UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Recognition of outer k-planar graphs

Author:
Ivan SHEVCHENKO

Supervisor:
Dr. Alexander WOLFF

A thesis submitted in fulfillment of the requirements
for the degree of Bachelor of Science

in the

Department of Computer Sciences and Information Technologies
Faculty of Applied Sciences

Lviv 2025

“*Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.*”

Dave Barry

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Recognition of outer k-planar graphs

by Ivan Shevchenko

Abstract

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

List of Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
WSF	**W**hat (it) **S**tands **F**or

Physical Constants

Speed of Light	$c_0 = 2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$ (exact)

List of Symbols

a	distance	m
P	power	W ($\mathrm{J\,s^{-1}}$)
ω	angular frequency	rad

For/Dedicated to/To my. . .

Chapter 1

Introduction

1.1 Motivation

Many different processes and concepts can be represented in the modern world using planar graphs. Throughout this field's long history, many efficient algorithms were developed for computers to deal with this abstraction.

Planarity, though, imposes massive restrictions on the graph, so the generalization to the graphs that are "almost" planar can be valuable. In the case of this work, the generalization to k-planar graphs is considered. A graph is k-planar if it can be drawn on the plane with each edge having at most k intersections.

Unfortunately, the task of recognizing even 1-planar graphs is known to be $\mathcal{NP}$-hard [**NP-hard-1p**], so it is not possible to efficiently test whether a graph is 1-planar or not.

Outerplanar graphs are a specialization of planar graphs in which all vertices lie on the same (outer) face. This subclass is interesting as some of the problems known to be $\mathcal{NP}$-complete for planar graphs can be solved for outer planar in polynomial time. Among them are chromatic number, Hamiltonian path and Hamiltonian circuit.

1.1.1 Problem formulation

There is an algorithm that tests outer k-planarity in $\mathcal{O}(k! \cdot n^{3k+\mathcal{O}(1)})$ time [9]. There are, however, the linear time algorithms for testing outer planarity [12] and outer 1-planarity [7, 1], which use some insights from low values of k, which is a significant improvement compared to the general algorithm.

For the value of k as small as two, testing would require at least $\mathcal{O}(n^6)$, which could be unbearable for big graphs.

Furthermore, all existing algorithms are purely combinatorial, thus gaining nothing from optimizations already discovered for solving other similar $\mathcal{NP}$-hard problems such as satisfiability (SAT) or integer linear programming (ILP).

1.2 Contributions

This thesis aims to reduce the asymptotic time complexity of such tests for small values of k.

Another approach that will be discovered in the thesis is to transform the outer k-planar test into a satisfiability (SAT) and an integer linear programming (ILP) problem for which highly optimized solvers exist. The results can be compared afterward to combinatorial methods.

A possible extension to this problem could be a modification of outer k-planar straight-line drawings to use simple curves (e.g., quadratic Bézier curves) for edges to separate the intersections.

1.3 Structure Of The Thesis

Chapter 2

Related Work

Already in the 1980s, researchers in the field of graph drawing acknowledged the importance of reducing the edge crossings for improving visualisation clarity [2]. The suspicion that a drawing with fewer edge crossings was easier to comprehend was later confirmed by several experimental studies [11]. These studies showed that such crossing minimising graph representations significantly improves humans' ability to interpret the imposed structure, particularly when dealing with complex or large graphs.

The ideal form of crossing minimising drawings – planar ones, has been focused on by the research community for a long time, with algorithms for recognition of planar graphs presented already in 1974 [8]. However, requiring the drawing to be completely crossing free imposes huge limitations on an underlying graph. While providing a clean structure, these restrictions are often too constraining for many real-world graphs, especially large ones. This has led to a growing interest in exploring graphs close to being planar. Such graphs allow a limited number of crossings while still retaining some of the beneficial structural properties of planar ones.

Despite the potential benefits of studying graphs beyond strict planarity, research in this area has been relatively sparse. In 2018, a survey by Didimo et al. [4] offered an overview of the state of the research on graph drawing beyond planarity. The survey highlighted key contributions and outlined the challenges within this area of graph theory.

