_1, p_2 on P_1, P_2, respectively. Bowen et al. [25] proved that PLR with fixed orientation is NP-hard even if the hinge graph is a tree. In our case and in the case of this reduction, hinges are only of size two, i.e., a realization will identify exactly two points on distinct polygons per hinge. This means that cyclic order around hinges is always kept by default. A polygonal linkage and its realization are shown in Figure 6.11.



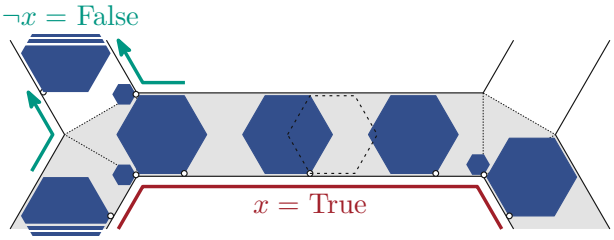
(a) Auxiliary hexagonal grid structure with embedded incidence graph



(b) False variable x. The connected wire on the top left is in a true state and therefore corresponds to the negated literal $\neg x$.



(c) Unsatisfied state (left) and one possible satisfied state (right) of the clause gadget



(d) True variable x. The wire is in a false state. The dotted hexagon indicates the overlap, if hexagons would have inconsistent states.

Figure 6.12: Auxiliary structure details used by Bowen et al. [25]. All Figures are recreations/adaptions from their paper. The incidence graph is embedded on a hexagonal grid **(a)**. The edges are short corridors in which the blue hexagons are fitted, hinged at white vertices. Hexagons in the variable cycle (red line, grey backdrop) have two states **(b)** and **(d)**. The clause gadget **(c)** requires one hexagon, which does not enter the junction.

Auxiliary Structure

The auxilliary structure mimics a hexagonal grid. This grid-like structure is obtained by using a hexagonal tiling of the plane and then shrinking every hexagon by a small amount to obtain narrow channels of fixed height between two hexagons. These *large* hexagons are initially considered to have a fixed position and will later be hemmed in by a frame construction, which limits their actual position to be close to this fixed position. At the corners of the hexagons, three such channels meet to form a junction. The union of all channels and junctions forms the grid-like structure. The hardness of PLR with fixed orientation was proven via a reduction from planar 3-SAT. An instance of planar 3-SAT consists of a Boolean formula ϕ over a set V of variables given as a set of disjunctive clauses C and an incidence graph $G_\phi = (V \cup C, E)$, where $v \in V$ appears as a literal in $c \in C$ if and only if $(v, c) \in E$. A representation of the incidence graph G_ϕ is fitted into the grid of the auxiliary structure, see Figure 6.12a.

A variable v of ϕ is represented in this grid as an alternating cycle of channels and

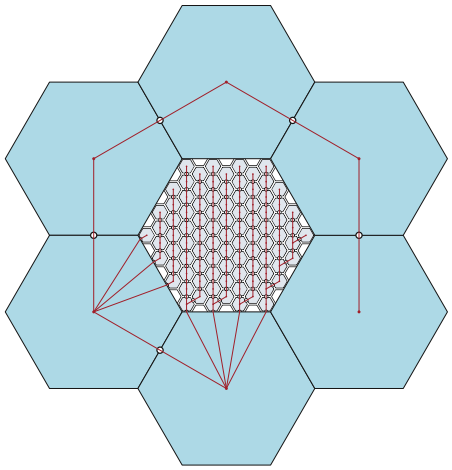
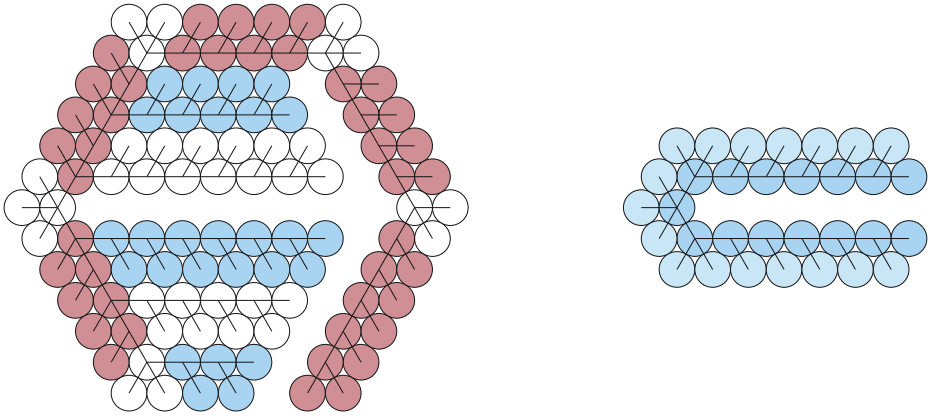


Figure 6.13: The composition of the rigid structure as a realization of a polygonal linkage. Hinges are indicated by points. The figure is a recreation of a similar figure in [25], augmented with lines, which emphasize that the hinge graph of the auxiliary structure is indeed a tree

junctions, indicated with a grey fill in Figure 6.12a. In such a cycle the channels can be filled with smaller hexagons, connected to the large hexagon on the side of the channel – which is on the "inside" of the variable cycle – via a junction. In a channel, one corner of each of the small hexagons is connected to the large hexagon via a hinge such that the small hexagon can be "flipped" around this junction. Due to the chosen size, the hexagon can be realized in one of two states, i.e., flipped (almost) completely to one or the other side, see Figures 6.12b and 6.12d. The distance of the hinges of neighboring small hexagons is chosen in such a way that the state of one hexagon determines the state of all hexagons in the channel, see Figure 6.12d. At each junction, we add an even smaller hexagon with a hinge to the corner of that large hexagon, which is adjacent to the channels on either side of the junction in the variable cycle. This propagates the state of the hexagons in one channel through the junction into the other channel and so throughout the entire cycle. See Figures 6.12b and 6.12d for a detailed explanation.

Wire gadgets are alternating paths of channels and junctions, which use the same mechanism to propagate the state of the hexagons in the channels and therefore admit two states overall.

Wire gadgets can be connected to a variable cycle at every junction using the third unoccupied channel and adding a second very small hexagon in the junction. A wire gadget is considered to transmit the value *true*, if and only if, part of the small hexagon at the first channel of the wire gadget enters into the junction. By placing the small hexagon on one or the other corner (cf. Figures 6.12b and 6.12d) the truth value which



(a) Graph T_k and its UDC (b) Graph T'_k and its UDC

Figure 6.14: The 2-stable approximations of a long thin rhombus **(a)** and a regular hexagon **(b)**. All Figures are recreations from Bowen et al. [25].

is transmitted can be "inverted", in order to guarantee the correct transmission of truth values for both positive and negative literals.

For every clause in ϕ, three such wire gadgets are connected to the variable cycles of the occurring variables and the wires are routed to meet in a junction. This junction contains a small hexagon connected to the corner of a large hexagon via a long and very thin rhombus (in place of a line segment) such that an overlap-free realization is only possible, if at least one connected wire gadget has no hexagon entering the junction and is therefore in a true state, see Figure 6.12c.

In order to guarantee a sufficiently rigid placement of these hexagons, the entire construction is surrounded by a set of six huge hexagons, and the hexagons acting as the faces of the hexagonal grid are column-wise connected (cf. Figure 6.13). This confines the position of the large hexagons to a position, very close to their fixed position [25].

Note that the hinge graph of the auxiliary structure is indeed a tree. In particular note that we can replace the hexagons with outerplanar graphs and replace the hinges between them with paths of length one to three and the entire construction remains an outerplanar graph. The same way we can replace all hexagons with trees and replace the hinges with vertex paths of length one to three and the entire construction remains a tree.

For a more detailed description, a full construction and the proof of the semi-rigid placement we refer to the original paper of Bowen et al. [25].

Approximating the Auxilliary Structure

In order to prove the NP-hardness for recognition of unit disk contact graphs, Bowen et al. [25] created λ-stable approximations of the basic building blocks of the auxiliary structure (hexagons of varying sizes and long thin rhombi).

Definition 6.1. *A graph G is a λ-stable approximation of a region P in the plane if, for every UDR $\mathcal{D}$ of G, there exists a congruence transformation $f : \mathbb{R}^2 \to \mathbb{R}^2$ such that the union U of all unit disks in the UDR has a Hausdorff distance $d_H(f(P), U) \leq \lambda$.*

Bowen et al. described the construction procedures for two graphs T_k, T'_k, which are 2-stable approximations of a long thin rhombus and a regular hexagon. These two graphs are shown in Figure 6.14. For the details of these constructions, we again refer to Bowen et al. [25]. The construction of our building blocks using outerplanar graphs in Section 6.4.3 and embedded trees Section 6.4.2 is based on these graphs.

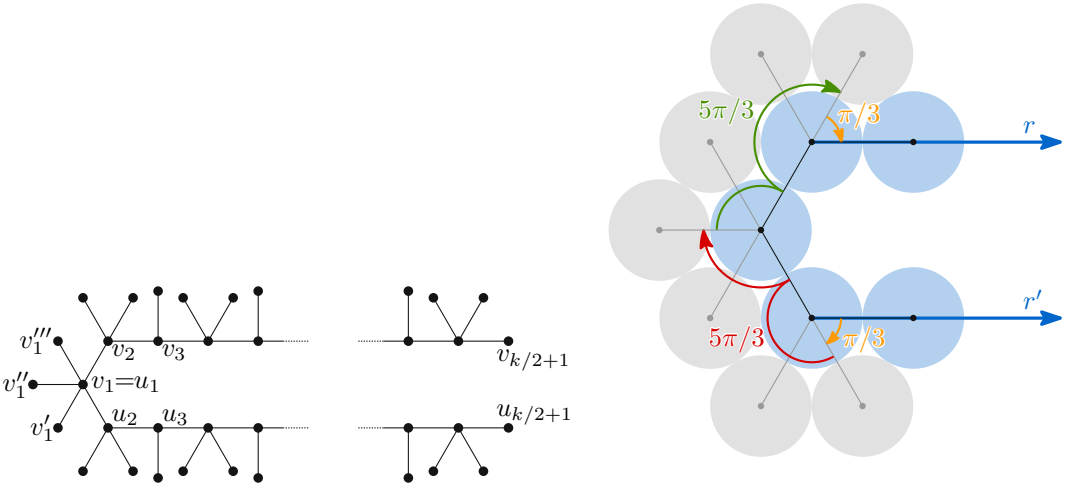
It remains to discuss how the hinges are modeled. Two λ-stable approximations G, G' of two polygons P, P' are connected with a path of vertices of constant length, if there exists a hinge $h \in H$, with one point on the boundary of P and the other on the boundary of P'. The exact length of the path is dependent on the location of the hinge. If the hinge is not placed on a corner of either polygon, they are simply connected via a single cut vertex, and with a path of length three otherwise, in order to facilitate more movement, which mimics the possibility of polygons to rotate around hinges. The union of the UDC of two thus connected graphs G, G', remains a constant factor approximation of a congruent copy of the realization of $P \cup P'$.

6.4.2 Recognizing UDRs is NP-hard for Embedded Trees

We prove that recognition of unit disk graphs is hard for embedded trees by providing embedded trees T_k^R and T_k^H, which are X- and Y-stable approximations of a rhombus of width k and a hexagon of side length k, respectively. Recall that embedded trees have a fixed rotational order of neighbors around every vertex and any UDR of an embedded tree has to induce the same rotational order. Then, the NP-hardness follows immediately from the construction of Bowen et al. [25] sketched above.

We start with the construction of T_k^R (shown in Figure 6.15a). It consists of two augmented paths $v_1, \ldots, v_{k/2+1}$ and $u_1, \ldots, u_{k/2+1}$ of length $k/2 + 1$ placed opposite of each other, such that, the leaf neighbors point away from the other chain. Note that the largest possible distance between the center of the disks of v_1 and $v_{k/2+1}$ is upper bounded by k in any UDR of T_k^H. The two chains are connected by identifying v_1 and u_1. Then we additionally add three leaf neighbors v'_1, v''_1 and v'''_1 to $v_1 = u_1$ (in that clockwise order). We now show that the two augmented paths "point towards each other" in every UDR of T_k^R.

Lemma 6.8. *In every UDR of T_k^R the rays r (starting at v_2 through v_3) and r' (starting at u_2 through u_3) intersect.*



(a) Graph T_k^R (b) Angle restrictions at the connection

Figure 6.15: Construction of the graph T_k^R (a) out of two augmented paths of length k. At the point of connection the angle restrictions (b) force the two rays r and r' to intersect. Note that the Figure displays an unreachable boundary case, where the sum of inner angles is exactly 2π.

Proof. Using both Lemma 6.1 and Lemma 6.2 as shown in Figure 6.15b we can show that the sum of outer angles at v_2, $v_1 = u_1$ and u_2 is at least 4π. Since there are three vertices, the sum of inner angles is 6π minus the sum of outer angles, i.e., at most 2π. If this sum would be exactly 2π the rays would be parallel. Therefore r and r' intersect. □

We use this to prove that T_k^R is a $\mathcal{O}(1)$-stable approximation of a long thin rhombus. We will do so by proving that T_k^R is in fact a $\mathcal{O}(1)$-stable approximation of a straight line segment of length k, which is contained as the long diagonal of a long thin rhombus of width k. The fact that T_k^R is a $\mathcal{O}(1)$-stable approximation of a long thin rhombus, will be a corollary of the following theorem.

Lemma 6.9. *The graph T_k^R is a X-stable approximation of a straight line segment ℓ_h of length k.*

Proof. We assume that $v_1''v_1$ is horizontal (i.e. the connection between the central leaf neighbor of v_1 and v_1). We place a copy of ℓ_h horizontally with its left endpoint at v_1. We will prove that any point on ℓ_h has distance at most X to a point of the UDR of T_k^R and any point of the UDR has distance at most X a point on ℓ_h.

We do so by first showing that either the polyline through $v_2, \ldots, v_{k/2+1}$ or the polyline through $u_2, \ldots, u_{k/2+1}$ has to be x- monotone in any UDR of T_k^R. The width of the bounding box of this x-monotone polyline will then force the other polyline to be also mostly horizontal and will bound the possible vertical distance to ℓ_h.

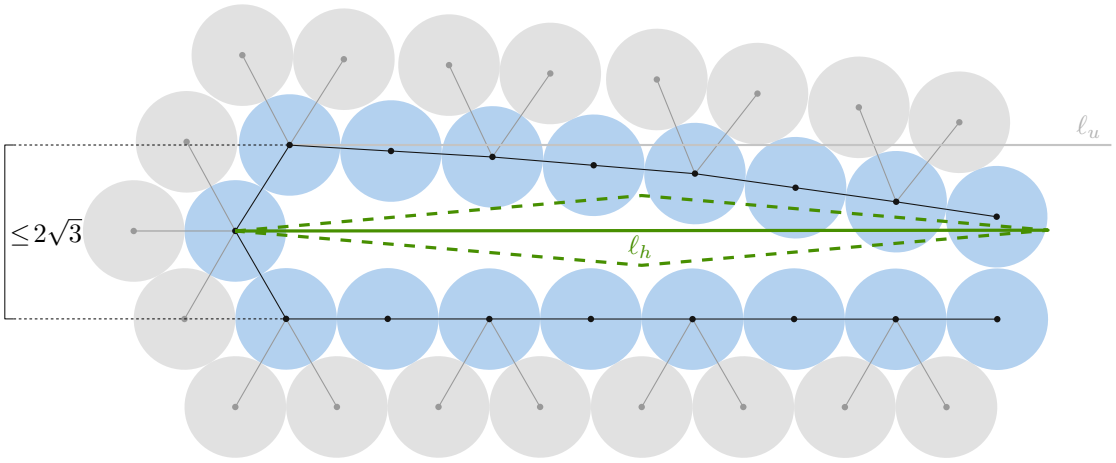


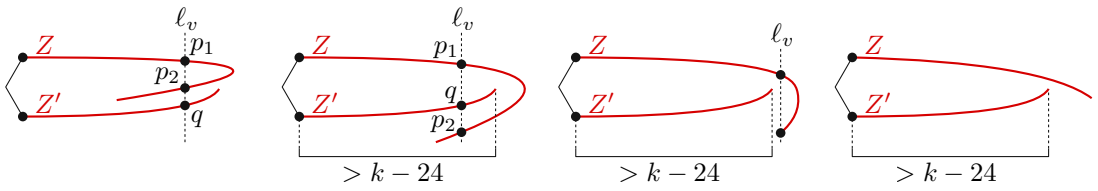
Figure 6.16: A UDR of T_{16}^R. The rhombus of width 16 and the horizontal line segment ℓ_h of length 16 are overlayed in green. The upper augmented path is displayed with small inward bends, the lower chain is displayed in its most straight (unreachable) configuration. The vertical distance between any point on the upper and lower chain is at most $2\sqrt{3}$. Note that the singular leaf neighbors attached to odd indexed vertices of the chains have been omitted to make the figure more legible.

By Lemma 6.8 we know that in any UDR the two rays r and r' intersect. We can use Lemma 6.4 to see that in any UDR of T_k^R the center of the disks of $v_2, \ldots, v_{k/2+1}$ are all contained within the halfplane below a horizontal line ℓ_u through v_2 and $u_2, \ldots, u_{k/2+1}$ are contained in the halfplane above a horizontal line through u_2. In other words, the upper augmented path can extend only a very limited amount to the top (without wrapping around the lower path, which is analyzed below) and vice versa, the lower path is very limited in its extension to the bottom.

We now have to investigate how far the upper path can extend downwards and vice versa. Consider the subgraph induced by $v_3, \ldots, v_{k/2+1}$ and their leafs, which is an augmented path of length $k/2-1$. Let Z be the polyline through the centers of the disks of $v_3, v_4, v_6, v_8, \ldots, v_{k/2}, v_{k/2+1}$. Let Z' be the polyline that is similarly defined on the vertices $u_3, u_4, u_6, u_8, \ldots, u_{k/2}, v_{k/2+1}$. Both polylines have a maximal length of $k-4$. By Observation 6.2 both polylines are convex in any UDR of T_k^R. In any UDR of T_k^R both polylines must not cross.

Assume Z is not x-monotone. This implies the existence of a vertical line ℓ_v, which intersects Z twice. The line ℓ_v can either (a) also intersect Z' or (b) not intersect Z'. If ℓ_v (a) also intersects Z', we distinguish two cases. We label the intersections from top to bottom along ℓ_v starting at the first intersection with Z as p_1, the second as p_2 and the first intersection with Z' as q (we ignore any intersection of ℓ_v and Z' above p_1).

If the top-down order of intersection points along ℓ_v is (a.1) p_1, p_2, q (see Figure 6.17a) then Z is folded in on itself and placed between itself and Z'. However since no additional



(a) Z folded in-between (b) Z folded under Z' (c) Z folds downwards (d) Both x-monotone

Figure 6.17: The two chains Z and Z'. If at least one chain is not x-monotone then (a) neither can be folded between itself and the other chain, (b-d) if Z' is not x-monotone it has a width of at least $k - 24$.

disk can be placed between the disks of the two augmented paths, this is impossible.

If the top-down order of intersection points along ℓ_v is (a.2) p_1, q, p_2 (see Figure 6.17b) then Z' is compressed horizontally and Z bypasses it on the right, then doubles back. With the same argument as in the first case, Z' can now not fold in on itself (it would need to be placed between itself and Z). Therefore in this case Z' cannot cross Z and is x-monotone which implies that the sum of inner angles is larger than $(k/2 - 4)\pi$.

If (b) there is no line that intersects Z twice and also Z' (Figure 6.17c), then Z' again has to be x-monotone (it still cannot fold between itself and Z').

In cases (a.2) and (b) the height of the axis aligned rectangular bounding box of Z' to be at most $2\sqrt{3}$ in height. By Lemma 6.7 the width of its axis aligned rectangular bounding box is at least $2(k/2 - 1) - 14 - 2\sqrt{3} = k - 16 - 2\sqrt{3} > k - 20$.

Any UDR of the other augmented path has to traverse at least this length horizontally and any extension downwards (and thereby away from the horizontal line ℓ_h) is at most its maximum length minus this value. The longest path in an augmented path starting at v_1 ends at $v_{k/2+1}$ and is of length $k/2+1$. Therefore the largest possible distance of such a path in a UDR is $2(k/2+1-1) = k$. Since the UDR of the other (possibly not x monotone) path has to traverse at least the width of the axis-aligned rectangular bounding box of the x-monotone path, its extension downwards (and thereby away from the horizontal line ℓ_h) is at most its maximum length minus this value, i.e., $k - (k - 20) = 20$.

We conclude that every point has at most a vertical distance of 20 and a horizontal distance of 20 to a point on ℓ_h and vice versa. Therefore T_k^R is a 40-stable approximation of ℓ_h. □

The reduction of Bowen et al. [25] uses long thin rhombi of constant height instead of line segments. Note that any long thin rhombus of width k and small constant height contains a line segment of length k. We therefore state the following corollary.

Corollary 6.2. *The graph T_k^R is a 40-stable approximation of a rhombus of height $\varepsilon < 1$ and width k.*

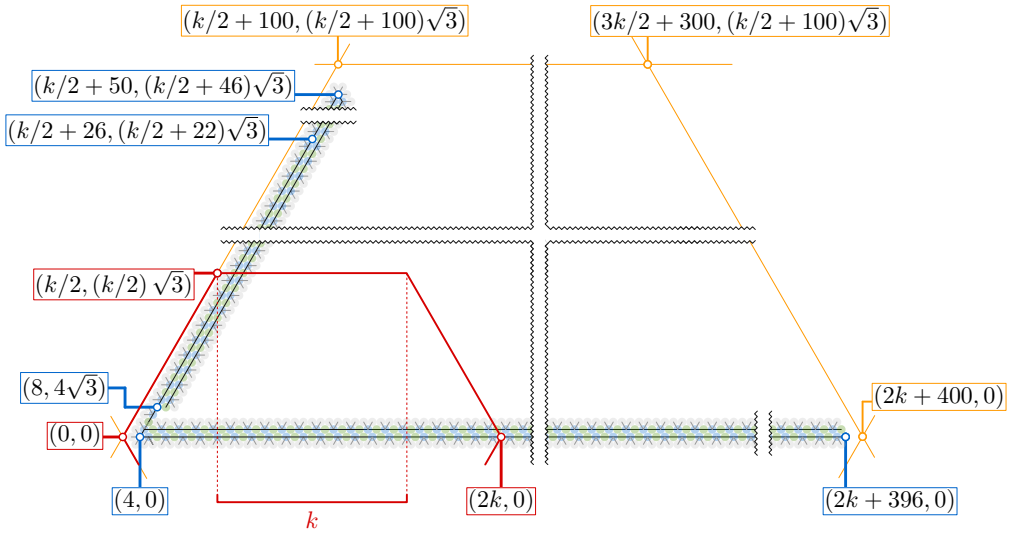
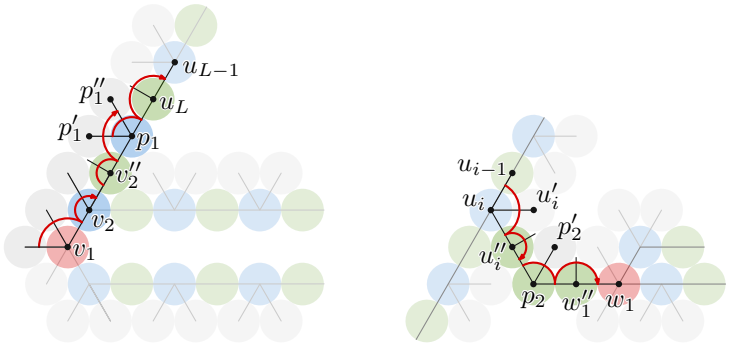


Figure 6.18: The two graphs G_1 and G_2 (connected with a Type-1 connector). Exact coordinates for relevant point are shown. Red coordinates are relevant points of H, blue coordinates are relevant points for T_k^H. Yellow coordinates are relevant for a larger hexagon P, which will contain all disks of T_k^H. The construction is not shown to proportion. Both G_1 are longer than shown (indicated with ellipses indicators in the form of wiggle lines) and k in practice will be significantly larger than shown here.

It should be noted that the bound of Lemma 6.9, while being a generous overestimation is still constant. The reader might see the similarity between our construction and the corresponding graph of Bowen et al. [25]. We will now present the construction for T_k^H, where the differences are more significant. The construction of a λ-stable approximation T_k^H of a hexagon of side length k builds on the previous construction of T_k^R. Note that this construction includes a series of large constant numbers. These are used to exceed the possible but constant compaction that is possible within the construction. Many of these numbers are chosen "sufficiently" large and do not aim to provide a tight bound, i.e., there is no claim that the presented gadget is of the minimal possible size, but simply that is a C-stable approximation for a possibly large, but constant C.

The construction of T_k^H is symmetrical. We describe the construction of the upper half, the lower half is then obtained by mirroring the upper construction at the x-axis. The hexagon we are approximating is assumed to be placed with two edges parallel to the x-axis, the left-most point at $(0,0)$ and the right-most point at $(2k,0)$.

The subgraph G_1. We first place one $G_1 = T_{2k+392}^R$, such that, its first vertex (shared by both augmented paths) is placed at $(4,0)$ and we assume that it points to the right. It approximates a horizontal line of length $2k+392$ starting at $(4,0)$ and ending at $(2k+396,0)$. This is also illustrated in Figure 6.18. Note that by construction, any UDR of G_1 is contained entirely to the right of a line with positive slope $\pi/3$ through the point



(a) Type-1 connector (b) Type-2 connector

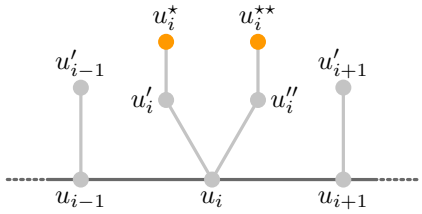
Figure 6.19: The two different connector types used in the construction of T_k^H.

$(0,0)$ and similarly entirely to the right of a line with negative slope $-\pi/3$ through the same point. Further note that the longest path of vertices in G_1 whose disks (or more precisely their centers) form an x-monotone polyline is starting at a leaf of v_1 and ending at a leaf of the last vertex of one of the augmented paths. There is a maximal horizontal distance of $2k+396$ between two points in disks of a UDR of G_1. Therefore all disks of a UDR of G_1 are entirely to the left of a line with negative slope $-\pi/3$ through the point $(2k+400,0)$. They are also contained entirely to the left of a line with positive slope $\pi/3$ through the same point.

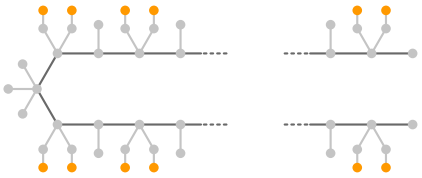
The subgraph G_2 and the Type-1 connector. Next we place a $G_2 = T_{k+84}^R$, which is connected to G_1 at the second vertex of an augmented path of G_1, using a Type-1 connector (Figure 6.19a). We now describe the Type-1 connector. Let v_1 be the first vertex of G_1 and let v_2 be the second vertex on one of the augmented paths (with its two leaf vertices v_2' and v_2''). Let u_L and u_{L-1} be the last and second to last vertex of one augmented path of G_2, respectively. We add a leaf vertex u_L' to u_L. Let u_{L-1}', u_{L-1}'' be the two leafs of u_{L-1}. We add a vertex p_1, with two leaf vertices p_1' and p_1'', which is connected to v_2'' and u_L. We also add a single additional leaf to v_2'' and fix all rotational orders of neighbors such that this leaf points to the left, i.e., the clockwise order of vertices around v_2'' is v_2, then the new leaf and then p_1. The following lemma implies that $u_1''u_1$ points upwards at most at a $\pi/3$ angle and in fact the angle has to be slightly smaller.

Lemma 6.10. *Let q_1 be a point on the same height as u_L and to its right. Then $\measuredangle u_{L-1}u_Lq_1 < \pi/3$.*

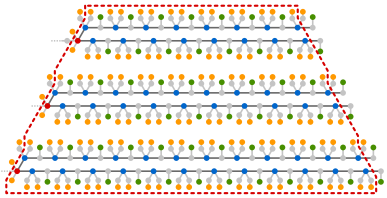
Proof. Through three applications of Lemma 6.2 we obtain a lower bound on the sum of outer angles at v_1, v_2, v_2'', p_1 and u_L (see Figure 6.19a). Note that the angle $\measuredangle p_1'p_1p_1''$ is double counted and we can subtract its minimal value of $\pi/3$ to obtain a lower bound of $3 \cdot 5\pi/3 - \pi/3 = 14\pi/3$. The lemma follows. □



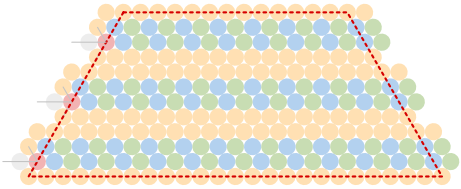
(a) Additional leaf neighbors



(b) Augmented chains with new neighbors



(c) Independent set highlighted in graph



(d) UDR of the independent set

Figure 6.20: The augmented chains of the λ-stable approximation T_k^R are augmented (a-b), such that, the core graphs contain a sufficiently large independent set (c) whose UDR forms a tight packing of unit disks (d).

A direct consequence of the previous Lemma is that all disks of G_2 are to the right of a line with positive slope $\pi/3$ through the point $(0, 0)$, which is the same line we used for our observation about G_1. By construction, the first vertex u_1 of G_2 has its center at most at a height of $(k/2 + 20)\sqrt{3}$. Any point in a disk of a UDR of G_2 is at most 3 units higher than this. Therefore all disks of a UDR of G_2 are entirely below a horizontal line through $(k/2 + 100, (k/2 + 100)\sqrt{3})$ and $(3k/2 + 300, (k/2 + 100)\sqrt{3})$ (again illustrated in Figure 6.18).

The core graphs $H_1, \ldots, H_{k/8+4}$ and the Type-2 connector. Lastly we place an even number of $k/8 + 4$ graphs $H_1, \ldots, H_{k/8+4}$. All of these graphs are T^R graphs with different lengths. However they are extended with another set of vertices. The extension is made by adding more leaf neighbors. Let v be a vertex of an augmented path of one such graph, which has two leaf neighbors v' and v''. We add two more vertices $v^\star$ and $v^{\star\star}$ and connect v' to $v^\star$ and v'' to $v^{\star\star}$, respectively (see also Figures 6.20a and 6.20b). Now for any augmented path, the vertices v_i, $v_i^\star$, $v_i^{\star\star}$ for all even i and the vertices v'_j for all odd j form an independent set. We will call the total set of all of these disks the *independent set of the core graphs*, which is highlighted for a set of three core graphs in Figure 6.20c. All disks of the different independent sets of these core graphs have to be pairwise interior disjoint (Figure 6.20d). This will be used to bound the compression of these disks in a UDR. It is also important to note that any UDR of an augmented path can be transformed into a UDR of an augmented path, which includes the newly added vertices $v_i^\star$ and $v_i^{\star\star}$, since the disks of $v_i^\star$ and $v_i^{\star\star}$ can be placed exactly on top of the disks of v'_i and v''_i, respectively and then moved a very small distance away from the disk of v_i.

Now we describe the connection of the core graphs to G_2. All core graphs are connected to the augmented chain of G_2, which was not connected to G_1. The first core graph H_1 is connected to the second leaf neighbor (the one pointing downwards in the figure) of the second vertex fo the augmented path of G_2 and is of length $k-12$. Then every next graph is connected four vertices later and with a length 8 units greater than the core graph before. In particular the graph H_i is connected to the $(4i-2)$-th vertex of the augmented chain and is of length $k+8i-20$.

We now describe the Type-2 connector and how a core graph is connected using this connector. Let u_j be the vertex of the augmented path of G_2, such that, the core graph H_i is connected with a connector to u_i''. Let w_1 be the first vertex of G_3 with its three leaf neighbors w_1', w_1'' and w_1'''. We add one vertex p_2 with one leaf vertex p_2'. The vertex p_2 is connected to u_i'' and w_1''. Further we add one leaf neighbor to u_i'' and w_1''. The embedding is defined, such that, all leafs are on the same side (we refer again to Figure 6.19b).

