Efficient Algorithms for Graph-Theoretic and Geometric Problems

Floderus, Peter

2015

Efficient Algorithms

for Graph-Theoretic and Geometric Problems

Peter Floderus

LUND
UNIVERSITY

Faculty of Science
Centre for Mathematical Sciences
Mathematics

Centre for Mathematical Sciences
Lund University
Box 118
S–221 00 Lund
Sweden

E-mail: pflo@maths.lth.se

Printed in Sweden

ISSN 1404-0034
ISBN 978-91-7623-277-4

Abstract

This thesis studies several different algorithmic problems in graph theory and in geometry. The applications of the problems studied range from circuit design optimization to fast matrix multiplication.

First, we study a graph-theoretical model of the so called ”firefighter problem”. The objective is to save as much as possible of an area by appropriately placing firefighters. We provide both new exact algorithms for the case of general graphs as well as approximation algorithms for the case of planar graphs.

Next, we study drawing graphs within a given polygon in the plane. We present asymptotically tight upper and lower bounds for this problem

Further, we study the problem of Subgraph Isormorphism, which amounts to decide if an input graph (pattern) is isomorphic to a subgraph of another input graph (host graph). We show several new bounds on the time complexity of detecting small pattern graphs. Among other things, we provide a new framework for detection by testing polynomials for non-identity with zero.

Finally, we study the problem of partitioning a 3D histogram into a minimum number of 3D boxes and it's applications to efficient computation of matrix products for positive integer matrices. We provide an efficient approximation algorithm for the partitioning problem and several algorithms for integer matrix multiplication. The multiplication algorithms are explicitly or implicitly based on an interpretation of positive integer matrices as 3D histograms and their partitions.

Acknowledgements

First of all, I want to thank my advisor, Andrzej Lingas for all his help. He has been very helpful and patient with me during these five years and his importance to this thesis cannot be overstated.

Another important person is Therese Biedl that I visited in Canada, her enthusiasm was a continuous source of inspiration during both my stays at University of Waterloo and she has had a big influence on my research interests.

I would also like to thank my other co-authors, Christos Levcopolous for his inspiration and helpfulness with my writing, Dzmitry Sledneu for his help filling out my knowledge gaps, Jesper Jansson whom I visited in Japan and showed me several interesting problems that kept me awake at night.

During my stay at the Mathematics department in Lund, I met many nice people, Anna-Maria, Jan-Fredrik, Alexandru and Joe are just a few of too many to mention, but all of them played an important role in the completion of this thesis.

Finally, I would like to thank my family, Britt, Per, Anders and Lotta, all my friends and of course Jenny, you are all very important to me.

Contents

Preface

The order of the papers presented is chronological. The first part about the Firefighter Problem was written in 2010. The second part about Graph Drawing, was written during my stay at University of Waterloo in 2011. The third part consists of two papers, both regarding the Subgraph Isomorphism problem, which were written during 2012 and 2013. The final part is about matrix multiplication using a geometric decomposition, it was written during 2014.

The papers comprising the thesis have been published as following:

1. Peter Floderus, Andrzej Lingas and Mia Persson, *Towards More Efficient Infection and Fire Fighting*, International Journal of Foundations of Computer Science, 2013, volume 24, number 1, pages 3-14.

2. Therese C. Biedl and Peter Floderus, *Drawing Planar Graphs on Points Inside a Polygon*, Mathematical Foundations of Computer Science (MFCS) 2012 - 37th International Symposium, Bratislava, Slovakia, August 27-31, 2012, Proceedings, Pages 172-183.

3. Peter Floderus, Miroslaw Kowaluk, Andrzej Lingas and Eva-Marta Lundell, *Induced Subgraph Isomorphism: Are Some Patterns Substantially Easier Than Others?*, Computing and Combinatorics (COCOON) - 18th Annual International Conference, Sydney, Australia, August 20-22, 2012. Proceedings, Pages 37-48.

4. Peter Floderus, Miroslaw Kowaluk, Andrzej Lingas and Eva-Marta Lundell, *Detecting and Counting Small Pattern Graphs*, Algorithms and Computation - 24th International Symposium (ISAAC), Hong Kong, China, December 16-18, 2013, Proceedings, Pages 547-557.

5. Peter Floderus, Jesper Jansson, Christos Levcopoulos, Andrzej Lingas and Dzmitry Sledneu, *3D Rectangulations and Geometric Matrix Multiplication*, Algorithms and Computation - 25th International Symposium (ISAAC), Jeonju, Korea, December 15-17, 2014, Proceedings, Pages 65-78.

Introduction

0.1 The Firefighter Problem

This section is concerned with the so called "Firefighter Problem". The problem models a fire spreading or other similar processes, for example, a disease spreading in a population or a computer virus in a network.

The problem is formulated as follows: Given a graph, each vertex can be in one of three states: *on fire, protected* or *vulnerable*. Initially, a distinguished source vertex is on fire and the remaining vertices are vulnerable.

The fire spreads throughout the graph in discrete time steps, at each time step, the fire spreads from all vertices on fire to all adjacent vulnerable vertices. We can place firefighters at vulnerable vertices to make them protected, preventing the fire to spread to them. A budget integer B, bounds how many firefighters we can place at each time step. The firefighters are placed before the fire is spreading, so we can save a vertex adjacent to a burning vertex by placing a firefighter on it. After a vertex has caught fire, there is no way to save it. Once there are no vulnerable vertices adjacent to burning vertices, the game is over. We call the vertices not on fire at the end of this process *saved*.

The objective is to find a sequence of vulnerable vertices to place firefighters on, that maximizes the number of saved vertices without exceeding the budget at any time step.

Input A graph $G = (V, E)$, a vertex $s \in V$ and a budget integer B.

Output An ordered sequence of disjoint subsets of V (indicating where to place firefighters at each time step i) $S = S_1, S_2, \ldots S_k, |S_i| \leq B, i = 1, 2, \ldots, k$, such that the vertices in each set S_i are vulnerable at time i and the number of saved vertices is maximized.

There is another version of this problem studied in the literature, called the *spreading model*. The definition is the same except for one difference: the firefighters also spread at each time step in the same manner as the fire, protecting all adjacent vertices

not already on fire. In case a vulnerable vertex is adjacent to both a vertex protected by a firefigher as well as a vertex on fire, the spreading of the firefighter has priority. The problem of finding an optimal placement is known to be NP-hard irregardless of which model is used and the results in the paper are concerned with both models of the problem. Note that the spreading model and the standard model are equivalent on trees.

First, we show a reduction of the Firefighter Problem in the spreading model to a problem called "Maximum Coverage with Group Budgets". This immediately yields several already known approximation results on the problem for trees in the standard model and for general graphs in the spreading model.

We show further results for general graphs in the spreading model. Since the problem remains NP-hard in this model, the time complexity for any exact algorithm is likely to be exponential. Specifically, we provide an exact $2^{O(\sqrt{n}\log n)}$-time algorithm in the spreading model. Due to the aforementioned equivalence between the spreading model and the standard model on trees, we also obtain an $O(n^{\sqrt{n}})$-time algorithm for trees in the standard model. This algorithm is significantly more straightforward and simpler to implement than previously known subexponential algorithms for this problem.

We also show a lower bound on the approximation factor of any approximation algorithm for the Firefighter Problem on weighted directed graphs in the spreading model. This is done by a reduction from the so called ”Budgeted Maximum Cover” problem.

In the standard model, we study the problem on Planar Graphs. If for example, a forest is on fire, the topology of the terrain can often be described using a plane graph, which motivates the assumption. Using an assumption on the size of the budget depending on the maximum degree of the input planar graph, we show that at least $\frac{1}{3}$ of the vertices in the graph can be saved.

Finally, we present an approximation-time trade-off for the so called Directed Layered Graphs in the standard model. For an integer k, we can either approximate the optimal solution within $1 - \frac{1}{k}$ or we can approximate it within $\frac{1}{k}$. The former approximation (which for large k is the most accurate) takes time $n^{O(\sqrt{n})}$, whereas the latter only requires $O(n^2)$ time.

0.2 Poly-line Graph Drawing with constraints

This section deals with *Graph Drawing*, which in a nutshell aims to draw a graph using points and lines on the plane to represent the vertices and edges. A usual constraint is to require the drawing to be free of crossings if the graph is planar (and supplied with a planar embedding). An example of a more specific constraint is to require the drawing to be done using only straight lines. There are many other different constraints studied in the literature as well.

It is also common to introduce some measurement of quality, for instance, the area of the smallest square containing the drawing, or the smallest side length of any square containing the drawing.

We study the problem of drawing a planar graph inside a given polygon when the vertex locations are fixed beforehand. The edges of the drawing must then be drawn as sequences of straight-line segments, connected at the endpoints, to form so called *Polylines* or "Polygonal Chains". The connection between two such segments is called a bend, and the total number of bends for all edges in the drawing will be used as a minimization objective.

Input A planar graph $G = (V, E)$ of size n, a set S of points in the plane with rational coordinates and an injective mapping $\rho : V \to S$, a polygon P with k corners such that all points of S are strictly in the interior of P.

Output A drawing of G with the vertices placed on the points specified by ρ such that all edges are drawn without crossings as connected, finite sequences of straight line segments completely inside P.

We provide bounds on the number of bends needed as a function of both n and k. Namely, we show that it is always possible to draw a graph according to the constraints using at most $O(kn^2)$ bends in total. We also derive a lower bound, stating that there exists graphs that require at least $\Omega(kn^2)$ bends. So, in the asymptotic sense, the bounds proven are tight.

The method used to show the lower bounds is probabilistic, we do not construct any explicit graph achieving the lower bound. The upper bound however is shown by providing an algorithm that draws for a given polygon all the edges according to the constraints.

0.3 Subgraph Isomorphism

This section is comprised of two papers, both concerned with different versions of the *Subgraph Isomorphism* problem. The problem is one of the classic NP-complete problems: Given two graphs, one called the host graph and another called the pattern graph, the objective is to decide if the host graph contains a subgraph isomorphic to the pattern graph.

Input A pair of graphs G, H called "host" and "pattern", with vertex and edge sets V_G, V_H and E_G, E_H respectively.

Output "yes" if there exists an injective mapping $\rho : V_H \to V_G$ such that $\{u, v\} \in E_H \Rightarrow \{\rho(u), \rho(v)\} \in E_G$, and "no" otherwise.

If we instead of the implication in the defintion require equivalence (i.e. $\{u, v\} \in E_H \Leftrightarrow \{\rho(u), \rho(v)\} \in E_G$), the resulting problem is called *Induced* Subgraph Isomorphism. We distinguish between three variants of both versions of the problem:

- The detection variant, which only requires to state whether such a mapping exists or not.
- The finding variant, which requires to report such a mapping ρ (provided it exists).
- The counting variant, which either asks for the total number of such mappings or the total number of subgraphs that are images of the pattern graph under such mappings.

0.3.1 Induced subgraph isomorphism: Are some patterns substantially easier than others?

This paper is mainly concerned with the time complexity of the induced subgraph isomorphism problem. Two main results are presented, one concerning detection and the other dealing with counting.

The detection result shows a lower bound in terms of an independent set in the pattern graph. Specifically, we show that any pattern graph containing a maximum independent set of size k that is disjoint from other maximum independent sets is at least as difficult to detect as an independent set of size k. An interesting example is a cycle of even length, which only has two maximum independent sets (which are disjoint), each containing half of the vertices. It follows that detecting whether a graph contains an induced cycle of length $2l$ requires at least as many operations as detecting an independent set of size l. Other applications include such natural pattern graphs as odd paths and complete bipartite graphs.

The second result is concerned with the counting version of the induced subgraph isomorphism problem (which naturally includes detection). This result is similar in nature to the previous one in the sense that it also determines a lower bound in terms of a smaller independent set. However, it is concerned with the topology of the pattern graph, showing that no specific topology is substantially more difficult than any other. Specifically, for *any* connected pattern graph H on k vertices, we can create a subdivision of H by subdividing each edge of H into a path of length four and attaching a path of length three to each vertex of degree one, such that this subdivision is at least as difficult to detect as an independent set of size k.

Finally, the paper shows a few results on fixed pattern graphs of size 4. We show that the pattern graphs "diamond" and "paw" are at least as difficult to detect as a triangle. See Figure 1. These results resolve conjectures on the detection problem for the respective pattern graphs.

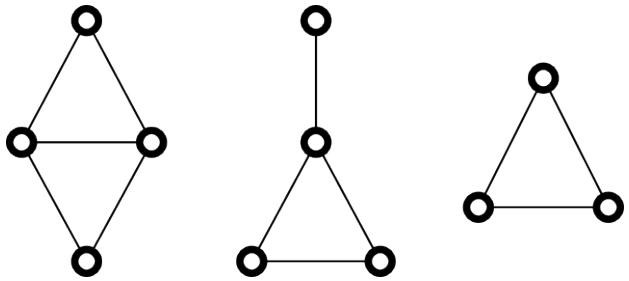

Figure 1: From left: diamond, paw and triangle

0.3.2 Detecting and Counting small pattern graphs

The second paper is concerned with both the detection and counting variants of the subgraph isomorphism problem when the pattern graph is of constant size.

The main contribution is an innovative technique for detecting an induced subgraph, we consider a multivariate polynomial where the monomials are in one to one correspondence with the induced instances of different pattern graphs in the host graph. The idea is to use the evaluation of the polynomial as an implicit method for detecting a pattern graph. The polynomial evaluation relies on a lemma by DeMillo-Lipton-Schwartz-Zippel, which uses randomization, so the algorithm is randomized.

A lot of the previous progress on subgraph isomorphism for fixed pattern graphs relies on fast matrix multiplication. Unfortunately fast matrix multiplications algorithms involve huge overheads, which makes them impractical. In contrast, our algorithm should be more practical as it does not rely on fast matrix multiplication.

We also present an algorithm for the counting variant of the standard subgraph isomorphism problem. We measure the time complexity of our algorithmic solution both in terms of the size of the host graph, the size of the pattern graph as well as the size of an independent set in the pattern graph. Our algorithm improves/generalizes the previously known upper time bounds for the counting variant of the problem. This result, however relies on fast algebraic algorithms for matrix multiplication.

A weighted version of the counting problem is also considered: For a vertex weighted host graph, count the number of subgraph isomorphism mappings that minimize the total weight of the image of the pattern graph. We obtain an algorithm with a running time dependent on both the size of an independent set in the pattern graph as well as the sizes of the host and pattern graphs. In contrast to the previous result, this algorithm is combinatorial i.e, it does not rely on fast matrix multiplication.