2.1 Hardness of relaxation

Most relaxations of strict planarity dramatically increase the complexity of recognising such graphs. So, the general problem of minimising edge crossings in a graph drawing was known to be computationally intractable already in 1983 when Garey and Johnson [6] demonstrated that the CROSSING NUMBER problem determining whether a given graph can be drawn with at most k crossings, is NP-complete. Their proof relies on a reduction from the OPTIMAL LINEAR ARRANGEMENT problem, which is known to be NP-hard.

Minimising the amount of local crossing is also hard. Korzhik and Mohar [10] showed that testing the 1-planarity, recognising whether a graph can be drawn with at most one crossing per edge, is NP-complete. Later, Cabello and Mohar [3] showed that testing 1-planarity is still NP-complete even for near-planar graphs, graphs that can be obtained from planar by adding a single edge.

Given the complexity of recognising k-planarity, researchers considered exploring more restrictive classes of graphs, hoping that imposed limitations could simplify the recognition. One of the considered classes is outer k-planar graphs, a subclass of k-planar graphs in which all vertices lie on the outer face of a drawing.

2.2 Efficient recognition of some outer k-planar graph

Despite the general recognition problem for outer k-planar graphs remains NP-hard, efficient algorithms have been developed for specific values of k.

For $k = 0$, the recognition task simplifies to an outerplanarity test. Regocnition can be accomplished by augmenting the graph with a new vertex connected to all existing ones and testing whether the resulting graph is planar. An alternative approach, described in [12], introduces the concept of 2-reducible graphs, which are totally disconnected or can be made totally disconnected by repeated deletion of edges adjacent to a vertex with a degree at most two. The outerplanarity test proposed in this study is based on an algorithm for testing 2-reducibility.

In the case of $k = 1$, two research groups independently presented linear-time algorithms in [7] and [1]. Both methodologies utilise the SPQR decomposition of a graph for their testing mechanisms. Notably, the latter solution extends the graph to a maximal outer 1-planar configuration if such a drawing exists, unlike the former, which employs a bottom-up strategy which does not require any transformations to the original graph.

For all other values of k, no research had been conducted until recently when a group of researchers proposed an algorithm for the general case, which we discuss in the next section.

2.3 Recognizing general outer k-planar graphs

For a given outer k-planar drawing of a graph, Firman et al. [5] proposed a method for constructing triangulation with the property that each edge of the triangulation is crossed by at most k edges of the graph. Since the edges of the triangulation do not necessarily belong to the original graph, they are termed *links* to distinguish them from the graph's original edges. The construction is done recursively.

Initially, the algorithm selects an edge on the outer face and labels it as the active link. At each recursive step, the active link partitions the graph into two regions: a left part already triangulated and a right part not yet explored. The objective of each step is to triangulate the right portion. To achieve this, a splitting vertex is chosen within the right region, dividing it into two smaller subregions. The splitting vertex is selected so that the two new links, connecting the split vertex with the endpoints of the active link, are each intersected by at most k edges, which allows including them into the triangulation. The algorithm then recurses, treating each of these newly formed links as the active one.

Later, Kobayashi et al. [9] extended this approach to address the recognition problem for outer k-planar graphs. In contrast to the triangulation task, where the drawing is provided, the recognition problem requires determining whether a given graph admits an outer k-planar drawing. Although the core idea remains analogous, the absence of a drawing requires the exploration of all possible configurations. Here, each recursive step verifies whether the unexplored right portion of the graph can be drawn as an outer k-planar graph that is compatible with the left part.

Moreover, instead of relying on recursion, the method utilises a dynamic programming approach. This framework combines solutions of smaller subproblems retrieved from a table to solve larger ones. To populate this table, the algorithm iterates over all possible configurations corresponding to different recursion steps. Several parameters characterise each such configuration. The first parameter is the active link – a pair of vertices that divides the graph into left and right regions. The second parameter is a set of vertices in the right part, which is not uniquely determined as in the triangulation case. Additionally, the configuration depends on the order in which edges intersect the active link and the number of intersections on the right side for each one of them. These parameters are used to ensure that the drawing of the right portion is compatible with the left part. For each configuration, the algorithm considers all possible ways to split the right region further. For each of them, the method checks whether these splits are compatible with each other and the left part of the drawing.

Using the restriction on the number of edges crossing each link, the authors demonstrated that for a fixed k, the number of possible right subgraphs grows only polynomially with respect to the size of the graph. They preceded by arguing that the overall time complexity of the algorithm is $2^{O(k \log k)} n^{3k+O(1)}$, showing that the algorithm is efficient for a fixed parameter k.