Lemma 6.11. *Let q_2 be the crossing point of the two rays $\overrightarrow{u_{i-1}u_i}$ and $\overrightarrow{w_1w_1''}$. Then $\measuredangle u_{j-1}q_2w_1 > \pi/3$.*

Proof. We use Lemma 6.2 twice as shown in Figure 6.19b, to obtain a lower bound of $10\pi/3$ on the sum of outer angles (which are now to the right) at u_i, u_i'', p_2 and w_1''. The lemma immediately follows. □

By Lemma 6.11, the line segment approximated by any core graph is horizontal or has a negative slope. Therefore any disk can extend at most 40 units to the top (by Lemma 6.9). Since the horizontal line through $(k/2{+}100, (k/2{+}100)\sqrt{3})$ and $(3k/2{+}300, (k/2{+}100)\sqrt{3})$ lies far more than 80 units vertically above the connection point of any core graph, no point in a disk of such a core graph can be higher than this line (even when factoring in a compression of 40 units of G_2). Similarly, since the line segment approximated by G_1 is far more than 80 units longer than the length of any line segment approximated by any core graph, every disk of a core graph is entirely contained above G_1 and to the left of the two lines with slopes $\pi/3$ and $-\pi/3$ through $(2k+400, 0)$.

We mirror the entire construction at the x axis (except G_1, which is already placed centrally). Now consider the polygon P defined by the following six points:

- $(0, 0)$
- $(k/2+100, (k/2+100)\sqrt{3})$
- $(3k/2+300, (k/2+100)\sqrt{3})$
- $(k/2+100, -(k/2+100)\sqrt{3})$
- $(3k/2+300, -(k/2+100)\sqrt{3})$
- $(2k+400, 0)$

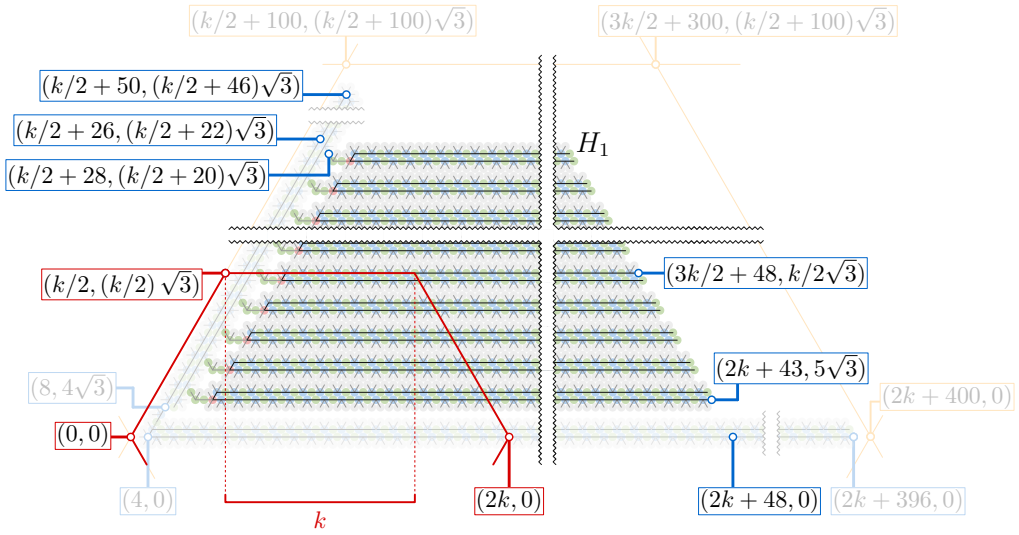


Figure 6.21: The graph G_3 (connected with a Type-2 connector to G_2) and the core graphs (connected with a Type-2/3 connector). Exact coordinates for relevant point are shown. The construction is not shown to scale. The graphs G_1, G_2, and G_3 are far longer than shown (indicated with ellipses indicators in the form of wiggle lines, note that there are two such indicators on these graphs). The core graphs are longer than they are shown (indicated with ellipses indicators in the form of wiggle lines) and k in practice will be significantly larger than shown here. The hexagon P is again shown in yellow.

It is obvious that P has a constant, albeit quite large, Hausdorff distance of at most 400 (the actual Hausdorff distance is in fact smaller than that). to the hexagon H of side length k.

As argued above, all disks are contained within the lines defining the boundaries of P and we state the following observation as a result.

Observation 6.4. *In any UDR of T_k^H all disks of G_1, G_2, G_3, all connectors and all core graphs are completely contained within P.*

We have now bounded the extension of disks towards the outside. It remains to prevent a possible compaction to the inside. Here we observe that the core graphs as displayed in Figure 6.21 form a complete packing except for one free row between G_1 and the last H graph. Note that for every disk in this complete packing, there is one disk in the independent set of the core graphs. None of these disks can overlap. Next note that by construction, there is one at least core graph whose connection point to G_2 is higher than $k/2\sqrt{3}$ in any UDR of T_k^H and which is therefore entirely above the hexagon of side length k. Thirdly note that even if the complete packing is shifted downwards to use the free space of the entire row, and to the left to extend beyond the boundary of the hexagon of side length k (red in the figures), it is still large enough to cover the entire hexagon.

And finally note that the largest vertical distance between two the lines approximated by two such core graphs is at most their distance in the densest packing, which is $4\sqrt{3}$ plus the distance between the line segment approximated by the last H graph and G_1, which is $5\sqrt{3}$. In total the distance between two such approximated line segments is at most $9\sqrt{3} < 16$. If we factor in the maximum distance the nearest disk of one of these graphs can have to the line segment approximated by its graph (40 units, Lemma 6.9), we arrive at an overestimation of the distance between two approximated line segment of 96 units.

Therefore we state the following observation.

Observation 6.5. *No point in the hexagon H is more than a constant distance of 96 away from a point contained in a disk of a UDR of a core graph.*

With the construction completed, we now state the central lemma, which combines the two observations and states the NP-hardness of the problem.

Lemma 6.12. *The graph T_k^H is a 400-stable approximation of a hexagon H of side length k.*

Proof. One direction of this lemma follows from the fact that by Observation 6.5, the UDR of T_k^H covers H with at most constant sized gaps, which means that the distance from any point of H to a point contained in a disk of the UDR of T_k^H is at most 91. The other direction follows from Observation 6.4, which states that all disks are contained within a shape, and that shape has at most a constant Hausdorff distance of 400 to H, far exceeding the distance of the other direction, but still remaining constant. □

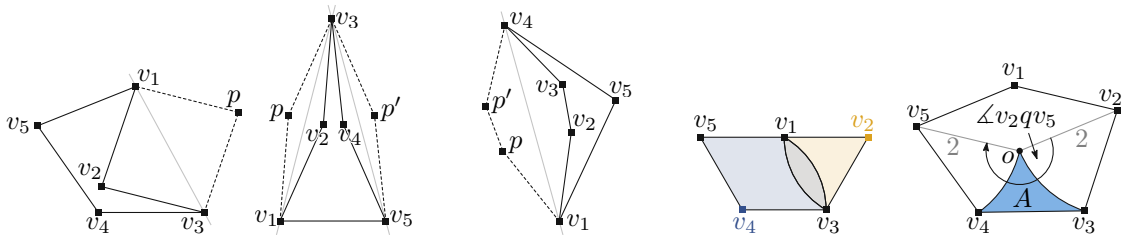
Note that a UDR of T_k^H extends quite far beyond the hexagon it is approximating. However by simply using the larger hexagon P as the one that is being approximated, one could easily obtain an $\mathcal{O}$-stable approximation of a regular hexagon, whose UDR is in fact entirely contained within the hexagon it is approximating.

In this section we have presented two graphs T_k^R and T_k^H, which are a 40-stable approximation of a rhombus of width k (Lemma 6.9) and a 400-stable approximation of a hexagon of side length k (Lemma 6.12). Using the method described above to model the hinges, NP-hardness follows from Bowen et al. [25].

Theorem 6.1. *The recognition of unit disk graphs is NP-hard for embedded trees.*

6.4.3 Recognizing UDRs is NP-hard for Outerplanar Graphs

In the previous section, we showed how we can build graphs, whose UDR approximate a rhombus and a hexagon of different sizes within a constant error using trees with a prescribed rotational order of neighbors around every vertex. The rotational order restriction is crucial, since the augmented paths rely on all leafs being placed on the same side of the construction so the the lemmata of Section 6.3 can be applied. In



(a) (b) (c) (d) (e)

Figure 6.22: (a-c) Transformation into a convex pentagon in the proof of Lemma 6.14. (d) All angles in P are strictly smaller than π. (e) Points in A have distance larger than 2 to v_2 and v_5. Angle $\measuredangle v_2ov_5$ would need to be smaller than π, such that, q has distance larger than 2 to v_4 and v_5.

this section, we start by providing initial lemmata, which show how we can force the embedding in a bi-connected graph using small cycles. Using this we present a $\mathcal{O}(1)$-stable approximation of a line segment (and rhombus) of width k. This is a critical component for the $\mathcal{O}(1)$-stable approximation. The rest of the section will show how all other required components (the connectors, the independent set in the core graphs and the placement of the core graphs) can be achieved.

Forcing embedding with small cycles.

If we do not prescribe a rotational order, we will have to force the placement of all leafs on the same side through the graph structure. We can do so for example by creating/adding cycles to the graph. One example is the following observation, which follows easily from the triangle inequality.

Observation 6.6. *Any point q placed inside a triangle P, whose sides have length at most 2 has a distance of at most 2 to two (or more) corners of P.*

Now consider a graph that consists of a triangle between a, b and c and has a fourth vertex d, which is connected only to a. Since the distance between centers of connected disks in a UDR is at most two and between not connected disks it has to be larger than two, Observation 6.6 implies that the center of the disks of vertex d cannot be placed within the triangle between the centers of a, b and c. Therefore the rotational order of disks around a is fixed in any UDR (up to mirroring the entire UDR).

However, while it is obvious that such a placement is impossible for a 3-cycle, it is also impossible, for a 4- and 5-cycle, although this is less obvious. We state the following two lemmata, which similarly imply the impossibility of placing leafs inside 4- or 5-cycles in a UDR.

Lemma 6.13. *Any point q placed inside a quadrilateral P, whose sides have length at most 2 has a distance of at most 2 to two (or more) corners of P.*

Proof. Let $v_i, i \in \{1,2,3,4\}$ be the four corners of a quadrilateral P. Let A_i be a closed disk of radius 2 around v_i. Since the sides of P are of length at most two, we know that the triangle $\triangle v_1v_2v_4$ is contained in A_1. Similarly the triangle $\triangle v_3v_2v_4$ is contained in A_3. Since q is contained in $\triangle v_1v_2v_4$ or $\triangle v_3v_2v_4$, q has distance at most 2 to either v_1 or v_3. With a similar argument q has a distance of at most 2 to either v_2 or v_4. □

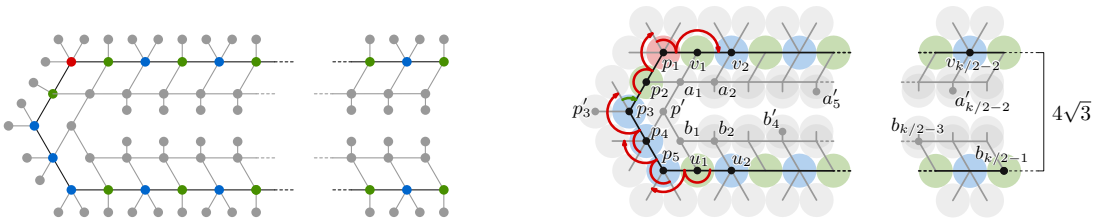
Lemma 6.14. *Any point q placed inside a pentagon P, whose sides have length at most 2 has a distance of at most 2 to two (or more) corners of P.*

Proof. Let $v_i, i \in \{1,2,3,4,5\}$ be the five corners of P. First we show that we can assume that all inside angles $\measuredangle v_{i+1}v_iv_{i-1}$ of the pentagon defined by the circle centers in the UDR are smaller or equal than π, i.e., the pentagon is convex.

Assume there is only one angle $\gamma = \measuredangle v_{i+1}v_iv_{i-1} > \pi$ and, without loss of generality, let $i = 2$ (Figure 6.22a). Let p be the image of v_2 mirrored along the line $\overline{v_1v_3}$. Then $\measuredangle v_3pv_1 < \pi$. Observe that $P \subseteq \pentagon v_1pv_3v_4v_5$. Additionally $\overline{v_1v_3}$ is the bisector of $\overline{v_2p}$. Since all points $q \in P$ are on the same side of $\overline{v_1v_3}$ as v_2, we have $|qv_2| \leq |qp|$. Next assume there are two non-consecutive angles larger than π and they are without loss of generality at v_2 and v_4 (Figure 6.22b). We mirror v_2 at $\overline{v_1v_3}$, yielding p and v_4 at $\overline{v_3v_5}$ yielding p'. Observe that $P \subseteq \pentagon v_1pv_3p'v_5$. Again all points $q \in P$ (as well as p') are on the same side of $\overline{v_1v_3}$ as v_2, we have $|qv_2| \leq |qp|$. Similarly all points $q \in P$ (as well as p) are on the same side of $\overline{v_3v_5}$ as v_4, we have $|qv_4| \leq |qp'|$. Now assume there are two consecutive angles larger than π and they are without loss of generality at v_2 and v_3 (Figure 6.22c). In this case we mirror v_2 and v_3 along $\overline{v_1v_4}$. Let p and p' be the mirror images of v_2 and v_3, respectively. Observe that $P \subseteq \pentagon v_1pp'v_4v_5$. Again all points $q \in P$ are on the same side of $\overline{v_1v_4}$ as v_2 and v_3, we have $|qv_2| \leq |qp|$ and $|qv_3| \leq |qp'|$. Therefore it is sufficient to consider convex pentagons.

We assume that the segment v_3v_4 is horizontal and that v_1 lies above v_3v_4. Further we assume that the distance between q and v_1 is smaller than from q to any other corner. Towards a contradiction we assume that q has a distance larger than 2 to all $v_i, i \in \{2,3,4,5\}$.

We can now further restrict the possible position of q within P. In particular let A be the closed area bounded by v_3v_4, a circular arc C around v_5 with radius 2 and a circular arc C' around v_2 with radius 2. Let o be the meeting point of C and C' contained in P (if both intersection points lie outside of P, no point in P has a distance larger than 2 to both v_2 and v_5 and we are done). If there is a valid position for q in A, then The point q cannot be contained within C or C'. Therefore $q \in A$ (Figure 6.22e) and $q \in \triangle v_4v_5o$ since A is contained within this triangle. If $|ov_3| \leq 2$ (or $|ov_4| \leq 2$), then no point in A has a distance to v_3 (or v_4) larger than 2, which would be a contradiction to our assumption. Therefore we assume both distances to be larger than 2. Since $|v_2o| = 2$, $|v_2v_3| \leq 2$ and $|v_3o| > 2$, we know that $\measuredangle v_2ov_3 < \pi/3$ and similarly $\measuredangle v_4ov_5 < \pi/3$. Also since $|v_3o| > 2$, $|v_4o| > 2$ and $|v_3v_4| \leq 2$, we know that $\measuredangle v_3ov_4 < \pi/3$. This implies o is in the triangle $\triangle v_2v_2v_5$ and therefore $|v_5o| + |ov_2| \leq |v_5v_1| + |v_1v_5|$. Since $|v_5o| + |ov_2| = 4$,



(a) Graph G_k^R (b) UDR of G_k^R

Figure 6.23: The (a) graph structure of the 50-stable approximation G_k^R and (b) a UDR of the graph. The angle restrictions implied by the structure are highlighted in red.

we have $|v_5v_1| + |v_1v_2| \geq 4$. By assumption $|v_5v_1| + |v_1v_2| \leq 4$ and we finally obtain $|v_5v_1| + |v_1v_2| = 4$. If this inequality is strict, this is a contradiction. □

A consequence of Lemmata 6.13 and 6.14 is that in a UDR of a graph of a 4- or 5-cycle, any disk of a vertex which is adjacent to at most one of the vertices of the cycle cannot be placed "inside" the cycle, that is, it cannot be placed such that its center is contained within the polygon defined by the centers of the cycle. This also applies also if the 4- or 5-cycle is an induced subgraph of a larger graph.

A $\mathcal{O}(1)$-stable approximation of a rhombus.

With these results established, we again provide a λ-stable approximation G_k^R of a long thin rhombus of width k. We again want to place two augmented paths opposite of each other so close, that, in any UDR of G_k^R, neither of the two paths can be folded between itself and the other augmented path. Here we will refer to the two augmented paths as $v_1, \ldots, v_{k/2-1}$ and $u_1, \ldots, u_{k/2-1}$. Note that the indices are smaller than for T_k^R, which is due to a different labeling scheme. The construction is similar to T_k^R, however it is wider to accommodate for a set of 4- and 5-cycles, which will force the placement of leafs to the outside.

The first vertices v_1 and u_1 are connected with five vertices $p_1, \ldots, p_5$ in this order from v_1 to u_1. The vertices $p_1, \ldots, p_5$ are of degree 5, 3, 4, 4 and 4 respectively. Now we place a second chain of vertices per path, $a_1, \ldots, a_{k/2-1}$ and $b_1, \ldots, b_{k/2-1}$. Every a_i (b_i) is connected to v_i (u_i) as well as to a_{i-1} (b_{i-1}) and a_{i+1} (b_{i+1}). a_1 is also connected to p_2. Further one additional vertex p' is placed and connected to a_1, b_1 and p_4. And lastly, we add a degree-one leaf a'_i (b'_i) to any a_i (b_i). For an illustration we refer to Figures 6.23b and 6.23a.

It is crucial to note that all faces of G_k^R are 3-, 4- or 5-cycles. Moreover, any disk that is not part of a cycle, is adjacent to at most one vertex of the cycle. By Observation 6.6 and Lemmata 6.13 and 6.14, we know that any disk placed inside a UDR of a 3-, 4- or 5-cycle has to overlap at least two disks of that cycle. Therefore all faces of the straight line drawing induces by a UDR of a G_k^R have to be empty and the outer face is fixed.

The possible permutation of leaf neighbors is not a problem for our arguments, which only rely on the fact that all leaf neighbors of the augmented paths are placed on the same side (which will again be called the "outside"). We will use these leafs again to apply Lemmata 6.1 and 6.2 and place angle restrictions on the realization.

Lemma 6.15. *In every UDR of G_k^R the rays r (starting at v_1 through v_2) and r' (starting at u_1 through u_2) intersect.*

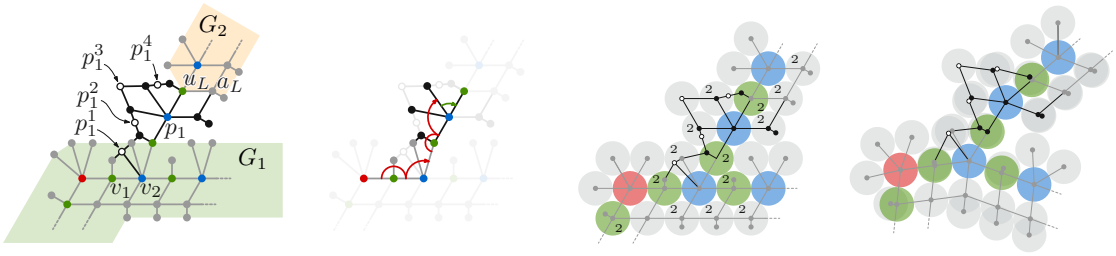
Proof. Using Lemma 6.1 once and Lemma 6.2 five times, as shown in Figure 6.15b, we can show a lower bound for the sum of outer angles at angle sum $v_1, p_1, p_2, p_3, p_4, p_5$, and u_1, which is $5(5\pi/3) + \pi/3 - 2(\pi/3) = 8\pi$ (we have double counted the angles $\measuredangle p_4'' p_4 p_4'$ and $\measuredangle p_5'' p_5 p_5'$ and therefore subtract the minimum values these angles need to have due to Lemma 6.1). The sum of inner angles at $v_1, p_1, p_2, p_3, p_4, p_5$, and u_1 is therefore less than 6π. If this sum would be exactly 6π the rays would be parallel. Therefore r and r' intersect. □

Since we know that all leafs are on the outer side of the augmented paths and the two starting edges of the augmented paths point towards each other (due to Lemma 6.15). We also observe that it is again impossible for any UDR of the augmented paths together with all added vertices to be folded between itself and the UDR of the opposing augmented path. Therefore we state the following Lemma analogue to Lemma 6.9.

Lemma 6.16. *The graph G_k^R is a 50-stable approximation of a line segment of length k.*

Proof. Note that the two arms of G_k^R both contain an augmented path as an induced subgraph. We assume that $v_1 v_2$ is horizontal at a height of $2\sqrt{3}$. By Lemma 6.4 all centers of disks of vertices of the upper arm are below a line at height $2\sqrt{3}$. By Lemma 6.15, the ray $\overrightarrow{u_1 u_2}$ intersects the ray $\overrightarrow{v_1 v_2}$ and therefore u_2 is above u_1. By construction, the angles shown in Figure 6.23b are maximal and v_1 is therefore at least at a height of $-2\sqrt{3}$. By Lemma 6.4 all centers of disks of vertices of the upper arm are below a line at height $-2\sqrt{3}$. Since neither UDR of the augmented paths can be folded in-between itself and the other arm, we know that at least one augmented path has to be realized in an x-monotone manner and, by Lemma 6.7, the width of the axis aligned rectangular bounding box of any UDR of this augmented path is at least $2(k/2-1) - 14 - (4\sqrt{3}) = k - 16 - 4\sqrt{3} > k - 23$. The longest path in an augmented path (which is an induced subgraph of G_k^R) starting at p_3 ends at $v_{k/2-1}$ and is of length $k/2 + 2$. Therefore the largest distance between p_3 and a point of the augmented path is at most $k + 4$. Any UDR of the other augmented path has to travel at least the width of the axis aligned rectangular bounding box of the UDR of the x-monotone augmented path. Only then can it bypass that other augmented path and travel a remaining distance vertically. Therefore the maximal vertical distance is at most $k + 4 - (k - 23) = 27$. We again add the maximum horizontal and maximum vertical distance and arrive at a value of $27 + 23 = 50$.

With the same argument as for T_k^R, we can therefore state that G_k^R is a 50-stable approximation of a line segment of length k. □



(a) The connection (b) Angle restriction (c) Tight packing (d) Correct realization

Figure 6.24: The Type-1^o connector connects the two subgraphs G_1 and G_2 (a) by adding additional vertices and edges, i.e., the black vertices and edged in (a). This connector still contains the necessary subgraph to enforce the same angle restrictions as the Type-1 connector (b). A tight packing of the corresponding disks in a UDR is shown in (c), where disks, which represent two disks that are placed almost exactly at the same spot are marked with a small 2. The distances between disks are very small and would not be visible. A realization with exaggerated distances and correct overlap is shown in (d).

Similar to the tree construction we also again observe the following corollary.

Corollary 6.3. *The graph G_k^R is a 50-stable approximation of a long thin rhombus of width k.*

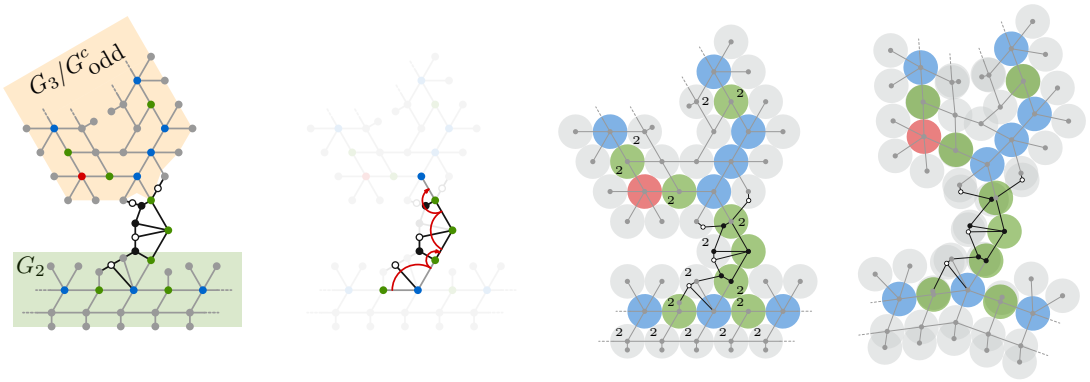
A $\mathcal{O}(1)$-stable approximation of a regular hexagon.

Next we reuse the construction of T_k^H to create G_k^H. It is again symmetrical and made up of one central G_1 to which a G_2 is attached at a maximal angle of $\pi/3$ and in turn several graphs are attached at a maximal angle of $2\pi/3$. The constant values added to the lengths of these G_k^Hs are slightly different, however it is unsurprising that a very similar construction can be achieved.

The crucial part that has to be replicated is the connectors. Therefore we start by presenting the three different connectors, which force the same angle restrictions and at the same time also force the correct rotational order for all relevant vertices, since we cannot prescribe this order.

The connectors. We describe the construction of the two types of connectors analogue to the construction of the Type-1 and Type-2 connector. Both constructions are based around the idea of placing different graph G^R and connecting them with additional vertices. Some additional vertices are placed to connect vertices of different graphs, while others form connections between vertices of the same graph.

We start by describing the construction of the Type-1^o connector (the outerplanar analogue of the Type-1 connector), which is shown in Figure 6.24. This is the connector that will connect the central graph G_1 to the first appended graph G_2 (see Figure 6.24a). While both are connected to a leaf of the second vertex of an augmented path, due to



(a) The connection (b) Angle restriction (c) Tight packing (d) Correct realization

Figure 6.25: The Type-2^o connector connects the two subgraphs G_2 and G_3 or an odd indexed core graph (a) by adding additional vertices and edges, i.e., the black vertices and edged in (a). This connector still contains the necessary subgraph to enforce the same angle restrictions as the Type-2 connector (b). A tight packing of the corresponding disks in a UDR is shown in (c), where disks, which represent two disks that are placed almost exactly at the same spot are marked with a small 2. the distances between disks are very small and would not be visible. A realization with exaggerated distances and correct overlap is shown in (d).

the difference in labeling, the second vertex of a G^R is "farther along" the graph. We again connect v_2'' and u_L by introducing a vertex p with two leaf neighbors p' and p'' (which are only leaf neighbors for now and will be further connected). We also still add the additional leaf neighbors v_2''' and u_L' to v_2'' and u_L, respectively. This is similar to the construction of the Type-1 connector. Next we connect v_2''' and v_2' with an edge.

Then we add four additional vertices p_1^1, p_1^2, p_1^3 and p_1^4, which act as bridges between existing vertices. Specifically the vertices create the following connections:

- p_1^1 is connected to v_1' and v_2',
- p_1^2 is connected to v_2''' and p_1',
- p_1^3 is connected to p_1' and p_1'' and
- p_1^4 is connected to p_1'' and u_L'.

All newly added vertices and connections are shown in Figure 6.24a.

Note that this connects the sequence of 4-cycles of the augmented path of G_1 to the sequence of 4-cycles of the augmented path of G_2 via a sequence of 3-, 4- and 5-cycles. Observation 6.6 and Lemmata 6.13 and 6.14 force the newly added vertices p_1^1, p_1^2, p_1^3 and p_1^4 as well as the two vertices p_1' and p_1'' and the vertex u_L' to be all locally on the same

side of the polyline through the centers of v_1, v_2, v_2'', p_1 and u_L. It also forces the chain of 4-cycles of the augmented path of G_2 containing u_L to be locally on the right side of that polyline, and thereby forces the entire orientation of G_2 (Note that technically leaf vertices of nodes of an augmented path, which are connected to the same vertex and are not further connected can placed in reverse order, but since they are indistinguishable, this is irrelevant).

Now with the placement of the leafs forced, the angle restriction analogous to Lemma 6.10 simply follows again from two applications of Lemma 6.2 and one application of Lemma 6.1 (see Figure 6.24b).

In a tight packing, the resulting UDR leads exactly to the $\pi/3$ angle restriction, which is required (Figure 6.24c). Any UDR would introduce very small distances to avoid unwanted adjacencies. An exaggerated realization, which introduces visible non-overlap, is shown in Figure 6.24d.

The adaptions for the connectors of Type-2^o are created in a similar fashion. The same steps:

(a) addition of new vertices, which connect the chain of 4-cycles of the two augmented paths

(b) observation of present angle restrictions using Lemmata 6.13 and 6.14

(c) presentation of the angle restriction in a tight packing UDR and

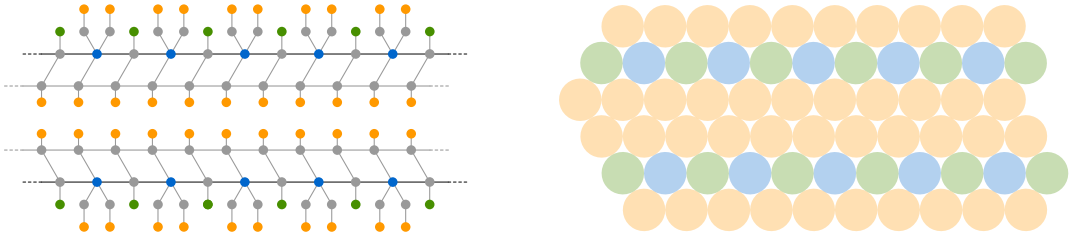
(d) illustration of a correct representation using exaggerated distances

are illustrated in Figure 6.25.

Therefore we can also replicate Type-2 connectors with the properties stated in Lemma 6.11.

The independent set in the core graphs. To adaption made to the G^R core graphs is made in the same fashion as for the T^R core graphs, i.e., any leaf neighbor of a vertex with two leaf neighbors is augmented with another leaf neighbor. The independent set is however larger since we also include all a_i' and b_i' vertices. This yields three rows of a complete packing per augmented path or six rows for the entire graph G^R. The adaption is illustrated in Figure 6.19, which shows the graph structure and the resulting packing of disks in similar coloring.

The placement of the core graphs. The core graphs are connected to G_2 via Type-2^o connectors. However as described above, the tightest packing of the independent set of the core graphs leads to 6 rows of disks, rather then 4 as was the case for the trees. Note that this is again a set of vertices of the augmented path of G_2, such that, every vertex of the set has two leaf neighbors. We can therefore use the Type-2 connector for all core graphs. As a result there are less core graphs and the connection between the core graphs and G_2 is made every sixth vertex, instead of every fourth. As a result the core graphs



(a) Graph structure (b) UDR of independent set

Figure 6.26: Adaption of an augmented path of the outerplanar λ-stable approximation G_k^R, such that it contains a sufficiently large independent set (a), which results in a tight packing of six rows (b).

can again be connected such that the tight packing of disks of the independent set has at most a constant distance to G_1.

With the connectors established and the possibility to create an independent set of sufficient size, which can be sufficiently tightly packed, the construction of G_k^H is done in the same manner as for T_k^H, although the constant in the length of the different subgraphs will differ. Nevertheless the construction retains the same properties, so we can state the following lemma.

Lemma 6.17. *The graph G_k^H is a $\mathcal{O}(1)$-stable approximation of a hexagon of side length k.*

Using again the method described in Section 6.4.2 to model the hinges, NP-hardness follows from Bowen et al. [25].