0.4 3D Rectangulations and Matrix Products

This section is concerned with two problems and an unexpected relationship between them. The problems are called: *Minimum Polyhedron Rectangulation* and *Matrix Multiplication*.

The first problem, Mimimum Polyhedron Rectangulation, concerns finding a partition of a rectilinear polyhedron into as few as possible 3D rectangles. Finding the minimum partition is known to be NP hard for general polyhedra.

We consider a restriction of the problem to a special case of polyhedra called rectilinear 3D histograms, a generalization of rectilinear 2D histograms to three dimensions. The complexity and approximation status of this restricted problem is still open.

Input 3D rectilinear histogram H.

Output A partition of H into a minimum number of 3D rectangles.

In the first half of the paper, we present an approximation algorithm that finds a partition of a 3D rectilinear histogram with m corners that is within 4 times the size of the optimal partition in $O(m \log(m))$ time.

The second problem deals with computing the product of two matrices as fast as possible and the focus is on the time complexity of an algorithm computing the product.

Input Two $n \times n$ integer matrices A, B.

Output A $n \times n$ matrix C such that $A \times B = C$.

The naive multiplication algorithm takes $O(n^3)$ time. The exponent of fast matrix multiplication algorithms for two $n \times n$ matrices is denoted with ω. The currently best known bound for $\omega \leq 2.372$, due to Le Gall [46]. The known fast matrix multiplication algorithms are based on algebra and recursion which involve large overheads in the asymptotic running time. This makes them competitive only for extremely large input matrices.

We show that if at least one of the input matrices can be decomposed into the sum of relatively few uniform rectangular matrices, then the product of the matrices can be efficiently computed using a simple to implement algorithm. Interpret a matrix M with positive integer entries as a 3D histogram and denote the size of a optimal partition of the corresponding histogram with r_M.

In the second part of this paper we present two matrix multiplication algorithms with a running time dependent on the size of the minimum partitions, i.e. r_A, r_B. The first algorithm uses our 4-approximation algorithm to obtain decompositions of the corresponding 3D histograms and then it explicitly uses these decompositions to compute the product of the matrices. The algorithm has a running time of $\tilde{O}(n^2 + r_A r_B)$, for

two $n \times n$ matrices A, B, where the $\tilde{O}$ notation supresses any polylogarithmic factors. The second algorithm doesn't make explicit use of the partitions, instead it uses vertical slices of the two histograms to compute the product and the resulting running time becomes $\tilde{O}(n^2 + n \min\{r_A,\ r_B\})$.

We also give a generalization of the latter upper bound in terms of the minimum cost of a spanning tree of the slices, where the distance between a pair of slices corresponds to the cost of transforming one slice into the other.

Our matrix multiplication algorithms is superior to the current fast matrix multiplication algorithms only when the value of r_A or r_B is substantially smaller than $n^{\omega-1} \approx n^{1.372}$.

Chapter 1

The Firefighter problem

1.1 Introduction

The firefighter problem was first considered by Hartnell [33]. The objective is to determine a deployment of firefighters which maximizes the sum of weights of saved vertices. Later, several other variants of the firefighter problem have also been studied.

Since the first definition, the problem has received significant attention [2, 9, 25, 33]. We model the underlying network by a directed or undirected graph $G = (V, E)$ with a distinguished vertex s called the source node (or the root), and nonnegative vertex weights. Each vertex $v \in V$ is either *on fire*, *protected*, or *vulnerable*, where the latter implies that v is neither protected nor on fire. At time 0 a fire breaks out at the source node, at each subsequent time step a firefighter may be placed on a vulnerable vertex to protect it. At each subsequent time step the fire also spreads to all vulnerable vertices that are adjacent to a burning vertex. At some time, when the fire can no longer spread, the process ends and all the vertices which are not on fire are considered to be *saved*.

Anshelevich et al. [2] generalized the classical firefighter problem to include the possibility of placing up to B firefighters at a single time step. They also considered the dual problem where the objective is to minimize the budget constraint B in order to save a given set of nodes $T \subseteq V$. Furthermore, they also introduced the so called *spreading model* [2] for the firefighter problem and its dual. In the spreading model, if a node u is protected and v is a vulnerable neighbour of u at time step t, then at the next time step $t + 1$, the node v also becomes protected. Note that in this model a firefighter prevails over possible neighbouring nodes on fire and the adjacent vulnerable node is protected in the subsequent step.

The introduction of the spreading model stems from the fact that the firefighter problem can also model a diffusive process, such as an infection, which spreads through a network. The objective is to stop this infection by using targeted vaccinations (see

e.g. [2]). Hence, the firefighting problem in the spreading model is highly relevant to health care efficiency.

Related work

The firefighter problem appeared to be NP-hard even when restricted to trees [25]. It is hard to approximate within n^{α} in polynomial-time for general undirected graphs, for any $\alpha < 1$ while it admits an $(1 - e^{-1})$-approximation in polynomial-time for trees [9]. The greedy heuristic on trees which places a firefighter on a vulnerable vertex that saves the largest number of vertices is known to achieve $\frac{1}{2}$-approximation factor [34]. Even a subexponential, $2^{O(\sqrt{n}\log n)}$-time, exact algorithm has been designed for trees on n vertices [9].

In the spreading model, the firefighter problem is much more feasible. For general graphs it can be approximated within $1 - e^{-1}$ in polynomial-time [2].

Contributions

To begin with, we observe that essentially all the known approximability results for the firefighter problem on trees [9, 34] as well as those on general graphs in the spreading model [2] immediately follow from the known corresponding results for the so called maximum coverage problem with groups [10].

Our main results are concerned with general graphs in the spreading model.

We provide a very simple exact $2^{O(\sqrt{n}\log n)}$-time algorithm. In the special case of trees, where the standard and spreading model are equivalent, our algorithm is substantially simpler than that exact subexponential algorithm for trees presented in [9].

On the other hand, we show that the firefighter problem on weighted directed graphs in the spreading model cannot be approximated within a constant factor better than $1 - 1/e$ unless NP $\subseteq$ DTIME$(n^{O(\log\log n)})$.

We also present several results in the standard model.

Firstly, we obtain two approximation results in terms of the degree of the source vertex for planar graphs, assuming that at least two firefighters can be placed in a single step.

Secondly, we derive two trade-offs between approximation factors for polynomial-time solutions and the time complexity of exact or nearly exact solutions for instances of the firefighter problem for the so called directed layered graphs studied in [2].

A reduction to a variant of the Maximum Coverage Problem

A restriction of the cardinality variant of the problem of **Maximum Coverage with Group Budgets (MCG)** has been defined in [10] as follows. There are given subsets $S_1, S_2, ..., S_m$ of a ground set X, disjoint subsets $G_1, ..., G_l$ of $\{S_1, ..., S_m\}$ called groups and a positive integer k. The objective is to find $H \subseteq \{S_1, ..., S_m\}$ such that $|H| \leq k$, $|H \cap G_i| \leq 1$ for $i = 1, ..., l$ and the number of elements in X covered by sets in H is maximized.

Chekuri and Kumar proved the following fact in [10].

Fact 1 *The standard greedy heuristic for minimum set cover (stopped when k sets are already included in the cover) yields $\frac{1}{2}$ approximation for the restriction of the cardinality variant of MCG. This problem can be also approximated within $1-e^{-1}$ by linear programming techniques.*

Corollary 1.1 *The firefighter problem with budget B on trees as well as the firefighter problem with budget B for general (directed or undirected) graphs in the spreading model can be approximated within $1 - e^{-1}$ in polynomial time. Also, the standard greedy heuristic that iterates picking a vertex that saves the largest number of not yet saved vertices yields $\frac{1}{2}$ approximation in both cases.*

Proof: Given an instance of the firefighter problem with budget B on trees, we define a corresponding instance of the cardinality variant of MCG as follows.

We root the input tree T at the distinguished vertex and for each other vertex v define S_v as the set of all descendants of v, including v, in the tree. Thus, the ground set is the set of all vertices of T different from the source vertex, and the family of sets consists of the aforementioned sets S_v. We partition the family into groups by accounting into the same group all sets S_v where v share the same level of T. Finally, we set the parameter k to B.

Any feasible solutions to the resulting instance of MCG which covers q vertices is in one-to-one correspondence with a placement of firefighters in T which saves q vertices (place the firefighters on those v for which S_v are in the set cover) and *vice versa* (account to the cover all S_v where a firefighter is placed on v).

This proves the corollary for the firefighter problem on trees.

The proof for general (directed or undirected) graphs G in the spreading model is analogous. It relies on the observation that the set of vertices saved by a placement of firefighters in the spreading model is a union of the sets saved by single firefighter placements included in the placement.

Let d be the maximum distance of a vertex from the fire source in the graph. For $1 \leq r \leq d$, and each vertex v of G, we define S_v^r as the set of all vertices of G that

become directly or indirectly saved if we place a firefighter on the vertex v in the r-th step. Note that v is included into S_v^r and this set can be easily computed in time polynomial in the size of G. For each r, the sets S_v^r form a separate group. □

1.2 Firefighting in the spreading model

1.2.1 A subexponential-time algorithm

Our subexponential-time algorithm for general graphs in the spreading model relies on the two following lemmata.

Lemma 1.2 *After the j-th step, all vertices within distance at most j from the source vertex in the spreading model are either burnt or (directly or indirectly) saved.*

Proof: Let v be a vertex at distance of j from the source. If the fire has not reached v during the j steps then for any shortest path from the source to v, there is a step $1 \leq t \leq j$ at which the $t+1$st vertex on the path has been directly or indirectly saved. It follows from the assumed model that v must be saved too. □

Lemma 1.3 *After the $2\sqrt{n}+1$-st step in any optimal solution all vertices at distance of $2\sqrt{n}+1$ from the source are saved.*

Proof: Suppose that there is a vertex v at distance of $2\sqrt{n}+1$ from the source that is not saved after the $2\sqrt{n}+1$-st step. It follows from Lemma 1.2 that it is burnt after this step. Thus, there must be a shortest path P of length $2\sqrt{n}+1$ from the source to v that is totally burnt after the $2\sqrt{n}+1$-st step.

On the other hand, among the $\sqrt{n}$ firefighters placed during the first $\sqrt{n}$ steps, there exists at least one, say placed at t-th step that saves uniquely at most $\sqrt{n}$ of the vertices in the optimal solution.

Now, if we move the firefighter placed in the t-th step to the $t+1$-st vertex on P (counting from the source) then we save at least $2\sqrt{n}+1-t \geq \sqrt{n}+1$ new vertices and let to burn at most those $\sqrt{n}$ vertices previously uniquely saved by this firefighter. We obtain a contradiction with the optimality of the solution. □

Corollary 1.4 *Any optimal solution in the spreading model for a graph with n vertices and a distinguished source vertex places at most $2\sqrt{n}+1$ firefighters.*

Proof: By Lemma 1.3, directly after the $2\sqrt{n}+1$-st step, all the vertices at distance of $2\sqrt{n}+1$ from the source are saved. Hence, the more remote vertices will be saved too. This implies that that there is no need to place firefighters in the next steps. □

Now, we are ready to derive our main result in this section.

Theorem 1.5 *An optimal solution in the spreading model for a graph with n vertices and a distinguished source vertex can be found in time $O(n^{2\sqrt{n}+3}) = 2^{O(\sqrt{n}\log n)}$.*

Proof: By Corollary 1.4, it is sufficient to enumerate all valid placements of at most $2\sqrt{n}+1$ firefighters, for each of them compute the number of saved vertices, and chose the placement maximizing the number of saved vertices. There are $O(n^{2\sqrt{n}+1})$ such placements and the computation of the number of saved vertices for a given placement takes time $O(n^2)$. □

1.2.2 A lower bound

The *budgeted maximum coverage* problem (BMC for short) is as follows. For a budget k and a family S of sets defined over a domain of n weighted elements, each set having an associated cost, find a subset S' of S such that the total cost of sets in S' does not exceed k and the total weight of elements covered by S' is maximized.

Khuller et al. proved the following approximability hardness result on BMC [41].

Fact 2 *The unit cost version of the budgeted maximum coverage problem cannot be approximated within a constant factor better than $1-1/e$ unless NP $\subseteq$ DTIME$(n^{O(\log\log n)})$.*

Lemma 1.6 *There is a polynomial-time many-one reduction ϕ of the unit cost version of the budgeted maximum coverage problem to the firefighter problem on weighted directed graphs in the spreading model such that for an instance I of the maximum coverage problem the maximum number of the elements covered under budget k is equal to the maximum weight of saved vertices in $\phi(I)$ minus 1 (or even equal in case all elements are covered).*

Proof: Let S be the family of sets in I. Form a layered directed graph $G(I)$ with a distinguished source vertex s as follows. In the bottom layer put vertices in one-to-one correspondence with the elements in the domain. In the next layer put vertices in one-to-one correspondence with the sets in S. Direct from each of them edges to all vertices in the bottom layer corresponding to elements covered by the associated set in S. Now, connect the source s with each of the vertices v on the next to the bottom layer by a unique directed path of length k from s to v. Set the weights of the vertices on the bottom level to one and the weights of all remaining vertices to zero.

Consider a solution to I covering q elements with k sets. Suppose first that it does not cover all elements. Place a firefighter on each of the unique paths connecting the source with a vertex on the next to the bottom level corresponding to a set in the solution to I so no two firefighters are placed at the same distance from s and no firefighter is placed on s. Such a placement is possible since the paths have length k and it saves $q + \frac{k(k+1)}{2}$ vertices of total weight q. Finally, we can save one more vertex of weight 1

corresponding to an uncovered element by placing a firefighter on it in the last $k+1$st step.

On the other hand, since placement of a firefighter on an ancestor of a vertex is never worse than such a placement on the vertex, we may easily transform an optimal solution to $G(I)$ to a normalized one which places k firefighters on the unique paths connecting the source s with the next to the bottom layer during the first k steps. Such a normalized placement saves at most $\frac{k(k+1)}{2}$ vertices on these paths and, say t, vertices on the bottom layer. In the last, $k+1$st step, it places a firefighter on a not yet saved vertex at the bottom level. It follows that the total weight of saved vertices is $t+1$. By picking the k sets corresponding to the vertices on the next to the bottom level saved by the k firefighters placed in the first k steps, we obtain a family of k sets covering t elements.