2.4 Our contribution

A common drawback of the methods described above is the lack of practical validation. Although these algorithms have been analysed and discussed in a theoretical context, they have not been implemented or empirically tested. This thesis addresses this gap by implementing the most recent recognition algorithm and introducing two alternative approaches based on reductions to Integer Linear Programming (ILP) and Satisfiability (SAT). The performance and efficiency of these methods are then evaluated, demonstrating their practical applicability and limitations.

Chapter 3

Theoretical Background

Chapter 4

Proposed Solution

4.1 Problem Formulation

4.1.1 ILP Solver

The result of the integral linear program should be an arrangement of vertices on a line. Given the arrangement of the vertices, we can definitively identify whether two edges cross each other. Indeed, consider two edges, uv and st. Without loss of generality, we can assume that u is located before v in the arrangement, s before t, and u before s. Under these assumptions, the edges uv and st cross if and only if s is located before v and t after v.

The arrangement is represented using the so-called "ordering variables". For every pair of vertices u and v, we create a binary variable $a_{u,v}$ that defines the order in which they appear in the arrangement. $a_{u,v} = 1$ indicates that u is located before v and vice versa, and $a_{u,v} = 0$ indicates that v is located before u or u and v is the same vertex.

For these variables, it is crucial to ensure transitivity. That is, if $a_{u,v} = 1$ and $a_{v,w} = 1$ which means u is located before v and v is located before w, then u must be located before w, so the following should hold $a_{u,w} = 1$. This can be ensured by the following constraint: $a_{u,w} \geqslant a_{u,v} + a_{v,w} - 1$. This constraint will restrict the value of $a_{u,w}$ if and only if both $a_{u,v}$ and $a_{v,w}$ equal 1. Otherwise, the constraint will have no impact on the system at all.

Having the arrangement of the vertices, we can now deduce for each pair of edges uv and st whether they intersect or not. Naturally, there are 24 different arrangements of the vertices, as demonstrated in figure **??**. Among them, there are only eight in which the edges intersect. Thus, the edge uv crosses the edge st if and only if one of the following holds:

- $a_{u,s} = 1$, $a_{s,v} = 1$, and $a_{v,t} = 1$
- $a_{u,t} = 1$, $a_{t,v} = 1$, and $a_{v,s} = 1$
- $a_{v,s} = 1$, $a_{s,u} = 1$, and $a_{u,t} = 1$
- $a_{v,t} = 1$, $a_{t,u} = 1$, and $a_{u,s} = 1$
- $a_{s,u} = 1$, $a_{u,t} = 1$, and $a_{t,v} = 1$
- $a_{t,u} = 1$, $a_{u,s} = 1$, and $a_{s,v} = 1$
- $a_{s,v} = 1$, $a_{v,t} = 1$, and $a_{t,u} = 1$
- $a_{t,v} = 1$, $a_{v,s} = 1$, and $a_{s,u} = 1$

To describe this in the linear program, we create a binary variable $c_{uv,st}$ for every pair of edges uv and st. $c_{uv,st} = 0$ indicates that the edge uv does not cross st. To ensure the correctness of these values, we impose eight constraints on each variable, one for each case from above. For example, considering the case $a_{u,s} = 1$, $a_{s,v} = 1$, and $a_{v,t} = 1$, we impose the following restriction: $c_{uv,st} \geqslant a_{u,s} + a_{s,v} + a_{v,t} - 2$, which ensures that $c_{uv,st}$ equals 1 whenever the vertices are ordered as $usvt$. Making this for each case restricts $c_{uv,st}$ to the value 1 if the vertices are arranged in one of the eight "intersecting" configurations, ensuring that $c_{uv,st} = 0$ is possible only if uv does not cross st.

The algorithm's objective is to minimize the maximal number of crossings per edge. This value can be written as follows: $\max_{uv \in E(G)} \sum_{st \in E(G)} c_{uv,st}$. Unfortunately, it cannot represent an objective function for a linear program as the max operation is not linear. To solve this, we introduce a new integer variable k. To ensure that it is equal to the objective value, we impose the following constraint on k: $k \geqslant \sum_{st \in E(G)} c_{uv,st}$ for each edge $uv \in E(G)$.