Theorem 6.2. *The recognition of unit disk graphs is* NP*-hard for outerplanar graphs.*

6.5 Recognition Algorithm for Caterpillars

In this section, we propose a linear-time algorithm that recognizes if an input caterpillar graph $G = (V, E)$ admits a UDR or not; it uses ideas similar to an algorithm in a conference version of a publication by Klemz et al. [73], which aimed to recognize caterpillar graphs that admit a UDC in linear time; this result, however, was reduced to a conjecture in the journal version [72], due to a flaw in their original proof. This flaw does not pose an obstacle for our approach, which is constructive and provides a representation if one exists. However, we need to address several new issues as we show that a larger class of graphs admits a UDR compared to a UDC. Clearly, if G contains a vertex of degree at least 6, then due to the unit disk packing property, it does not admit a UDR. Hence, every realizable caterpillar must have maximum degree $\Delta \leq 5$. Moreover, it is easy to observe that all caterpillars with $\Delta \leq 4$ admit a UDC (and thus a UDR), as also noted by Klemz et al. [73]. Not every caterpillar with $\Delta = 5$, however, is realizable as UDR. We show that two consecutive degree-5 vertices on the spine path B_G cannot be realized. Surprisingly, the absence of two consecutive vertices of degree 5 in a caterpillar with

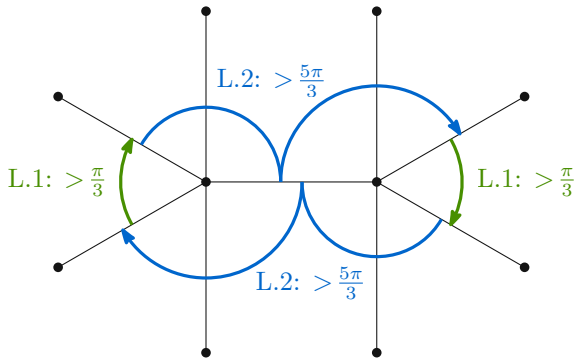


Figure 6.27: Illustration of the impossibility of two consecutive degree-5 vertices in a unit disk caterpillar. The applications of lemmata, which provide angle bounds are indicated in green and blue.

$\Delta = 5$ is necessary and sufficient, leading to a characterization of caterpillars, which admit a UDR.

First we prove an auxiliary Lemma, which allows us to assume certain distances in a UDR.

Lemma 6.18. *If B_G contains two adjacent degree-5 vertices u, v, then it does not admit a UDR.*

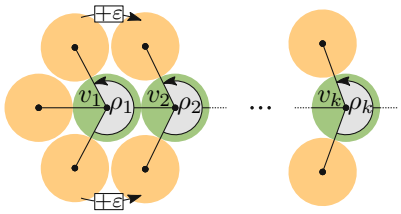
Proof. We obtain the Lemma statement immediately from two applications of Lemma 6.1 and two applications of Lemma 6.2. This is illustrated in Figure 6.27. □

Now we will present an algorithm, which realizes a caterpillar as a UDR, decides that no such realization exists.

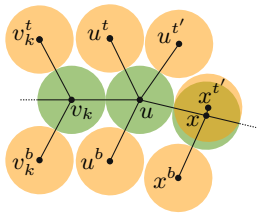
6.5.1 The Algorithm

As a preprocessing step we augment all spine vertices of degree 3 or lower with additional degree-1 neighbors such that they have degree 4. Consider a chain $v_1, \ldots, v_n$ of spine vertices. Now assume all vertices are exactly of degree 4. We place them on a horizontal line. For each $1 \leq i \leq n$ at disk $d(v_i)$, we place its leaf neighbor disks $d(v_i^t), d(v_i^b)$ first at the top and then at the bottom of $d(v_i)$, see Figure 6.28a such that the clockwise angle $\measuredangle v_i^t v_i v_i^b = \frac{4\pi}{3} - 2i\varepsilon$. The rotational ε offset avoids adjacencies between subsequent leaf disks on the same side of the spine. While these offsets can add up, we can choose ε small enough for every finite caterpillar (a polynomial function of the size of the caterpillar, which can simply be computed based on this size) such that this is negligible.

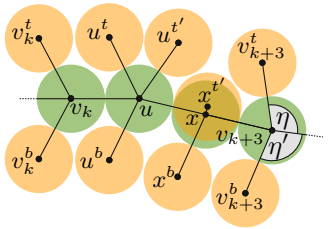
Now we assume that at least one vertex is of degree 5. To keep the entire construction of the spine x-monotone, whenever we encounter a degree-5 vertex $u = v_{k+1}$ after a degree-4 vertex v_k, we place $d(u^{t'})$ of its additional leaf $u^{t'}$ alternatingly on the top or the bottom



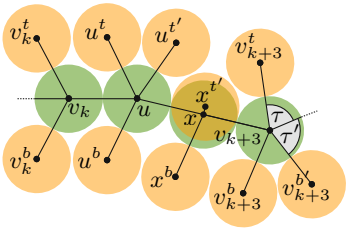
(a) ε-offsets. $\rho_k = \frac{4\pi}{3} - 2k\varepsilon$



(b) Placement of x after encountering u



(c) $\eta = \eta'$ and $\eta + \eta' = \frac{4\pi}{3} - (2k+6)\varepsilon$



(d) $\tau = \tau'$ and $\tau + \tau' = \pi - (2k+7)\varepsilon$

Figure 6.28: Chains of degree-4 vertices are placed in a dense packing formation with small offsets **(a)**. A degree-5 vertex places an additional leaf on one side **(b)**. The next vertex v_{k+3} can again be placed with the desired angle of just over $\frac{2\pi}{3}$ between two neighbors **(c)**. Placement of v_{k+3} is possible if its degree is 5 **(d)**. Note that, the rotational offset angles are exaggerated, for better readability.

side with a $\frac{\pi}{3} + \varepsilon$ rotational offset to $d(u^t)$ (or $d(u^b)$). We will assume that we placed the disk at the top. Therefore $\measuredangle u^{t'} u u^b \leq \pi - (2k+1)\varepsilon$, i.e., they form an almost horizontal connection, see Figure 6.28b.

If the next vertex $x = v_{k+2}$ has also degree five, then due to Lemma 6.18 we know that the sequence is not realizable. Otherwise, we place $d(x)$ such that it is touching $d(u)$ with a $\frac{\pi}{3} + \varepsilon$ rotational offset to $d(x)$, see Figure 6.28b. We place $d(x^b)$ at the planned position relative to $d(x)$ at the bottom, i.e., with a $\frac{\pi}{3} + (k+2)\varepsilon$ counterclockwise offset relative to the x-axis, however, we place $d(x^t)$ almost exactly on top of $d(x)$ with a very small shift of $\frac{\varepsilon}{Cn}$ orthogonal to $\overline{ux}$, for some large constant C. This prevents touching of $d(u)$ and $d(x^t)$, without creating an adjacency between $d(u^{t'})$ and $d(x^t)$.

From this point onwards, we consider the direction of $\overline{ux}$ to be the direction in which we extend the spine of the caterpillar. Any following disk $d(v_{k+3})$ can be placed again in the new extension direction touching $d(x)$. Its leaf disks $d(v_{k+3}^t)$ and $d(v_{k+3}^b)$ can be placed in their planned positions, i.e., with a clockwise or counterclockwise offset of $\frac{\pi}{3} + (k+3)\varepsilon$ relative to the new extension direction, respectively, which results in a clockwise angle $\measuredangle u_t u u_b \leq \frac{4\pi}{3} - (2k+6)\varepsilon$. Note that v_{k+3} can have a degree of four (Figure 6.28c) or five (Figure 6.28d) and that at this point, if v_{k+3} has degree five, we can immediately repeat this procedure. As a postprocessing step, we remove all degree-1 vertices that were added in the preprocessing step.

Correctness If a caterpillar contains consecutive degree-5 vertices we reject it as it has no UDR by Lemma 6.18. In any other case, the algorithm above can represent it in a way such that at any point the angle between a spine vertex and its left-most top and left-most bottom leaf is less than π. As long as this property holds, we can always add a new spine vertex. Moreover, if the sequence is extended by a spine vertex of at most degree 4, this property immediately holds again. If the sequence is extended by a spine vertex of degree 5, a spine vertex of degree at most 4 must follow. Once we place this degree 4 vertex appropriately, cf. Figure 6.28b, then the property immediately holds again.

Then from the above description of the algorithm and the correctness analysis we obtain the following theorem.

Theorem 6.3. *Let $G = (V, E)$ be a caterpillar graph. G admits a UDR if and only if G does not contain any two adjacent degree-5 vertices in the spine path B_G of G. This property can be tested in linear time and if a UDR exists then it can be constructed in linear time.*

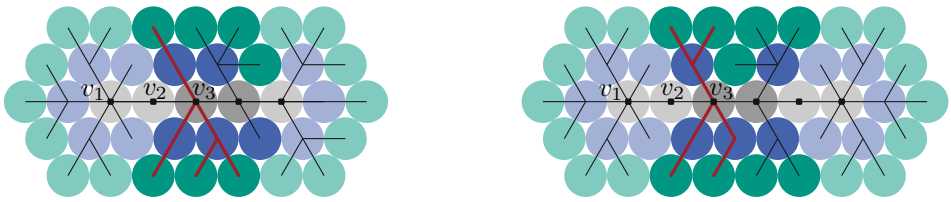
6.6 Weak UDCs of Lobsters on the Triangular Grid

We have shown that recognition of UDRs is NP-hard for outerplanar graphs and linear-time solvable for caterpillars, which mirrors the results for UDCs and weak UDCs; it leaves the recognition complexity for (non-embedded) trees as an open question for both UDRs and UDCs. For weak UDCs, however, recognition has been proven NP-hard for trees [38]. In order to investigate the complexity of weak UDCs further, we zoom in on the gap between trees and caterpillars and investigate the graph class of lobsters.

The *spine* of a weak UDC of a lobster G is the polyline defined by connecting the centers of all disks belonging to the vertices of B_G in order. A weak UDC is *straight*, if its spine is a straight line segment. Similarly, a weak UDC is x- or y-monotone, if its spine is x- or y-monotone. Since we consider weak UDCs with contacts between non-adjacent disks permitted, we focus our attention on weak UDCs placed on a triangular grid that admits tightly packed unit disks (similar to previous work on weak UDCs [38]). Finally we call the set of grid positions, which are occupied by a disk in a weak UDC A on the grid, the *occupation pattern* induced by A.

6.6.1 Straight Spine Lobsters

Since any caterpillar G admits a weak UDC if and only if it admits a straight weak UDC [38] we investigate lobster graphs, which admit a straight weak UDC. These are not all lobsters, since any simple lobster graph containing a non-spine vertex of degree 6 only admits a non-straight weak UDC. We observe that already for this restricted subclass, a greedy placement scheme similar to Cleve's approach [38] for caterpillars is not possible; shown by the following example.



(a) G has 18 possible weak UDCs (b) G' has 12 possible weak UDCs

Figure 6.29: The subgraphs of G and G' induced by their first three spine vertices are equal, however, depending on the following vertices a different realization of the neighbors of v_3 is necessary.

We specify two lobster graphs G and G', see Figure 6.29. It can be checked via exhaustive enumeration that G admits 18 different weak UDCs, while G' admits only 12. The subgraphs induced by their first three spine vertices and all of their (non-spine) descendants are identical, however the realization of the descendants of v_3 (highlighted in red) is unique for both graphs (up to symmetry) and dependent on the structure of the graphs beyond this point. We can therefore not simply scan over the spine in a greedy manner and fix all positions for the disks of descendants of a spine vertex and then continue on to the next. It is, however, still possible to do this in linear time with dynamic programming. The requirements for this are actually less strict, as it is already sufficient to have an x-monotone rather than a straight spine, which we show in the next section.

6.6.2 Monotone Weak UDCs

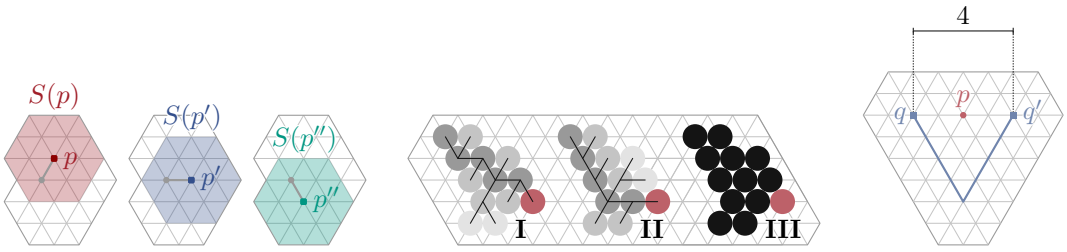
If a lobster can be realized as an x-monotone weak UDC, we can compute such a weak UDC with a linear-time dynamic programming algorithm. The dynamic program uses the following three observations.

Lemma 6.19. *The number of possible placements of a spine vertex v_i and its descendants is constant for a fixed position of v_{i-1}, i.e., the previous spine vertex.*

Proof. For a grid position p, let $S(p)$ be the set of all grid positions with a grid distance to p of at most 2. Due to the monotonicity, the vertex v_i can only be placed at one of three possible grid positions, which are adjacent to v_{i-1}. For each such position p every placement of v_i and its descendants corresponds to a subset of the constant size set $S(p)$, which yields again a constant number of placements, as shown in Figure 6.30a. □

In the following Lemma, we refer to the subgraph of a lobster graph G induced by the last k spine vertices and their descendants as the k-appendix of G.

Lemma 6.20. *Let G, G' be two lobster graphs such that they have the same k-appendix A. Further let G admit a weak UDC U and let $G' \setminus A$ admit a weak UDC W' such that the occupation pattern induced by U over the last 8 spine vertices of $G \setminus A$ and the occupation*



(a) (b) (c)

Figure 6.30: Examples for the proofs of **(a)** Lemma 6.19, which shows the constant sized sets $S(p)$ for three possible placements of the next spine vertex, **(b)** Lemma 6.20, which displays two different weak UDCs (I and II) of the same graph placing their last spine vertex at the same spot (red disk), but inducing the same occupation pattern (III) and **(c)** Lemma 6.21, highlighting the at most constant distance between q and q', which are (in terms of x-coordinate) the minimum and maximum grid position able to occupy p with a descendant.

pattern induced by W' over the last 8 spine vertices of $G' \setminus A$ are equal. Then G' admits a weak UDC.

Proof. Assume without loss of generality, that in U the disk of the first spine vertex of A was placed one grid position to the right of the disk of the last spine vertex of $G \setminus A$. We now translate the weak UDC of A, i.e., the subpart of U induced by the spine vertices of A and their descendants such that the disk of the first spine vertex of A is placed one grid position to the right of the last spine vertex of W', and as such constructing a new weak UDC U'. Since occupation pattern induced by U over the last 8 spine vertices of $G \setminus A$ and the occupation pattern induced by W' over the last 8 spine vertices of $G' \setminus A$ are equal, as illustrated in Figure 6.30b, U' is a valid weak UDC of G'. □

Lemma 6.21. *For a fixed grid position p, which could be occupied by a spine vertex v_i or one of its descendants, the number of spine vertices of a strictly x–monotone weak UDC, which could also occupy this position by themselves or by a descendant is constant. Moreover, the graph distance between the first and the last such vertex is constant.*

Proof. Clearly the number of grid positions inside a circle of radius 2 are constant. Now let q be the grid position with the smallest and q' the grid position with the largest x position such that a spine vertex placed at these positions could still occupy p with a descendant (see Figure 6.30c). Since the weak UDC is strictly x–monotone, every next spine vertex on the path from q to q' must be placed to the right of the previous one. In particular this means that only a constant number of spine vertices can have a distance less or equal to 2 in x-direction to p. Therefore only a constant number of previously placed spine vertices could have possibly been placed a this grid position or placed one

of their descendants on it and clearly q and q' have constant distance, namely at most 8. □

With these three lemmata we obtain the following theorem.

Theorem 6.4. *Using dynamic programming it can be checked in linear time if a (non-embedded) lobster graph admits an x-monotone weak UDC on the triangular grid.*

Proof. The dynamic program starts at the first spine vertex v_1, which is placed at an arbitrary fixed position, and enumerates all possible placements of the descendants of v_1. Due to Lemma 6.19, this can be done in $O(1)$ time. Every single placement yields a pattern of occupied positions on the grid. Note that two placements, which yield the same occupation pattern and place v_1 at the same position relative to that pattern (up to translation and rotation) are indistinguishable, when considering the obstruction they pose for the rest of the weak UDC, due to Lemma 6.20. We save all possible occupation patterns together with the position of the previously placed spine vertex as a record. After processing the first spine vertex, we therefore clearly have a constant number of records.

When the dynamic program moves on from the $(i-1)$-th to the i-th spine vertex v_i it needs to place the new spine vertex at one of only three possible grid positions, due to the x-monotonicity. Every such position p has a constant set of grid positions, which are at most at a grid distance of 2 to p. Note that these grid positions can be occupied by descendants of previous spine vertices, however, due to Lemma 6.21 these spine vertices have at most a constant distance in the graph to v_i. Therefore the dynamic program needs to only remember the exact occupation pattern for a constant number of previously placed spine vertices.

In particular, this means that for all possible occupation patterns (a constant number per spine vertex) of all relevant previous spine vertices (a constant number) we need to save all possible placements (a constant number per spine vertex and occupation pattern).

This results in an overall constant time to process a single spine vertex. □

6.7 Conclusion

We have investigated the existing complexity gap for the recognition problem of UDRs and weak UDCs. In addition to the open problems for various graph classes in different settings (recall Table 6.1 in Section 6.1) – which includes the hardness of the recognition of unit disk graphs and unit disk contact graphs for non-embedded trees – there are two main open questions. First, we have investigated weak UDCs of lobsters on the triangular grid, however, it is not clear if every lobster, which admits a weak UDC, also admits a monotone weak UDC and moreover if it does so on the grid. Second, it seems reasonable to assume that our enumeration approach can be extended to graph classes beyond lobsters, which admit a weak UDC at least on the triangular grid. And finally,

the entire concept of UDRs and UDCs can be lifted and investigated in higher dimensions, while weak UDCs on grids are interesting at least in the 3-dimensional case, as grid-like unit sphere packings exist.

CHAPTER 7

Computing Schematic Curve Arrangements for Nonogram Puzzles

This chapter is (partially) based on the following publications:
[22]: de Nooijer et al. – Removing Popular Faces in Curve Arrangements (GD'23, to appear)
[88]: de Nooijer et al. – Removing Popular Faces in Curve Arrangements (EuroCG'22)

Let $\mathcal{A}$ be a set of curves which lie inside the area bounded by a closed curve, called the *frame*. All curves in $\mathcal{A}$ are either *closed*, or *open* with endpoints on the frame. We refer to $\mathcal{A}$ as a *curve arrangement*, see Figure 7.1a. We consider only *simple* arrangements, where no three curves meet in a point and there are only finitely many total intersections, which are all crossings (no tangencies).

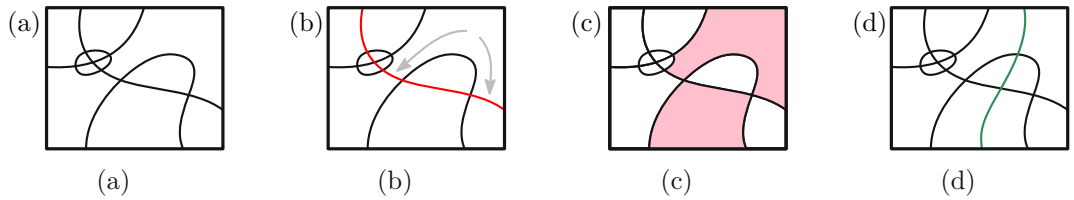
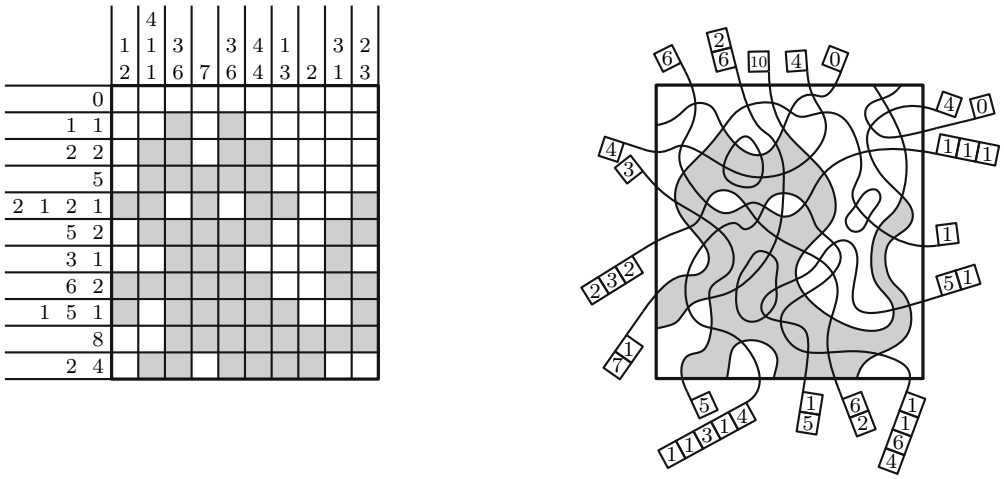


Figure 7.1: (a) A curve arrangement in a rectangular frame. (b) The top right face is incident to two disconnected segments of the red curve, making it *popular*. (c) All popular faces are highlighted. (d) After inserting an additional curve, no more popular faces remain.



(a) (b)

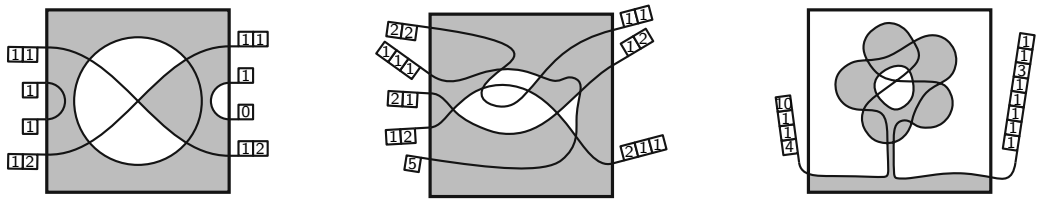
Figure 7.2: Two nonogram puzzles in solved state. (a) A classic nonogram. (b) A curved nonogram.

The arrangement $\mathcal{A}$ can be seen as an embedded multigraph whose vertices are crossings of curves and whose edges are *curve segments*. $\mathcal{A}$ subdivides the region bounded by the frame into *faces*. A face is *popular* when it is incident to multiple curve segments belonging to the same curve in $\mathcal{A}$ (see Figures 7.1b–c). We study the Nonogram 1-Resolution (N1R) problem: can one additional curve ℓ be inserted into $\mathcal{A}$ such that no faces of $\mathcal{A} \cup \{\ell\}$ are popular (see Figure 7.1d)?

7.1 Related Work

Our question is motivated by the problem of generating *curved nonograms*. Nonograms, also known as *Japanese puzzles*, *paint-by-numbers*, or *griddlers*, are a popular puzzle type where one is given an empty grid and a set of *clues* on which grid cells need to be colored. A clue consists of a sequence of numbers specifying the numbers of consecutive filled cells in a row or column. A solved nonogram typically results in a picture (see Figure 7.2a). There is quite some work in the literature on the difficulty of solving nonograms [13, 18, 35].

Van de Kerkhof et al. [70] introduced *curved* nonograms, in which the puzzle is no longer played on a grid but on an arrangement of curves (see Figure 7.2b). In curved nonograms, clues specify numbers of filled faces of the arrangement in the sequence of faces incident to a common curve on one side. Van de Kerkhof et al. focus on heuristics to automatically generate such puzzles from a desired solution picture by extending curve segments to a complete curve arrangement.



(a) (b) (c)

Figure 7.3: Three types of curved nonograms of increasing complexity [70], shown with solutions. (a) *Basic* puzzles have no popular faces. (b) *Advanced* puzzles may have popular faces, but no self-intersections. (c) *Expert* puzzles have self-intersecting curves. We can observe closed curves (without clues) in (a) and (c).

Nonogram complexity.

Van de Kerkhof et al. observed that curved nonograms come in different levels of complexity — not in terms of how hard it is to *solve* a puzzle, but how hard it is to understand the rules (see Figure 7.3). They state that it would be of interest to generate puzzles of a specific complexity level; their generators can currently do this only by trial and error.

- *Basic* nonograms are puzzles in which each clue corresponds to a sequence of distinct faces. The analogy with clues in classic nonograms is straightforward.
- *Advanced* nonograms may have clues that correspond to a sequence of faces in which some faces may appear multiple times because the face is incident to the same curve (on the *same* side) multiple times. When such a face is filled, it is also counted multiple times; in particular, it is no longer true that the sum of the numbers in a clue is equal to the total number of filled faces incident to the curve. This makes the rules harder to understand.
- *Expert* nonograms may have clues in which a single face is incident to the same curve on *both* sides. They are even more confusing than advanced nonograms. Expert nonograms are only suitable for experienced puzzle freaks.

It is easy to see that arrangements with self-intersecting curves correspond exactly to expert puzzles. The difference between basic and advanced puzzles is more subtle; it is exactly the presence of *popular faces* in the arrangement.

One possibility to generate nonograms of a specific complexity would be to take an existing generator and modify the output. Recently, Brunck et al. [29] have investigated how popular faces in a nonogram might be removed by reconfiguring and/or reconnecting parts of curves at small local areas, which they call switches (e.g. around curve crossings), and they have proved that this problem is NP-hard. As an alternative, one may try to get rid of the popular faces by adding extra curves that cut the popular faces into smaller

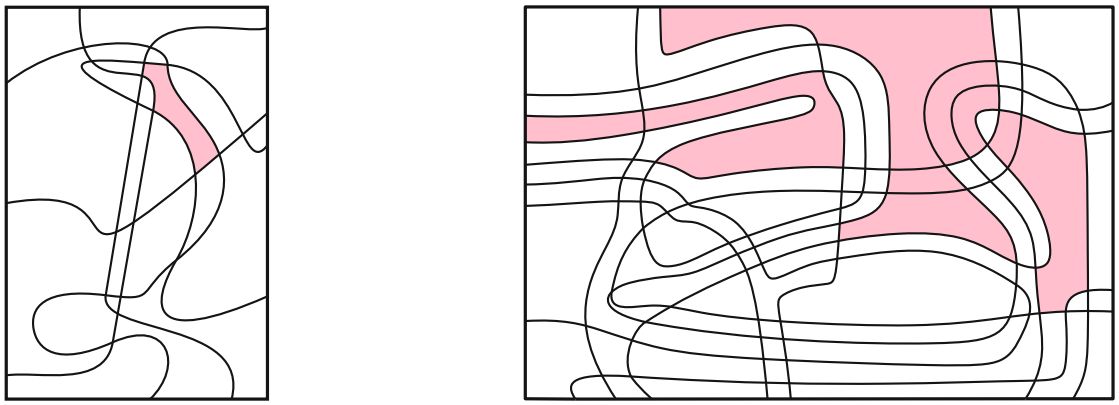


Figure 7.4: Real puzzles (without clues) with all popular faces highlighted.

pieces. In this chapter, we explore what we can do by inserting a single new curve into the arrangement. Clearly, inserting curves will not remove self-intersections, so we focus on changing advanced puzzles into basic puzzles; i.e., removing all popular faces.

7.2 Results

After discussing in Section 7.3 how a singular face is resolved, we show in Section 7.4 that deciding whether we can remove all popular faces from a given curve arrangement by inserting a single curve – which we call the N1R problem – is NP-complete. However, often the number of popular faces is small, see Figure 7.4. Hence, we are also interested in the problem parametrized by the number of popular faces k. we show in Section 7.5 that the problem can be solved by a randomized algorithm in FPT time.

7.3 Resolving one popular face by adding a single curve

As a preparation, we analyze how a single bad face F can be resolved. If F is visited three or more times by some curve, it cannot be resolved with a single additional curve ℓ, and we can immediately abort. Otherwise, there are *popular* edges among the edges of F, which belong to a curve that visits F twice. As a visual aid, we indicate each such pair of edges by connecting them with a red curve (a *curtain*), see Figure 7.5a or 7.8b.

Observation 7.1. *To ensure that a popular face F becomes unpopular after insertion of a single curve ℓ into the arrangement, it is necessary and sufficient that the curve ℓ has the following properties.*

1. *It visits the face F exactly once;*
2. *It does not enter or exit through a popular edge;*
3. *It separates each pair of popular edges. In other words, ℓ cuts all curtains.* □

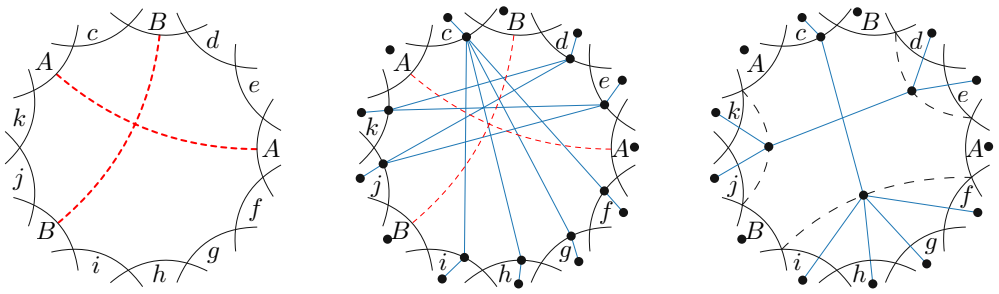


Figure 7.5: Resolving a popular face F. (a) Curtains model duplicate edges. (b) Possible ways how ℓ can pass through F. (c) A more compact representation

The ways how ℓ can traverse a popular face F can be modeled as a graph: We place a vertex on every edge of F except the popular edges. We then connect two such vertices u, v if for every curtain c, the endpoints of c alternate with the vertices u and v around F, as shown in Figure 7.5b. This representation can be condensed as shown in Figure 7.5c and explained in Section 7.5.3.

In our arguments, we often use the dual graph $\mathcal{A}^d$ of a curve arrangement $\mathcal{A}$, where every face of $\mathcal{A}$ is represented by a vertex and edges represent faces which share a common boundary segment (not just a common point). A curve ℓ traversing $\mathcal{A}$ and crossing a sequence of faces $F_1, \ldots, F_k$ in that order can be expressed as a path $P = (F_1, \ldots, F_k)$ in $\mathcal{A}^d$.

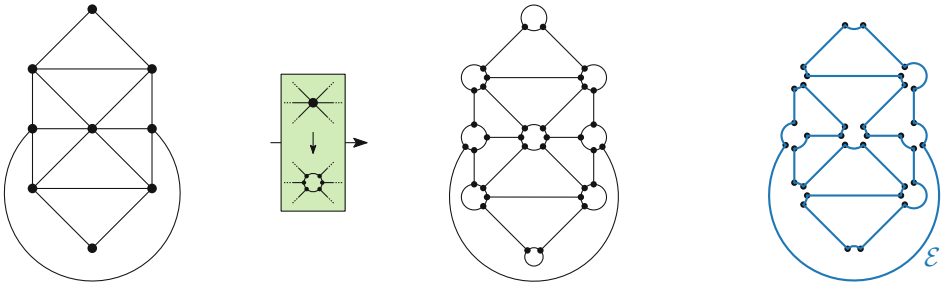
7.4 N1R is NP-complete

In order to prove NP-hardness, we reduce from *Planar Non-intersecting Eulerian Cycle*. This reduction assumes ℓ to be a closed loop, but it can easily be adapted to work for an open curve ℓ' starting and ending at the frame.

7.4.1 Non-intersecting Eulerian cycles

An Eulerian cycle in a graph is a closed walk that contains every edge exactly once. An Eulerian cycle in a graph embedded into the plane (a plane graph) is *non-intersecting* if every pair of consecutive edges $(a, b), (b, c)$ is adjacent in the radial order around b. Intuitively, an Eulerian cycle is non-intersecting if it can be drawn without repeated vertices after replacing each vertex by a small cycle linking the incident edges in circular order (see Figures 7.6a and 7.6b). The Eulerian cycle has to visit all of the original edges, but it does not have to cover the small vertex cycles (see Figure 7.6c). The following problem was proved to be NP-complete by Bent and Manber [17, Theorem 1].

Problem 7.1 (Planar Non-Intersecting Eulerian Cycle PNEC)**.** *Given a planar graph embedded into the plane graph G, decide whether G contains a non-intersecting Eulerian cycle.*



(a) (b) (c)

Figure 7.6: Vertices of the graph G (a) are replaced by cycles (b). A non-intersecting cycle drawn on the modified graph visiting all original edges (c).