In case, a (optimal) solution to I covers all n elements we do not need to place any firefighter in the $k+1$st step. Then, on the other hand, the firefighters placed in the first k steps of any normalized optimal solution to $G(I)$ induce a family of k sets covering $t \geq n-1$ elements while the total weight of saved vertices is n. □

Fact 2 combined with the proof of Lemma 1.6 yields the following lower bound.

Theorem 1.7 *The firefighter problem on weighted directed graphs in the spreading model cannot be approximated within a constant factor better than* $1 - 1/e$ *unless NP* $\subseteq$ *DTIME*$(n^{O(\log \log n)})$.

1.3 Firefighting in the standard model

1.3.1 A simple subexponential-time algorithm for trees

Since in the case of trees, there is no difference between the standard model and the spreading model, our simple subexponential-time algorithm for general graphs in the spreading model also works for trees in the standard model. This yields both a much simpler subexponential algorithm as well as analysis for the firefighter problem on trees than those presented in [9].

Furthermore, if we adopt the derivation of the subexponential algorithm in the spreading model to the standard firefighter model constrained to trees then we can decrease the constant in the exponent of the upper bound derived in the spreading model substantially.

The following counterpart of Lemma 1.2 is obvious.

Lemma 1.8 *In the standard model for trees with a distinguished source vertex, after the* j*-th step, all vertices within distance at most* j *from the source vertex are either burnt or (directly or indirectly) saved.*

The following counterpart of Lemma 1.3 can be simply obtained by replacing Lemma 1.2 with Lemma 1.8 in the body of the proof.

Lemma 1.9 *After the $2\sqrt{n}+1$-st step in any optimal solution in the standard model for a tree with a distinguished source vertex, all vertices at distance of $2\sqrt{n}+1$ from the source are saved.*

Similarly, the proof of the following counterpart of Corollary 1.4 can be obtained by replacing Lemma 1.3 with Lemma 1.9 in the body of the proof.

Corollary 1.10 *Any optimal solution in the standard model for trees with n vertices and a distinguished source vertex places at most $2\sqrt{n}+1$ firefighters.*

Finally, the proof of the following counterpart of Theorem 1.5 can be obtained by replacing Corollary 1.4 with Corollary 1.10 in the body of the proof of Theorem 1.5 and observing that computing the number of saved vertices for a given placement of firefighters requires time linear in n in case of trees.

Theorem 1.11 *An optimal solution in the standard model for a tree with n vertices and a distinguished source vertex can be found in time $O((\frac{n}{4})^{\sqrt{n}}n^{3/2})$.*

Proof: By Corollary 1.10, it is sufficient to enumerate all valid placements of at most $2\sqrt{n}+1$ firefighters, for each of them compute the number of saved vertices and chose the placement maximizing the number of saved vertices. We may w.l.o.g consider only the placements that for $i=1,...,2\sqrt{n}+1$ place at most one firefighter at distance of exactly i from the source. The number of the latter placements is at most $(n/(2\sqrt{n}+1))^{2\sqrt{n}+1} \leq n^{\sqrt{n}+\frac{1}{2}}2^{-2\sqrt{n}-1}$. It remains to observe that the computation of the number of saved vertices for a given placement takes time $O(n)$. □

1.3.2 Approximate firefighting on planar graphs

Planar graphs and their planar embeddings termed as *plane graphs* seem to be a very natural model of a network for the applications of the firefighter problem. If we allow for placement of more than one firefighter in a single step, we can obtain non-trivial approximation results based on the good separator properties of planar graphs.

We can rephrase Lemma 2 from [50] for our purposes in terms of plane graphs as follows.

Lemma 1.12 *Let G be a plane graph with nonnegative vertex costs summing to W. Suppose G has a rooted spanning tree T of radius r. Then the vertices of G can be partitioned into three sets A, B, C such that no edge joins a vertex in A with a vertex in B, neither A nor B has total cost exceeding $2W/3$, and C consists of vertices on two*

paths towards the root of T starting from vertices v and u on a common face and ending at their lowest common ancestor. Furthermore, C can be completed to a simple cycle by the diagonal joining v with u, and the vertices in A lie outside the cycle while those in B inside the cycle.

Theorem 1.13 *The firefighter problem on planar graphs with a source vertex r of degree $deg(r)$ and budget $B \geq \max\{2, \lceil deg(r)/2 \rceil\} + 1$ can be approximated within $\frac{1}{3}$ in polynomial-time.*

Proof: Let G be a planar graph on n vertices with a source vertex r in which fire starts and let G' be its planar embedding.

Construct the breadth-first search tree BT of G' with the root at r. By Lemma 1.12, there are two vertices $v,\ u$ in BT such that the set C of vertices on the paths $P_v,\ P_u$ from v and u to the lowest common ancestor of v and u in BT splits G' into two parts each having at least $1/3$ of the total weight of G'.

Now if the aforementioned lowest common ancestor is different from r, for $j = 1, 2, ...$ iterate the following step:
Place firefighters on the at most two vertices of P_v and P_u that are at distance of j from r.
It follows from Lemma 1.12 that the part of G' between these two paths of total weight at least $1/3$ of that of G is saved in this way.

Suppose in turn that the lowest common ancestor is at r. In case the number children of r between P_v and P_u is smaller than $deg(r)/2$ then in the first step, i.e., for $j = 1$, we place at most $\lceil deg(r)/2 \rceil + 1$ firefighters on the aforementioned children and the two children on P_v and P_u, and then proceed analogously as in the previous case. Finally, in case the aforementioned number of children is greater than $deg(r)/2$ we proceed as follows. We place at most $\lceil deg(r)/2 \rceil + 1$ firefighters on all the children of r outside the cycle induced by the paths P_v and P_u and the children of r on these two paths and proceed analogously as in the consecutive steps. In this way, we save all the vertices outside the cycle whose total weight is at least $\frac{1}{3}$ of that of G. □

1.4 Tradeoffs for firefighting on directed layered graphs

The dual, budget variant of the firefighter problem for the so called directed layered graphs have been studied in [2].

A *directed layered graph* G with a source s is one whose vertices can be partitioned into l layers such that s is the only vertex in the 0 layer and for each directed edge (u, v) there is $0 < i \leq l$ where u belongs to the layer $i - 1$ and v belongs to the layer i.

Note that for a vertex on a layer i placements of firefighters in time steps greater than i cannot help in saving the vertex.

Definition 1.14 The (standard) firefighter problem for a directed graph G with a source vertex and the upper bound B on the number of vertices on which firefighters can be placed at a single time step is denoted by $FF_B(G)$.

By enumerating all feasible solutions to $FF_1(G)$, we obtain the following lemma.

Lemma 1.15 *Let G be a layered directed graph on n vertices, with l layers and a source vertex. An optimal solution to $FF_1(G)$ can be computed in time $O(n^2 + \Pi_{j=1}^{l}|V_j(G)|) \leq O(n^2 + \binom{n}{l})$, where $V_j(G)$ is the set of vertices of G on the layer j.*

By the following lemma and the known results on the approximability hardness of the classical firefighter problem for general undirected graphs, the hard instances have to have small radius. In the lemma, the *eccentricity* of a vertex means the maximum length of a shortest directed path from the vertex to another vertex in the graph.

Lemma 1.16 *Let G be a directed graph on n vertices with a source vertex and let d be the eccentricity of the source. $FF_1(G)$ can be approximated within n/d.*

Proof: Let P be a shortest directed path from the source vertex to a most distance vertex from it in G. Note that P has length d. In time step i, place a firefighter on the i-th vertex of P. Observe that all the d vertices on P different from the source vertex will be saved. □

By Lemmata 1.15, 1.16, we obtain the following tradeoff between the approximability and the exact time complexity in case of the classical firefighter problem on layered directed graphs.

Theorem 1.17 *Let G be a layered directed graph on n vertices with a source vertex. For each positive integer $k \geq 2$, $FF_1(G)$ can be approximated within n/k in polynomial time or an optimal solution to $FF_1(G)$ can be computed in time $O(n^k)$.*

Proof: Let d be the eccentricity of the source in G. If $k < d$ then by Lemma 1.16, we obtain an approximation within n/k in polynomial time. Otherwise, we obtain an optimal solution in time $O(n^k)$ by Lemma 1.15. □

The following combinatorial lemma valid for any directed graph with a source will be useful.

Lemma 1.18 *Let G be a directed graph on $n \geq 4$ vertices with a source vertex and let $\alpha > 1$. If the eccentricity of the source in G not less than $\lceil \alpha\sqrt{n} \rceil$ then there is $i \in \{1, ..., \lceil \alpha\sqrt{n} \rceil\}$ such that the number of vertices of G at distance of i from the root is at most $\lfloor 2i/\alpha^2 \rfloor$.*

Proof: If the theorem does not hold then the number of vertices of G different from the source vertex is not less than $\sum_{i=2}^{\lceil \alpha\sqrt{n} \rceil} 2i/\alpha^2$ which in turn is not less than $\frac{2}{\alpha^2} \times ((\lceil \alpha\sqrt{n} \rceil + 1)\lceil \alpha\sqrt{n} \rceil/2 - 1)$. Since the latter value is clearly greater than n, we obtain a contradiction. □

By using Lemma 1.18, we can also obtain another tradeoff for layered directed graphs between very close approximability in subexponential time and constant approximability in polynomial time.

Theorem 1.19 *Let G be a layered directed graph on n vertices with a source vertex. For any integer $k > 0$, an optimal solution to $FF_1(G)$ can be approximated within $(1 - \frac{1}{k})$ in time $n^{O(\sqrt{n})}$ or it can be approximated within $\frac{1}{k}$ in time $O(n^2)$.*

Proof: We may assume without loss of generality that the number of layers in G is not less than $\lceil \alpha\sqrt{n} \rceil$ since otherwise $FF_1(G)$ can be solved exactly in time $n^{O(\sqrt{n})}$ by Lemma 1.15. Apply Lemma 1.18 with $\alpha = \sqrt{2}$ to G. Let i be the number of the layer of G which has at most i vertices. Consider an algorithm which places during the first i time steps firefighters on the vertices on the i-th layer. If the number of vertices of G on the layers $j \geq i$ is at least $\frac{1}{k}$ of the optimum then the algorithm yields an $\frac{1}{k}$ approximation. Otherwise, more than $1 - \frac{1}{k}$ of the saved vertices are placed on the layers 1 through $i - 1$ in an optimal solution to $FF_1(G)$. Thus, there is at least one placement of at most $\binom{n}{i-1}$ feasible placements of firefighters on these levels which saves at least $1 - \frac{1}{k}$ of the optimal number of vertices. Such a placement can be detected in time $n^{O(\sqrt{n})}$. □

Chapter 2

Graph Drawing

2.1 Introduction

Drawing, or *embedding*, a planar graph in the plane is a well studied problem, and proofs that we can draw planar graphs with straight lines have been around longer than computers [63, 24]. One of the first algorithmic result was that all planar graphs have such a drawing using polynomially bounded integer coordinates [28, 59]. Many other graph drawing results and models of straight-line drawings have been developed since.

If we introduce the possibility to draw the edges as sequences of straight line segments, i.e., as poly-lines with *bends*, then we may be able to incorporate other constraints on the drawing. The typical measure of quality is then the number of bends used while satisfying such constraints. For example, in some applications such as geographic visualization, vertices (which may represent cities) should be placed at or near a given location. Hence the following *point-embedding problem* is of interest: Given a planar graph G, a set of points S and an injective mapping $V \to S$, can we draw G without crossing such that the vertices are at the specified locations?

It is quite clear that not all graphs have such drawings with straight lines, and it can always be done if we allow sufficiently many bends. Pach and Wenger [55] studied this point-embedding problem and gave bounds on how many bends may be needed. They showed that it can always be done with $O(n^2)$ bends, and $\Omega(n^2)$ bends are required, even for a matching, for some mappings of vertices to points. Since then, some variants of the point-embedding problem have been studied, for example when the mapping of the vertices to points can be arbitrary [39], or when it is only partially restricted [4, 18].

Here we explore the point-embedding problem with the additional restriction that the drawing must be within a given polygon. Thus, we are given a planar graph G, a point set S, a polygon P that contains all points of S in its interior, and an injective mapping from the vertices to S. We would like to create a planar graph drawing of G

that lies entirely inside P and that has the vertices at the specified locations. Since we require the points of S to be in the strict interior of P, it is clear that such a drawing always exists if we allow bends, but how many bends are needed?

We cam across this problem when studying maps, and give two motivating examples. Consider Figure 2.1(left), which shows a hand-drawn (1910) cartogram of the United States, with states skewed so that the area reflects population. The adjacencies in this map are hilariously wrong (especially for Pennsylvania), but nevertheless, the reader has no difficulty in recognizing the United States, simply because the boundary is correct. We are currently doing research on how to create cartograms that use given boundaries and other identifying features such as rivers, and encountered the problem studied in the current paper as a sub-problem. A second example is in Figure 2.1(right), which shows a flight map of some intra-Canadian flights. Does the flight from Toronto to Sault Ste. Marie enter US airspace? How about the one from North Bay to Thunder Bay?[1] This flight map is drawn with straight lines, while flights paths are often zig-zag lines depending on location of control towers. So this map cannot be trusted to answer the question, and a map that distinguish clearly between flights that remain entirely inside Canada and those that do not may be useful.

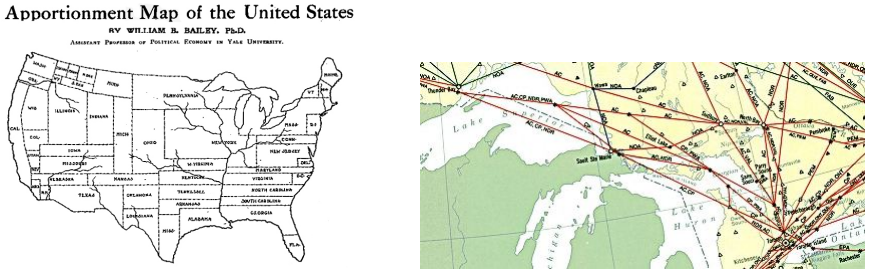

Figure 2.1: (left) A cartogram of the United States. (right) A flight map of parts of Canada.

