Ukrainian Catholic University

Bachelor Thesis

Recognition of outer k-planar graphs

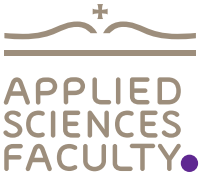
Author:
Ivan Shevchenko

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 $\quad c_0 = 2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$ (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 [2], 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 [4]. There are, however, the linear time algorithms for testing outer planarity [5] and outer 1-planarity [3, 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

Let's cite! The Einstein's journal paper [**einstein**] and the Dirac's book [**dirac**] are physics-related items. The [**Chien_2019**] is a conference paper, [**cpu_freq**] is an example of the *Internet source*, and [**gpudirect**] – **manual**.

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:

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

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 $a_{u,v} \equiv$ TRUE and $a_{v,w} \equiv$ TRUE implies $a_{u,w} \equiv$ 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 $e_1, e_2, \ldots, e_{k+1}$ 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] Alexander Grigoriev and Hans L. Bodlaender. "Algorithms for Graphs Embeddable with Few Crossings per Edge". In: *Algorithmica* 49.1 (Sept. 2007), pp. 1–11. ISSN: 1432-0541. DOI: 10.1007/s00453-007-0010-x.

[3] 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.

[4] Yasuaki Kobayashi, Yuto Okada, and Alexander Wolff. *Recognizing 2-Layer and Outer k-Planar Graphs*. unpublished manuscript. 2024.

[5] 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.