7.4.2 NP-completeness reduction

We will present a polynomial-time reduction from PNEC to N1R, i.e., we will create a curve arrangement $\mathcal{A}$ containing popular faces based on a planar input graph G of PNEC, such that there exists a curve ℓ for which $\mathcal{A} \cup \ell$ contains no popular faces if and only if G contains a non-intersecting Eulerian cycle. We assume that G is 2-edge-connected and all vertices have even degree, because otherwise, G clearly cannot contain an Eulerian cycle. We also replace every self-loop with a path of length two, without affecting the existence of a planar non-intersecting Eulerian cycle.

The reduction is gadget based. We will represent every vertex $v \in V$ with a vertex gadget $\mathcal{N}(v)$ and every edge $e = (u, v) \in E$ with an edge gadget $\mathcal{L}(e)$ or $\mathcal{L}(u, v)$. Both gadgets are sets of curves starting and ending at the frame, and $\mathcal{A} = \bigcup_{v \in V} \mathcal{N}(v) \cup \bigcup_{e \in E} \mathcal{L}(e)$.

Vertex gadgets.

The vertex gadgets consist of curves in one of three basic shapes shown in Figure 7.7a, which we call beakers. We place one beaker per incident edge of v, at the position of v, all rotated, such that their *bases* (the lower ends in Figure 7.7a) overlap in a specific pattern. The *opening* of each beaker (the upper ends in Figure 7.7a) will point outwards. We use three variants of the vertex gadget, depending on the vertex degree.

The vertex gadget for a degree-two vertex is simply made up of two overlaying Type-I beakers (see Figure 7.7b). Since ℓ must cross the two curtains c_1 and c_2, it must connect the two points p_1, p_2 by crossing the overlap of the two beakers (a face of degree two, marked in green in Figure 7.7b). Since by Observation 7.1, ℓ can enter any beaker only once, the routing of ℓ as shown in Figure 7.7c is forced and corresponds exactly to the traversal of a planar non-intersecting Eulerian cycle through a vertex of degree two.

The vertex gadget $\mathcal{N}(v)$ for a degree-four vertex v consists of four Type-I beakers, one per incident edge, which form the intersection pattern of Figure 7.7d. Since ℓ must cross the four curtains, it must enter or exit the gadget at least four times through the thick blue edges in Figure 7.7e and the vertices p_1, p_2, p_3, p_4 of the dual graph $\mathcal{A}^d$. Since

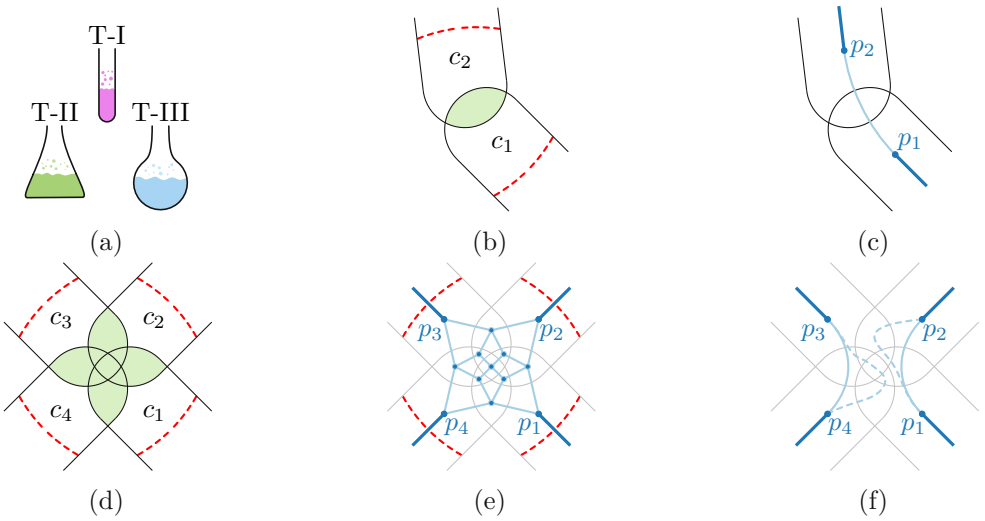


Figure 7.7: (a) Basic beaker curve shapes. (b) Degree two gadget (2 Type-I beakers) and (c) its forced resolution. (d) Degree four gadget (4 Type-I beakers). (e) Dual graph of the degree 4 gadget. (f) A possible curve ℓ in light blue; some alternative routings, which connect the same endpoints, in dashed light blue.

ℓ cannot cross itself, there are only two possibilities how ℓ can pass through $\mathcal{N}(v)$: It can connect p_1 with p_2 and p_3 with p_4, as in Figure 7.7f, or p_1 with p_4 and p_2 with p_3. Both possibilities can be realized by routings of ℓ, and they correspond precisely to the ways how a non-intersecting Eulerian cycle can pass through the edges incident to v. Note that the exact routing of ℓ can vary inside $\mathcal{N}(v)$ (indicated by the dashed lines in Figure 7.7).

The vertex gadget for a vertex v of degree $d \geq 6$ is more complex. We place $d-1$ Type-II beakers $c_1, \ldots, c_{d-1}$ symmetrically around the location of v (Figure 7.8a). Each beaker intersects four adjacent beakers (two on each side), with the exception that $c_{d/2-1}$ and $c_{d/2+1}$ (dark green curves in Figure 7.8a) do not intersect. We place an additional Type-III beaker c_d (the light green curve in Figure 7.8a) that surrounds all bases of the Type-II beakers and protrudes between c_{d-1} and c_1, such that the intersection pattern of Figure 7.8a arises.

All popular faces and curtains in $\mathcal{N}(v)$ are shown in Figure 7.8b. The dual of the construction is shown in Figure 7.8c. The curtains in the green faces force ℓ to pass from these faces to the adjacent small faces with the blue boundaries. This constrains ℓ to pass through a chain of faces as shown in Figure 7.8d. The passages from these faces to other neighboring faces can now be excluded, and the corresponding edges have been removed from the dual graph in Figure 7.8d.

The curtains in the openings of the beakers force the outer blue endpoints of ℓ. The endpoint in beaker c_i will be called p_i. Now we analyze which of these endpoints can be connected with each other. We see that in most cases, p_i can only be connected to p_{i-1} or p_{i+1} without going through another endpoint. The exception is $p_{d/2-1}$ and $p_{d/2+1}$,

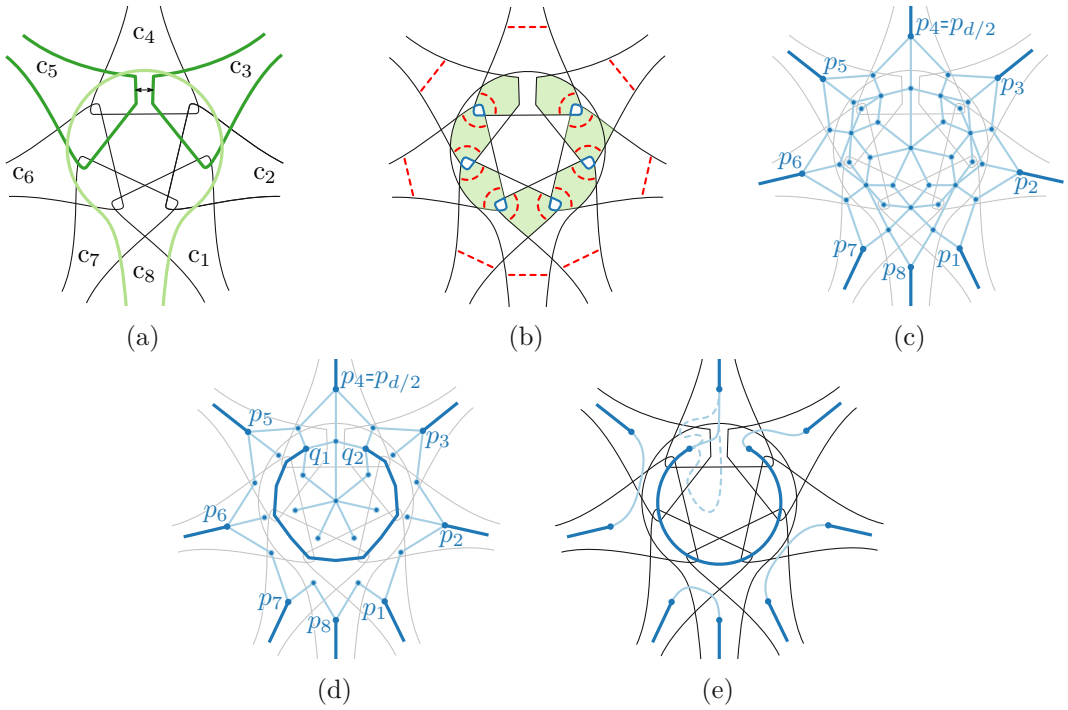


Figure 7.8: (a) Vertex gadget $\mathcal{N}(v)$ for a degree-8 vertex v, (b) its curtains and (c) dual graph. (d) Highlighted light green faces in (b) force the dark blue connections in the dual graph and restrict it. (e) One of two symmetric possibilities for the splitting curve ℓ. The dashed lines show different possible routings of ℓ.

which can be connected via the inner loop from q_1 to q_2. However, this connection would cut off $p_{d/2}$ from the remaining points. We conclude that the visits of ℓ to $\mathcal{N}(u)$ must match endpoints p_i that are adjacent in the circular order. There are two matchings, which correspond to the two possibilities how a non-intersecting Eulerian cycle can visit v. Both possibilities can be realized by routings of ℓ; one is shown in Figure 7.8e, and the other is symmetric.

We now have placed vertex gadgets for all vertices. They require ℓ to connect to an endpoint in each opening of a beaker. With these openings, we will now construct the edge gadgets.

Edge gadgets.

Let $e = (u, v) \in E$ be an edge in G. Then there are two vertex gadgets $\mathcal{N}(u)$ and $\mathcal{N}(v)$ already placed. In particular, we placed one beaker in the gadgets per incident edge at u or v, i.e. two per edge.

We now elongate the open ends of these beakers and route them along the edge e according to the embedding of G given in the input (recall that G is a plane graph) until they

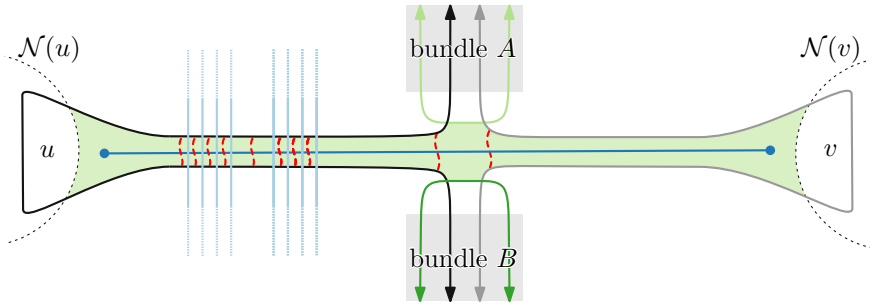


Figure 7.9: Edge gadget $\mathcal{L}(u, v)$ connecting two beakers from $\mathcal{N}(u)$ and $\mathcal{N}(v)$ with two additional curves. Open ends of all curves are collected into two bundles of parallel curves that lead into the incident faces. The inside of the two beakers are connected via a chain of popular faces in $\mathcal{L}(u, v)$ (shaded in light green). Other bundles, like the two groups of four light blue curves in the left half, can freely cross either beaker.

almost meet at the center point of e. We bend the ends of each beaker outward, routing them into the two faces incident to e. Additionally, we place two more curves on top (shown in green), forming the intersection pattern of Figure 7.9. This results in two *bundles* A and B, each consisting of four parallel curves. (The light blue curves in the left half are not part of the gadget; they are two bundles that come from other gadgets.)

This connects a popular face in $\mathcal{N}(u)$ to one in $\mathcal{N}(v)$, forming one big popular face in $\mathcal{L}(u, v)$. An arbitrary number of curves may cross the opening of a beaker. The face is then simply divided into a chain of consecutive popular faces. In each of these faces, except the left- and right-most faces, which contain the blue endpoints, ℓ has to leave through two specific edges in order to cut the curtains. This forces ℓ to pass straight through $\mathcal{L}(u, v)$ from $\mathcal{N}(u)$ to $\mathcal{N}(v)$ along the thin dark-blue horizontal axis.

It remains to describe how the open ends of the curves in the bundles are routed to the frame (since all curves other than ℓ have start and end at the frame). This is not difficult because these bundles can cross quite freely without creating popular faces. Each bundle consists of a unique set of curves, except for the two bundles from one edge gadget. A bundle that originates from the edge gadget $\mathcal{L}(e)$ can thus cross any bundle from a different edge gadget without creating popular faces. It can cross a different edge gadget $\mathcal{L}(e')$ by passing over one of its beakers, as shown with the light-blue curves.

Since G does not contain self-loops, we route each bundle along a path in the dual of G to the outer face of G, and then connect it to the frame, see Figure 7.10. A popular face might only be created when a bundle crosses the other bundle from the same edge gadget. In this case, we reroute the bundle in parallel to the second bundle until it hits the frame.

The curves in a bundle run in parallel. Two bundles originating from an edge gadget $\mathcal{L}(e)$ have different curves as their outside curves. Hence, no popular faces are created between two bundles, and we can make the following statement.

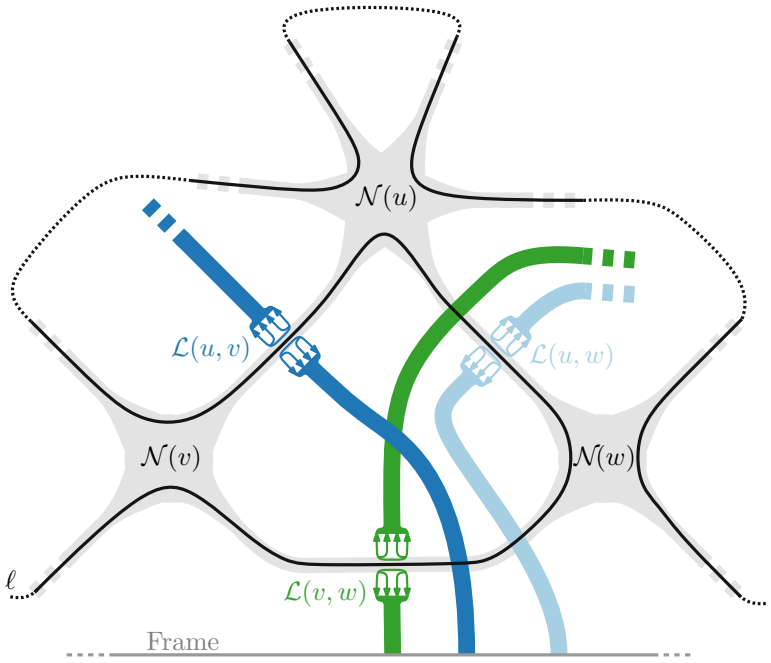


Figure 7.10: Schematic representation of three vertex and edge gadgets. The bundles are routed through beakers of other edges ending at the frame (partially shown at the bottom of the figure). A possible routing of ℓ is shown with a black curve.

Observation 7.2. *All popular faces in $\mathcal{A}$ are contained in vertex gadgets and edge gadgets (the faces with dashed red curtains in Figures 7.7b, 7.7d, 7.8b, and 7.9).*

The next theorem follows from the construction and the resulting correspondence between resolving curves and non-crossing Eulerian cycles.

Theorem 7.1. *N1R is* NP*-complete.*

Proof. Assume we are given a non-intersecting Eulerian cycle $\mathcal{E}(G)$ of G as a permutation of the edges. We now show how to construct the curve ℓ. We choose a random edge $e = (u, v)$ in the permutation and start drawing ℓ at the endpoint inside the beaker of $\mathcal{L}(u, v)$ originating from $\mathcal{N}(v)$ along the thin blue axis crossing all popular faces as described above.This will end at an open endpoint inside $\mathcal{N}(u)$. At this point we either connect to the left or right endpoint according to the next edge in $\mathcal{E}(G)$ (which is possible, since $\mathcal{E}(G)$ is non-intersecting). We make this connection either directly or via the forced inner circular part of ℓ in $\mathcal{N}(u)$ if one of the two involved endpoints is in beaker $c_{d(u)/2}$ (where $d(u)$ is the degree of u). These connections are made as described above.Now we are again at an endpoint in a beaker. We can repeat this procedure until we cycle back to e, at which point we will have reconnected to our starting point. Since $\mathcal{E}(G)$ by definition visits every edge exactly once, before cycling back to the first edge, we know

that every popular face in the edge gadgets is split. Moreover, since $\mathcal{E}(G)$ visits every vertex v exactly $\delta(v)/2$ times and $\mathcal{E}(G)$ is non-intersecting, we know that one of the two possible ways of connecting all endpoints in $\mathcal{N}(v)$ can be chosen to resolve all popular faces in $\mathcal{N}(v)$ (and all other vertex gadgets). Since by Observation 7.2 there are no other popular faces in $\mathcal{A}$, $\mathcal{A} \cup \ell$ does not contain any popular faces.

Now assume we are given a curve ℓ, such that $\mathcal{A} \cup \ell$ does not contain any popular faces. The order, in which ℓ traverses all edge gadgets gives us a permutation of all edges. Since both variants of resolving all popular faces in a vertex gadget connect any endpoint only to an endpoint in a neighboring beaker, consecutive edges in this permutation share an endpoint and both belong to the same face and this permutation is a non-intersecting Eulerian cycle. We have shown that there exists a curve ℓ, such that, $\mathcal{A} \cup \ell$ contains no popular faces, if and only if G contains a non-intersecting Eulerian cycle and therefore N1R is NP-hard.

It is easy to see that N1R is in NP. The input arrangement can be represented as a plane graph in which the edges are marked as belonging to the different curves or to the frame boundary. The certificate is an extension of this arrangement by a resolving curve ℓ. Since ℓ cannot visit a face more than once, the certificate is of polynomial size. It can be easily verified in polynomial time whether it is valid, in particular, whether it contains no popular faces.

□

7.4.3 Adaption to open curves.

The reduction assumes that ℓ is a closed loop. It can be adapted to work for open curves, i.e., we can create the arrangement $\mathcal{A}$, for which there exists an open curve ℓ' starting and ending at the frame, such that $\mathcal{A} \cup \ell'$ does not contain any popular faces, if and only if G contains a planar non-intersecting Eulerian cycle $\mathcal{E}(G)$.

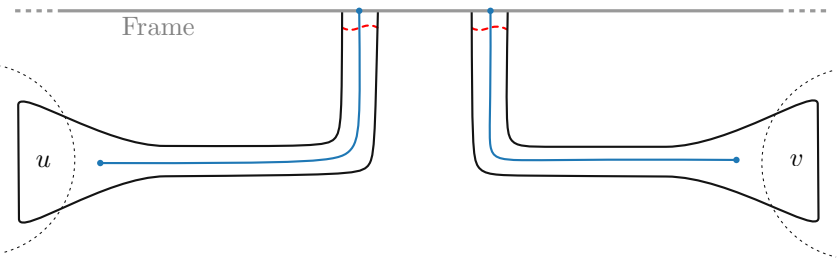


Figure 7.11: By routing both ends of an open beaker in parallel to the frame, we force ℓ to start (or end) at the frame between the two connection points of the beaker.

The reduction creates $\mathcal{A}$ in the same fashion as above, except that we do not add the edge gadget for exactly one edge $e_o = (v_o, u_o)$ on the outer face of G. Instead, the open curves of the openings of the beakers in $\mathcal{N}(u_o)$ and $\mathcal{N}(v_o)$, which would normally form the edge gadget $\mathcal{L}(u_o, v_o)$ are simply connected to the frame. This forces ℓ' to start (and end) at the frame between the points at which the beakers connect to the frame, in order

to properly split the popular face in the opening of these beakers. It is now easy to see that all other properties still hold and N1R remains NP-complete even when ℓ' can be an open curve.

7.5 Randomized FPT-algorithm for N1R

In this section, we show that N1R with k popular faces can be solved by a randomized algorithm in $O\left(2^k \text{poly}(n)\right)$ time, placing N1R in the class randomized FPT when parameterized by the number k of popular faces. We model N1R as a problem of finding a simple cycle (i.e., a cycle without repeated vertices) in a modified dual graph G, subject to a constraint that certain edges must be visited.

Problem 7.2 (Simple Cycle with Edge Set Constraints SNESC)**.** *Given an undirected graph $G = (V, E)$ and k subsets $S_1, S_2, \ldots, S_k \subseteq E$ of edges, find a simple cycle, if it exists, that contains exactly one edge from each set S_i.*

We start with the dual graph of the given curve arrangement $\mathcal{A}$. We replace the vertex corresponding to the i-th popular face f with a set S_i of edges modeling the ways how an additional curve can cut all curtains of f, as described in Section 7.3 and shown in Figure 7.5b. To be specific, we place a vertex on each curve segment s bounding f and connect it to the vertex of the face that is adjacent to f across s. Further we connect two such vertices on curve segments if a curve entering f through one segment and exiting through the other would cut all curtains of f. The latter connecting edges, which run through f, form the set S_i. There is a one-to-one correspondence between the simple cycles containing exactly one edge of every set S_i and the resolving curves for $\mathcal{A}$.

We will describe a randomized algorithm for Problem 7.2, extending an algorithm of Björklund, Husfeld, and Taslaman [24]. We first state the original result.

Theorem 7.2 (Theorem 1 [24])**.** *A shortest simple cycle through k given vertices or edges in an undirected n-vertex graph can be found by a randomized algorithm in time $2^k n^{O}(1)$ with one-sided error of exponentially small probability in n.*

The adapted result following from our algorithm is as follows. The main distinction is that we do not look for a cycle through a set of vertices or edges, but instead we have multiple sets of edges, of which we have to visit exactly one edge each.

Theorem 7.3. *The SNESC problem on a graph with n vertices and $m \geq n$ edges can be solved in $O\left(2^k m n^2 \log \frac{2m}{n} \cdot |V(S_1)| \cdot W\right)$ time and $O(2^k n + m)$ space with a randomized Monte-Carlo algorithm, with probability at least $1 - 1/n^W$, for any $W \geq 1$. Here, $V(S_1)$ denotes the set of vertices of the edges in S_1 (note that S_1 can be chosen to be the smallest*

set among $S_1, \ldots, S_k$). The model of computation is the Word-RAM[1] *with words of size* $\Theta(k + \log n)$.

The algorithm finds the cycle with the smallest number of edges if it exists (with high probability).

If $\mathcal{A}$ has n faces, the graph G has $O(n)$ vertices and $m = O(n^2)$ edges. The quadratic blow-up of m results from the construction as shown in Figure 7.5b. The number of edges can be reduced to $O(n)$, as shown in Figure 7.5c and discussed in Section 7.5.3.

The number k of popular faces is the same as the number k of edge sets S_i. With the alternative algorithm with polynomial space, since $k \leq n$, we get:

Corollary 7.1. *The N1R problem with k popular faces in a curve arrangement with n faces can be solved in expected time $O(2^k \text{poly}(n))$ and $O(kn)$ space.* □

We first give a high-level overview of the algorithm. We start by assigning random weights to the edges from a sufficiently large finite field $\mathbb{F}_q$ of characteristic 2. Such a field exists for every size q that is a power of 2. In a field of characteristic 2, the law $x + x = 0$ holds, and therefore terms cancel when they occur an even number of times. The weight of a *walk* (with vertex and edge repetitions allowed) is obtained by multiplying the edge weights of all visited edges. Our goal is now to compute the sum of weights all closed walks, of given length, that satisfy the edge set constraints. The characteristic-2 property will ensure that the unwanted walks, those which are not simple, cancel, while a simple closed walk makes a nonzero contribution and leads to a nonzero sum with high property. The crucial idea is that, while these sets of closed walks can be very complicated, we can compute the aggregated sum of their weights in polynomial time. We have to anchor these walks at some starting vertex b, and we choose b to be one of the vertices incident to an edge of S_1.

More precisely, for each such vertex b, and for increasing lengths $l = 1, 2, \ldots, n$, the algorithm computes the quantity $\hat{T}_b(l)$, which is the sum of the weights of all closed walks that

- start and end at b,
- have their first edge in S_1,
- use exactly one edge from each set S_i (and use it only once),
- and consist of l edges.

We consider the edge weights as variables and regard $\hat{T}_b(l)$ as a function of these variables. The result is a polynomial where each term is a product of l variables (possibly with repetition), and hence the polynomial has degree l, unless all terms cancel and it is the zero polynomial. We apply the following lemma, which is a straightforward adaptation of a lemma of Björklund et al. [24].

[1] A model of computation in which every memory cell contains a single word of bounded finite size w, which can be accessed in constant time.

Lemma 7.1. *(a) Suppose there exists a simple cycle of length l that satisfies the edge set constraints and that goes through an edge of S_1 incident to b. Then the polynomial $\hat{T}_b(l)$ is homogeneous of degree l and is not identically zero.*

(b) If there is no such cycle of length $\leq l$, the polynomial $\hat{T}_b(l)$ is identically zero.

Proof. (a) By assumption, there is a simple cycle among the walks whose weights are collected in $\hat{T}_b(l)$, and we easily see that the monomial corresponding to such a walk occurs with coefficient 1. Hence $\hat{T}_b(l)$ is not identically zero. By definition, $\hat{T}_b(l)$ is a sum of weights of walks of length l, and hence it is clear that it is homogeneous of degree l.

(b) For the second statement of the lemma, we have to show that $\hat{T}_b(l)$ is zero if there is no simple walk of length l or shorter. Since the field has characteristic 2, it suffices to establish a matching among those closed walks that satisfy the edge set constraints but don't represent simple cycles. Let W be such a walk. We will map W to another walk $\phi(W)$ that uses the same multiset of edges, by reversing (flipping) the order of the edges of a subpath between two visits to the same vertex v. Our procedure closely follows Björklund et al. [24], but we correct an error in their description.

$$\begin{aligned} W = W_0 &= 12341565143234546178657 1 = 1[23415651432]345461786571 \\ W_0' = W_1 &= 12345461786571 = 123[454]61786571 \\ W_1' = W_2 &= 123461786571 = 1234[61786]571 \end{aligned}$$

$$\begin{aligned} W &= 1234156514323454[61786]571 \\ \phi(W) &= 1234156514323454[68716]571 \end{aligned}$$

Figure 7.12: Mapping a nonsimple cycle W to another cycle $\phi(W)$. The start vertex is $b = 1$.

An example of the procedure is shown in Figure 7.12. We look for the first vertex v that occurs several times on the walk. (During this whole procedure, the occurrence of b at the start of the walk is never considered.) We look at the piece $[v \dots v]$ between the first and last occurrence of v and flip it. If the sequence of vertices in this piece is not a palindrome, we are done. Otherwise, we cut out this piece from the walk. The resulting walk will still visit an edge from each S_i because such an edge cannot be part of a palindrome, since it is visited only once. Since the resulting walk W' is shorter than l, and thus shorter than L, by assumption, it cannot be a simple cycle, and it must contain repeated vertices.

We proceed with W' instead of W. Eventually we must find a piece $[v \dots v]$ that is not a palindrome. We flip it in the original walk W, and the result is the walk $\phi(W)$ to which W is matched. (The flipped piece $[v \dots v]$ does not necessarily start at the first

occurrence of v in the original sequence W, because such an occurrence might have been eliminated as part of a palindrome.[2])

It is important to note that after cutting out a palindrome $[v \dots v]$, all repeated vertices must come *after* the vertex v. Hence, when the procedure is applied to $\phi(W)$, it will perform exactly the same sequence of operations until the last step, where it will flip $\phi(W)$ back to W.

Since the first edge of the walk is unchanged, the condition that this edge must belong to S_1 is left intact. This concludes the proof of Lemma 7.1. □

The lemma is based on the fact that each term in the polynomial $\hat{T}_b(l)$ represents some closed walk. A term coming from a walk that visits a vertex twice can be matched with another walk, which traverses a loop in the opposite direction and contributes the same term. Since the field has characteristic 2, these terms cancel. A term coming from a simple walk does not cancel.

In case (a) of Lemma 7.1, it follows from the Schwartz-Zippel Lemma [113, Corollary 1] that, for randomly chosen weights in $\mathbb{F}_q$, $\hat{T}_b(l)$ is nonzero with probability at least $1 - \text{degree}/|\mathbb{F}_q| = 1 - l/q \geq 1 - n/q$.

Thus, if we choose $q > n^2$, we have a success probability of at least $1 - 1/n$ for finding the shortest cycle when we evaluate the quantities $\hat{T}_b(l)$ for increasing l until they become nonzero. The success probability can be boosted by repeating the experiment with new random weights.

In the unlikely case of a failure, the algorithm may err by not finding a solution although a solution exists, or by finding a solution that is not shortest. The last possibility is not an issue for our original problem, where we just ask about the existence of a cycle, of arbitrary length.

In the following, we will discuss how we can compute the quantities $\hat{T}_b(l)$, and the runtime and space requirement for this calculation. We describe the method for actually recovering the cycle after we have found a nonzero value in Section 7.5.5.

7.5.1 Computing sums of path weights by dynamic programming

We cannot compute the desired sums $\hat{T}_b(l)$ directly, but have to do this incrementally via a larger variety of quantities $T_b(R, l, v)$ that are defined as follows:

For $R \subseteq \{1, 2, \dots, k\}$ with $1 \in R$, $v \in V$, and $l \geq 1$, we define $T_b(R, l, v)$ as the sum of the weights of all walks that

- start at b,

[2]Here our procedure differs from the method described in [24], where the flipped subsequence extends between the first and last occurrence of v in W. In this form, the mapping ϕ is not an involution. The proof in [24], however, applies to the method as described here. This inconsistency was confirmed by the authors (Nina Taslaman, private communication, February 2021).

- have their first edge in S_1,
- end at v,
- consist of l edges,
- use exactly one edge from each set S_i with $i \in R$ (and use it only once),
- contain no edge from the sets S_i with $i \notin R$.

The walks that we consider here differ from the walks in $\hat{T}_b(l)$ in two respects: They end at a specified vertex v, and the set R keeps track of the sets S_i that were already visited. The quantities $\hat{T}_b(l)$ that we are interested in arise as a special case when we have visited the full range $R = \{1, \ldots, k\}$ of sets S_i and arrive at $v = b$:

$$\hat{T}_b(l) = T_b(\{1, \ldots, k\}, l, b).$$

We compute the values $T_b(R, l, v)$ for increasing values $l = 1, \ldots, n$. The starting values for $l = 1$ are straightforward from the definition.

To compute $T_b(R, l, v)$ for $l \geq 2$, we collect all stored values of the form $T_b(R', l-1, u)$ where (u, v) is an edge of G and R' is derived from R by taking into account the sets S_i to which (u, v) belongs. We multiply these values with the edge weight w_{uv} and sum them up. If (u, v) is in some S_i but $i \notin R$, we don't use this edge. Formally, let $I(u, v) := \{\, i \mid (u, v) \in S_i \,\}$ be the index set of the sets S_i to which (u, v) belongs. Then

$$T_b(R, l, v) = \sum_{\substack{(u,v)\in E \\ I(u,v)\subseteq R}} w_{uv} \cdot T_b(R \setminus I(u, v), l-1, u) \tag{7.1}$$

7.5.2 Finite Field Computations

Each evaluation of the recursion (7.1) involves additions and multiplications in the finite field $\mathbb{F}_q$, where $q = 2^s > n$ is a power of 2.

We will argue that it is justified to regard the time for these arithmetic operations as constant, both in theory and in practice. The error probability can be controlled by choosing the finite field sufficiently large, or by repeating the algorithm with new random weights.

For implementing the algorithm in practice, arithmetic in $\mathbb{F}_{2^s}$ is supported by a variety of powerful libraries. see for example [106]. The size that these libraries conveniently offer (e.g., $q = 2^{32}$) will be mostly sufficient for a satisfactory success probability.

For the theoretical analysis, we propose to choose a finite field of size $q \geq n^2$, leading to a failure property less than $1/n$, and to achieve further reductions of the failure property by repeating the algorithm W times.