The topic of drawing at fixed locations inside a polygon is also related to the *local routing* problem in VLSI design (see for example [47].) Here the modules of a chip have been placed already, and the routes of connections between the pins of the modules must be placed in the remaining free space. However, the research on local routing is quite different from our research for two reasons: (1) Pins are located on the modules, and hence at the boundary of the drawing region. This makes planar drawings impossible in almost all cases. In contrast, we demand that points are strictly inside P, precisely so that a planar drawing is always feasible. (2) The VLSI community focuses on minimizing area as main objective. In contrast, we focused on minimizing the num-

[1] If yes, then due to the recently passed Bill C-42, data about the passengers may be forwarded to the US government, and passengers on the US no-fly list may be denied boarding. So the question may be of interest to some people.

ber of bends. (Area considerations of the resulting drawings are an interesting problem, but this remains for future studies.)

We give in this paper upper and lower bounds on the number of bends needed for a planar drawing on a given set of points inside a polygon. Our results resemble the ones by Pach and Wenger [55], but (as is to be expected) also include the size of the bounding polygon. Presume we are given an n-vertex planar graph G and a k-sided polygon P with points S inside P, and a mapping from V to S. A drawing with $O(nk)$ bends per edge would be quite easy to achieve: Take the drawing with $O(n)$ bends per edge from [55] that ignores P, and then re-route each segment of an edge to be a polyline inside P with at most k bends. We show here that we can use far fewer bends than that: there always exists a planar drawing of G inside P with vertices at pre-specified points of S that has $O(k + n)$ bends per edge. We also show that this is tight: $\Omega(n)$ edges need $\Omega(k + n)$ bends for some choice of G, S and P.

Our lower bound builds directly on the lower bound of Pach and Wenger, endowed with a suitable polygon; see Section 2.2. Our upper bound is also somewhat similar to the upper-bound method used in [55] where all edges are routed in parallel channels around the drawing. However, instead of "channels" we use cyclic levels, similarly as was done by Bachmeier et al. [3]. We then map the cyclic levels to line segments inside the polygon that contain the points of S and satisfy some other conditions to make such a mapping possible.

To minimize the number of bends, we need to minimize the number of times an edge needs to do a "turn", i.e., a change of direction from clockwise to counter-clockwise or vice versa in a cyclic level drawing. We show that any edge needs at most two turns. This is also of interest for so-called *upward drawings* (where directed graphs are drawn such that edges go monotonically from smaller to a larger y-coordinate). Not every planar directed acyclic digraph (dag) has an upward drawing with straight lines, and testing whether it does is NP-hard [30]. Our results imply that any planar dag has a planar drawing where the y-coordinate of each tail is smaller than the y-coordinate of the head, and every edge consists of at most 3 y-monotone pieces.

2.2 Lower bound

In this section, we argue our lower bounds: for some n-vertex graph graph G, point set S inside a k-sided polygon P and mapping $V \to S$, $\Omega(n)$ edges have $\Omega(n + k)$ bends each.

Our lower bound builds directly on the lower bound given by Pach and Wenger for the point-embedding problem [55]. They showed that for any planar graph that has a matching of size m, and any random assignment of the vertices to points in convex position, almost surely there are at least $\frac{m}{20}$ matching-edges that have at that have at least $m/(40)^2$ bends each.

For our lower bound, let G be the graph that is a matching of size $m := n/2$. As bounding polygon B, we use a k-sided polygon that has two regions R_1 and R_2 with $\Omega(k)$ *link-distance*, i.e., any path from a point in R_1 to a point in R_2 that stays inside B has $k/2 - 1$ bends. See Figure 2.2. We choose the points in S such that there are $n/2$ points in each of these two regions, and the points in S are in convex position.

Now choose a random assignment from the vertices of G to the points of S. By Pach and Wenger's result, almost surely at least $\frac{m}{20}$ edges have at least $m/(40)^2$ bends. Let E' be those edges, and let $E'' \subseteq E'$ be all those edges in E' that connect a point in R_1 with R_2. By construction, any edge in E'' has $\Omega(k)$ bends because of the link-distance, and $\Omega(m) = \Omega(n)$ bends because it is in E', and hence $\Omega(n + k)$ bends total. So all that remains to do is to bound the size of E''.

We know $|E'| \geq \frac{m}{20}$ almost surely. Any edge connects the two different regions with probability $\approx 1/2$. The expected number of edges in E'' is hence $\geq \frac{1}{2}\frac{m}{20}$. Since the variance of whether an edge connects two difference regions is $\approx \frac{1}{4}$, so the variance of $|E''|$ is $\approx \frac{1}{4}\frac{m}{2}$. Hence by Chebyshev's inequality the probability that $|E''| < \frac{1}{2}\frac{m}{20} - \sqrt{m/80}\sqrt{\frac{1}{4}\frac{m}{20}}$ is at most $80/m$. So the probability that $|E''| < m/80 = n/160$ goes to 0 as n goes to infinity. In conclusion, almost surely at least $n/160$ edges have at least $n/40 + (k-6)/2$ bends each, which proves:

Theorem 2.1 *Let $G = (V, E)$ be a plane graph that contains a perfect matching and P be the bounding polygon with points in it as in Figure 2.2. Then any random mapping of vertices to points requires $\Omega(n)$ edges to bend $\Omega(n + k)$ times almost surely.*

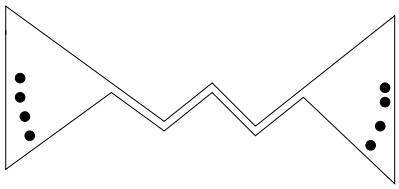

Figure 2.2: The bounding polygon used in Theorem 2.1; it has $k/2 - 1$ reflex vertices, and any path from R_1 to R_2 must detour around all of them.

2.3 Upper bound

To obtain an upper bound of $O(n + k)$ bends per edge, we need some intermediate results. We first give an overview of the algorithm here, and then explain the individual steps below.

So presume we are given a graph G, a point set S, a bounding polygon P, and an injective mapping of V to S. We first create a ordered set of disjoint line segments

L_S (which we call *skewed levels*) inside P that contain all points of S (with at most one point per segment) and have some other useful properties. The injective mapping $V \to S$ then naturally gives an injective mapping $V \to L_S$.

Next, we consider what we call *cyclic levels*, which is a set L_C of disjoint line segments on rays from the origin. We have $|L_C| = |L_S|$, and a natural 1-to-1 correspondence $L_S \leftrightarrow L_C$, which gives an injective mapping $V \to L_C$. We want a *cyclic level drawing* of G (previously studied in [3]) where each vertex is placed somewhere on the cyclic level that it maps to. Our objective is to minimize the number of turns done by each edge. We show that every graph has a cyclic level drawing such that every edge has at most two turns.

With this drawing in hand, we then easily obtain a drawing of G on S inside P by mapping the cyclic levels back to the skewed levels. Since every edge has at most two turns, it crosses every level at most 3 times, and hence has $O(|L_S|) = O(k+n)$ bends.

2.3.1 Cyclic level drawings and turns

We first start by explaining the cyclic level drawings. A *cyclic level set* $L_C = \{l_1, \ldots, l_M\}$ is a set of disjoint line segments such that l_i lies on the ray from the origin with angle $2i\pi/M$. See Figure 2.3. Let b_i be the endpoint of l_i closer to the origin, and t_i be the other endpoint. Note that $\{b_1, \ldots, b_M\}$ and $\{t_1, \ldots, t_M\}$ form a polygon (with one hole), which has a natural quadrangulation defined by the segments. For ease of description, we will assume that all cyclic levels have the same length and distance from the origin, so the quadrangles formed by them are trapezoids.

In the following section, we are given a set of cyclic levels, a planar graph G, and an injective mapping of vertices to cyclic levels. We want a poly-line drawing of G such that each vertex is placed on its cyclic level (but it does not matter where along that level.) Furthermore, the drawing of G should be entirely within the polygon defined by the endpoints of the levels. Cyclic level drawings of planar graphs were first studied by Bachmeier et al. [3] with the objective of testing whether a planar graph has such a drawing such that directed edges make no turns at all. In our work, we create such drawings where every edge makes at most 2 turns. Here, a *turn* is a change of direction of the edge with respect to the cyclic levels. More precisely, we say that an edge makes a turn whenever it crosses the same cyclic level twice without crossing any other cyclic level inbetween; see Figure 2.3.

Theorem 2.2 *Given a planar graph $G = (V, E)$, a set of cyclic levels L_C and an injective mapping $l : V \to L_C$. Then G can be drawn inside the polygon formed by the cyclic levels such that any vertex v is drawn somewhere on $l(v)$, and any edge has $O(|L_C|)$ bends and at most 2 turns.*

The proof of this theorem proceeds in two steps. We first prove a weaker result, where we assume that G has a Hamiltonian cycle $C = \{v_1, v_2 \ldots v_n\}$. In this case we

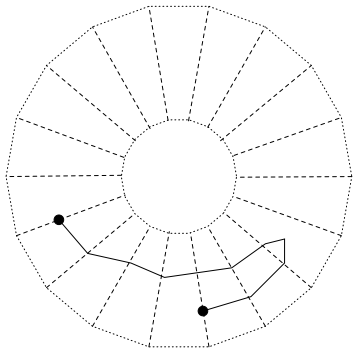

Figure 2.3: Cyclic levels (dashed), the polygon defined by them (dotted), and an edge drawn with one turn.

use a technique similar to the one used by Kaufmann and Wiese [39], who studied how to draw a planar graph on a point set but without specifying which vertex goes to which point.

Our drawing is illustrated in Figure 2.4. To draw C, we imagine the interior of each level l_i to be subdivided with with n points, say $p_1^{l_i}, \ldots, p_n^{l_i}$ at equal distance, in order towards the origin. Draw v_1 on the outermost point $p_1^{l(v_1)}$ of its level, v_2 on the second point $p_2^{l(v_2)}$ on its level, and so on. To route the edge (v_1, v_2), assume first that $l(v_1) > l(v_2)$. Then route the edge by adding a bend on every level i with $l(v_1) > i > l(v_2)$, placing the bend somewhere between p_1^i and p_2^i on level i. Thus we form a counter-clockwise arc from v_1. If $l(v_2) > l(v_1)$ then we route similarly but in clockwise direction from v_1, adding bends on every level i with $l(v_2) > i > l(v_1)$. All other edges of the Hamiltonian cycle are routed similar: go either counter-clockwise or clockwise along the level, in such a way that none of the added edge-segments pass through the trapezoid between the last and first level.

Finally, to close up the Hamiltonian cycle, we need to route edge (v_1, v_n); we do so by routing counter-clockwise from $l(v_1)$ (outside all points) and clockwise from $l(v_n)$ (inside all points) until both parts reach the trapezoid between the last and first level. There the parts can be connected with a segment, without introducing a crossing, since this trapezoid is empty.

Now we need to draw all remaining edges, each of which is inside or outside of the Hamiltonian cycle. Let (v_i, v_j) be an inside edge, $i < j$. Route it by going clockwise from v_i, always staying near the ith point on each level, until we are past the smallest level ℓ used by $v_{i+1}, \ldots, v_{j-1}$. Similarly route clockwise from v_j. Finally add one bend at a point just beyond ℓ and connect the two routes; edge (v_i, v_j) makes a turn at this bend. Route the outside edges similar, except go counterclockwise. All edges can be routed in this fashion with at most one turn per edge. So we have:

Lemma 2.3 *A planar Hamiltonian graph has a cyclic level drawing where edges of the Hamiltonian cycle have no turn and all other edges have at most one turn.*

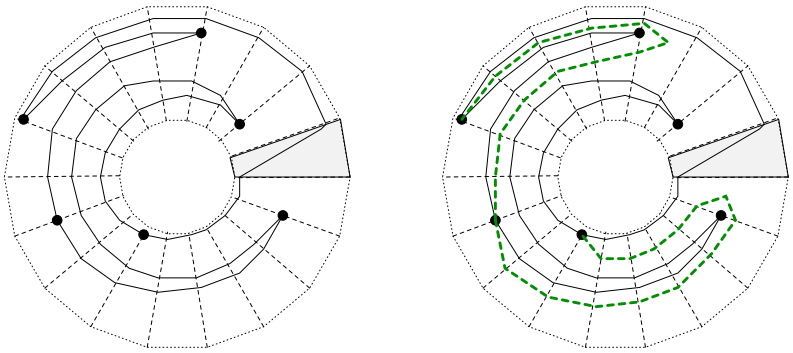

Figure 2.4: (Left) The drawing of the Hamiltonian cycle. Only (v_n, v_1) is allowed to use the trapezoid between the last and first level. (Right) Adding inner and outer edges (thick dashed, green) with one turn.

Now we turn to graphs that are not Hamiltonian. It is well known that a planar triangulated graph is Hamiltonian if it does not contain a *separating triangle*, i.e., a triangle that has vertices both inside and outside. Moreover, a Hamiltonian cycle can be found in linear time [12]. Both Pach and Wenger [55] and Kaufmann and Wiese [39] describes methods to augment G to make it Hamiltonian in linear time. We use the method in [39], which removes a separating triangle by subdividing an edge in it, and then connect the subdivision vertex to the third vertex on the two adjacent faces. Doing this to all separating triangles results in a Hamiltonian graph that has $O(n)$ vertices and every edge has been subdivided at most once.

Thus presume we have made graph G into a Hamiltonian graph G'. Add extra cyclic levels and assign the subdivision vertices to them in an arbitrary manner. Draw G' on these cyclic levels as explained above. Then remove the added edges and restore the original edges by replacing subdivision vertices by bends. All desired properties of the resulting drawing of G are easily verified, except for the number of turns. A naive argument would say that an edge e of G has at most 3 turns, since e may have consisted of two edges e_1 and e_2 in G', each of those could have acquired one turn in G', and removing the subdivision vertex v' that was common to e_1 and e_2 might add another turn. However, in fact e has at most two turns:

- If e_1 belonged to C, then it had no turn in G', hence we have at most 2 turns total in e. Similarly we have at most 2 turns if e_2 belonged to C.

- Not both e_1 and e_2 can be outside edges. For the subdivision vertex v' has degree 4 and e_1 and e_2 are not consecutive at v', so if both e_1 and e_2 are outside edges, then there is only one edge at v' that could be on or inside the Hamiltonian cycle, but there must be at least two (the two on the Hamiltonian cycle.)

  Similarly not both e_1 and e_2 can be inside edges.

- So the only remaining case is if e_1 is an inside edge and e_2 an outside edge, or vice versa. Since we route inside edges clockwise and outside edges counter-clockwise, then e does not acquire a turn when removing the subdivision vertex v'.

We have thus proved Theorem 2.2: every planar graph has a drawing on cyclic levels such that every edge turns at most twice.