So, the integer linear program can be described as follows:

$$\begin{array}{lrll}
\textbf{minimize} & k & & \\
\textbf{subject to} & k \geqslant & \displaystyle\sum_{st \in E(G)} c_{uv,st}, & \forall uv \in E(G) \\
& c_{uv,st} \geqslant & a_{u,s} + a_{s,v} + a_{v,t} - 2, & \forall uv, st \in E(G) \\
& c_{uv,st} \geqslant & a_{u,t} + a_{t,v} + a_{v,s} - 2, & \forall uv, st \in E(G) \\
& c_{uv,st} \geqslant & a_{v,s} + a_{s,u} + a_{u,t} - 2, & \forall uv, st \in E(G) \\
& c_{uv,st} \geqslant & a_{v,t} + a_{t,u} + a_{u,s} - 2, & \forall uv, st \in E(G) \\
& c_{uv,st} \geqslant & a_{s,u} + a_{u,t} + a_{t,v} - 2, & \forall uv, st \in E(G) \\
& c_{uv,st} \geqslant & a_{t,u} + a_{u,s} + a_{s,v} - 2, & \forall uv, st \in E(G) \\
& c_{uv,st} \geqslant & a_{s,v} + a_{v,t} + a_{t,u} - 2, & \forall uv, st \in E(G) \\
& c_{uv,st} \geqslant & a_{t,v} + a_{v,s} + a_{s,u} - 2, & \forall uv, st \in E(G) \\
& a_{u,w} \geqslant & a_{u,v} + a_{v,w} - 1, & \forall u, v, w \in V(G) \\
& c_{uv,st} \in & \{0, 1\}, & \forall uv, st \in E(G) \\
& a_{u,v} \in & \{0, 1\}, & \forall u, v \in V(G)
\end{array}$$

4.1.2 SAT Solver

Another approach to solving this problem is to check for a specific k whether the given graph is outer-k-planar. This check can be encoded as a boolean satisfiability problem. This problem asks whether it is possible to assign logic values TRUE or FALSE so that all disjunctive clauses are satisfied. A disjunctive clause is a single literal or a disjunction of several. Literal is either a variable or a negation of a variable, with the former being the positive and the latter the negative literal.

Similarly to the ILP algorithm described in 4.1.1, this algorithm uses the same "ordering variables" $a_{u,v}$ for each pair of vertices u and v that represent the arrangement of the vertices. If the boolean variable $a_{u,v}$ is TRUE, the vertex u is located before the vertex v and vice versa otherwise.

Similarly, these variables must account for transitivity, which means that for every triple of vertices u, v, and w $a_{u,v} \equiv \text{TRUE}$ and $a_{v,w} \equiv \text{TRUE}$ implies $a_{u,w} \equiv \text{TRUE}$. This can be written as follows: $a_{u,v} \wedge a_{v,w} \rightarrow a_{u,w}$. Expanding the implication, this transforms into $\overline{a_{u,v} \wedge a_{v,w}} \vee a_{u,w}$. After applying De Morgan's law, we receive $\overline{a_{u,v}} \vee \overline{a_{v,w}} \vee a_{u,w}$, which represents a clause in the SAT problem.

The next step is to represent the crossing variables $c_{uv,st}$ in terms of the ordering ones for each pair of edges uv and st. Similarly to the ILP algorithm, we can restrict $c_{uv,st}$ to TRUE if uv and st cross by adding new clauses to the problem. The clauses are constructed by making the implications for each of the eight intersecting cases shown in figure **??**, expanding them, and applying De Morgan's law. For example, for the case $a_{u,s} = 1$, $a_{s,v} = 1$, and $a_{v,t} = 1$, we start with the logical equation as follows: $a_{u,s} \wedge a_{s,v} \wedge a_{v,t} \rightarrow c_{uv,st}$. Afterwards, we expand the implication: $\overline{a_{u,s} \wedge a_{s,v} \wedge a_{v,t}} \vee c_{uv,st}$. Finally, we apply De Morgan's law: $\overline{a_{u,s}} \vee \overline{a_{s,v}} \vee \overline{a_{v,t}} \vee c_{uv,st}$ receiving one of the eight clauses for $c_{uv,st}$.