We will describe some elementary approach for setting up the finite field $\mathbb{F}_q$ of size $q \geq n^2$, considering that we can tolerate a runtime and space requirement that is linear or a small polynomial in n. Various methods for arithmetic in finite fields $\mathbb{F}_{2^s}$ are surveyed in Luo, Bowers, Oprea, and Xu [81] or Plank, Greenan, and Miller [105]. The natural way to

represent elements of $\mathbb{F}_{2^s}$ is as a polynomial modulo some fixed irreducible polynomial $p(x)$ of degree s over $\mathbb{F}_2$. The coefficients of the polynomial form a bit string of s bits. Addition is simply an XOR of these bit strings.

We propose to use the split table method, which is simple and implements multiplication by a few look-ups in small precomputed tables, see [81, Section 3.3.1] or [105, Section 6.5].

Let $C = \lceil (\log_2 n)/2 \rceil$, and let $q = 2^{4C}$. Thus, $q \geq n^2$, as required.

According to [115, Theorem 20.2], an irreducible polynomial $p(x)$ of degree $d = 4C$ over $\mathbb{F}_2$ can be found in $O(d^4) = O(\log^4 n)$ expected time, by testing random polynomials of degree d for irreducibility. (Shoup [115] points out that this bound is not tight; there are also faster methods.)

Multiplication of two polynomials modulo $p(x)$ can be carried out in the straightforward "high-school" way in $O(d^2) = O(\log^2 n)$ steps, by elementwise multiplication of the two polynomials, and reducing the product by successive elimination of terms of degree larger than d.

We now consider the bitstring of length d as composed of $r = 4$ chunks of size C. In other words, we write the polynomial $q(x)$ as

$$q(x) = q_3(x)x^{3C} + q_2(x)x^{2C} + q_1(x)x^C + q_0(x),$$

where q_3, q_2, q_1, q_0 are polynomials of degree less than C. Addition takes $O(r)$ time, assuming an XOR on words of length $C = O(\log n)$ can be carried out in constant time. Multiplication is carried out chunk-wise, using $2r - 1 = 7$ multiplication tables. The j-th table contains the products $r(x)r'(x)x^{jC}$, for all pairs $r(x), r'(x)$ of polynomials of degree less than C, for $j = 0, 1, \ldots, 2r - 2$. Multiplication in the straightforward way takes then $O(r^3)$ time. Each multiplication table has $2^C \times 2^C = O(n)$ entries of $r = 4$ words, and it can be precomputed in $O(nd^2) = O(n \log^2 n)$ time by multiplying in the straightforward way.

Thus, after some initial overhead of $O(n \log^2 n)$ time, which is negligible in the context of the overall algorithm, with $O(n)$ space, arithmetic in $\mathbb{F}_q$ can be carried out in $O(1)$ time in the Word-RAM model, where arithmetic, logic, and addressing operations on words with $\Theta(\log n)$ bits are considered as constant-time operations.

7.5.3 Adapting the Edge Set Constraints

Here we give more details of this construction.Figure 7.5c. We cut off each run of consecutive edges, like *fghi*, by an additional edge (shown dotted in Figure 7.5c) and place a single terminal node there.

One has to take care that the cycle that is found does not use two such terminal edges in succession, like the edges crossing f and h, because such a cycle would not correspond to a valid curve. This constraint must be added to the problem definition, and the recursion (7.1) must be modified accordingly.

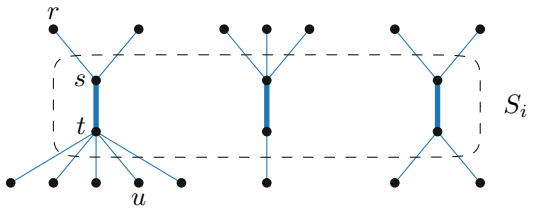


Figure 7.13: A set S_i and its connecting edges

The sets S_i have a special structure. Each set S_i consists of a few vertex-disjoint edges (the thick edges in Figure 7.13), which we call *central edges*. We call the edges connecting the central edges to the remainder of the graph the *peripheral edges*. We don't want to consider cycles that use two such peripheral edges (of the same S_i) in succession, without going through the central edge. We impose this as an extra condition on the walks whose weights we accumulate.

It is easy to incorporate this condition in the recursion of Section 7.5.1: Instead of the quantities $T_b(R, l, v)$, we work with quantities $\tilde{T}_b(R, l, v, p)$ that depend on an additional parameter p. This is one bit that tells whether the last edge of the walk has traversed a peripheral edge in the direction towards the central edge. If this is the case, we force the walk to use the central edge in the next step.

In this way, the additional condition incurs a blow-up of at most a factor 2 in the size of the dynamic programming tables and in the runtime.

We now argue that Lemma 7.1 still holds for these modified quantities. The proof of Lemma 7.1 goes through for the following reason. The conditions on allowed walks ensure that an endpoint of a central edge, like the vertex s in Figure 7.13), is always part of a subpath consisting of a central edge surrounded by two peripheral edges, like (r, s, t, u) or (u, t, s, r). Such a subpath cannot be part of a palindrome, since (s, t) belongs to a special set S_i, and for the same reason, s can never be a repeated vertex of a walk. If the bijection constructed in the proof reverses a subpath that goes through the vertex s, this is no problem because the reversed traversal does not violate the extra condition.

7.5.4 Runtime and space

The finite field additions and multiplications in (7.1) take constant time, as shown in the previous section. Similarly, the set operation $R \setminus I(u, v)$ on subsets of $\{1, \ldots, k\}$ and the test $I(u, v) \subseteq R$ can be carried out in constant time, using bit vectors. Thus, for a fixed starting vertex $b \in V(S_1)$ and fixed R, going from $l - 1$ to l by the recursion (7.1) takes $O(m)$ time in total, because each edge (u, v) appears in at most one of the sums on the right-hand side. The overall runtime is $O(|V(S_1)|2^k mn)$.

As mentioned after Lemma 7.1, the probability that the algorithm misses the shortest simple path is at most $1/n$. To amplify the probability of correctness, we repeat the computation W times, reducing the failure probability to $1/n^W$.

We consider each starting vertex b separately, and do not need to store entries for lengths $l-1$ or shorter when proceeding from l to $l+1$; thus the space requirement is $O(2^k n)$.

7.5.5 Recovering the solution

The algorithm, as described so far, works as an oracle that only gives a yes-no answer (and a length l), but it does not produce the solution. To recover the solution, we will call the oracle repeatedly with different inputs.

Suppose the algorithm was successful in the sense that some number $\hat{T}_b(l)$ turned out to be nonzero, after finding only zero values for all smaller values of l. By Lemma 7.1, we conclude that there exists a simple cycle through b satisfying the edge set constraints, possibly (with small probability) shorter than l.

We will find this cycle by selectively deleting parts of the edges and recomputing $\hat{T}_b(1), \hat{T}_b(2), \ldots, \hat{T}_b(l)$ for the reduced graph to see whether this graph still contains a solution. In this way, we will determine the successive edges of the cycle, and, as we shall see, we will know the cycle after at most $4n \log_2 \frac{4m}{n}$ iterations.

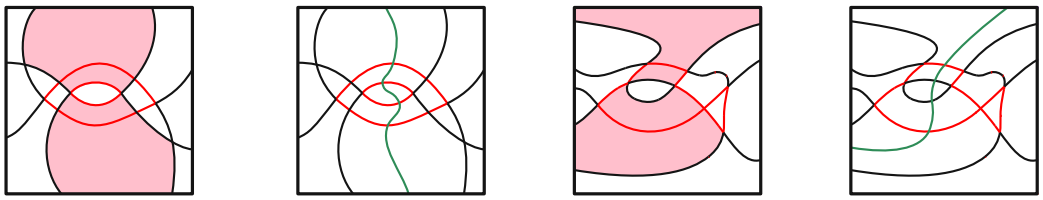
The first edge out of b is an edge of S_1, and thus we start by looking for the first edge among these edges. (The other edge incident to b, by which we eventually return to b, is not in S_1, and thus there is no confusion between the two edges incident to b.) In the general step, we have determined an initial part of the cycle up to some vertex u, and we locate the outgoing edge among the edges (u, v) incident to u, excluding the edge leading to u that we have already used.

In general, for a vertex u of degree d_u, we have a set of $d_u - 1$ potential edges. We locate the correct edge by binary search: We split the potential edges into two equal parts, and query whether a cycle still exists when one or the other part is removed. It may turn out (with small probability) that none of the two subproblems yields a positive answer. In this case, we repeat the oracle with new random weights. Since the success probability is greater than $1/2$, we are guaranteed to have a positive answer after at most two trials, in expectation. (We mention that such repeated trials may be necessary only when the length l for which we are looking is not the shortest length of a feasible cycle. Otherwise one can show, using arguments from the proof of Lemma 7.1,that the polynomial for the original problem is the sum of the polynomials for the two subproblems. Thus, at least one of the two subproblems must give a positive answer.) After at most $2\lceil \log_2(d_u - 1) \rceil$ successful queries, we have narrowed down the search to a single outgoing edge uv, and we continue at the next vertex v.

For a walk W of length l, we use, in expectation, less than $Q := \sum_{u \in W} 4(1 + \log_2 d_u)$ queries, where $\sum_{u \in W} d_u \leq 2m$. Q can be bounded by

$$Q \leq 4l \left(1 + \log_2 \tfrac{2m}{l}\right) \leq 4n \left(1 + \log_2 \tfrac{2m}{n}\right) = 4n \log_2 \tfrac{4m}{n},$$

as claimed.



(a) (b) (c) (d)

Figure 7.14: Input (a,c) and resulting output (b,d) generated by the implementation; the green curve is the curve with the smallest number of crossings that resolves the popular faces.

During this procedure, it may also turn out that a solution with fewer than l edges exists. In this case, we know that we must have been in the unlikely case that the original algorithm failed to produce a nonzero value for the shortest solution l. We simply adjust l to the smaller value and continue.

Note that this procedure is guaranteed to produce a simple cycle, although not necessarily the shortest one. The algorithm selects a branch only when the corresponding polynomial is nonzero, which implies that a simple cycle exists in that branch.

Some simplifications are possible. For edges that are known to belong to every solution (for example if some set S_i contains only one edge), we can assign unit weight, thus saving random bits, reducing the degree of the polynomial, and increasing the success probability. Edges that would close a loop can be discarded. When the graph is sparse and $m = O(n)$, we can simply try the $d_u - 1$ edges one at a time instead of performing binary search, at no cost in terms of the asymptotic runtime.

Figure 7.14 shows initial results of an implementation of our algorithm on two small test instances; see also [99].

7.6 Conclusion

In light of our NP-hardness and randomized FPT-algorithm, a natural next step is a deterministic parameterized algorithm. There are $O(n)$ local possibilities of resolving a single popular face, however, this does not immediately lead to an $O(n^k)$ algorithm (which would place N1R in XP), since we might need to branch additionally over all possible connections between these solutions through the dual of $\mathcal{A}$, which can have an unbounded size.

Additionally there are multiple interesting question which are natural extensions of our result. First, it would be interesting to investigate an optimization version of N1R, where one tries to resolve as many popular faces as possible. Second, we would be interested in proving an upper bound on the number of resolution curves which are needed at most (relative to the number of curves in the input). And third, we hope to extend the algorithm to a setting in which more than one resolution curve is used.

CHAPTER 8

Conclusion

In this thesis, we have investigated five different problems involving schematization. Since these problems were quite diverse, some open problems specific to the topic of each chapter have already been given at the end of the respective chapter. Here we aim to give a short summary of the topics discussed in this thesis and some overarching themes.

Two chapters in the first half of this thesis have tackled the topic of schematization of transit networks, which emphasize connectivity within the network and graph-theoretical distance between stations over geographical accuracy and relative distances. Our work aims to broaden the possible uses of such maps, by providing an automated method of including user defined shapes into these maps, a task that is often used for advertisement or design-purposes. We have also provided an automated method for a generalization of the traditional design of such maps, by allowing (ir)-regular k-linear designs. These chapters exists within the context of a rich literature, which provides similar extensions to the classical metromap design problem. The clear direction within this field of reserach is the unification of existing approaches and results into working and usable implementations, that can be used by designers. While recent work [28] has provided the combination of various tasks into one toolchain, the simultaneous consideration of station placement, edge routing, station labeling and inclusion of tariff zones should be one of the next steps.

In the second half of this thesis we have focused on more general problems involving schematic representations. This included the computation of schematic separators between polygonal elements, the schematic representations of graphs using geometric primitives, specifically unit disks and the more combinatorial problem of preventing undesirable properties in puzzles played on a schematic representation of an image in the form of a curve arrangement. While all three of the presented chapters have interesting further questions, there is an aspect of schematization, which can be of interest on a general level to such geometric problems. Consider the set of restricted directions, which are available for the schematic transit maps. This restriction of possible directions can often be imposed on geometric problems involving polygonal domains. One example from

literature is the result of Hershberger and Snoeyink [63], which we used in Chapter 5 as a subroutine. The authors here consider their problem of computing minimum length paths of a given homotopy class also under the restriction, that every piecewise linear part of such a path has to be parallel to one of just a small number of directions. They extend their algorithmic result to this case, however it is not entirely clear if other problems in general will be similarly solvable or if they in- or decrease in complexity when such restrictions are applied.

While most of the problems that were tackled in this thesis turned out to be NP-hard we still provided, one way or the other, a method of tackling the problem, be it on a subclass of inputs, or through heuristics and parameterized algorithms. We thereby hope to contribute our small part to illustrate a commonly emphasized sentiment in the theoretical computer science community: "Just because it's NP-hard, doesn't mean you can't do anything about it!"

APPENDIX A

Supplementary Material for Chapter 3

A.1 Supplementary Material

This appendix includes a table for each of the six networks with the metric values as well as a table of final solutions (after one hour) for reference. The different parameters are the number of available directions (k) and the orientation system (**A**ligned, **R**egular, **I**rregular). Metrics are the number of bends, sector deviation (total and per edge), distortion per edge, the runtime in seconds and the optimality gap in percent (listed as number between 0 and 1). For one set of objective function weights and linearity k combination the best value across all orientation systems of every metric is marked in bold. Instances that reached the time limit are marked with a clock. Finally we show a comparison of the quality metrics between the three variants of fixing the drawing directions $\mathcal{C}$ for all instances and linearities and the weight vector $(f_1, f_2, f_3) = (10, 5, 1)$.

Table A.1: Metric results for the Vienna network.

weights	instance	Vienna								
	linearity	$k=3$			$k=4$			$k=5$		
	orientation system	A	R	I	A	R	I	A	R	I
$(3,2,1)$	bends	**11.0**	13.0	**11.0**	**19.0**	20.0	23.0	26.0	**22.0**	29.0
	sector deviation	23.0	**14.0**	35.0	6.0	8.0	**4.0**	8.0	14.0	**6.0**
	↳ per edge	0.43	**0.26**	0.65	**0.11**	0.15	0.07	0.15	0.26	**0.11**
	distortion per edge	**0.69**	0.84	0.7	**0.32**	0.39	0.42	**0.57**	0.58	**0.57**
	time in seconds	687.0	**262.0**	2661.0	**116.0**	1403.0	209.0	**804.0**	3181.0	2739.0
	optimality gap	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$(10,5,1)$	bends	**9.0**	10.0	10.0	**19.0**	**19.0**	**19.0**	25.0	**24.0**	26.0
	sector deviation	22.0	**20.0**	36.0	**6.0**	9.0	8.0	**7.0**	11.0	**7.0**
	↳ per edge	0.41	**0.37**	0.67	**0.11**	0.17	0.15	**0.13**	0.2	**0.13**
	distortion per edge	**0.7**	0.79	0.74	**0.32**	0.41	0.41	0.59	0.6	**0.54**
	time in seconds	84.0	**74.0**	319.0	**45.0**	60.0	47.0	**109.0**	382.0	168.0
	optimality gap	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table A.2: Metric results for the Montreal network.

weights	instance	Montreal								
	linearity	$k=3$			$k=4$			$k=5$		
	orientation system	A	R	I	A	R	I	A	R	I
$(3,2,1)$	bends	10.0	**9.0**	10.0	14.0	**12.0**	13.0	**13.0**	15.0	15.0
	sector deviation	8.0	**6.0**	14.0	5.0	6.0	**1.0**	7.0	7.0	**2.0**
	↳ per edge	0.3	**0.22**	0.52	0.19	0.22	**0.04**	0.26	0.26	**0.07**
	distortion per edge	**0.68**	0.78	0.87	0.5	**0.42**	0.46	0.69	0.55	**0.53**
	time in seconds	21.0	**6.0**	11.0	12.0	**10.0**	13.0	13.0	43.0	**7.0**
	optimality gap	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$(10,5,1)$	bends	**9.0**	**9.0**	10.0	**11.0**	12.0	13.0	**13.0**	15.0	15.0
	sector deviation	9.0	**7.0**	15.0	8.0	6.0	**1.0**	7.0	7.0	**2.0**
	↳ per edge	0.33	**0.26**	0.56	0.3	0.22	**0.04**	0.26	0.26	**0.07**
	distortion per edge	**0.64**	0.78	0.88	0.5	**0.42**	0.46	0.69	0.55	**0.53**
	time in seconds	**6.0**	**6.0**	8.0	**5.0**	8.0	9.0	15.0	56.0	**5.0**
	optimality gap	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table A.3: Metric results for the Washington network.

weights	instance	Washington								
	linearity	$k=3$			$k=4$			$k=5$		
	orientation system	A	R	I	A	R	I	A	R	I
$(3,2,1)$	bends	21.0	24.0	**17.0**	40.0	**33.0**	37.0	**46.0**	49.0	49.0
	sector deviation	27.0	**25.0**	31.0	10.0	16.0	**8.0**	6.0	6.0	**0.0**
	↳ per edge	0.51	**0.47**	0.58	0.19	0.3	**0.15**	0.11	0.11	**0.0**
	distortion per edge	1.18	**1.12**	1.24	**0.35**	0.42	0.43	0.79	0.81	**0.44**
	time in seconds	**599.0**	2212.0	3617.0	**104.0**	290.0	161.0	11.0	44.0	**10.0**
	optimality gap	0.0	0.0	0.03	0.0	0.0	0.0	0.0	0.0	0.0
$(10,5,1)$	bends	18.0	19.0	**15.0**	**29.0**	**29.0**	30.0	42.0	**37.0**	45.0
	sector deviation	**30.0**	35.0	34.0	18.0	18.0	**15.0**	8.0	12.0	**3.0**
	↳ per edge	**0.57**	0.66	0.64	0.34	0.34	**0.28**	0.15	0.23	**0.06**
	distortion per edge	**1.13**	**1.13**	1.24	**0.42**	**0.42**	0.46	0.81	0.8	**0.46**
	time in seconds	**13.0**	65.0	80.0	**36.0**	68.0	45.0	12.0	**8.0**	10.0
	optimality gap	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table A.4: Metric results for the Sydney network.

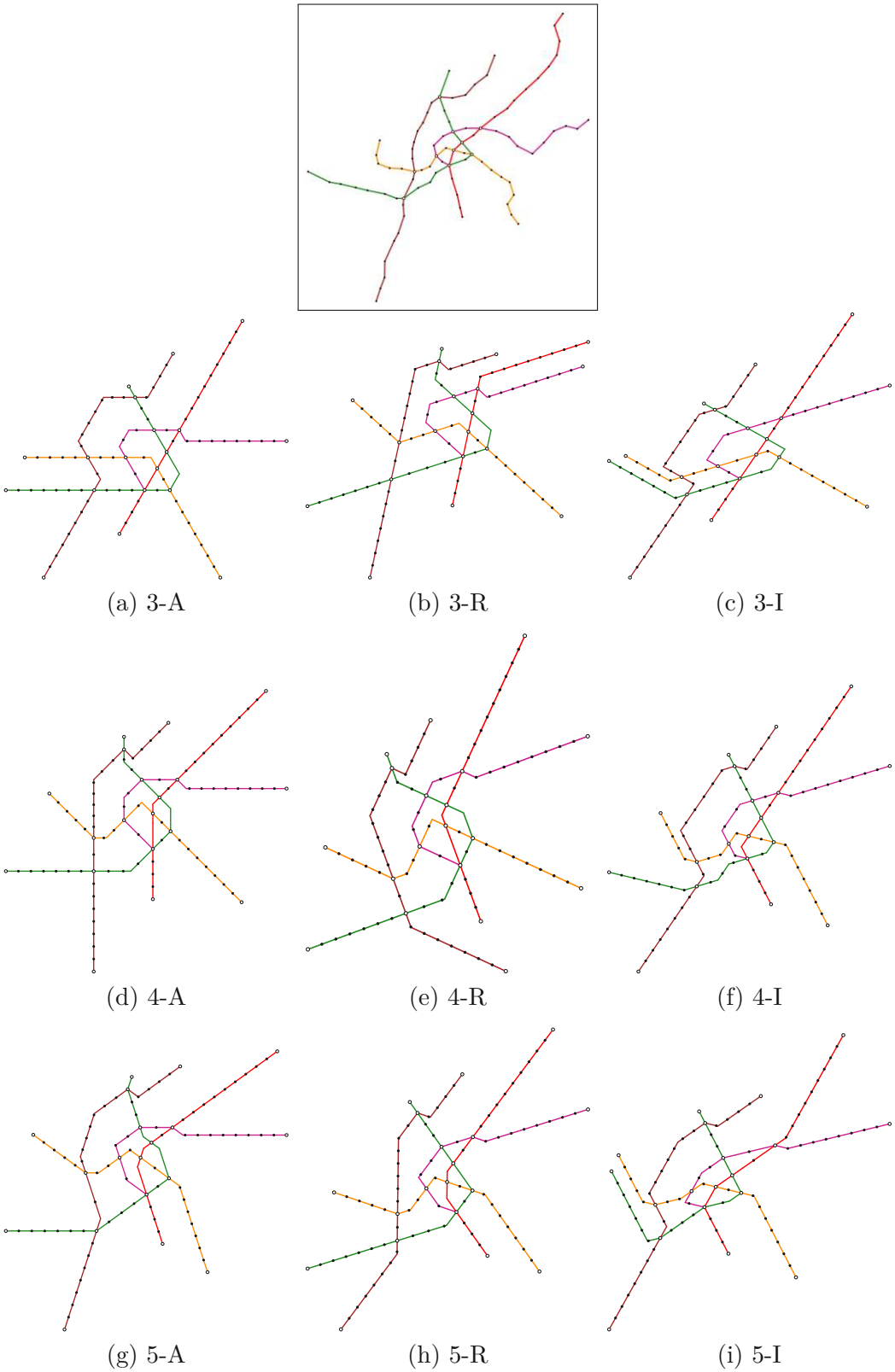
weights	instance	Sydney								
	linearity	$k=3$			$k=4$			$k=5$		
	orientation system	A	R	I	A	R	I	A	R	I
$(3,2,1)$	bends	42.0	43.0	**35.0**	**49.0**	**49.0**	**49.0**	**63.0**	64.0	72.0
	sector deviation	41.0	32.0	**29.0**	11.0	**8.0**	18.0	12.0	**10.0**	14.0
	↳ per edge	0.42	0.33	**0.3**	0.11	**0.08**	0.19	0.12	**0.1**	0.14
	distortion per edge	1.0	**0.78**	1.03	**0.31**	0.38	0.62	0.63	0.71	**0.43**
	time in seconds	🕘	🕘	🕘	🕘	🕘	🕘	🕘	🕘	🕘
	optimality gap	0.2	0.2	**0.16**	0.13	0.12	**0.1**	0.11	0.08	**0.07**
$(10,5,1)$	bends	**31.0**	32.0	35.0	48.0	**47.0**	**47.0**	**59.0**	**59.0**	**59.0**
	sector deviation	36.0	32.0	**24.0**	**8.0**	10.0	17.0	**12.0**	**12.0**	18.0
	↳ per edge	0.37	0.33	**0.25**	**0.08**	0.1	0.18	**0.12**	**0.12**	0.19
	distortion per edge	0.88	**0.84**	1.02	**0.29**	0.33	0.59	0.65	0.68	**0.49**
	time in seconds	🕘	🕘	🕘	🕘	🕘	🕘	🕘	🕘	2216.0
	optimality gap	0.09	0.07	**0.05**	**0.01**	0.02	**0.01**	0.01	0.01	**0.0**

Table A.5: Metric results for the Berlin network.

weights	instance	Berlin								
	linearity	$k=3$			$k=4$			$k=5$		
	orientation system	A	R	I	A	R	I	A	R	I
$(3,2,1)$	bends	–	–	–	141.0	**116.0**	149.0	147.0	**145.0**	160.0
	sector deviation	–	–	–	64.0	**54.0**	141.0	52.0	**35.0**	46.0
	↳ per edge	–	–	–	0.22	**0.18**	0.48	0.18	**0.12**	0.16
	distortion per edge	–	–	–	0.4	**0.37**	0.68	0.78	0.81	**0.52**
	time in seconds	–	–	–	🕘	🕘	🕘	🕘	🕘	🕘
	optimality gap	–	–	–	0.28	**0.23**	0.4	0.25	**0.2**	0.28
$(10,5,1)$	bends	–	–	–	**107.0**	**107.0**	123.0	**125.0**	128.0	139.0
	sector deviation	–	–	–	69.0	**62.0**	98.0	**52.0**	60.0	59.0
	↳ per edge	–	–	–	0.24	**0.21**	0.33	**0.18**	0.2	0.2
	distortion per edge	–	–	–	**0.36**	0.4	0.64	0.8	0.82	**0.54**
	time in seconds	–	–	–	🕘	🕘	🕘	🕘	🕘	🕘
	optimality gap	–	–	–	0.21	**0.15**	0.3	**0.16**	0.21	0.26

Table A.6: Metric results for the London network.

weights	instance	London								
	linearity	$k=3$			$k=4$			$k=5$		
	orientation system	A	R	I	A	R	I	A	R	I
$(3,2,1)$	bends	–	–	–	144.0	**143.0**	151.0	**174.0**	177.0	304.0
	sector deviation	–	–	–	68.0	71.0	**63.0**	**54.0**	66.0	409.0
	↳ per edge	–	–	–	0.21	0.22	**0.2**	**0.17**	0.21	1.28
	distortion per edge	–	–	–	**0.38**	0.4	0.45	0.68	0.66	**0.48**
	time in seconds	–	–	–	🕘	🕘	🕘	🕘	🕘	🕘
	optimality gap	–	–	–	**0.19**	0.26	0.27	**0.25**	**0.25**	0.68
$(10,5,1)$	bends	–	–	–	132.0	**121.0**	149.0	**156.0**	165.0	262.0
	sector deviation	–	–	–	74.0	81.0	**72.0**	**52.0**	56.0	410.0
	↳ per edge	–	–	–	**0.23**	0.25	**0.23**	**0.16**	0.17	1.28
	distortion per edge	–	–	–	0.4	**0.39**	0.41	0.7	0.67	**0.5**
	time in seconds	–	–	–	🕘	🕘	🕘	🕘	🕘	🕘
	optimality gap	–	–	–	**0.16**	0.17	0.27	**0.17**	0.22	0.7

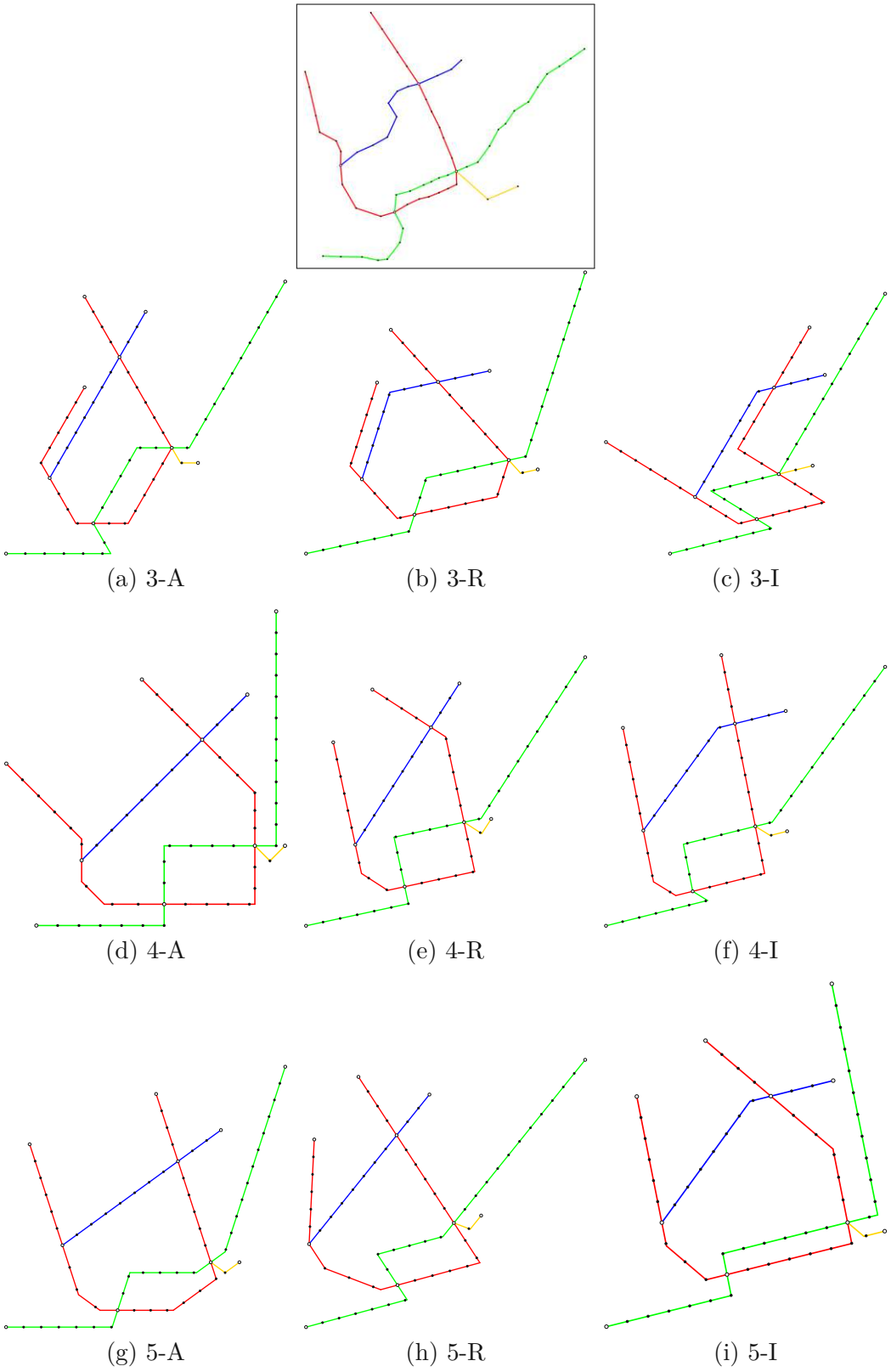


(a) 3-A (b) 3-R (c) 3-I

(d) 4-A (e) 4-R (f) 4-I

(g) 5-A (h) 5-R (i) 5-I

Figure A.1: Examples of Vienna generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