2.3.2 Skewed levels inside P

We now show how to create skewed levels inside P that contain the given points S on distinct level. The ultimate goal is to be able to map the cyclic level drawing obtained above onto these levels, which will require the following conditions:

Definition 2.4 Let P be a polygon. A set L_S of line segments is called a *skewed level set inside* P if there exists polygons $P'' \subset P' \subseteq P$ and a triangulation $\mathcal{T}$ of $P' - P''$ such that:

1. the dual of triangulation $\mathcal{T}$ is a cycle,
2. every segment in L_S lies on an interior edge of $\mathcal{T}$,
3. for every interior edge of $\mathcal{T}$, there is exactly one edge of L_S that lies on it.

Put differently, the segments in L_S are all disjoint, and connecting their endpoints up in order gives a polygon with exactly one hole and for which the segments define a convex quadrangulation. See also Figure 2.5.

Note that the dual of the triangulation defines a cyclic order of the skewed levels. They could hence be viewed as a set of cyclic levels, except that they have been deformed as to fit inside P; we will exploit this correspondence later.

Lemma 2.5 *Let P be a polygon and S a set of points in the interior of P. Then there exists a skewed level set of size $O(|P| + |S|)$ inside P such that each point in S is on one of the levels, and no level contains two points of S.*

Proof: To prove this, it will be helpful to assume that no three points of $S \cup P$ lie on a line, unless they all belong to S. This is not a restriction: We can change P by

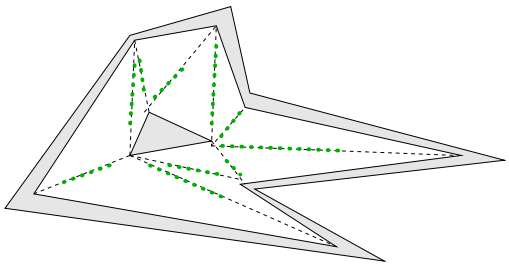

Figure 2.5: A polygon P with skewed levels (thick dotted, green) that are on the inner edges of a triangulation (dashed) of the polygon $P' - P''$.

moving points of P inward. Recall that S is strictly inside P, so with a sufficiently small movement we still have all of S inside the new polygon, and a skewed level set inside the new polygon is also one inside P. We will also assume that P is simple (has no holes); if there is a hole then we can remove it by removing a thin channel from P that connects from the outside to the hole and does not contain a point in S. We can do this for all holes without affecting the asymptotic size of the bounding polygon.

Now triangulate P; none of the triangulation edges contains a point of S by the above. On each triangulation edge, place another *subdivision point.* Inside each face of the triangulation place a *face-point* and connect it to all polygon-corners and subdivision points of the face. We choose the positions of face-points in such a way that none of the lines incident to a face-point contains a point in S, and such that no line through two points of S contains a face-point.

The connections from face-points to the subdivision points form a tree T (which is a subdivision of the dual tree of the triangulation.) Observe that $P - T$ is a polygon with a (degenerate) hole, and it has $O(k)$ corners. Also, the triangulation edges of P and the edges from the face points form a triangulation of the polygon $P - T$, and it is easy to see that the dual of this triangulation is a cycle (effectively, we are "walking around tree T".) See Figure 2.6.

For each point in $s \in S$, we now construct one line segment $l(s)$. By construction s is inside $P - T$, so it belongs to one of the triangles F in the constructed triangulation of $P - T$. One side of F belongs to either P or T; let c be the corner of F that is *not* on that edge. Let $l(s)$ be the maximal open line segment that goes through s and c and stays within F.

Point c is either a corner of P or a face point. By our assumptions on P and construction of face points, the line segment $l(s)$ therefore contains only one point from S. Our set of skewed levels now consists of these $O(n)$ line segments $l(s)$ for $s \in S$, as well as the $O(k)$ line segments in the triangulation of $P - T$ that belong to neither P

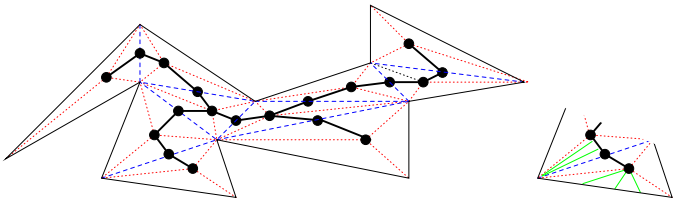

Figure 2.6: A polygon (solid, black) with its triangulation (dashed, blue), the subdivision and face points (black dots), the tree between them (bold, black), and edges from the face points to the corners (dotted, red). (Right) A close-up shows how to add one skewed segment per point in S (marked by an x.)

nor T. One easily verifies that this is a skewed level set.[2] □

2.3.3 Mapping the levels

We are now ready for the main result, where we show that we can create a mapping from a drawing on cyclic levels onto the skewed levels obtained when triangulating the polygon.

Theorem 2.6 *Any planar graph G can be drawn with vertices at prespecified points S inside a given polygon P such that any edge has at most $O(|V| + |P|)$ bends.*

Proof: Start by creating a skewed level set L_S inside P for the point set S according to Lemma 2.5. Next create a set of cyclic levels L_C of cardinality $|L_S|$. Map the (cyclically ordered) set L_S to L_C, and with it, obtain a mapping from V to L_C. Create a drawing of G on L_C that has at most two turns per edge and respects this mapping (Theorem 2.2.)

Now we explain how to map this drawing back onto the skewed levels. Assume vertex v is drawn on cyclic level l_c at the point that is a λ-proportion away from the inner endpoint (i.e., if $l_c = [b, t]$ with b closer to the origin, then v is drawn at $\lambda b + (1-\lambda)t$). Let l_s be the level that corresponds to l_c, i.e., l_s contains the point s on which we are supposed to draw v. Retract the ends of l_s in such a way that point s is a λ-proportion away from the endpoint closer to the "inner polygon" P''. Call the resulting line segment l'_s. See Figure 2.7.

Let l_c^1 and l_c^2 be two consecutive cyclic levels; the area R_c between them is a trapezoid. Let l_s^1 and l_s^2 be the corresponding skewed levels, and $(l_s^1)'$ and $(l_s^2)'$ be their retractions as explained above (we only retract those levels that contain a vertex.) The area R_s between $(l_s^1)'$ and $(l_s^2)'$ is a convex quadrilateral since l_s^1 and l_s^2 were on sides of

[2]This uses a degenerate polygon for P'', which is allowed. One could make it non-degenerate by thickening T into T' and then re-triangulating $P - T'$.

a triangle by definition of a skewed level set. We now map from R_c to R_s in the obvious way, by mapping the four corners of the trapezoid to the corresponding corners in the convex quadrilateral, and mapping all other points by interpolation. See Figure 2.7.

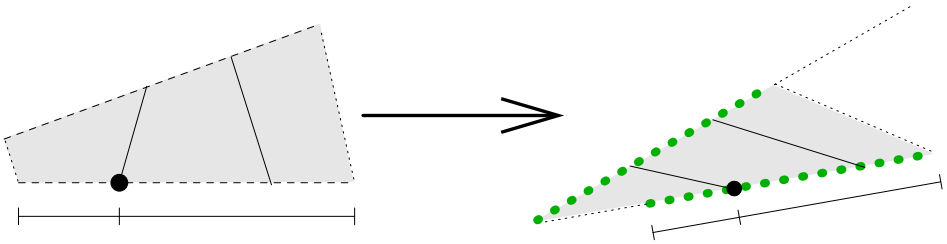

Figure 2.7: Mapping the cyclic levels to skewed levels that have been retracted such that each vertex v lands on its corresponding point s.

This maps each vertex to its desired point in S by construction. Let p be any point on l_c^1 where an edge e crossed l_c^1. This point maps to some point p' on $(l_s^1)'$, which will (usually) become a bend in the resulting drawing of e. Any bend that e had inside R_c (e.g., because e made a turn inside R_c) will be mapped into a bend in R_s as well. We complete the drawings of edges by putting in straight-line segments between these created bends and endpoints.

Since the drawing inside R_s is a linear mapping of the drawing in R_c, we do not create any crossing inside R_s. Since each quadrilateral of consecutive skewed levels was inside a face of a triangulation of $P' - P''$, no crossings can occur between two segments in two different quadrilaterals. Therefore we obtain a planar drawing.

In the cyclic level drawing, edges had $O(|L_C|) = O(n + k)$ bends. Mapping to the skewed levels does not introduce new bends (all edges already had bends where they crossed levels.) So every edge had $O(n + k)$ bends, which proves the theorem. □

2.3.4 Counting the bends

In the previous section, we were only concerned with asymptotic bounds on the number of bends. However, it is possible to give more precise bounds for the actual construction (if we do not consider the additional corners/vertices resulting from making polygon P simple or making graph G Hamiltonian.) The main observation is that bends only happen at turns or when an edge crosses a level, and we can bound these events.

Lemma 2.7 *For any set S of n points inside a simple k-sided polygon P, there exists a set of $n + 5k - 12$ skewed levels.*

Proof: Every point in S gives rise to one skewed level. In addition, we added skewed levels from the triangulation of P, and we count their number now.

The triangulation of P had $k-3$ edges. Each of those obtains a subdivision point and hence gives rise to $2(k-3)$ skewed levels. The triangulation also had $k-2$ interior faces. Each of those obtains a face point which is connected to three corners, hence adding $3(k-2)$ skewed levels. In total we hence have $2(k-3)+3(k-2) = 5k-12$ skewed levels. □

Since every edge makes at most 2 turns, it can traverse each level at most 3 times. For each traversal, the edge may only bend $n+5k-122$ times, for a total of at most $3n+15k-36$ bends plus one bend at each turn, which gives the precise bounds for Hamiltonian graphs.

Theorem 2.8 *Any Hamiltonian planar graph G can be drawn with vertices at prespecified points S inside a given simple polygon P such that any edge has at most $3|V|+15|P|-34$ bends.*

2.4 Conclusions and open problems

In this paper, we studied the problem of drawing a planar graph with vertices at specified locations inside a given polygon. We provide lower bounds for the number of bends, and give a construction that matches these lower bounds asymptotically.

Our construction was done with the theoretical objective of matching the lower bounds; we make no claims as to it being aesthetically pleasing or useful in practice. In particular our method of dealing with holes in the polygon by simply forbidding some region to be used at all would be unsatisfactory in a practical setting. One should also apply post-processing heuristics to remove many unnecessary bends.

We leave some open questions:

- Our lower bound uses a polygon that is unlikely to occur in practice. Can better bounds be shown for special polygons? For example, if a polygon can be split into K convex pieces (not necessarily triangles), can we create planar drawings at specified points inside the polygon with $O(n+K)$ bends per edge?
- What are area considerations? In particular, what area is required for our drawings, presuming all corners of P and points in S are at integer coordinates (say of size $O(n)$)?

Chapter 3

Subgraph Isomorphism

3.1 Are some patterns easier than others?

3.1.1 Introduction

The *induced subgraph isomorphism* problem is to detect if a host graph has an induced subgraph that is isomorphic to a pattern graph. Its counting variant asks for the number of induced subgraphs of the host graph isomorphic to the pattern graph. The well known independent set and clique problems are special cases of the induced subgraph isomorphism problem which consequently is generally NP-complete [29]. When the pattern graph is of fixed size, induced subgraph isomorphism can be solved in polynomial time even by exhaustive search.

In the literature, there are only a few examples of pattern graphs of fixed size k for which the induced subgraph isomorphism admits lower asymptotic time upper bound in terms of the number of vertices of the host graph than those known for the k-clique problem (for general host graphs). The oldest and most striking example is P_4, a path on four vertices, which can be detected in $O(n + m)$ time, where $n,\ m$ stand for the number of vertices and edges in the host graph [15]. The other is P_3, the path on three vertices which can be detected in $O(n+m)$ time [61], the third example is the diamond, obtained by removing a single edge from K_4, which can be detected in $O(n + m^{3/2})$ time [22] (cf. [42]). The fourth example is a paw which is a triangle connected to the fourth vertex by an edge, i.e., $K_3 + e$. It can be detected in $O(n^{2.376})$ time

(In fact, by considering the complement graph, the analogous bounds hold for the pattern graphs consisting of two adjacent vertices and one or two isolated vertices, or two incident edges and one isolated vertex, respectively.) In comparison, K_3 and K_4 can be detected and counted in $O(n^{2.376})$ time [38] and $O(n^{3.334})$ time [22], respectively. Interestingly, such a gap between the time upper bounds for P_4 and K_4, for

P_3 and K_3, and for the diamond and K_4, respectively, is not possible in the counting variant by [42, 44].

In the extreme case, when the pattern graph is a set of k isolated vertices, the induced subgraph isomorphism is equally hard as the k-clique problem if the time complexity is a function of the number of vertices. (This is in sharp contrast with the general subgraph isomorphism problem which for the aforementioned pattern becomes trivial.) Therefore, we can naturally pose the following conjecture:
There exists a constant C such that the time complexity of the problems of detecting (counting) the induced subgraphs isomorphic to a given k-vertex pattern graph in an n-vertex host graph is lower bounded by that of detecting (counting) the induced subgraphs of an n-vertex graph isomorphic to an independent set on k/C vertices.
In the counting variant, one could strongly conjecture $C = 1$ while in the detection variant the smallest value of C that one could conjecture is 2 in view of the result on P_4 [15].

By time complexity in the conjecture and throughout the paper, we mean the worst-case asymptotic time complexity in terms of the number n of vertices in the host graph under the assumption that the size of pattern graph is fixed. This allows for reductions of fixed independent set problems to induced fixed subgraph problems which are linear with respect to the number of vertices but not necessarily preserve graph sparsity so their time complexity can be even quadratic in the number of vertices.

Importantly, we assume arbitrary host graphs in the conjecture. Otherwise, one can easily come up with examples of classes of host graphs for which the topology of the pattern graph on k vertices affects the complexity of induced subgraph isomorphism. E.g., counting independent sets of size k for $k \geq 5$ in a planar graph does not seem to be an easy task while counting occurrences of K_k for $k \geq 5$ in a planar graph is trivial.

A related conjecture would be to claim that the hardness of induced subgraph isomorphism depends on the maximum sizes of an independent set and a clique in the pattern graph.

In the context of our conjecture, let us recall that the problems of detecting an independent set on k vertices and detecting a clique on k vertices in a host graph on n vertices are known to be W[1]-hard in the theory of parametrized complexity and believed to require $n^{\Omega(k)}$ time [20].