The last step in the construction of the problem is to count the number of crossings for each edge. The goal of this solver is to check whether the number of crossings can be smaller or equal to some constant k for each edge. To ensure this, we can build a set of clauses that prevent the problem from being satisfiable if the value k is too small. To do so, for every edge e_0, we consider all combinations of $e_1, e_2, \dots, e_{k+1}$ for each of which we construct the following clause: $\overline{c_{e_0,e_1}} \vee \overline{c_{e_0,e_2}} \vee \cdots \vee \overline{c_{e_0,e_{k+1}}}$. Doing so, we ensure that for each edge e_0, no $k+1$ different edges intersect e_0, which effectively means that each edge has at most k crossings if all clauses are satisfied.

Chapter 5

Experiments and Results

Chapter 6

Conclusions

Appendix A

Appendix

Bibliography

[1] Christopher Auer et al. “Outer 1-Planar Graphs”. In: *Algorithmica* 74.4 (Apr. 2016), pp. 1293–1320. ISSN: 1432-0541. DOI: 10.1007/s00453-015-0002-1.

[2] Carlo Batini, Enrico Nardelli, and Roberto Tamassia. “A layout algorithm for data flow diagrams”. In: *IEEE Transactions on Software Engineering* SE-12 (1986), pp. 538–546. URL: https://api.semanticscholar.org/CorpusID:12932438.

[3] Sergio Cabello and Bojan Mohar. “Adding One Edge to Planar Graphs Makes Crossing Number and 1-Planarity Hard”. In: *SIAM Journal on Computing* 42.5 (2013), pp. 1803–1829. DOI: 10.1137/120872310. eprint: https://doi.org/10.1137/120872310. URL: https://doi.org/10.1137/120872310.

[4] Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. “A Survey on Graph Drawing Beyond Planarity”. In: *ACM Comput. Surv.* 52.1 (Feb. 2019). ISSN: 0360-0300. DOI: 10.1145/3301281. URL: https://doi.org/10.1145/3301281.

[5] Oksana Firman et al. “Bounding the Treewidth of Outer k-Planar Graphs via Triangulations”. In: *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*. Ed. by Stefan Felsner and Karsten Klein. Vol. 320. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 14:1–14:17. ISBN: 978-3-95977-343-0. DOI: 10.4230/LIPIcs.GD.2024.14. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.GD.2024.14.

[6] M. R. Garey and D. S. Johnson. “Crossing Number is NP-Complete”. In: *SIAM Journal on Algebraic Discrete Methods* 4.3 (1983), pp. 312–316. DOI: 10.1137/0604033. eprint: https://doi.org/10.1137/0604033. URL: https://doi.org/10.1137/0604033.

[7] Seok-Hee Hong et al. “A Linear-Time Algorithm for Testing Outer-1-Planarity”. In: *Algorithmica* 72.4 (Aug. 2015), pp. 1033–1054. ISSN: 1432-0541. DOI: 10.1007/s00453-014-9890-8.

[8] John E. Hopcroft and Robert Endre Tarjan. “Efficient Planarity Testing”. In: *J. ACM* 21 (1974), pp. 549–568.

[9] Yasuaki Kobayashi, Yuto Okada, and Alexander Wolff. “Recognizing 2-Layer and Outer k-Planar Graphs”. In: *Proc. 41st Annu. Sympos. Comput. Geom. (SoCG)*. Ed. by Oswin Aichholzer and Haitao Wang. LIPIcs. To appear. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. URL: https://arxiv.org/abs/2412.04042.

[10] Vladimir P. Korzhik and Bojan Mohar. “Minimal Obstructions for 1-Immersions and Hardness of 1-Planarity Testing”. In: *Journal of Graph Theory* 72.1 (2013), pp. 30–71. DOI: https://doi.org/10.1002/jgt.21630. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jgt.21630. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jgt.21630.

[11] Helen Purchase. “Which aesthetic has the greatest effect on human understanding?” In: *Graph Drawing*. Ed. by Giuseppe DiBattista. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 248–261. ISBN: 978-3-540-69674-2.

[12] Manfred Wiegers. “Recognizing outerplanar graphs in linear time”. In: *Graph-Theoretic Concepts in Computer Science*. Ed. by Gottfried Tinhofer and Gunther Schmidt. Springer, 1987, pp. 165–176.