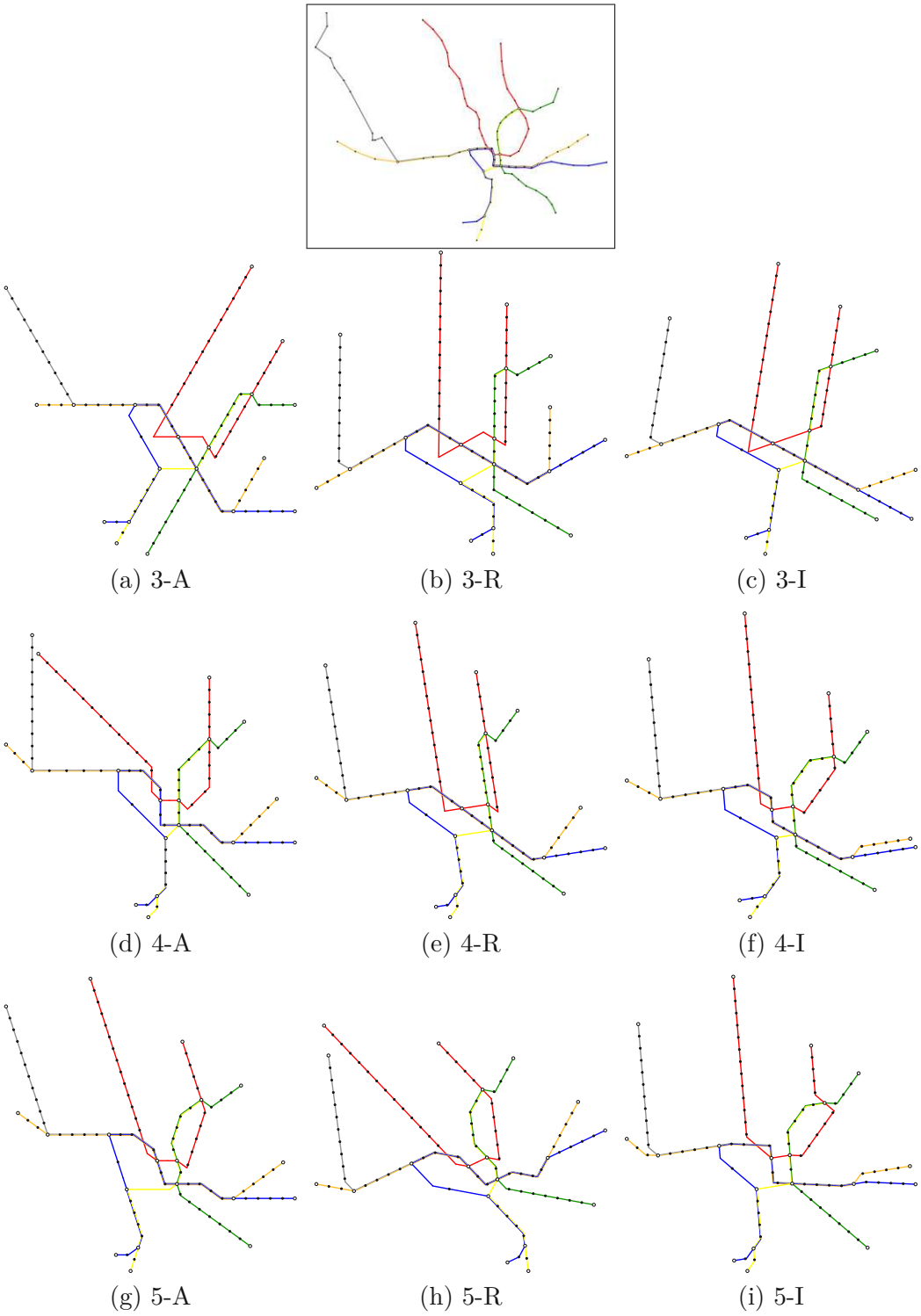


(a) 3-A (b) 3-R (c) 3-I

(d) 4-A (e) 4-R (f) 4-I

(g) 5-A (h) 5-R (i) 5-I

Figure A.2: Examples of Montreal generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

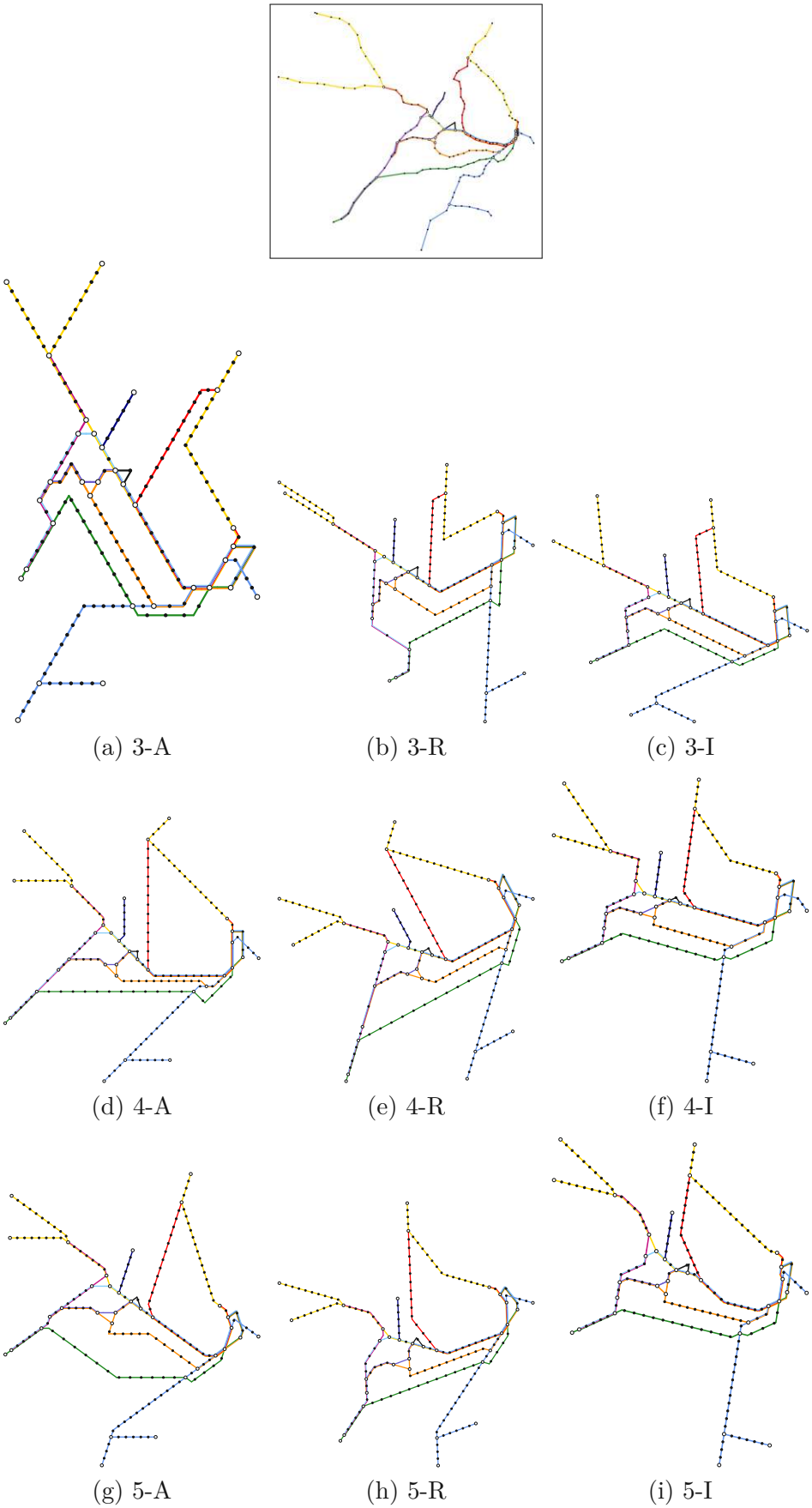


(a) 3-A (b) 3-R (c) 3-I

(d) 4-A (e) 4-R (f) 4-I

(g) 5-A (h) 5-R (i) 5-I

Figure A.3: Examples of Washington generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

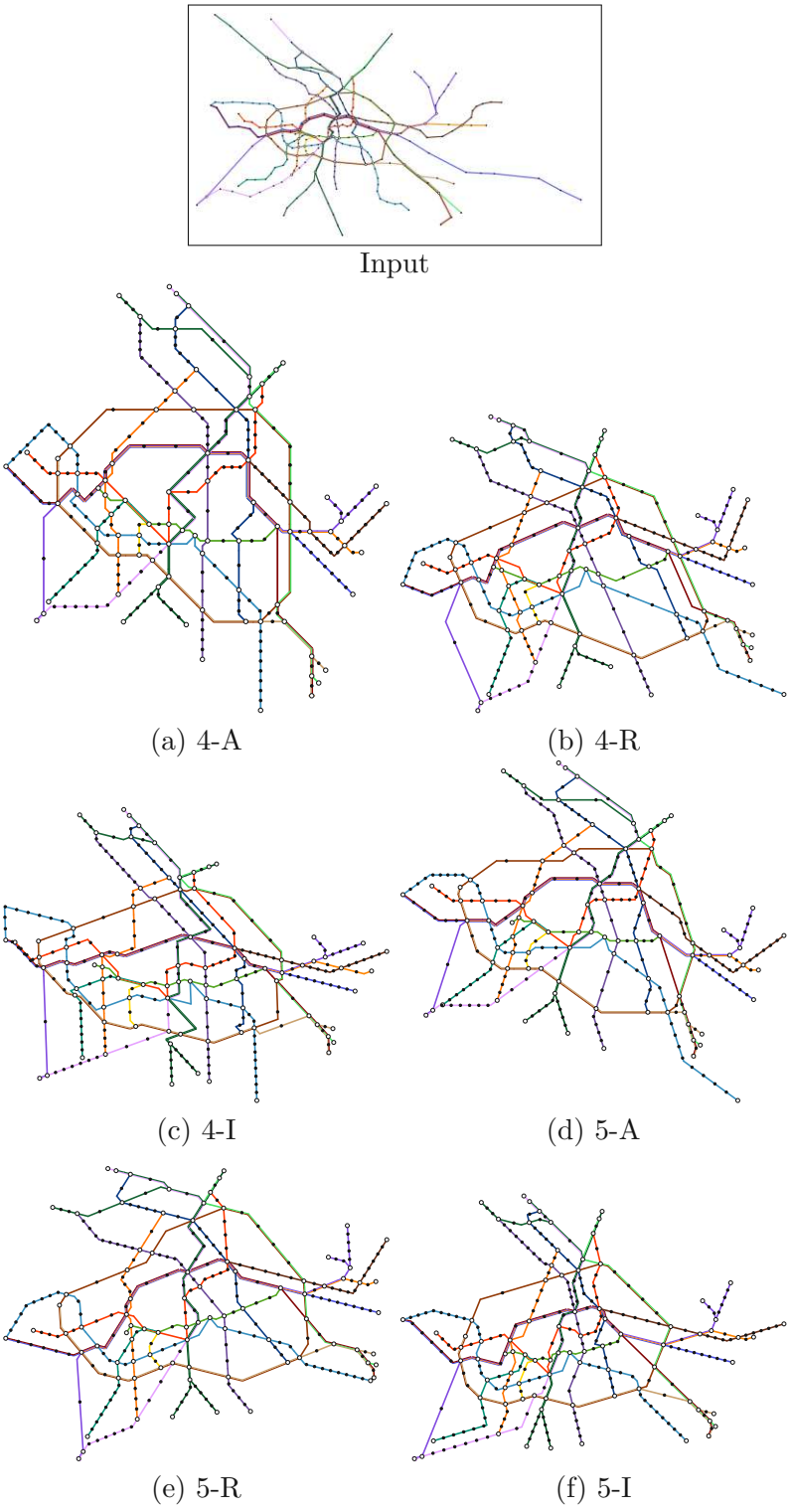


(a) 3-A (b) 3-R (c) 3-I

(d) 4-A (e) 4-R (f) 4-I

(g) 5-A (h) 5-R (i) 5-I

Figure A.4: Examples of Sydney generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.



Input

(a) 4-A (b) 4-R

(c) 4-I (d) 5-A

(e) 5-R (f) 5-I

Figure A.5: Examples of Berlin generated with objective function weights $(f_1, f_2, f_3) = (3, 2, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

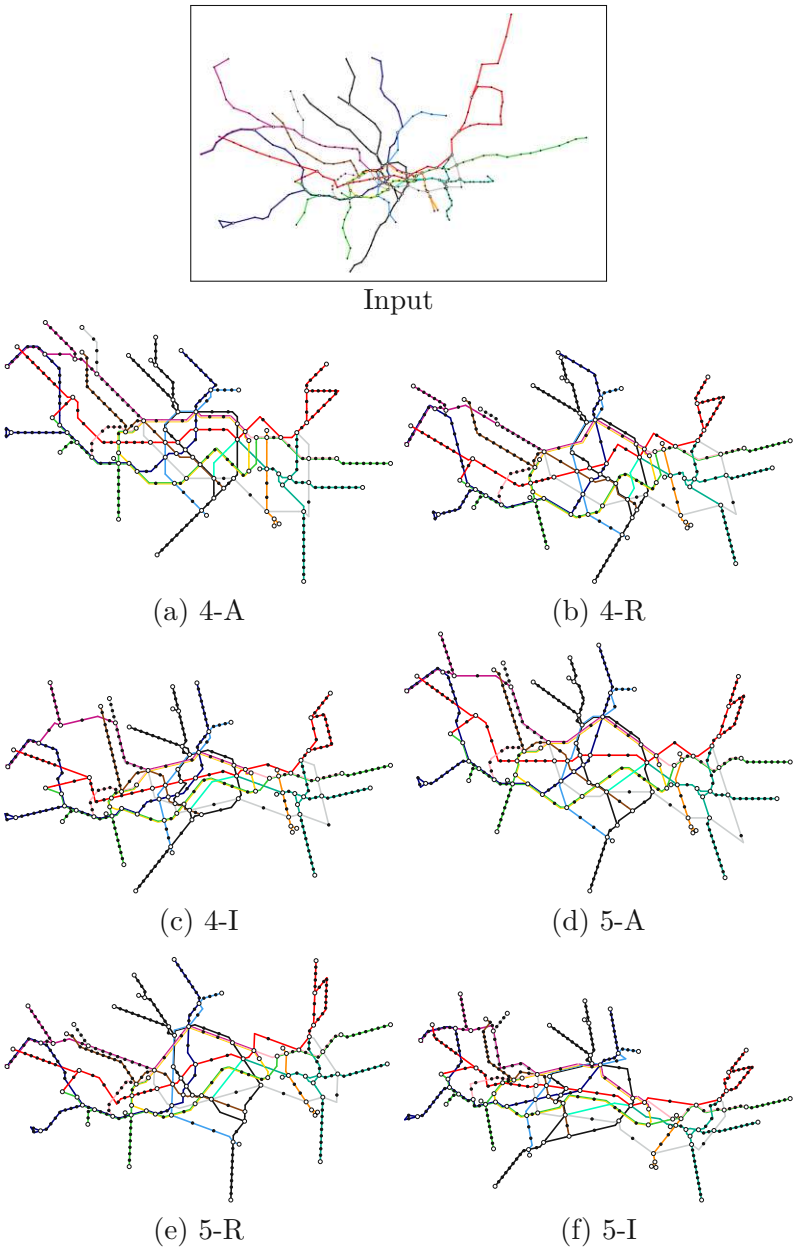
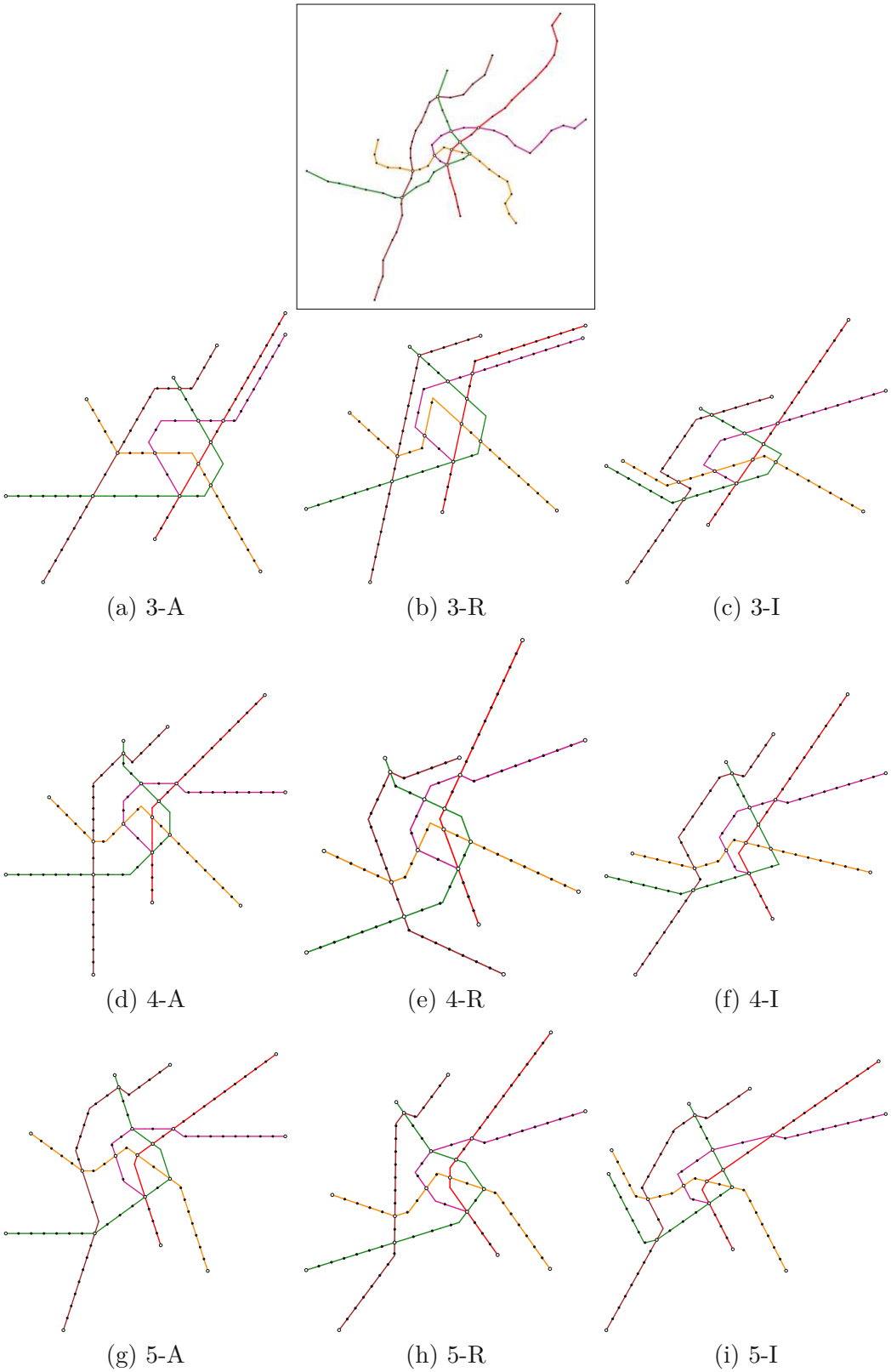


Figure A.6: Examples of London generated with objective function weights $(f_1, f_2, f_3) = (3, 2, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

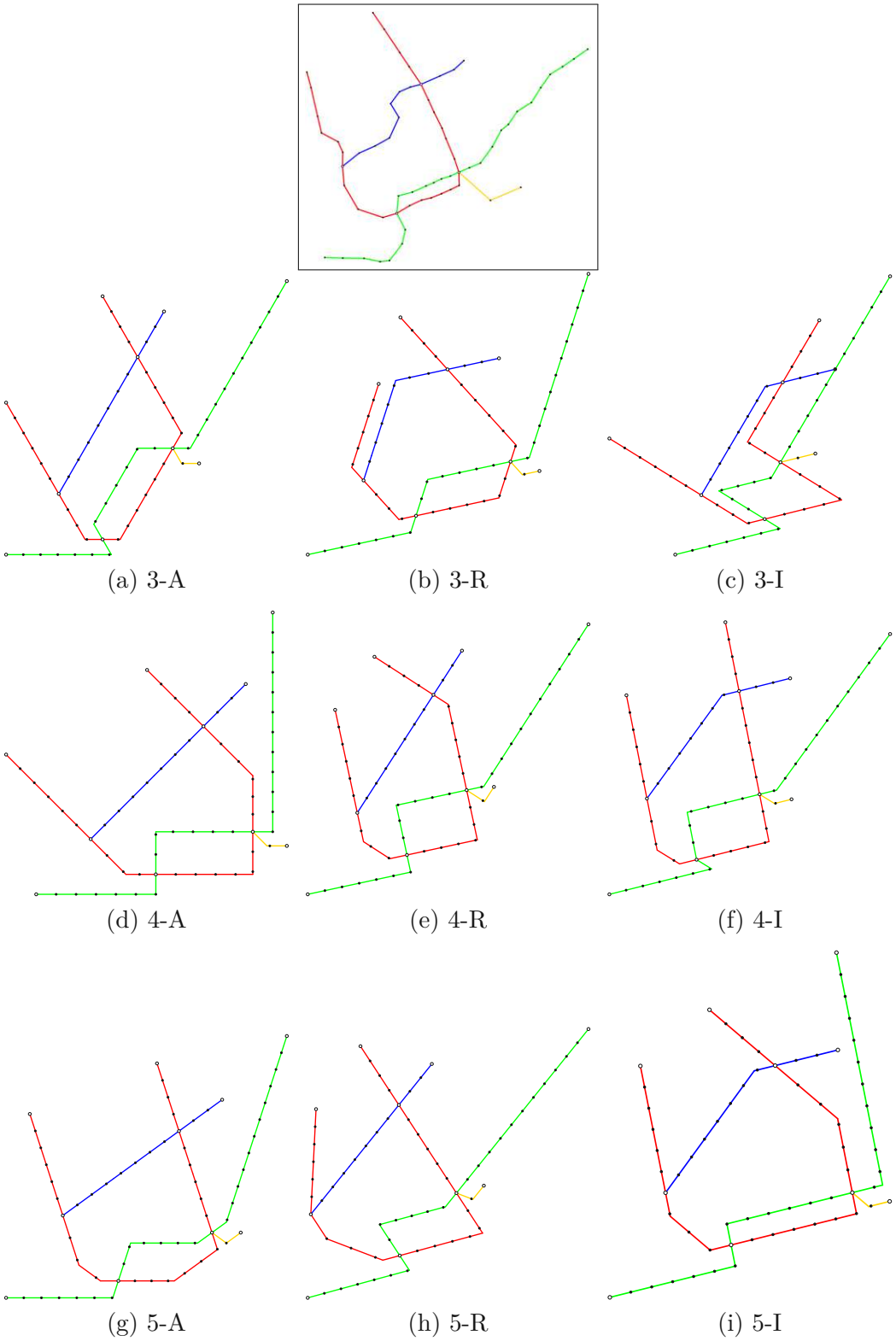


(a) 3-A (b) 3-R (c) 3-I

(d) 4-A (e) 4-R (f) 4-I

(g) 5-A (h) 5-R (i) 5-I

Figure A.7: Examples of Vienna generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

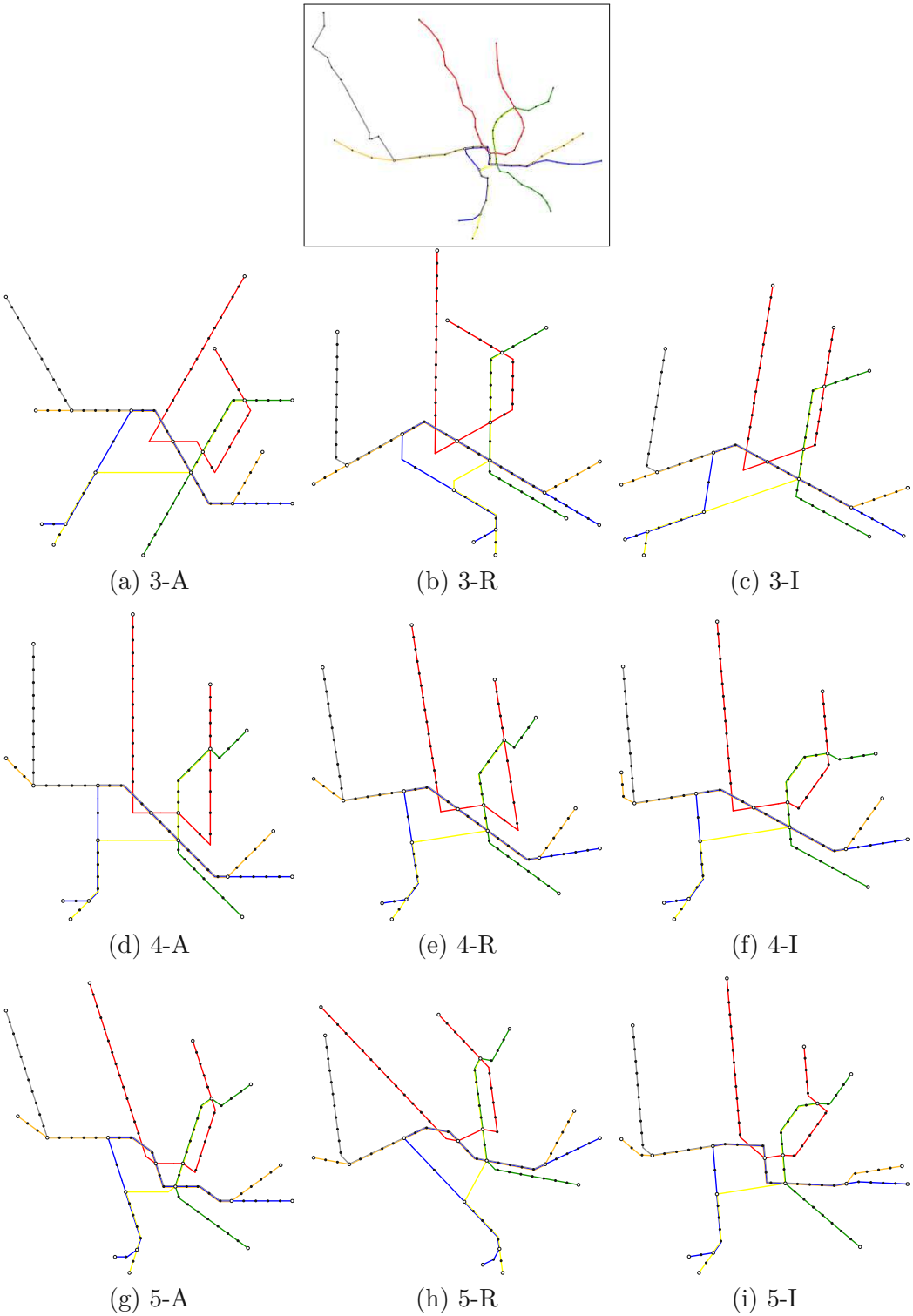


(a) 3-A (b) 3-R (c) 3-I

(d) 4-A (e) 4-R (f) 4-I

(g) 5-A (h) 5-R (i) 5-I

Figure A.8: Examples of Montreal generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

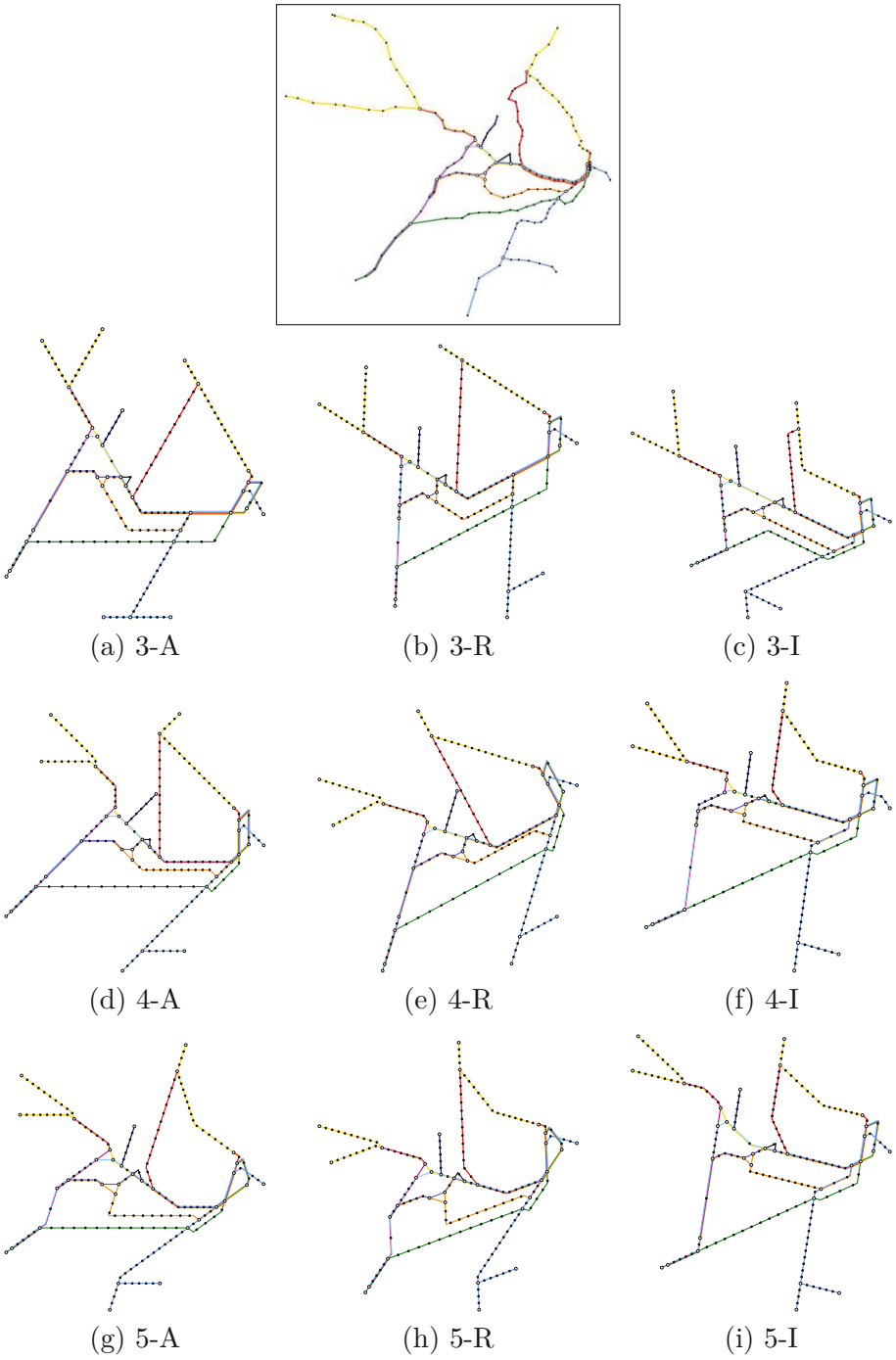


(a) 3-A (b) 3-R (c) 3-I

(d) 4-A (e) 4-R (f) 4-I

(g) 5-A (h) 5-R (i) 5-I

Figure A.9: Examples of Washington generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

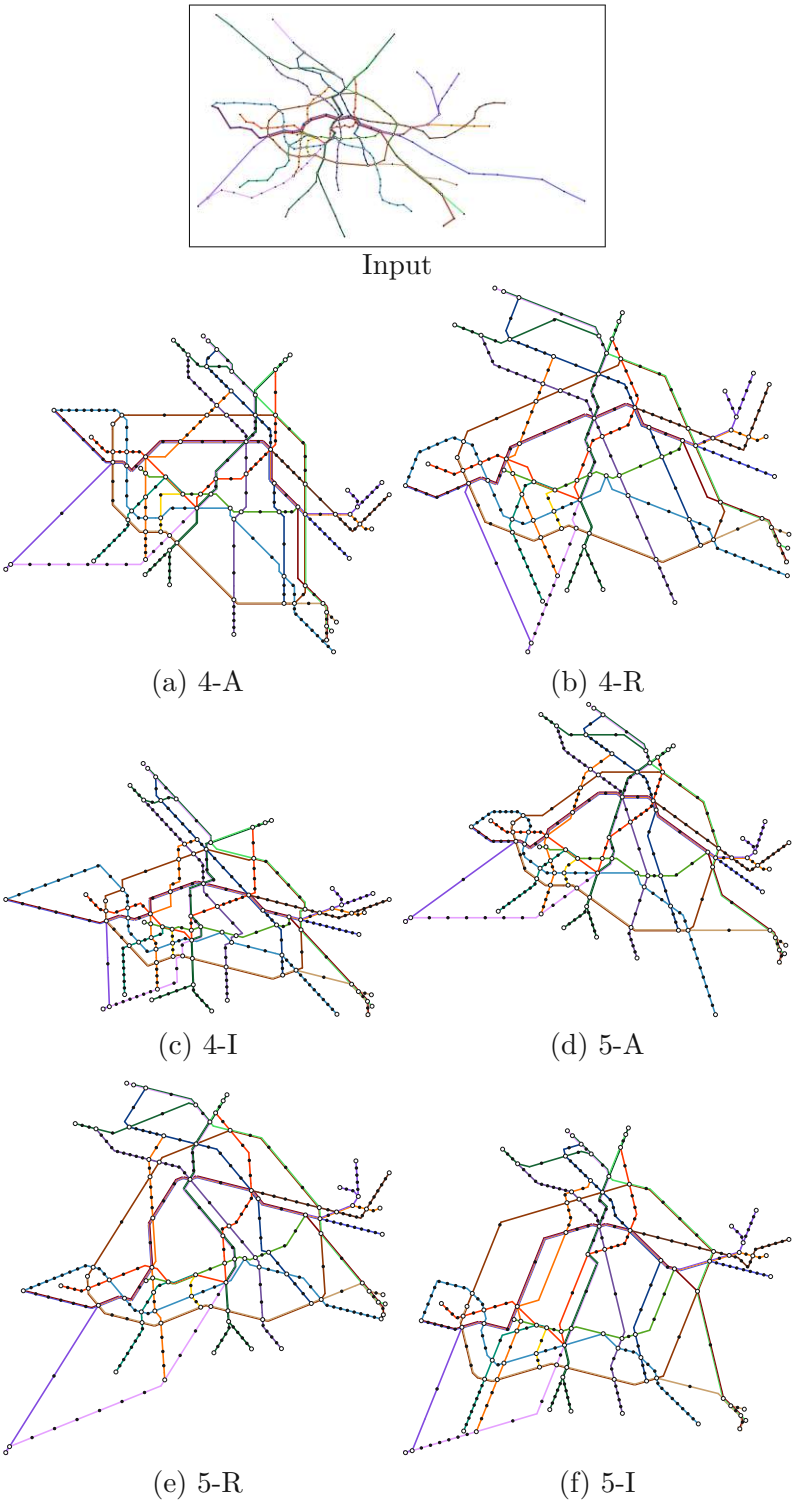


(a) 3-A (b) 3-R (c) 3-I

(d) 4-A (e) 4-R (f) 4-I

(g) 5-A (h) 5-R (i) 5-I

Figure A.10: Examples of Sydney generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.