Known results supporting the conjecture. Already in 1985, Nešetřil and Poljak showed in [53] that the detection and counting versions of the induced subgraph isomorphism with fixed pattern graph on k vertices are easily reducible to the corresponding versions of the k-clique problem (or, equivalently, the k-independent set problem) in $O(kn^2)$ time, where n is the number of vertices of the host graph.

More recently, Chen and Flum [11] adapted the reduction of log clique to log chordless path due to Papadimitriou and Yannakakis [56] to show that detecting an induced path of length $4k - 1$ is not easier than (i.e., its time complexity is lower bounded by

subgraph	time complexity	reference
K_4 $(4K_1)$	$\mathcal{O}(n^{3.334})$ $(\mathcal{O}(m^{1.682}))$	Eisenbrand-Grandoni [22]
$K_4 \setminus e$ $(K_2 + 2K_1)$	$\mathcal{O}(m^{3/2})$	Eisenbrand-Grandoni [22]
C_4 $(2K_2)$	$\mathcal{O}(n^{3.334})$	Eisenbrand-Grandoni [22]
$K_3 + e$ $(P_3 + K_1)$	$\mathcal{O}(n^{\omega})$	Olariu [54]
$K_{1,3}$ $(K_3 + K_1)$	$\mathcal{O}(m^{(\omega+1)/2})$ $\mathcal{O}(n^{3.334})$	Kloks et al. [42] Eisenbrand-Grandoni [22]
P_4 (P_4)	$\mathcal{O}(n + m)$	Corneil et al. [15]

Table 3.1: Known upper time bounds for detecting induced subgraphs on 4 vertices in an undirected, unweighted graph on n vertices. The complement pattern graphs are given in parentheses.

that of) detecting an independent set on k vertices.

They also showed that a induced cycle on $4k$, C_{4k}, is not easier to detect than an independent set on k vertices. A stronger result for C_5 showing that it is not easier to detect than K_3 (equivalently, $3K_1$) is folklore. Also, it is easy to observe that the claw on four vertices, i.e., a graph consisting of three edges all sharing the same vertex, is not easier to detect than K_3.

In [42], Kloks, Kratsch and Müller showed that in the induced case if the occurrences of some pattern graph on 4 vertices can be counted in $T(n)$ time then the occurrences of any other pattern graph on 4 vertices can be counted in $O(n^{\omega} + T(n))$ time, where ω is the exponent of fast matrix multiplication known to be not greater than 2.376 [14]. Recently, Kowaluk et al. generalized the aforementioned result in [44] by showing that the knowledge of the number of occurrences of any pattern graph on k vertices as an induced subgraph is sufficient to compute the number of occurrences of any other pattern graph on k vertices both as induced and non-necessarily induced subgraph in time $O(n^{\omega(\lceil (k-2)/2 \rceil, 1, \lfloor (k-2)/2 \rfloor)})$, where $\omega(p, q, r)$ is the exponent of fast arithmetic matrix multiplication of an $n^p \times n^q$ matrix by an $n^q \times n^r$ matrix [13, 36].

The aforementioned generalization is interesting solely for fairly small k in view of the following fact: the detection and counting versions of the induced subgraph isomorphism problem for k-vertex pattern graphs can be solved in time $O(n^{\omega(\lfloor k/3 \rfloor, \lceil (k-1)/3 \rceil, \lceil k/3 \rceil)})$ [22] (cf. [42, 53]).

Examples of surprisingly fast algorithms for fixed size induced subgraph isomorphism.

The most striking result is clearly that on detection of induced P_4 in $O(n+m)$ time [15].

The aforementioned $O(n + m)$-time algorithm for the detection of induced P_3 [61]

and $O(n + m^{3/2})$-time algorithm for the detection of induced diamond have been generalized to an $O(n + m^{(k-1)/2})$-time algorithm for the detection of induced K_k with a single missing edge, denoted by $K_k \setminus e$, by Vassilevska in [61]. Observe that P_3 and the diamond can be denoted as $K_3 \setminus e$ and $K_4 \setminus e$, respectively. By considering the complement graph, we obtain also an analogous time bound $O(n^{k-1})$ for the detection of the pattern graph consisting of a pair of adjacent vertices and $k-2$ isolated vertices, denoted by $K_2 + (k-2)K_1$. The generalized upper bound $O(n^{k-1})$ for $K_k \setminus e$ and $K_2 + (k-2)K_1$ is however subsumed for $k > 5$ by that universal upper bound $O(n^{\omega(\lfloor k/3 \rfloor, \lceil (k-1)/3 \rceil, \lceil k/3 \rceil)}$ [22] valid for all pattern graphs on k vertices.

Furthermore, in a recent manuscript [35], the authors prove an analogous $O(n + m^{(k-1)/2})$-time bound for the problem of detection of induced path on k vertices, P_k.

Our contributions. We present two main results on the hardness (i.e., the time complexity) of detecting and counting induced subgraphs of fixed size.

For *detection*, we provide a substantially more general and stronger result than those on chordless induced path and cycle from [11].

We show that any fixed pattern graph with a maximum independent set of size k that is disjoint from other maximum independent sets is not easier to detect as an induced subgraph than an independent set of size k. It follows in particular that an induced path on $2k-1$ vertices is not easier to detect than an independent set on k vertices and that an induced cycle on $2k$ vertices is not easier to detect than an independent set on k vertices. We can also conclude that an induced complete bipartite graph $K_{p,q}$ is not easier to detect than an independent set on $\max\{p, q\}$ vertices.

Our second result is concerned with both *detection* and *counting*. It can be regarded as a generalization of the aforementioned results on chordless induced path and cycle [11], basically showing that no pattern topology is easier to detect or count.

For an arbitrary pattern graph H on k vertices with no isolated vertices, let H' be the subdivision of H obtained from H by splitting each edge into a path of length four and attaching a distinct path of length three at each vertex of degree one. We show that H' is not easier to detect or count than an independent set on k vertices, respectively.

Finally, we show that the diamond and paw are not easier to detect as induced subgraphs than an independent set on three vertices.

Organization. In the next section, we present our lower bound on detecting induced subgraphs isomorphic to restricted pattern graphs in terms of the size of independent set that is not easier to detect. In Section 3.1.3, we provide our lower bound on detecting and counting induced subgraphs isomorphic to restricted pattern graphs in terms of the size of independent set that is not easier to detect or count, respectively. In Section 3.1.4, we present simple lower bounds implying that the diamond and paw are not easier to detect as induced subgraphs than a triangle. We conclude with final remarks.

3.1.2 Lower bounds on detecting induced subgraphs

Chen and Flum demonstrated the hardness of the induced path and induced cycle problems in [11]. We can state precisely their results as follows.

Fact 1. *Let G be an arbitrary graph on n vertices and m edges. In $O(kn^2 + k^2m)$ time, one can construct a graph G' on $O(kn)$ vertices and $O(kn^2 + k^2m)$ edges such that G' has an induced subgraph isomorphic to a path on $4k - 1$ vertices iff G has an independent set of cardinality k.*

Similarly, one can construct a graph G'' on $O(kn)$ vertices and $O(kn^2 + k^2m)$ edges such that G'' has an induced subgraph isomorphic to a cycle on $4k$ vertices iff G has an independent set of cardinality k.

In this section, we provide a general equivalence which works in case of detection for arbitrary pattern graphs with a maximum independent set disjoint from other maximum independent sets. It supports our conjecture if such a maximum independent set is relatively large. Our equivalence also subsumes that of Chen and Flum in the particular case of odd paths and even cycles.

Theorem 3.1 *Let G be an arbitrary graph on n vertices and m edges, and let H be a pattern graph on h vertices. Suppose that there is a maximum independent set of size k that is disjoint from all other maximum independent sets in H. In $O(kn^2+hkn+k^2m)$ time, one can construct a graph G^* on $O(h + kn)$ vertices and $O(kn^2 + hkn + k^2m)$ edges such that G^* has an induced subgraph isomorphic to H iff G has an independent set of cardinality k.*

Proof: Let $G = (V, E)$ and $H = (V_H, E_H)$. Next, let S be a maximum independent set that is disjoint from the other maximum independent sets in H.

G^* consists of $k = |S|$ cliques $G^*(i)$ on $V \times \{i\}$, where $i \in S$, and the subgraph H' of H induced by all vertices in V_H outside S. (Note that $H \cap G^* = H'$.) Additionally, G^* contains the following edges between the k cliques and H'. Two vertices (v, i), (u, j) from two different cliques $G^*(i)$ and $G^*(j)$ form an edge if $\{v, u\} \in E$ or $v = u$. Each vertex l of H' is connected by an edge with each vertex of each clique $G^*(i)$, where $\{l, i\} \in E_H$ and $i \in S$. There are no other edges in G^*. See Fig. 1 for an example.

Suppose that G has an independent set $\{v_1, v_2, ..., v_k\}$ on k vertices. Then, we map each vertex $i \in S$ on the vertex (v_i, i). Next, we map each vertex in $V_H \setminus S$ on itself. The image of H under this mapping is easily seen to induce a subgraph isomorphic to H in G^*.

Conversely, suppose that G^* has an induced subgraph H^* such that there is an isomorphism between H and H^*.

Consider a maximum independent set U of H^*. Let U'' be the subset of U outside of H' and let U' be the subset of U within H'.

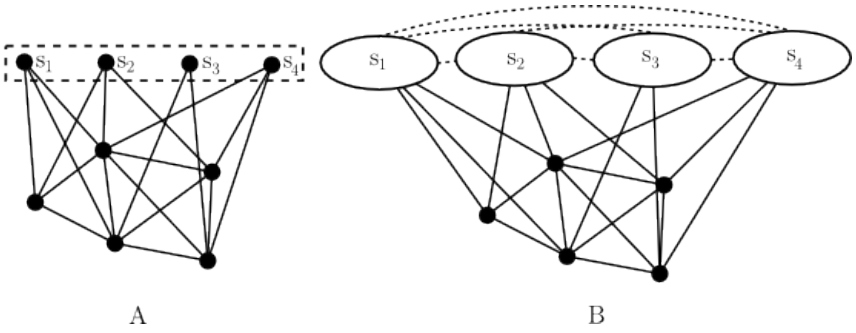

Figure 3.1: An example of a pattern graph (A) with a maximum independent set disjoint from others marked and the corresponding graph G^* (B), where the large vertices represent cliques and the dotted lines represent the edges of G between vertices from different cliques.

Since U'' is an independent set, then its vertices are in disjoint cliques $G^*(i)$. Let U''_S be the set of i for which there is a node of U'' in $G^*(i)$.

Now consider H. Observe that $V_H \setminus S$ is the set of vertices of H'. $U' \subset V_H \setminus S$ and $U''_S \subset S$ together form a maximum independent set of H. It properly intersects S, which yields a contradiction, unless $U''_S = S$ or $U''_S = \emptyset$. In the former case, we are done.

It remains to consider the situation where for each maximum independent set U of H^*, $U''_S = \emptyset$. This would however mean that H^* has all its maximum independent sets in the common subgraph H' of G^* and H. Consequently, H would have at least one more maximum independent set (S is outside H') than H^*. This would contradict the isomorphism between H^* and H.

□

Note that a path on $4k-1$ vertices as well as a cycle on $4k$ vertices have an independent set of cardinality $2k$. Odd paths as well as even cycles have at most two maximum independent sets, and they are always disjoint. Thus, Theorem 3.1 provides stronger lower bounds in terms of the size of independent set than Fact 1 in the particular case of induced odd paths and even cycles.

Corollary 3.2 *If H is a fixed pattern graph with a maximum independent set of cardinality k which is disjoint from other maximum independent sets (e.g., a path on $2k-1$ vertices or a cycle on $2k$ vertices) then the asymptotic complexity of the detection of an induced subgraph isomorphic to H in terms of the number of vertices of the host graph is not less than that of an independent set on k vertices.*

It is folklore that the detection of K_3 can be easily reduced to that of claw, i.e. $K_{1,3}$, by considering the complement graph expanded by an auxiliary vertex connected to all

vertices in the complement graph.

We obtain immediately the following much more general corollary from Theorem 3.1 and Corollary 3.2.

Corollary 3.3 *The time complexity of the detection of an induced subgraph isomorphic to the complete bipartite graph $K_{q,r}$ is lower bounded by that of an independent set on $\max\{q, r\}$ vertices.*

3.1.3 Lower bounds on detecting and counting induced subgraphs

We can also generalize the equivalence of Fact 1 to work not only for paths and cycles but for subdivisions of arbitrary pattern graphs without isolated vertices too, importantly both in case of detection and counting. The subdivisions replace each edge with a path with three additional inner vertices and attach an additional path at each vertex of degree one. Our next result basically shows that no pattern topology is easier to detect or count.

Theorem 3.4 *Let G be an arbitrary graph on n vertices and m edges, and let H be a pattern graph with h vertices and l edges and no isolated vertices. Next, let H_d be the subdivision of H obtained by placing three auxiliary vertices on each edge of H, and attaching at each leaf, i.e., vertex of degree 1, of H a distinct additional path of length three. In $O(hn^2 + ln + h^2m)$ time, one can construct a graph $G(h)$ on $O(hn)$ vertices and $O(hn^2 + ln + h^2m)$ edges such that the induced subgraphs of $G(h)$ isomorphic to H_d are in one-to-one correspondence with the independent sets in G of cardinality h.*

Proof: We form a graph $G(h)$ which basically consists of h copies of a clique on V, linked according to G. The h copies are additionally linked via auxiliary vertices in one-to-one correspondence with the edges of H. Furthermore, a path on three additional vertices is attached to each clique copy that corresponds to a leaf of H.

Let $G = (V, E)$ and $H = (V_H, E_H)$, and let L be the set of leaves in H. The vertex set of the i-th clique copy is $V \times \{i\}$ for $i \in V_H$. The set $V(h)$ of vertices of $G(h)$ is the union of $V \times V_H$ with the sets $\{a_{ij}, b_{ij}, c_{ij}\}$, where i and j, $i < j$, are adjacent vertices of H, and the sets $\{a_i, b_i, c_i\}$, where i is a leaf of H.

The set $E(h)$ of edges of $G(h)$ is the union of the following edge sets (see Fig. 2 for an illustration): $\bigcup_{i \in V_H} \{\{(u, i), (v, i)\} | u, v \in V \& u \neq v\}$

$$\bigcup_{\{i,j\} \subset V_H} \{\{(u, i), (v, j)\} | i \neq j \& u, v \in V \& (u = v \vee \{u, v\} \in E)\}$$

$$\bigcup_{\{i,j\} \subset V_H} \{\{(u, i), a_{ij}\} | i < j \& (a_{ij}\ defined)\}$$

$$\bigcup_{\{i,j\} \subset V_H} \{\{a_{ij}, b_{ij}\} | (a_{ij}\ defined) \& (b_{ij}\ defined)\}$$

$$\bigcup_{\{i,j\}\subset V_H} \{\{b_{ij}, c_{ij}\} | (b_{ij}\ defined) \& (c_{ij}\ defined)\}$$

$$\bigcup_{\{i,j\}\subset V_H} \{\{c_{ij}, (v,j)\} | i < j \& (c_{ij}\ defined)\}$$

$$\bigcup_{i\in L} \{\{(u,i), a_i\}\} \quad \bigcup_{i\in L} \{\{a_i, b_i\}\} \quad \bigcup_{i\in L} \{\{b_i, c_i\}\}$$

Claim. *Suppose that an embedding ϕ of H_d in $G(h)$ satisfies the following three conditions:*

- *for $l \in V_H$, $\phi(l)$ is a vertex in $V \times \{l\}$,*
- *for any two adjacent vertices i and j of H, where $i < j$, ϕ maps the three vertices between i and j on the path onto the three vertices in $\{a_{ij}, b_{ij}, c_{ij}\}$ so to form a path $\{\phi(i), a_{ij}\}$, $\{a_{ij}, b_{ij}\}$, $\{b_{ij}, c_{ij}\}$, $\{c_{ij}, \phi(j)\}$,*
- *for any leaf of H_d, the path leading to the associated leaf i of H is mapped on $\{c_i, b_i\}$, $\{b_i, a_i\}$, $\{a_i, \phi(i)\}$.*

Then,
(a) $\phi(H_d)$ is a subgraph of $G(h)$ isomorphic to H_d,
(b) $\phi(H_d)$ is an induced subgraph in $G(h)$ iff $\bigcup_{l\in V_H}\{\phi_1(l)\}$ is an independent set in G, where $\phi_1(l)$ stands for the first coordinate of $\phi(l)$,
(c) each induced subgraph of $G(h)$ isomorphic to H_d can be defined as the image of such an embedding ϕ composed with an automorphism of H_d,
(d) if μ is another embedding of H_d in $G(h)$ satisfying the aforementioned conditions and both $\phi(H_d)$ and $\mu(H_d)$ are induced subgraphs of $G(h)$ then $\bigcup_{l\in V_H}\{\phi_1(l)\}$ and $\bigcup_{l\in V_H}\{\mu_1(l)\}$ are different independent sets.
The (a) part follows directly from the specification of ϕ. Also by the specification, the image $\phi(H_d)$ is not an induced subgraph of $G(h)$ iff for some $i,\ j \in V_{H_d}$, the vertices

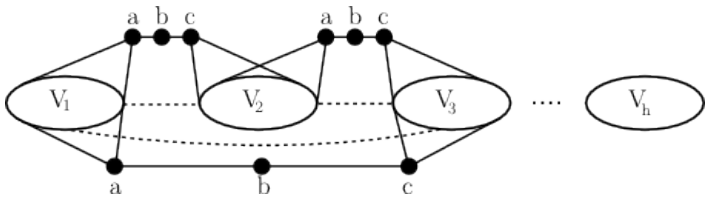

Figure 3.2: Example of the vertices and edges of $G(h)$, the large vertices $V_1, V_2, \ldots V_h$ represent the cliques of size n, one for each vertex in the pattern graph. The dotted lines represent the edges of G between vertices from different cliques.

$\phi(i)$ and $\phi(j)$ are adjacent in $G(h)$. Since each of them belongs to a different copy of the clique on V, this can only happen if $\phi_1(i) = \phi_1(j)$ or $\phi_1(i)$ is adjacent to $\phi_1(j)$ in G. In the first case, the set $\{\phi_1(1), ..., \phi_1(h)\}$ has size less than h, in the second one, it is not an independent set.

To prove part (c), consider an induced subgraph F of $G(h)$ isomorphic to H_d. The following observations will be useful:
(1) No vertex of F whose degree is at least three can be of the form $a_{i,j}$ or $b_{i,j}$, or $c_{i,j}$, or a_i, or b_i, or c_i since then either it would form a triangle with two vertices in the i-th or j-th copy of the clique on V or it would have degree at most two.
(2) All the vertices a_{ij}, b_{ij}, c_{ij} as well as all the vertices a_i, b_i, c_i, where i is a leaf of H, have to belong to F. It follows in particular that each leaf of H_d has to be mapped on some c_i in the isomorphism.

To see (2), denote by V_i the set $V \times \{i\}$ extended by the adjacent vertices a_{ij}, when $i < j$, and the adjacent vertices c_{ki}, when $i > k$, and halves of the in between vertices b_{ij}, as well as the vertices a_i, b_i, c_i in case $i \in L$.

Note that $V \times \{i\}$ can accommodate at most two vertices of F because the triangles do not occur in H_d. However, if $V \times \{i\}$ contains two vertices of F then the only additional vertices of F that can be accommodated by V_i are those placed at b_{ij}s counted as halves, again because of the absence of triangles, as well as b_i and c_i in case i is a leaf of H. If only one vertex of F is in $V \times \{i\}$ then V_i can accommodate additionally $1.5deg_H(i)$ vertices, plus three vertices in case i is a leaf of H, by fully using the vertices in $V_i \setminus V \times \{i\}$. The latter number of accommodated vertices is larger than that when $V \times \{i\}$ contains two vertices of F but for the case where $deg(i) = 1$, when the numbers are equal. In fact, each V_i has to accommodate the aforementioned maximum number in order to cover all vertices of F. Therefore, in particular all the vertices $a_i,\ b_i,\ c_i$ for leaves i of H have to be used by F. Hence, no b_{ij} can be used as a vertex of degree 1 which implies that each $V \times \{i\}$ contains exactly one vertex of F, and consequently all the vertices a_{ij}, b_{ij}, c_{ij} have to be used by F. This proves (2).

Suppose first that H is different from a simple cycle.

Consider a maximal path P with inner vertices of degree at most 2 in F. Suppose first that P has both endpoints of degree at least three. Let p_1 be the vertex on P within distance four from an endpoint p of P. By (1), p is in $V \times \{i\}$ for some i. If P does not continue from p through some a_{ij}, b_{ij}, c_{ij} or vice versa then by (2) the degree of p in H has to be larger than the number of such paths linked to $V \times \{i\}$. This in turn means that there is $l \in V_H$, where no all paths of this form linked to $V \times \{l\}$ are used by F. We obtain a contradiction by (2). We conclude that P continues to p_1 in $V \times \{j\}$ by a path in one of the two aforementioned forms. By iterating this argument for p_1 etc., we infer that the vertices of P corresponding to the vertices of H are in distinct $V \times \{l\}$, whereas the vertices between are mapped on some triples a_{ij}, b_{ij}, c_{ij}.

If P has an endpoint of degree 1 then it has some vertex c_l as an endpoint. Hence, the vertices b_l, a_l and some vertex p in $V \times \{l\}$ corresponding to a leaf of H have to follow. Then, we can continue with p similarly as in the previous case. In case the other endpoint is also of degree 1 then when P reaches a vertex in some $V \times \{q\}$ corresponding to another leaf of H, it has to have a_q, b_q and c_q as a suffix.

In case H_d is a simple cycle, we pick an arbitrary vertex of F in some $V \times \{l\}$ as the start and endpoint of a path P with inner vertices of degree 2 and proceed analogously as in the previous cases. Then every fourth following vertex of F will be also in some $V \times \{l\}$. If these vertices are not images of the original vertices of H, we need to compose an embedding ϕ satisfying the three conditions with an automorphism (shift) of H_d.

This completes the proof of the (c) part of the claim.

Now (d) follows by (b) from the fact that an embedding ϕ satisfying the three conditions is uniquely determined by the choice of the second coordinates of the clique vertices. The claim yields the theorem. □

Corollary 3.5 *Let H be a fixed graph on h vertices, and let its subdivision H_d be defined as in Theorem 3.4. The problems of detecting and counting induced subgraphs isomorphic to H_d have asymptotic time complexity in terms of the number of vertices of the host graph not less than those for the corresponding problems for independent set on h vertices, respectively.*

3.1.4 Simple lower bounds

We can expand the list of lower bounds on detection for pattern graphs on four vertices in terms of K_3 (for $K_{1,3}$ cf. Corollary 3.3) by the following ones for the diamond $K_4 \setminus e$ and the paw $K_3 + e$

Theorem 3.6 *Let $k \geq 4$. The time complexity of the detection of an induced subgraph isomorphic to $K_k \setminus e$ is lower bounded by that of an independent set on $k-1$ vertices (K_{k-1} equivalently).*

Proof: Augment an arbitrary host graph G with single copies of its vertices. For a copy v' of a vertex v, add edges between v' and all neighbours of v in G. Let G' denote the resulting graph, where the copy vertices form an independent set.

If G contains a K_{k-1} induced by $(u, ..., w)$ then G' contains the subgraph induced by $(u, ..., w, u')$ which is a $K_k \setminus e$. Conversely, if G' contains an induced $K_k \setminus e$ then the latter either includes a K_{k-1} of G or it is induced by a sequence $(u', v, ..., w, z')$. In the latter case, G contains the subgraphs induced by $(u, v, ..., w)$ and $(v, ..., w, z)$, both are K_{k-1}. □

By $K_k + e$, we denote a graph consisting of a clique on k vertices and an additional vertex connected by a single edge with the clique. In particular, the paw is $K_3 + e$.

Theorem 3.7 *Let $k \geq 3$. The time complexity of the detection of an induced subgraph isomorphic to $K_k + e$ is lower bounded by that of an independent set on k vertices (K_k equivalently).*

Proof: Augment an arbitrary host graph G with single copies of its vertices adjacent solely to their original counterparts to form a graph G'. Observe that G contains a K_k iff G' contains $K_k + e$. □

3.1.5 Final remarks

The fast universal algorithms for induced subgraph isomorphism with a fixed pattern graph given in [22, 42, 53] are generalizations of the method of detecting or counting triangles via fast matrix multiplication [38]. Our conjecture is strongly related to the fact that the aforementioned algorithms work equally well for all possible pattern graphs of a given size.

It is an interesting open problem if our lower bound on induced subgraph isomorphism in terms of the size of maximum independent set that is required to be disjoint from other maximum independent sets (i.e., Theorem 3.1) can be generalized by skipping the latter requirement.

3.2 Detecting and counting small pattern graphs

3.2.1 Introduction

The problems of detecting subgraphs or induced subgraphs of a graph that are isomorphic to another given graph are classical in algorithmics. They are generally termed as *subgraph isomorphism* and *induced subgraph isomorphism* problems, respectively.

In particular, they include special cases such as well-known NP-hard problems as the independent set, clique, Hamiltonian cycle or Hamiltonian path problems.

Recently, the detection and/or counting variants of subgraph isomorphism and/or induced subgraph isomorphism have found several applications, e.g., in bio-molecular networks [1], social networks [58], automatic design of processor systems and network security [32]. In these applications pattern graphs are typically of fixed size which allows for polynomial-time solutions.

The fastest known general algorithms for the detection and counting variants of subgraph isomorphism and induced subgraph isomorphism, where the pattern graph has k vertices while the host graph has n vertices, run in time $O(n^{\omega(\lfloor k/3 \rfloor, \lceil (k-1)/3 \rceil, \lceil k/3 \rceil)})$

[22, 42, 53], where $\omega(p,q,r)$ denotes the exponent of fast matrix multiplication for rectangular matrices of size $n^p \times n^q$ and $n^q \times n^r$, respectively [45]. For special graph classes faster algorithms are known (e.g., see [27, 44, 62]).

In the first part of our paper (Section 3.1.3), we study the detection variant of the induced subgraph isomorphism problem for pattern graphs of fixed size k, while in the second part (Section 3.1.4), we study counting variants of the general subgraph isomorphism problem for such pattern graphs. We denote the number of vertices and edges in the host graph by n and m.

Detection of small induced subgraphs: In the literature, besides the naive $O(n^k)$-time method for the induced subgraph isomorphism, the method reducing the problem to triangle detection, or counting, respectively, is known when the pattern graph is an *arbitrary* fixed graph on k vertices. The underlying triangle problem can be solved by fast (rectangular) matrix multiplication which yields the upper bound of $O(n^{\omega(\lfloor k/3 \rfloor, \lceil (k-1)/3 \rceil, \lceil k/3 \rceil)})$ for the induced subgraph isomorphism [22, 42, 53]. Because fast matrix multiplication algorithms rely on algebra, the aforementioned method can be classified as non-combinatorial. Since fast matrix multiplication algorithms involve large overheads, the method is not very practical. The other drawback is that it is not sensitive to the topology of the pattern graph and yields the same upper bound for any k-vertex pattern graph (e.g., K_k, i.e., the k-clique).

There are a few known examples of pattern graphs of fixed size k for which one succeeded to design specific algorithms for induced subgraph isomorphism yielding asymptotic time upper bounds in terms of n lower than those offered by the aforementioned triangle based method. The oldest and most striking example is P_4, a path on four vertices, which can be detected in $O(n+m)$ time [15].

The other is P_3, the path on three vertices which can be also detected in $O(n+m)$ time [61], the third example is the diamond, obtained by removing a single edge from K_4, which can be detected in $O(n+m^{3/2})$ time [22] (cf. [42]). The fourth example is a paw which is a triangle connected to the fourth vertex by an edge, i.e., $K_3 + e$. It can be detected in $O(n^{2.373})$ time [54, 64]. (Analogous upper bounds hold for the pattern graphs that are the complement to one of the aforementioned pattern graphs.)

Furthermore, an induced subgraph isomorphic to the generalized diamond $K_k - e$, i.e., K_k with a single edge removed, as well as an induced subgraph isomorphic to the path on k vertices, P_k, can be detected in $O(n^{k-1})$ time [35, 61] which improves the triangle based bound from [22] for $k \leq 5$.

For recent relative hardness results on detection of specific pattern graphs (e.g., $P_k,\ K_{p,q},\ K_{k-1}+e,\ K_k - e,\ C_4$) in the induced setting, see [26].