Input

(a) 4-A (b) 4-R

(c) 4-I (d) 5-A

(e) 5-R (f) 5-I

Figure A.11: Examples of Berlin generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

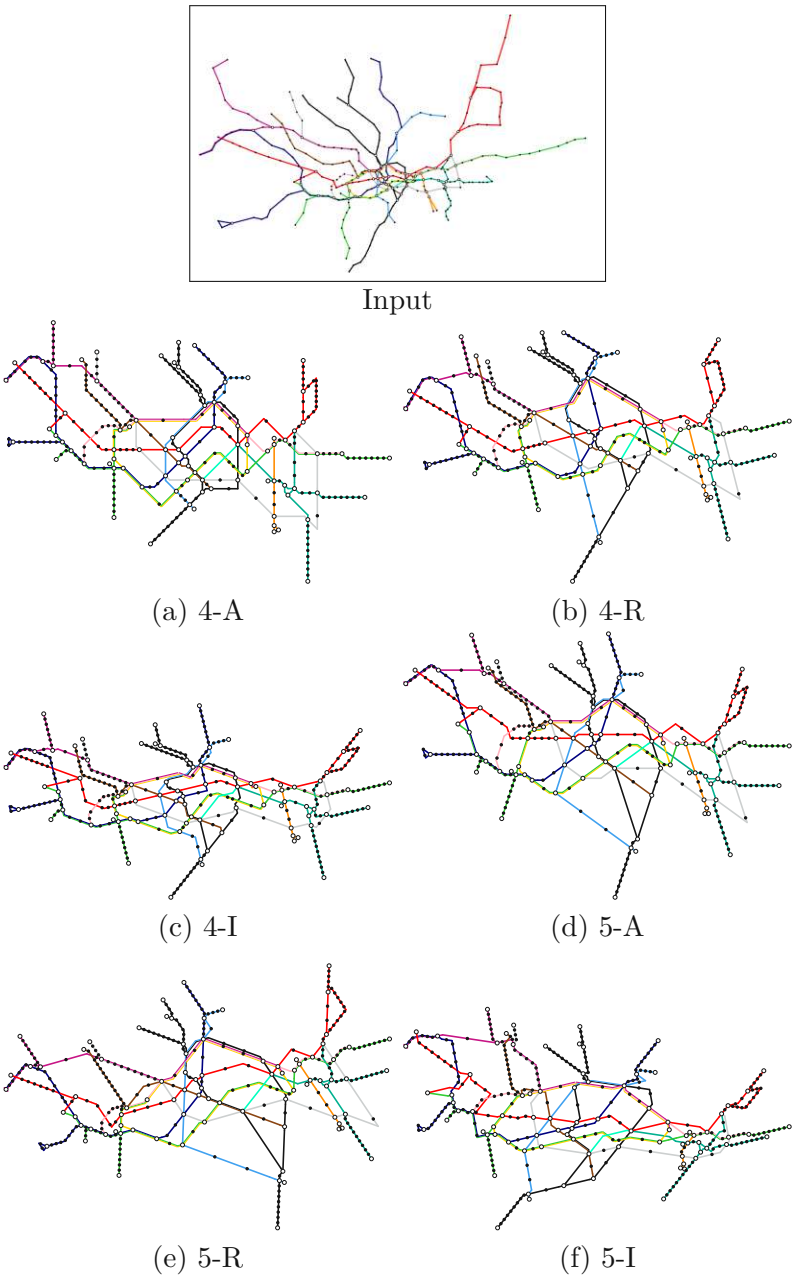


Figure A.12: Examples of London generated with objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$ for different $k \in \{3, 4, 5\}$ and aligned (k-A), regular (k-R) and irregular (k-I) orientation systems.

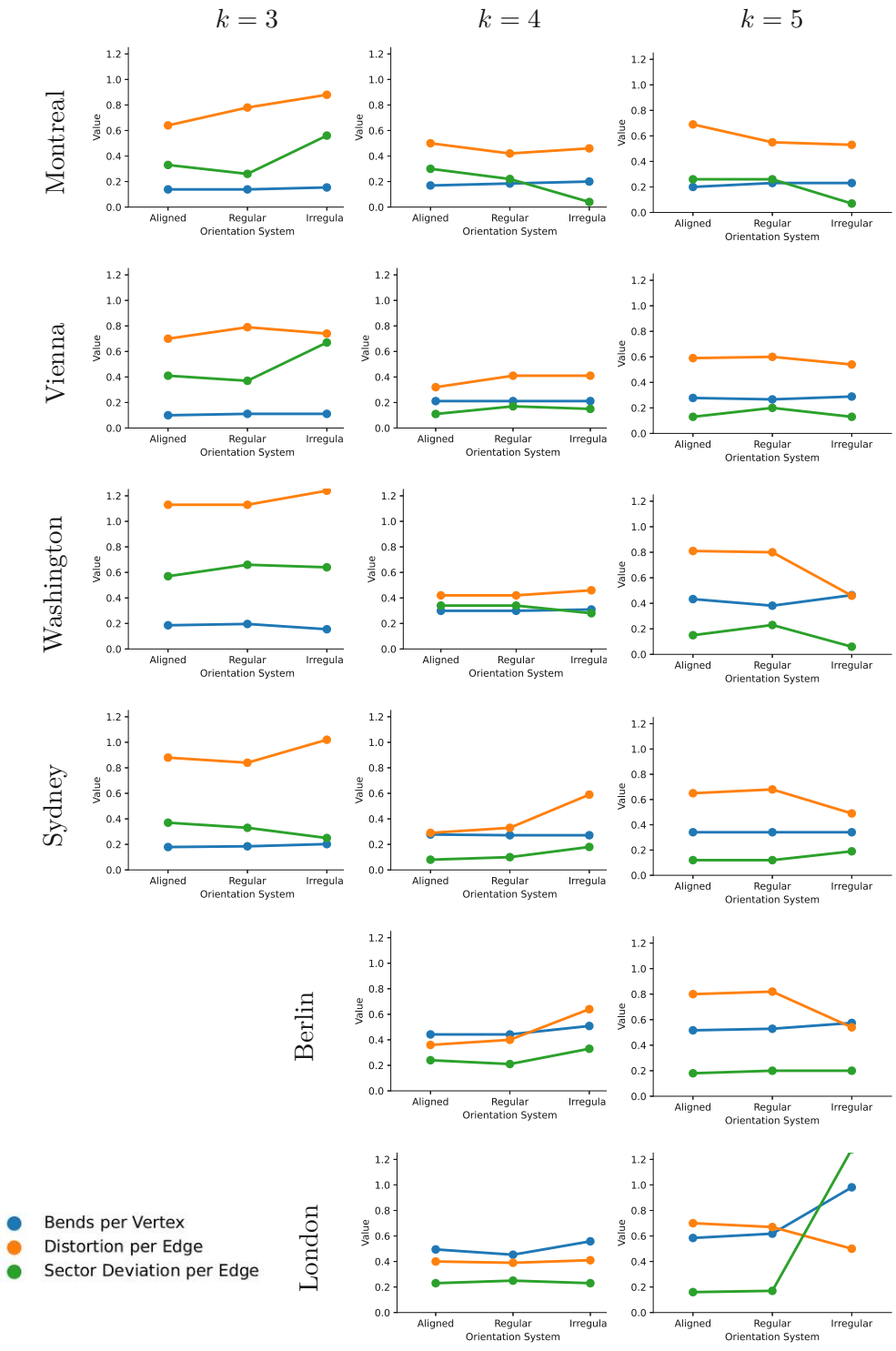


Figure A.13: Plots of the results of the experiments for the objective function weights $(f_1, f_2, f_3) = (10, 5, 1)$.

Bibliography

[1] Mikkel Abrahamsen, Anna Adamaszek, Karl Bringmann, Vincent Cohen-Addad, Mehran Mehr, Eva Rotenberg, Alan Roytman, and Mikkel Thorup. "Fast Fencing". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*. 2018, pages 564–573. DOI: 10.1145/3188745.3188878 (cited on pages 64, 65).

[2] Mikkel Abrahamsen, Panos Giannopoulos, Maarten Löffler, and Günter Rote. "Geometric Multicut". In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Volume 132. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 9:1–9:15. DOI: 10.4230/LIPIcs.ICALP.2019.9 (cited on pages 63, 64).

[3] Alok Aggarwal, Heather Booth, Joseph O'Rourke, Subhash Suri, and Chee-Keng Yap. "Finding Minimal Convex Nested Polygons". In: *Information and Computation* 83.1 (1989), pages 98–110. DOI: 10.1016/0890-5401(89)90049-7 (cited on pages 64, 66).

[4] Md Jawaherul Alam, Therese Biedl, Stefan Felsner, Michael Kaufmann, Stephen G Kobourov, and Torsten Ueckerdt. "Computing cartograms with optimal complexity". In: *Proceedings of the twenty-eighth annual symposium on Computational geometry*. 2012, pages 21–30 (cited on page 6).

[5] Sanjeev Arora and Boaz Barak. "Computational complexity: a modern approach". Cambridge University Press, 2009 (cited on page 14).

[6] Silvania Avelar. "Convergence Analysis and Quality Criteria for an Iterative Schematization of Networks". In: *Geoinformatica* 11 (2007), pages 497–513. DOI: 10.1007/s10707-007-0018-z (cited on page 42).

[7] Silvania Avelar and Lorenz Hurni. "On the Design of Schematic Transport Maps". In: *Cartographica* 41.3 (2006), pages 217–228. DOI: 10.3138/A477-3202-7876-N514 (cited on page 4).

[8] Silvania Avelar and Matthias Müller. "Generating Topologically Correct Schematic Maps". In: *Proceedings of the 9th International Symposium on Spatial Data Handling*. 2000, 4a.28–4a.35 (cited on page 42).

[9] Lukas Barth, Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. "A Topology-Shape-Metrics Framework for Ortho-Radial Graph Drawing". In: *CoRR* abs/2106.05734 (2021). arXiv: 2106.05734 (cited on pages 3, 18).

[10] Hannah Bast, Patrick Brosi, and Sabine Storandt. "Metro Maps on Flexible Base Grids". In: *17th International Symposium on Spatial and Temporal Databases*. New York, NY, USA: Association for Computing Machinery, 2021, pages 12–22. ISBN: 978-1-4503-8425-4. DOI: 10.1145/3469830.3470899 (cited on pages 18, 42, 50, 59–61).

[11] Hannah Bast, Patrick Brosi, and Sabine Storandt. "Metro Maps on Octilinear Grid Graphs". In: *Comput. Graph. Forum* 39.3 (2020), pages 357–367. DOI: 10.1111/cgf.13986 (cited on pages 18, 42, 50, 51, 59, 60).

[12] Kees Joost Batenburg and Walter A Kosters. "Solving Nonograms by combining relaxations". In: *Pattern Recognition* 42.8 (2009). Publisher: Elsevier, pages 1672–1683 (cited on page 7).

[13] Kees Joost Batenburg and Walter A. Kosters. "On the Difficulty of Nonograms". In: *ICGA Journal* 35.4 (2012), pages 195–205. DOI: 10.3233/ICG-2012-35402 (cited on page 152).

[14] Tobias Batik, Soeren Terziadis, Yu-Shuen Wang, Martin Nöllenburg, and Hsiang-Yun Wu. "Mixed Metro Maps with User-Specified Motifs". In: *29th International Symposium on Graph Drawing and Network Visualization (GD'21)*. Volume 12868. Lecture Notes in Computer Science. 2021, pages 304–317 (cited on pages 9, 39).

[15] Tobias Batik, Soeren Terziadis, Yu-Shuen Wang, Martin Nöllenburg, and Hsiang-Yun Wu. "Shape-Guided Mixed Metro Map Layout". In: *Comput. Graph. Forum* 41.7 (2022), pages 495–506. DOI: https://doi.org/10.1111/cgf.14695 (cited on pages 9, 39).

[16] Cristina Bazgan, Bruno Escoffier, and Vangelis Th Paschos. "Completeness in standard and differential approximation classes: Poly-(D) APX-and (D) PTAS-completeness". In: *Theoretical Computer Science* 339.2 (2005). Publisher: Elsevier, pages 272–292 (cited on page 6).

[17] Samuel W. Bent and Udi Manber. "On non-intersecting Eulerian circuits". In: *Discrete Applied Mathematics* 18.1 (1987), pages 87–94. DOI: 10.1016/0166-218X(87)90045-X (cited on page 155).

[18] Daniel Berend, Dolev Pomeranz, Ronen Rabani, and Ben Raziel. "Nonograms: Combinatorial questions and algorithms". In: *Discrete Applied Mathematics* 169 (2014), pages 30–42. DOI: 10.1016/j.dam.2014.01.004 (cited on pages 7, 152).

[19] Mark de Berg and Atlas F Cook. "Go with the flow: The direction-based Fréchet distance of polygonal curves". In: *Theory and Practice of Algorithms in (Computer) Systems (TAPAS'11)*. Volume 6595. LNCS. Springer, 2011, pages 81–91. DOI: 10.1007/978-3-642-19754-3_10 (cited on page 45).

[20] Sujoy Bhore, Fabian Klute, Maarten Löffler, Martin Nöllenburg, Soeren Terziadis, and Anaïs Villedieu. "Minimum Link Fencing". In: *The 37th European Workshop on Computational Geometry (EuroCG'21)*. 2021, 55:1–55:9. DOI: 10.48550/arXiv.2209.14804 (cited on pages 9, 63).

[21] Sujoy Bhore, Fabian Klute, Maarten Löffler, Martin Nöllenburg, Soeren Terziadis, and Anaïs Villedieu. "Minimum Link Fencing". In: *33rd International Symposium on Algorithms and Computation (ISAAC'22)*. Volume 248. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 34:1–34:14. DOI: 10.4230/LIPIcs.ISAAC.2022.34 (cited on pages 9, 63).

[22] Sujoy Bhore, Maarten Löffler, Soeren Nickel, and Martin Nöllenburg. "Unit Disk Representations of Embedded Trees, Outerplanar and Multi-legged Graphs". In: *29th International Symposium on Graph Drawing and Network Visualization (GD'21)*. Volume 12868. Lecture Notes in Computer Science. 2021, pages 304–317. DOI: 10.1007/978-3-030-92931-2_22 (cited on pages 10, 103, 151).

[23] Sujoy Bhore, Soeren Nickel, and Martin Nöllenburg. "Recognition of Unit Disk Graphs for Caterpillars, Embedded Trees, and Outerplanar Graphs". In: *The 37th European Workshop on Computational Geometry (EuroCG'21)*. 2021, 50:1–50:9 (cited on page 10).

[24] Andreas Björklund, Thore Husfeld, and Nina Taslaman. "Shortest Cycle through Specified Elements". In: *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '12. Kyoto, Japan: Society for Industrial and Applied Mathematics, 2012, pages 1747–1753. DOI: 2095116.2095255 (cited on pages 162–165).

[25] Clinton Bowen, Stephane Durocher, Maarten Löffler, Anika Rounds, André Schulz, and Csaba D. Tóth. "Realization of simply connected polygonal linkages and recognition of unit disk contact trees". In: *Graph Drawing and Network Visualization (GD'15)*. Volume 9411. LNCS. Springer International Publishing, 2015, pages 447–459. DOI: 10.1007/978-3-319-27261-0_37 (cited on pages 10, 103–105, 120–125, 128, 129, 134, 142).

[26] Heinz Breu and David G. Kirkpatrick. "On the complexity of recognizing intersection and touching graphs of disks". In: *Graph Drawing (GD'95)*. Edited by Franz J. Brandenburg. Volume 1027. LNCS. Springer, 1996, pages 88–98. ISBN: 978-3-540-49351-8. DOI: 10.1007/BFb0021793 (cited on page 103).

[27] Heinz Breu and David G. Kirkpatrick. "Unit disk graph recognition is NP-hard". In: *Comput. Geom. Theory Appl.* 9.1–2 (1998), pages 3–24. DOI: 10.1016/S0925-7721(97)00014-X (cited on pages 7, 10, 103–105, 120).

[28] Patrick Brosi. "A Toolchain for Generating Transit Maps from Schedule Data". In: *Schematic Mapping Workshop*. 2022 (cited on pages 59, 171).

[29] Florestan Brunck, Hsien-Chih Chang, Maarten Löffler, Tim Ophelders, and Lena Schlipf. "Reconfiguring Popular Faces". In: *Dagstuhl Reports (Seminar 22062)* 12.2 (2022). Edited by Maike Buchin, Anna Lubiw, Arnaud de Mesmay, Saul Schleimer, and Florestan Brunck, pages 24–34. ISSN: 2192-5283. DOI: 10.4230/DagRep.12.2.17 (cited on pages 7, 10, 153).

[30] Kevin Buchin, Wouter Meulemans, André van Renssen, and Bettina Speckmann. "Area-Preserving Simplification and Schematization of Polygonal Subdivisions". In: *ACM Trans. Spatial Algorithms and Systems* 2.1 (2016), 2:1–2:36. DOI: 10.1145/2818373 (cited on page 18).

[31] Sergio Cabello, Mark de Berg, and Marc van Kreveld. "Schematization of Networks". In: *Computational Geometry* 30.3 (2005), pages 223–238. DOI: 10.1016/j.comgeo.2004.11.002 (cited on page 42).

[32] Jug Cerovic. "One Metro World". Jug Cerovic, 2016 (cited on pages 39, 40, 42, 43).

[33] Jérémie Chalopin, Daniel Gonçalves, and Pascal Ochem. "Planar graphs have 1-string representations". In: *Discrete & Computational Geometry* 43.3 (2010), pages 626–647. DOI: 10.1007/s00454-009-9196-9 (cited on page 103).

[34] Steven Chaplick and Torsten Ueckerdt. "Planar Graphs as VPG-Graphs". In: *Graph Drawing (GD'12)*. Edited by Walter Didimo and Maurizio Patrignani. Volume 7704. LNCS. Springer, 2013, pages 174–186. ISBN: 978-3-642-36763-2. DOI: 10.1007/978-3-642-36763-2_16 (cited on page 103).

[35] Yen-Chi Chen and Shun-Shii Lin. "A fast nonogram solver that won the TAAI 2017 and ICGA 2018 tournaments". In: *ICGA Journal* 41.1 (2019), pages 2–14. DOI: 10.3233/ICG-190097 (cited on page 152).

[36] Man-Kwun Chiu, Jonas Cleve, and Martin Nöllenburg. "Recognizing embedded caterpillars with weak unit disk contact representations is NP-hard". In: *CoRR* abs/2010.01881 (2020). arXiv: 2010.01881 (cited on pages 104, 105).

[37] Daniel Chivers and Peter Rodgers. "Octilinear Force-Directed Layout with Mental Map Preservation for Schematic Diagrams". In: *Proceedings of Diagrammatic Representation and Inference (Diagrams'14)*. Volume 8578. LNCS. Springer, 2014, pages 1–8. DOI: 10.1007/978-3-662-44043-8_1 (cited on page 42).

[38] Jonas Cleve. "Weak Unit Disk Contact Representations for Graphs without Embedding". In: *CoRR* abs/2010.01886 (2020). arXiv: 2010.01886 (cited on pages 104, 105, 145).

[39] Gautam Das. "Approximation schemes in computational geometry." PhD thesis. University of Wisconsin, Madison, 1991 (cited on pages 64, 66).

[40] Gautam K Das, Guilherme D da Fonseca, and Ramesh K Jallu. "Efficient independent set approximation in unit disk graphs". In: *Discrete Applied Mathematics* 280 (2020). Publisher: Elsevier, pages 63–70 (cited on page 6).

[41] Tathagata Debnath and Mingzhou Song. “Fast Optimal Circular Clustering and Applications on Round Genomes”. In: *IEEE ACM Trans. Comput. Biol. Bioinform.* 18.6 (2021), pages 2061–2071. DOI: 10.1109/TCBB.2021.3077573 (cited on page 23).

[42] Asaf Degani. “A Tale of Two Maps: Analysis of the London Underground “Diagram””. In: *Ergonomics in Design: The Quarterly of Human Factors Applications* 21 (2013), pages 7–16. DOI: 10.1177/1064804613489125 (cited on page 41).

[43] Daniel Delling, Andreas Gemsa, Martin Nöllenburg, Thomas Pajor, and Ignaz Rutter. “On d-regular schematization of embedded paths”. In: *Comput. Geom. Theory Appl.* 47.3 (2014), pages 381–406. DOI: 10.1016/j.comgeo.2013.10.002 (cited on page 18).

[44] Thomas C. van Dijk, Arthur van Goethem, Jan-Henrik Haunert, Wouter Meulemans, and Bettina Speckmann. “Map Schematization with Circular Arcs”. In: *Proceedings of Geographic Information Science (GIScience'14)*. Volume 8728. LNCS. Springer, 2014, pages 1–17. DOI: 10.1007/978-3-319-11593-1_1 (cited on pages 2, 3).

[45] Thomas C. van Dijk and Dieter Lutz. “Realtime Linear Cartograms and Metro Maps”. In: *Proceedings of the Advances in Geographic Information Systems (SIGSPATIAL'18)*. 2018, pages 488–491. DOI: 10.1145/3274895.3274959 (cited on page 42).

[46] Daniel Dorling. “Area cartograms: their use and creation”. In: *Concepts and Techniques in Modern Geography (CATMOG)* (1996). Publisher: Univ. of East Anglia (cited on page 6).

[47] Rodney G Downey, Michael R Fellows, et al. “Fundamentals of parameterized complexity”. Volume 4. Springer, 2013 (cited on page 14).

[48] Christian A. Duncan and Michael T. Goodrich. “Planar Orthogonal and Polyline Drawing Algorithms”. In: *Handbook of Graph Drawing and Visualization*. Edited by Roberto Tamassia. Section: 7. CRC Press, 2013, pages 223–246 (cited on page 18).

[49] Tim Dwyer, Nathan Hurst, and Damian Merrick. “A Fast and Simple Heuristic for Metro Map Path Simplification”. In: *Proceedings of Advances in Visual Computing (ISVC'08)*. Volume 5359. LNCS. Springer, 2008, pages 22–30. DOI: 10.1007/978-3-540-89646-3_3 (cited on page 42).

[50] Peter Eades and David Rappaport. “The complexity of computing minimum separating polygons”. In: *Pattern Recognition Letters* 14.9 (1993), pages 715–718. DOI: 10.1016/0167-8655(93)90140-9 (cited on pages 64, 66).

[51] Jack Edmonds. “Paths, Trees, and Flowers”. In: *Canadian Journal of Mathematics* 17 (1965), pages 449–467. DOI: 10.4153/CJM-1965-045-4 (cited on page 100).

[52] Stefan Felsner. “Rectangle and Square Representations of Planar Graphs”. In: *Thirty Essays on Geometric Graph Theory*. Edited by János Pach. Springer, 2013, pages 213–248. DOI: 10.1007/978-1-4614-0110-0_12 (cited on page 103).

[53] Martin Fink, Herman Haverkort, Martin Nöllenburg, Maxwell J. Roberts, Julian Schuhmann, and Alexander Wolff. "Drawing Metro Maps using Bézier Curves". In: *Proceedings of Graph Drawing (GD'12)*. Volume 7704. LNCS. Springer, 2012, pages 463–474. DOI: 10.1007/978-3-642-36763-2_41 (cited on pages 3, 39).

[54] Martin Fink, Magnus Lechner, and Alexander Wolff. "Concentric Metro Maps". In: *Proceedings of the Schematic Mapping Workshop (SMW'14)*. 2014 (cited on page 39).

[55] Stefan Funke, Robin Schirrmeister, Simon Skilevic, and Sabine Storandt. "Compass-based navigation in street networks". In: *Web and Wireless Geographical Information Systems (W2GIS'15)*. Volume 9080. LNCS. Springer, 2015, pages 71–88. DOI: 10.1007/978-3-319-18251-3_5 (cited on page 40).

[56] Stefan Funke and Sabine Storandt. "Path shapes: an alternative method for map matching and fully autonomous self-localization". In: *Advances in Geographic Information Systems (SIGSPATIAL'11)*. ACM, 2011, pages 319–328. DOI: 10.1145/2093973.2094016 (cited on page 45).

[57] Michael R Garey, Ronald L Graham, and David S Johnson. "The complexity of computing Steiner minimal trees". In: *SIAM journal on applied mathematics* 32.4 (1977), pages 835–859. DOI: 10.1137/0132072 (cited on page 66).

[58] Ken Garland. "Mr. Beck's Underground Map". Harrow Weald: Capital Transport Publishing, 1994 (cited on pages 17, 43).

[59] Mari Ángeles Garrido, Claudia Iturriaga, Alberto Márquez, José Ramon Portillo, Pedro Reyes, and Alexander Wolff. "Labeling Subway Lines". In: *Proceedings of 12th International Symposium on Algorithms and Computation (ISAAC'01)*. Volume 2223. LNCS. Springer, 2001, pages 649–659. DOI: 10.1007/3-540-45678-3_55 (cited on page 42).

[60] Subir Kumar Ghosh. "Computing the visibility polygon from a convex set and related problems". In: *Journal of Algorithms* 12.1 (1991), pages 75–95. DOI: 10.1016/0196-6774(91)90024-S (cited on pages 64, 66).

[61] Arthur van Goethem, Wouter Meulemans, Andreas Reimer, Herman Haverkort, and Bettina Speckmann. "Topologically Safe Curved Schematisation". In: *The Cartographic Journal* 50.3 (2013), pages 276–285. DOI: 10.1179/1743277413Y.0000000066 (cited on page 3).

[62] Daniel Gonçalves, Lucas Isenmann, and Claire Pennarun. "Planar graphs as L-intersection or L-contact graphs". In: *Discrete Algorithms (SODA'18)*. SIAM. 2018, pages 172–184. DOI: 10.1137/1.9781611975031.12 (cited on page 103).

[63] John Hershberger and Jack Snoeyink. "Computing Minimum Length Paths of a Given Homotopy Class". In: *Computational Geometry: Theory and Applications* 4 (1994), pages 63–97. DOI: 10.1016/0925-7721(94)90010-8 (cited on pages 97, 99, 172).

[64] Petr Hliněný. “Classes and Recognition of Curve Contact Graphs”. In: *Journal of Combinatorial Theory, Series B* 74.1 (1998), pages 87–103. DOI: 10.1006/jctb.1998.1846 (cited on page 103).

[65] Petr Hliněný and Jan Kratochvíl. “Representing graphs by disks and balls (a survey of recognition-complexity results)”. In: *Discrete Mathematics* 229.1 (2001), pages 101–124. ISSN: 0012-365X. DOI: 10.1016/S0012-365X(00)00204-1 (cited on page 103).

[66] Klaus Jansen and Haiko Müller. “The minimum broadcast time problem for several processor networks”. In: *Theoretical Computer Science* 147.1-2 (1995), pages 69–85. DOI: 10.1016/0304-3975(94)00230-G (cited on page 66).

[67] Jan-Hinrich Kämper, Stephen G. Kobourov, and Martin Nöllenburg. “Circular-Arc Cartograms”. In: *IEEE Pacific Visualization Symposium (PacificVis'13)*. IEEE, 2013, pages 1–8. DOI: 10.1109/PacificVis.2013.6596121 (cited on page 3).

[68] Ross J. Kang and Tobias Müller. “Sphere and Dot Product Representations of Graphs”. In: *Discret. Comput. Geom.* 47.3 (2012), pages 548–568. DOI: 10.1007/s00454-012-9394-8 (cited on page 7).

[69] Richard M Karp. “Reducibility among combinatorial problems”. Springer, 2010 (cited on page 16).

[70] Mees van de Kerkhof, Tim de Jong, Raphael Parment, Maarten Löffler, Amir Vaxman, and Marc J. van Kreveld. “Design and Automated Generation of Japanese Picture Puzzles”. In: *Comput. Graph. Forum* 38.2 (2019), pages 343–353. DOI: 10.1111/cgf.13642 (cited on pages 7, 10, 152, 153).

[71] Kamil A Khan. “Solving nonograms using integer programming without coloring”. In: *IEEE Transactions on Games* 14.1 (2020). Publisher: IEEE, pages 56–63 (cited on page 7).

[72] Boris Klemz, Martin Nöllenburg, and Roman Prutkin. “Recognizing Weighted and Seeded Disk Graphs”. In: *J. Computational Geometry* 13.1 (2022), pages 327–376. DOI: 10.20382/jocg.v13i1a13 (cited on pages 7, 10, 104, 105, 142).

[73] Boris Klemz, Martin Nöllenburg, and Roman Prutkin. “Recognizing Weighted Disk Contact Graphs”. In: *Graph Drawing and Network Visualization (GD'15)*. Volume 9411. LNCS. Springer International Publishing, 2015, pages 433–446. DOI: 10.1007/978-3-319-27261-0_36 (cited on pages 104, 142).

[74] Paul Koebe. “Kontaktprobleme der konformen Abbildung”. In: *Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Klasse* 88 (1936), pages 141–164 (cited on pages 6, 103).

[75] Santosh Kumar, Ten H Lai, and Anish Arora. “Barrier coverage with wireless sensors”. In: *Proceedings of the 11th annual international conference on Mobile computing and networking*. 2005, pages 284–298 (cited on page 6).

[76] KVB. “Liniennetzpläne”. 2022 (cited on page 18).

[77] Kin Chung Kwan, Lok Tsun Sinn, Chu Han, Tien-Tsin Wong, and Chi-Wing Fu. "Pyramid of arclength descriptor for generating collage of shapes." In: *ACM Trans. Graph.* 35.6 (2016), pages 229–1. DOI: 10.1145/2980179.2980234 (cited on page 40).

[78] Zhilin Li and Weihua Dong. "A stroke-based method for automated generation of schematic network maps". In: *International Journal of Geographical Information Science* 24.11 (2010), pages 1631–1647. DOI: 10.1080/13658811003766936 (cited on page 42).

[79] Peter B. Lloyd. "From Modernism to Metro Maps: Mondrian, Beck, and Salomon". In: *Brief Encounters* 1.1 (2017), pages 1–17. DOI: 10.24134/be.v1i1.38 (cited on page 41).

[80] Andre Löffler, Thomas C. van Dijk, and Alexander Wolff. "Snapping Graph Drawings to the Grid Optimally". In: *Graph Drawing and Network Visualization.* 2016, pages 144–151 (cited on page 40).

[81] Jianqiang Luo, Kevin D. Bowers, Alina Oprea, and Lihao Xu. "Efficient Software Implementations of Large Finite Fields GF(2^n) for Secure Storage Applications". In: *ACM Trans. Storage* 8.Article 2 (2012), 2:1–2:27. ISSN: 1553-3077. DOI: 10.1145/2093139.2093141 (cited on pages 166, 167).

[82] Colin McDiarmid and Tobias Müller. "Integer realizations of disk and segment graphs". In: *CoRR* abs/1111.2931 (2011). arXiv: 1111.2931 (cited on pages 7, 104, 105).

[83] Damian Merrick and Joachim Gudmundsson. "Path Simplification for Metro Map Layout". In: *Proceedings of Graph Drawing (GD'06).* Volume 4372. LNCS. Springer, 2007, pages 258–269. DOI: 10.1007/978-3-540-70904-6_26 (cited on pages 18, 42).

[84] Kazuo Misue, Peter Eades, Wei Lai, and Kozo Sugiyama. "Layout Adjustment and the Mental Map". In: *J. Visual Languages and Computing* 6.2 (1995), pages 183–210. DOI: 10.1006/jvlc.1995.1010 (cited on page 27).

[85] Joseph SB Mitchell and Subhash Suri. "Separation and approximation of polyhedral objects". In: *Computational Geometry: Theory and Applications* 5.2 (1995), pages 95–114. DOI: 10.1016/0925-7721(95)00006-U (cited on pages 64, 66).

[86] Moskovsky Metropoliten. "Moscow Metro Map & Central Circle". 2019 (cited on pages 40, 42).

[87] Gabriele Neyer. "Line Simplification with Restricted Orientations". In: *Proceedings of the 6th International Workshop Algorithms and Data Structures (WADS'99).* Volume 1663. LNCS. Springer, 1999, pages 13–24. DOI: 10.1007/3-540-48447-7_2 (cited on pages 18, 42).

[88] Soeren Nickel and Martin Nöllenburg. "Drawing k-linear Metro Maps". In: *The 2nd Schematic Mapping Workshop.* 2019 (cited on pages 8, 17, 103, 151).

[89] Soeren Nickel and Martin Nöllenburg. “Towards Data-Driven Multilinear Metro Maps”. In: *Diagrammatic Representation and Inference - 11th International Conference (Diagrams'20)*. Volume 12169. Lecture Notes in Computer Science. 2020, pages 153–161. DOI: 10.1007/978-3-030-54249-8_12 (cited on pages 8, 17, 39, 41).

[90] Soeren Nickel, Max Sondag, Wouter Meulemans, Stephen G. Kobourov, Jaakko Peltonen, and Martin Nöllenburg. “Multicriteria Optimization for Dynamic Demers Cartograms”. In: *IEEE Trans. Vis. Comput. Graph.* 28.6 (2022), pages 2376–2387. DOI: 10.1109/TVCG.2022.3151227 (cited on pages 2, 6).

[91] Benjamin Niedermann and Jan-Henrik Haunert. “An algorithmic framework for labeling network maps”. In: *Algorithmica* 80.5 (2018). Publisher: Springer, pages 1493–1533. DOI: 10.1007/978-3-319-21398-9_54 (cited on pages 3, 37, 42, 59).

[92] Benjamin Niedermann and Ignaz Rutter. “An Integer-Linear Program for Bend-Minimization in Ortho-Radial Drawings”. In: *Graph Drawing and Network Visualization - 28th International Symposium, GD 2020, Vancouver, BC, Canada, September 16-18, 2020, Revised Selected Papers*. Edited by David Auber and Pavel Valtr. Volume 12590. Lecture Notes in Computer Science. Springer, 2020, pages 235–249. DOI: 10.1007/978-3-030-68766-3_19 (cited on pages 3, 18).

[93] Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. “Efficient Algorithms for Ortho-Radial Graph Drawing”. In: *Proceedings of Computational Geometry (SoCG '19)*. Volume 129. Leibniz International Proceedings in Informatics. 2019. DOI: 10.4230/LIPIcs.SoCG.2019.53 (cited on pages 3, 18, 41).

[94] Frank Nielsen and Richard Nock. “Optimal interval clustering: Application to Bregman clustering and statistical mixture learning”. In: *IEEE Signal Processing Letters* 21.10 (2014). Publisher: IEEE, pages 1289–1292 (cited on page 23).

[95] Martin Nöllenburg. “A Survey on Automated Metro Map Layout Methods”. In: *Proceeding of the 1st Schematic Mapping Workshop*. 2014 (cited on pages 3, 17, 18).

[96] Martin Nöllenburg. “Automated Drawing of Metro Maps”. Master's Thesis. Fakultät für Informatik, Universität Karlsruhe (TH), 2005 (cited on pages 3, 41).

[97] Martin Nöllenburg and Alexander Wolff. “A Mixed-Integer Program for Drawing High-Quality Metro Maps”. In: *Graph Drawing (GD'05)*. Edited by Patrick Healy and Nikola S. Nikolov. Volume 3843. LNCS. Springer Berlin Heidelberg, 2006, pages 321–333. DOI: 10.1007/11618058_29 (cited on pages 3, 8).

[98] Martin Nöllenburg and Alexander Wolff. “Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming”. In: *IEEE Transactions Visualization and Computer Graphics* 17.5 (2011), pages 626–641. DOI: 10.1109/TVCG.2010.81 (cited on pages 18, 19, 23, 24, 26–28, 31, 36, 39, 40, 42, 43, 60).

[99] Phoebe de Nooijer. “Resolving Popular Faces in Curve Arrangements”. Master’s thesis. Utrecht University, 2022 (cited on page 170).

[100] Phoebe de Nooijer, Soeren Nickel, Alexandra Weinberger, Zuzana Masárová, Tamara Mchedlidze, Maarten Löffler, and Günter Rote. “Removing Popular Faces in Curve Arrangements”. In: *The 38th European Workshop on Computational Geometry (EuroCG’22)*. 2022, 38:1–38:8 (cited on page 10).

[101] Phoebe de Nooijer, Soeren Nickel, Alexandra Weinberger, Zuzana Masárová, Tamara Mchedlidze, Maarten Löffler, and Günter Rote. “Removing Popular Faces in Curve Arrangements”. In: *31st International Symposium on Graph Drawing and Network Visualization (GD’23)*. (to appear) (cited on page 10).

[102] Mark Ovenden. “Paris Metro Style in Map and Station Design”. Harrow Weald: Capital Transport Publishing, 2008 (cited on page 41).

[103] Mark Ovenden. “Transit Maps of the world”. New York: Penguin Books, 2015 (cited on page 41).

[104] B Martin Pedersen. “Graphis diagram”. 1988 (cited on page 4).

[105] James S. Plank, Kevin M. Greenan, and Ethan L. Miller. “A Complete Treatment of Software Implementations of Finite Field Arithmetic for Erasure Coding Applications”. Technical report UT-CS-13-717. EECS Department, University of Tennessee, 2013 (cited on pages 166, 167).

[106] James S. Plank, Ethan L. Miller, Kevin M. Greenan, Benjamin A. Arnold, John A. Burnum, Adam W. Disney, and Allen C. McBride. “GF-Complete: A Comprehensive Open Source Library for Galois Field Arithmetic”. Revision 1.03. 2015 (cited on page 166).

[107] Maxwell J Roberts, Elizabeth J Newton, and Maria Canals. “Radi(c)al Departures: Comparing Conventional Octolinear Versus Concentric Circles Schematic Maps for the Berlin U-Bahn/S-Bahn Networks Using Objective and Subjective Measures of Effectiveness”. In: *Information Design Journal (IDJ)* 22.2 (2016). DOI: 10.1075/idj.22.2.04rob (cited on page 41).

[108] Maxwell J. Roberts. “Schematic maps in the laboratory”. In: *Proceeding of the 1st Schematic Mapping Workshop*. 2014 (cited on page 41).

[109] Maxwell J. Roberts. “Underground Maps Unravelled: Explorations in Information Design”. Self-published, 2012 (cited on page 2).

[110] Maxwell J. Roberts. “What’s your theory of effective schematic map design?” In: *Proceeding of the 1st Schematic Mapping Workshop*. 2014 (cited on pages 19, 39, 42, 43).

[111] Maxwell J. Roberts, Hannah Gray, and Jennifer Lesnik. “Preference versus performance: Investigating the dissociation between objective measures and subjective ratings of usability for schematic metro maps and intuitive theories of design”. In: *International Journal of Human-Computer Studies* 98 (2017), pages 109–128. DOI: 10.1016/j.ijhcs.2016.06.003 (cited on pages 18, 41).

[112] Maxwell J. Roberts, Elizabeth J. Newton, Fabio D. Lagattolla, Simon Hughes, and Megan C. Hasler. "Objective versus subjective measures of Paris Metro map usability: Investigating traditional octolinear versus all-curves schematics". In: *International Journal of Human-Computer Studies* 71 (2013), pages 363–386. DOI: 10.1016/j.ijhcs.2012.09.004 (cited on page 18).

[113] Jacob T. Schwartz. "Fast Probabilistic Algorithms for Verification of Polynomial Identities". In: *J. ACM* 27.4 (1980), pages 701–717. DOI: 10.1145/322217.322225 (cited on page 165).

[114] John D. Schwetman. "Harry Beck's London Underground Map: A Convex Lens for the Global City". In: *Transfers* 4.2 (2014). Publisher: Berghahn Journals, pages 86–103. DOI: 10.3167/TRANS.2014.040207 (cited on page 41).

[115] Victor Shoup. "A Computational Introduction to Number Theory and Algebra". 2nd. Cambridge University Press, 2008. DOI: 10.1017/CBO9781139165464 (cited on page 167).

[116] Jonathan Stott, Peter Rodgers, Juan Carlos Martínez-Ovando, and Stephen G. Walker. "Automatic Metro Map Layout Using Multicriteria Optimization". In: *IEEE Transactions Visualization and Computer Graphics* 17.1 (2011), pages 101–114. DOI: 10.1109/TVCG.2010.24 (cited on pages 39, 42).

[117] Jonathan M Stott and Peter Rodgers. "Automatic metro map design techniques". In: *Proceedings of the 22nd International Cartographic Conference*. 2005. DOI: 10.1109/TVCG.2010.24 (cited on page 42).

[118] Shigeo Takahashi, Ken Maruyama, Takamasa Kawagoe, Hsiang-Yun Wu, Kazuo Misue, and Masatoshi Arikawa. "Mental Map Preservation for Progressively Labeling Railway Networks". In: *International Journal of Art, Culture and Design Technologies (IJACDT)* 8.1 (2019), pages 31–50. DOI: 10.4018/IJACDT.2019010103 (cited on page 42).

[119] Roberto Tamassia. "On Embedding a Graph in the Grid with the Minimum Number of Bends". In: *SIAM J. Comput.* 16.3 (1987), pages 421–444. DOI: 10.1137/0216030 (cited on page 3).

[120] Michael J. Tarr and Steven Pinker. "Mental rotation and orientation-dependence in shape recognition". In: *Cognitive Psychology* 21.2 (1989), pages 233–282. DOI: 10.1016/0010-0285(89)90009-1 (cited on page 43).

[121] Soeren Terziadis and Martin Nöllenburg. "Towards Data-Driven Multilinear Metro Maps". In: *Cartogr. J. (Special Issue "Beck at 90 Years")* (). To appear (accepted with minor revision) (cited on pages 8, 17).

[122] The Schematic Mapping Workshop. "The 2nd Schematic Mapping Workshop". 2019 (cited on page 40).

[123] Chiara Vercellino, Paolo Viviani, Giacomo Vitali, Alberto Scionti, Andrea Scarabosio, Olivier Terzo, Edoardo Giusto, and Bartolomeo Montrucchio. "Neural-powered unit disk graph embedding: qubits connectivity for some QUBO problems". In: *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2022, pages 186–196 (cited on page 6).

[124] Cao An Wang. "Finding minimal nested polygons". en. In: *BIT Computer Science section* 31.2 (1991), pages 230–236. ISSN: 0006-3835, 1572-9125. DOI: 10.1007/BF01931283 (cited on pages 63, 64, 101).

[125] Cao An Wang and Edward P. F. Chan. "Finding the Minimum Visible Vertex Distance Between Two Non-Intersecting Simple Polygons". In: *Proceedings of the Second Annual ACM SIGACT/SIGGRAPH Symposium on Computational Geometry, Yorktown Heights, NY, USA, June 2-4, 1986*. ACM, 1986, pages 34–42. DOI: 10.1145/10515.10519 (cited on page 66).

[126] Yu-Shuen Wang and Ming-Te Chi. "Focus+Context Metro Maps". In: *IEEE Transactions Visualization and Computer Graphics* 17.12 (2011), pages 2528–2535. DOI: 10.1109/TVCG.2011.205 (cited on pages 39, 42, 44, 46).

[127] Yu-Shuen Wang and Wan-Yu Peng. "Interactive Metro Map Editing". In: *IEEE Transactions on Visualization and Computer Graphics* 22.2 (2016), pages 1115–1126. DOI: 10.1109/TVCG.2015.2430290 (cited on pages 42, 44).

[128] Yunhai Wang, Xiaowei Chu, Kaiyi Zhang, Chen Bao, Xiaotong Li, Jian Zhang, Chi-Wing Fu, Christophe Hurter, Oliver Deussen, and Bongshin Lee. "ShapeWordle: tailoring wordles using shape-aware archimedean spirals". In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2019). Publisher: IEEE, pages 991–1000 (cited on page 40).

[129] J. Mark Ware and Nigel Richards. "An ant colony system algorithm for automatically schematizing transport network data sets". In: *Proceedings of Evolutionary Computation (CEC'13)*. 2013, pages 1892–1900. DOI: 10.1109/CEC.2013.6557790 (cited on page 42).

[130] J. Mark Ware, George E. Taylor, Suchith Anand, and Nathan Thomas. "Automated Production of Schematic Maps for Mobile Applications". In: *Transactions in GIS* 10.1 (2006), pages 25–42. DOI: 10.1111/j.1467-9671.2006.00242.x (cited on page 42).

[131] Alexander Wolff. "Drawing Subway Maps: A Survey". In: *Informatik – Forschung und Entwicklung* 22.1 (2007), pages 23–44. DOI: 10.1007/s00450-007-0036-y (cited on page 17).

[132] Hsiang-Yun Wu, Benjamin Niedermann, Shigeo Takahashi, and Martin Nöllenburg. "A Survey on Computing Schematic Network Maps: The Challenge to Interactivity". In: *Proceeding of the 2nd Schematic Mapping Workshop*. 2019 (cited on pages 3, 18, 39).

[133] Hsiang-Yun Wu, Benjamin Niedermann, Shigeo Takahashi, Maxwell J. Roberts, and Martin Nöllenburg. "A Survey on Transit Map Layout from Design, Machine, and Human Perspectives". In: *Computer Graphics Forum (Special Issue of EuroVis 2020)* 39.3 (2020), pages 619–646. DOI: 10.1111/cgf.14030 (cited on pages 3, 17, 18, 39, 40, 42, 43).

[134] Hsiang-Yun Wu, Sheung-Hung Poon, Shigeo Takahashi, Masatoshi Arikawa, Chun-Cheng Lin, and Hsu-Chun Yen. "Designing and annotating metro maps with loop lines". In: *Proceedings of Information Visualisation*. IV '15. 2015, pages 9–14. DOI: 10.1109/iV.2015.14 (cited on pages 40, 42, 43).

[135] Hsiang-Yun Wu, Shigeo Takahashi, Chun-Cheng Lin, and Hsu-Chun Yen. "A Zone-Based Approach for Placing Annotation Labels on Metro Maps". In: *Proceedings of Smart Graphics (SG'11)*. Volume 6815. LNCS. Springer, 2011, pages 91–102. DOI: 10.1007/978-3-642-22571-0_8 (cited on page 42).

[136] Hsiang-Yun Wu, Shigeo Takahashi, Chun-Cheng Lin, and Hsu-Chun Yen. "Travel-Route-Centered Metro Map Layout and Annotation". In: *Computer Graphics Forum* 31.3 (2012), pages 925–934. DOI: 10.1111/j.1467-8659.2012.03085.x (cited on pages 40, 43).

[137] Kok-Hoo Yeap and Majid Sarrafzadeh. "Floor-Planning by Graph Dualization: 2-Concave Rectilinear Modules". In: *SIAM Journal on Computing* 22.3 (1993). _eprint: https://doi.org/10.1137/0222035, pages 500–526. DOI: 10.1137/0222035 (cited on page 6).

[138] Yuka Yoshida, Ken Maruyama, Takamasa Kawagoe, Hsiang-Yun Wu, Masatoshi Arikawa, and Shigeo Takahashi. "Progressive Annotation of Schematic Railway Maps". In: *Proceedings of Information Visualisation*. IV '18. 2018, pages 373–378. DOI: 10.1109/iV.2018.00070 (cited on page 42).

[139] ZERO PER ZERO. "City Railway System". 2021 (cited on pages 39, 40, 42, 43, 58).

Information about Ressources

Image Licenses

Images in this thesis with the following licenses have been used:

- CC0 1.0 Universal (CC0 1.0) Public Domain Dedication (link to license)
- Attribution 2.5 Generic (CC BY 2.5 Deed) (link to license)
- Attribution 4.0 International (CC BY 4.0) (link to license)
- Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) (link to license)

Software

The following software has been used for this thesis

- Implementation for Chapter 3 has been done using Python 3.
- Implementation for Chapter 4 has been done using Java.
- Functional analysis in Chapter 6 has been done using Mathematica.
- The largest part of figures and visualizations for this thesis has been done in IPE. This includes most figures in Chapters 1-3, all figures in Chapters 4-7 as well as every poster and presentation that was created by me during my 4 years of study. I would like to thank Otfried Cheong for creating and continuously supporting a software that is made freely accessible to the commuinity entirely for free and that has helped me structure my research for this thesis and beyond.

Soeren Terziadis — Curriculum Vitae

Contact & Personal Details

Full Name: Soeren Daniel Terziadis (né Nickel)
Date of Birth: 24.02.1991
Phone (): +31 6 23856399
Email (): sterziadis@ac.tuwien.ac.at
Website (): https://informatics.tuwien.ac.at/people/sterziadis
ORCID (): 0000-0001-5161-3841

Address:
Urkhovenseweg 502
5641 KV Eindhoven
The Netherlands

Education

Doctoral College: Logical Methods in Computer Science (Affiliated Student) — **Vienna, AT**
Technical University of Vienna [FWF Project W 1255] — *Mar 2021 - Jan 2024 [Expected]*

Ph.D. supervised by Martin Nöllenburg — **Vienna, AT**
Technical University of Vienna — *Oct 2019 - Jan 2024 [Expected]*

Master of Science in Logic and Computation — **Vienna, AT**
Technical University of Vienna — *Sep 2016 - Sep 2019*

Bachelor of Arts in Media Informatics and Information Science — **Regensburg, Germany**
University of Regensburg — *Sep 2013 - Sep 2016*

Employment

Postdoctoral Researcher — **The NETWORKS Project**
TU Eindhoven — *since Dec 2023*

Project Assistant — **WWTF Grant [10.47379/ICT19035]**
TU Wien — *Oct 2022 - Oct 2023*

University Assistant — **TU Wien Innovative Project**
TU Wien — *Oct 2019 - Oct 2022*

Teaching Assistant — **Algorithms and Datastructures**
TU Wien — *Mar 2018 - Jul 2018*

Awards & Scholarships

Best Poster Award — **GD**
The Witness Unit Disk Representability Problem — *2022*

3rd price: 2022 Graph Drawing Contest — **GD**
Manual Challenge: Planar Edge Length Ratio Minimizaiton (with J. Wulms, A. Villedieu) — *2022*

Travel Grant "Mittel zur Förderung von Auslandsbeziehungen" — **TU Wien**
Research stay at the Utrecht University — *2022*

Best Presentation Award — **EuroCG**
Removing Popular Faces in Curve Arrangements by Inserting one more Curve — *2022*

3rd price: 2021 Graph Drawing Contest — **GD**
Manual Challenge: Planar Edge Length Ratio Minimization (with J. Wulms, A. Villedieu) — *2021*

Best Presentation Award — **EuroCG**
Recognition of Unit Disk Graphs for Caterpillars, Embedded Trees, and Outerplanar Graphs — *2021*

Distinguished Young Alumn Award — **TU Wien**
Generating stable Demers Cartograms and Iso-Hexagon Cartograms (Master Thesis) — *2019*

1st price: 2017 Graph Drawing Contest **GD**
Creative Topic: Meal Ingredients (with G. Li, M. Nöllenburg, I. Viola, and H.-Y. Wu) *2019*

Best Presentation Award **Schematic Mapping Workshop**
Drawing k-linear Metro Maps *2018*

Research Visits

Talks given during visit are indicated with a microphone (🎤).

Visiting Dr. Fabian Klute **Barcelona (ES)**
Universitat Politècnica de Catalunya *one week in Mar 2023*
(🎤): "Minimum Link Fencing" (24.03.2023)

Visiting Prof. Marc van Kreveld **Utrecht (NL)**
Utrecht University *one month in May 2022*
(🎤): "Recognition of Unit Disk Graphs" (10.05.2022 Utrecht and 24.05.2022 Eindhoven)

Visiting Prof. Giuseppe Liotta **Perugia (IT)**
University of Perugia *two weeks in Mar 2022*

Attended Conferences – Workshops – PhD Schools

Talks given at conferences are indicated with a microphone (🎤).

31st International Symposium on Graph Drawing and Network Visualization **2023**
Isola delle Femmine (IT) *Sep 20 – Sep 22*
(🎤): "Removing Popular Faces in Curve Arrangements"

PhD school (colocated with Graph Drawing 2023) **2023**
Isola delle Femmine (IT) *Sep 19*

18th European Research Week on Geometric Graphs (invitation only) **2023**
Alcalá de Henares (ES) *Sep 4 – Sep 8*

39th European Workshop on Computational Geometry **2023**
Barcelona (ES) *Mar 29 – Mar 31*

33rd International Symposium on Algorithms and Computation **2022**
Seoul (KR) *Dec 19 – Dec 21*
(🎤): "Minimum Link Fencing"

17th European Research Week on Geometric Graphs (invitation only) **2022**
Leipzig (DE) *Aug 15 – Aug 19*

30th International Symposium on Graph Drawing and Network Visualization **2022**
Tokyo (JP, remote participation) *Sep 13 – Sep 16*

1st Workshop on Computational Cartography **2022**
Bonn (DE) *May 19 – May 20*

3rd Schematic Mapping Workshop **2022**
Bochum (DE) *Apr 21 – Apr 22*

38th European Workshop on Computational Geometry **2022**
Perugia (IT) *Mar 14 – Mar 16*
(🎤): "Removing Popular Faces from Curve Arrangements by Adding One More Curve"

29th International Symposium on Graph Drawing and Network Visualization **2021**
Tübingen (DE) *Sep 15 – Sep 17*
(🎤): "Unit Disk Representations of Embedded Trees, Outerplanar and Multi-legged Graphs"

PhD school (colocated with Graph Drawing 2021) **2021**
Tübingen (DE) *Sep 13 – Sep 14*

37th European Workshop on Computational Geometry **2021**
Saint-Petersburg (RU) *Jun 7 – Jun 11*
(🎤): "Unit Disk Representations of Embedded Trees, Outerplanar and Multi-legged Graphs"

36th European Workshop on Computational Geometry **2020**
Würzburg (DE) *Mar 16 – Mar 18*

CO@Work 2020 summer school **2020**
Berlin (DE) *Sep 14 – Sep 25*

11th International Conference on the Theory and Application of Diagrams **2020**
Tallin (EE) *Aug 24 – Aug 28*
(🎤): "Towards Data-Driven Multilinear Metro Maps"

27th International Symposium on Graph Drawing and Network Visualization **2019**
Průhonice/Prague (CZ) *Sep 18 – Sep 20*
(🎤): "Computing Stable Demers Cartograms"

PhD school (colocated with Graph Drawing 2019) **2019**
Prague (CZ) *Sep 16 – Sep 17*

16th European Research Week on Geometric Graphs (invitation only) **2019**
Strobl (AT) *Nov 18 – Nov 22*

9th Workshop on Graph Classes, Optimization, and Width Parameters **2019**
Vienna (AT) *Sep 23 – Sep 26*

2nd Schematic Mapping Workshop **2019**
Vienna (AT) *Apr 11 – Apr 12*
(🎤): "Drawing k-linear Metro maps"

34th European Workshop on Computational Geometry **2018**
Berlin (DE) *Mar 21 – Mar 23*

Teaching

Teaching Assistance

- Graph Drawing Algorithms (exercise project supervision)
  - SS'20: Student projects won two 1^{st} and one 2^{nd} prize in the 27th Annual Graph Drawing Contest
  - SS'21: Student projects won 1^{st} and 2^{nd} prize in the 28th Annual Graph Drawing Contest
  - SS'22: Student projects won 1^{st} prize in the 29th Annual Graph Drawing Contest
  - SS'23: (ongoing)
- Algorithmic Geometry (WS'19/20, WS'20/21, WS'21/22, WS'22/23)
- Wissenschaftliches Arbeiten (WS'20/21, SS'21, WS'21/22, SS'22, WS'22/23)
- Seminar in Algorithms, Graphs and Geometry (WS'20/21, WS'21/22, WS'22/23)
- Project in Computer Science (WS'20/21, WS'21/22)

Co-Supervision of Students

- Samantha Fuchs (Master Thesis: *SAT-based Optimization of Octolinear Metro Map Layouts*)
- Sebastian Adam (Master Thesis: *Elastic Set Visualization*)
- Peter Neubauer (Bachelor Thesis: *Exact/Heuristic Recognition of Monotone Lobster Graphs on a Grid*)
- Matthias Mayr (Bachelor Thesis: *Grid-based metro map layouts with estimated station location*)
- Lukas Fink (Bachelor Thesis: *Unit Disk Representations of Caterpillars*)
- Thomas Depian (Master Thesis: *Constrained External Labeling*), ongoing
- Matthias Eder (Master Thesis: *Geometric Railway Track Representation*), ongoing

Publications

Journal Articles

Tobias Batik, Soeren Terziadis, Yu-Shuen Wang, Martin Nöllenburg, and Hsiang-Yun Wu. **Shape-Guided Mixed Metro Map Layout**. *Comput. Graph. Forum*, 41(7):495–506, 2022. presented at Pacific Graphics 2022.

Soeren Nickel, Max Sondag, Wouter Meulemans, Stephen G. Kobourov, Jaakko Peltonen, and Martin Nöllenburg. **Multicriteria Optimization for Dynamic Demers Cartograms**. *IEEE Trans. Vis. Comput. Graph.*, 28(6):2376–2387, 2022.

Journal Articles under Review

Sujoy Bhore, Maarten Löffler, Soeren Nickel, and Martin Nöllenburg. **Unit Disk Representations of Embedded Trees, Outerplanar and Multi-legged Graphs**. *J. Graph Algorithms Appl.* Under review.

Soeren Nickel and Martin Nöllenburg. **Towards Data-Driven Multilinear Metro Maps**. *Cartogr. J. (Special Issue "Beck at 90 Years").* Under review.

Martin Nöllenburg, Manuel Sorge, Soeren Terziadis, Anaïs Villedieu, Hsiang-Yun Wu, and Jules Wulms. **Planarizing Graphs and Their Drawings by Vertex Splitting**. *J. Comput. Geom.* Under review.

Conference Publication

Sujoy Bhore, Fabian Klute, Maarten Löffler, Martin Nöllenburg, Soeren Terziadis, and Anaïs Villedieu. **Minimum Link Fencing**. In *33rd International Symposium on Algorithms and Computation (ISAAC'22)*, volume 248 of *LIPIcs*, pages 34:1–34:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Martin Nöllenburg, Manuel Sorge, Soeren Terziadis, Anaïs Villedieu, Hsiang-Yun Wu, and Jules Wulms. **Planarizing Graphs and Their Drawings by Vertex Splitting**. In *30th International Symposium on Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *Lecture Notes in Computer Science*, pages 232–246, 2022.

Sujoy Bhore, Maarten Löffler, Soeren Nickel, and Martin Nöllenburg. **Unit Disk Representations of Embedded Trees, Outerplanar and Multi-legged Graphs**. In *29th International Symposium on Graph Drawing and Network Visualization (GD'21)*, volume 12868 of *Lecture Notes in Computer Science*, pages 304–317, 2021.

Soeren Nickel and Martin Nöllenburg. **Towards Data-Driven Multilinear Metro Maps**. In *Diagrammatic Representation and Inference - 11th International Conference (Diagrams'20)*, volume 12169 of *Lecture Notes in Computer Science*, pages 153–161, 2020.

Matthias Hummel, Fabian Klute, Soeren Nickel, and Martin Nöllenburg. **Maximizing Ink in Partial Edge Drawings of k-plane Graphs**. In *27th International Symposium on Graph Drawing and Network Visualization (GD'19)*, volume 11904 of *Lecture Notes in Computer Science*, pages 323–336, 2019.

Soeren Nickel, Max Sondag, Wouter Meulemans, Markus Chimani, Stephen G. Kobourov, Jaakko Peltonen, and Martin Nöllenburg. **Computing Stable Demers Cartograms**. In *27th International Symposium on Graph Drawing and Network Visualization (GD'19)*, volume 11904 of *Lecture Notes in Computer Science*, pages 46–60, 2019.

Poster

Giuseppe Liotta, Maarten Löffler, Fabrizio Montecchiani, Alessandra Tappini, and Soeren Terziadis. **The Witness Unit Disk Representability Problem**. In *30th International Symposium on Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *Lecture Notes in Computer Science*, pages 483–486, 2022. Poster.

Tobias Batik, Soeren Terziadis, Yu-Shuen Wang, Martin Nöllenburg, and Hsiang-Yun Wu. **Mixed Metro Maps with User-Specified Motifs**. In *29th International Symposium on Graph Drawing and Network Visualization (GD'21)*, volume 12868 of *Lecture Notes in Computer Science*, pages 304–317, 2021. Poster.

Informal Conferences

Phoebe de Nooijer, Soeren Nickel, Alexandra Weinberger, Zuzana Masárová, Tamara Mchedlidze, Maarten Löffler, and Günter Rote. **Removing Popular Faces in Curve Arrangements**. In *The 38th European Workshop on Computational Geometry (EuroCG'22)*, pages 38:1–38:8, 2022.

Samantha Fuchs, Soeren Nickel, and Martin Nöllenburg. **Logic-based Computation of Metro Maps**. In *The 3rd Schematic Mapping Workshop*, 2022.

Julian Haumer, Soeren Nickel, Martin Nöllenburg, and Markus Wallinger. **Towards Automated Multilevel Schematic Maps**. In *The 3rd Schematic Mapping Workshop*, 2022.

Soeren Nickel, Martin Nöllenburg, Manuel Sorge, Anaïs Villedieu, Hsiang-Yun Wu, and Jules Wulms. **Planarizing Graphs and their Drawings by Vertex Splitting**. In *The 38th European Workshop on Computational Geometry (EuroCG'22)*, pages 14:1–14:8, 2022.

Sujoy Bhore, Fabian Klute, Maarten Löffler, Martin Nöllenburg, Soeren Terziadis, and Anaïs Villedieu. **Minimum Link Fencing**. In *The 37th European Workshop on Computational Geometry (EuroCG'21)*, pages 55:1–55:9, 2021.

Sujoy Bhore, Soeren Nickel, and Martin Nöllenburg. **Recognition of Unit Disk Graphs for Caterpillars, Embedded Trees, and Outerplanar Graphs**. In *The 37th European Workshop on Computational Geometry (EuroCG'21)*, pages 50:1–50:9, 2021.

Soeren Nickel and Martin Nöllenburg. **Drawing k-linear Metro Maps**. In *The 2nd Schematic Mapping Workshop*, 2019.

Matthias Hummel, Fabian Klute, Soeren Nickel, and Martin Nöllenburg. **Maximizing Ink in Partial Edge Drawings of k-plane Graphs**. In *The 34th European Workshop on Computational Geometry (EuroCG'18)*, pages 50:1–50:6, 2018.