The triangle based method was refined by the use of fast rectangular matrix multiplication in [22] a decade ago. Since then no new general approaches to induced subgraph isomorphism for fixed pattern graphs have been presented.

We present a new framework for detecting an induced subgraph of fixed size k

in a host graph on n vertices (Section 3.1.3). We associate a multivariate polynomial to a family of pattern graphs on k vertices that share both a subgraph on l vertices and the edges between the common subgraph and the remaining $k - l$ vertices outside the subgraph. The monomials of the polynomial are in one-to-one correspondence with the pattern graphs in the family and their coefficients are computed on the basis of the corresponding pattern graphs.

If all the coefficients but one share a prime factor p, we can detect the pattern graph corresponding to the monomial whose coefficient is *not* divisible by p, by verifying the polynomial for non-identity with zero over a field of characteristic p. The crucial part of our proof is showing that the polynomial can be evaluated in $O(n^{l+1})$ time, which enables us to use the DeMillo-Lipton-Schwartz-Zippel lemma for the verification of the polynomial.

By applying our method, we can list sixteen pattern graphs on five vertices that can be detected in $O(n^4)$ time. With the exception of P_5 [35] and $K_5 - e$ [61], our upper bounds of $O(n^4)$ are new and in particular improve the bounds yielded by the triangle based method (Corollary 3.17). We can obtain also the upper time bound of $O(n^{k-1})$ for plenty of pattern graphs on $k > 5$ vertices. Although for so large pattern graphs, we cannot improve the upper bounds yielded by the triangle based method, the application of our combinatorial method not relying on fast matrix multiplication can be still be of practical interest.

For all graphs on four vertices except K_4, $K_{1,3}$ and C_4, and their complements, our method yields the upper bound of $O(n^3)$, which is better than that yielded by the triangle based method.

Although our upper bounds for pattern graphs on four vertices do not improve the known bounds based on different specific methods [22, 42, 54], they have the advantage of not relying on the fast matrix multiplication algorithms and of being sensitive to the topology of the pattern graph. Similarly, for pattern graphs on three vertices, our method yields the upper bound of $O(n^2)$ in all the cases for which an upper time bound lower than that yielded by the triangle based method is known.

Our main technical contribution in the first part of our paper is as follows.

Let $\mathcal{H}_k$ denote the family of single representatives of all isomorphism classes of undirected graphs on k vertices, and let $\mathcal{H}_k(l)$ stand for its subfamily comprised of all graphs in $\mathcal{H}_k$ having an independent set of size at least $k - l$. Consider $H \in \mathcal{H}_k(l)$ and an induced subgraph H_{sub} of H on l vertices such that the $k - l$ vertices in $H \setminus H_{sub}$ form an independent set. Let $\mathcal{H}_k(H_{sub}, H)$ stand for the family of all supergraphs H' of H (including H) such that H' has the same vertex set as H, H_{sub} is also an induced subgraph of H', and the set of edges with endpoints in both H_{sub} and $H' \setminus H_{sub}$ is the same as that with endpoints in both H_{sub} and $H \setminus H_{sub}$ (see Fig 3.3(a,b)).

Finally, for each $H' \in \mathcal{H}_k(H_{sub}, H)$, let $B(H_{sub}, H, H')$ denote the number of isomorphisms between H_{sub} and an induced subgraph of H', say H^f_{sub}, that can be

extended to an isomorphism between H and the subgraph of H' consisting of H'_{sub}, all edges of H' incident to H'_{sub} and all the remaining vertices of H'.

We obtain the following result (Theorems 3.12, 3.13).

Let $F \in \mathcal{H}_k(H_{sub}, H)$, where $k = O(1)$. Suppose that there is a prime number p that is a factor of $B(H_{sub}, H, H')$ for all $H' \in \mathcal{H}_k(H_{sub}, H)$, except for $H' = F$. There is a randomized algorithm that detects if a graph on n vertices contains an induced subgraph isomorphic to F, with one-sided error of probability polynomially small in n (i.e., $O(n^{-\alpha})$ for $\alpha > 1$), in $O(n^{l+1})$ time. Importantly, for $k - l = 2$, $\mathcal{H}_k(H_{sub}, H)$ contains two graphs and it is sufficient to require that p is a prime factor of the number of automorphisms for the other graph in $\mathcal{H}_k(H_{sub}, H)$ but it is not a prime factor of the number of automorphisms of F.

The idea of associating a polynomial over a finite field to the sought structure has been already used by Edmonds to detect a perfect matching [21]. It appears in several recent papers that exploit also the idea of monomial cancellation [7, 43].

Counting subgraph isomorphisms for fixed pattern graphs: Vassilevska and Williams studied the counting variant of subgraph isomorphism under the assumption that the k-vertex pattern graph has an independent set of size s [62]. They designed combinatorial algorithms (i.e., not relying on fast matrix multiplication) for this counting problem running in time $O(f(s)n^{k-s+2})$ where f is an exponential or super-exponential function depending only on s. Subsequently, Kowaluk et al. [44] designed an algorithm for the corresponding detection problem using fast rectangular matrix multiplication and running in time $O(n^{\omega(\lceil (k-s)/2 \rceil, 1, \lfloor (k-s)/2 \rfloor)}) \leq O(n^{k-s+1})$ when $k = O(1)$. They also established an analogous upper bound for the counting variant when the size s of the independent set is 2.

By a *subgraph isomorphism* between the pattern graph and the host graph, we shall mean a one-to-one mapping of vertices in the pattern graph into vertices of the host graph that preserves vertex adjacency.

In the second part of our paper (Section 3.1.4), we present an algorithm for counting subgraph isomorphisms between a pattern graph with k vertices and an independent set of cardinality s and a host graph with n vertices. It runs in time $O(n^{\omega(\lceil (k-s)/2 \rceil, 1, \lfloor (k-s)/2 \rfloor)})$ which matches the upper bound for detection from [44] and largely extends that for counting showed only for $s \leq 2$ in [44].

Our algorithm relies on a solution to the so called $(k-s)$-neighbourhood problem from [44], which in turn relies on fast rectangular matrix multiplication.

We also consider a weighted version of the counting problem, where real weights are assigned to the edges and/or vertices of the host graph, and the task is to count the number of subgraph isomorphisms between the pattern graph with k vertices containing an independent set of cardinality s and the host graph with n vertices that minimize the total weight of the images of the pattern graph. For this more general counting problem we design a slightly slower combinatorial algorithm running in $O(n^{k-s+1} \log n)$ time

when $k = O(1)$.

In the literature, various weighted versions of the counting variants of subgraph isomorphisms have been studied solely in terms of the sizes of the pattern and host graphs, and without any explicit assumption on the size of independent set in the pattern graph [16, 62].

3.2.2 Preliminaries

An *isomorphism* between two graphs F and G is a one-to-one mapping f of the vertices of F onto vertices of G such that $\{u, v\}$ is an edge of F iff $\{f(u), f(v)\}$ is an edge of G. If $F = G$ then an isomorphism between F and G is called an *automorphism* of F. F is *isomorphic* to G if there is an isomorphism between F and G.

A *subgraph* of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. Such a subgraph G' of G is *induced* if $E' = (V' \times V') \cap E$.

A *subgraph isomorphism* between two graphs F and G is an isomorphism between F and a subgraph of G.

The *detection version* (or equivalently, the *decision version*) of the *subgraph isomorphism* problem is to decide for a host graph and a pattern graph if the host graph has a subgraph isomorphic to the pattern graph.

The *counting version* of subgraph isomorphism asks for reporting the total number of subgraphs of the host graph isomorphic to the pattern graph or just the total number of subgraph isomorphisms between these two graphs. The corresponding versions of *induced subgraph isomorphism* are defined analogously by replacing "subgraph" with "induced subgraph".

Let S be a subgraph of G with an order on its l vertices, or just an ordered subset of l vertices of G. The *S-neighbourhood type* of a vertex of G is a binary vector b with l coordinates such that $b(i) = 1$ iff v is adjacent to the i-th vertex of S for $i = 1, ..., l$.

The *l-neighbourhood problem* is to determine, for each ordered l-tuple α of vertices of G and each binary vector b with l coordinates, the number of vertices v in G outside α such that their α-neighbourhood type is b.

Fact 1 [44]. *The l-neighbourhood problem for a graph on n vertices can be solved in $O(n)$ time for $l = 1$ and in $O(2^l n^{\omega(\lceil l/2 \rceil, 1, \lfloor l/2 \rfloor)})$ time for $l \geq 2$.*

3.2.3 A new method of detecting small induced subgraphs

Let H be a pattern graph on k vertices in $\mathcal{H}_k(l)$ (see the introduction) and let H_{sub} be an induced subgraph of H on l vertices such that the $k - l$ vertices in $H \setminus H_{sub}$ form an independent set. Recall the definition of the family $\mathcal{H}_k(H_{sub}, H)$ of supergraphs of H and the definition of the quantities $B(H_{sub}, H, H')$, for $H' \in \mathcal{H}_k(H_{sub}, H)$, given

in the introduction. Let $\mathcal{SH}_k(H_{sub}, H)$ stand for the family of single representatives of all isomorphism classes in $\mathcal{H}_k(H_{sub}, H)$, i.e., one graph from each isomorphism class.

Finally, for a pattern graph H and a host graph G, let $SI(H, G)$ be the set of all subsets of $V(G)$ on $|H|$ vertices that induce a subgraph of G isomorphic to H. Next, let $PI(H, G)$ denote the multivariate polynomial $\sum_{S \in SI(H,G)} \prod_{v \in S} x_v$. We define the multivariate polynomial $P(H_{sub}, H, G)$ by $\sum_{H' \in \mathcal{SH}_k(H_{sub},H)} B(H_{sub}, H, H')PI(H', G)$.

To state our key lemma, for a prime number p, $H \in \mathcal{H}_k(l)$ and $H_{sub} \in \mathcal{H}_l$, we define the subset $\mathcal{H}_k^p(H_{sub}, H)$ of $\mathcal{H}_k(H_{sub}, H)$ as the set of all $H' \in \mathcal{H}_k(H_{sub}, H)$ for which $B(H_{sub}, H, H')$ is divisible by p. Also, recall that the *characteristic* of a ring or a field is the minimum number of 1 in a sum of ones that yields 0.

Lemma 3.8 *Let $H \in H_k(l)$ and let H_{sub} be an induced subgraph of H on l vertices such that the $k-l$ vertices in $H \setminus H_{sub}$ form an independent set. For a prime number p, a host graph G contains an induced subgraph isomorphic to a graph in $\mathcal{H}_k(H_{sub}, H) \setminus \mathcal{H}_k^p(H_{sub}, H)$ iff the polynomial $P(H_{sub}, H, G)$ is not identical to zero over a field of characteristic p.*

Proof: All the monomials with coefficients $B(H_{sub}, H, H')$, where $H' \in \mathcal{H}_k^p(H_{sub}, H)$, vanish over any field of characteristic p. On the other hand, those with the coefficient $B(H_{sub}, H, H')$, where $H' \in \mathcal{H}_k(H_{sub}, H) \setminus \mathcal{H}_k^p(H_{sub}, H)$, (if any) remain with a non-zero coefficient equal to $B(H_{sub}, H, H') \mod p$. It follows from the definition of the polynomial $P(H_{sub}, H, G)$ that it is not identical to zero iff G contains an induced subgraph isomorphic to a graph in $\mathcal{H}_k(H_{sub}, H) \setminus \mathcal{H}_k^p(H_{sub}, H)$. □

The following lemma on polynomial identity testing has been shown independently by DeMillo and Lipton, Schwartz, and Zippel.

Lemma 3.9 *[17, 60] Let $Q(x_1, x_2, ..., x_m)$ be a non-zero polynomial of degree d over a field of size r. Then, for f_1, f_2, ...,f_m chosen independently and uniformly at random from the field, the probability that $Q(f_1, f_2, ..., f_m)$ is not equal to zero is at least $1 - \frac{d}{r}$.*

The second part of Section 3.1.3 is devoted to the proof of the following key theorem.

Theorem 3.10 *Let p be a fixed prime number. For $H \in \mathcal{H}_k(l)$ and an induced subgraph H_{sub} of H on l vertices such that the $k-l$ vertices in $H \setminus H_{sub}$ form an independent set. The polynomial $P(H_{sub}, H, G)$ can be evaluated for a given assignment of values over a field $F_{p^{O(\log n)}}$ of characteristic p in $O(n^{l+1})$ time.*

By combining Lemmata 3.8, 3.9, and Theorem 3.10, we obtain our first main result.

Theorem 3.11 *Let $H \in \mathcal{H}_k(l)$, let H_{sub} be an induced subgraph of H on l vertices such that the $k-l$ vertices in $H \setminus H_{sub}$ form an independent set, and let p be a fixed*

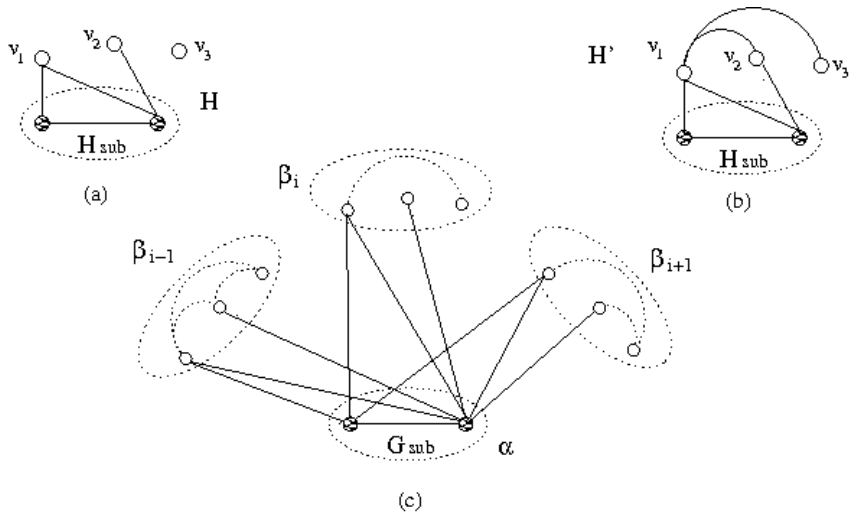

Figure 3.3: (a) An example of a graph H composed of the induced subgraph H_{sub} and the vertex set $\{v_1, v_2, v_3\}$ that forms an independent set in H. (b) An example of a supergraph H' of H_{sub} in $\mathcal{H}_k(H_{sub}, H)$. (c) An example of a set of $(k-l)$-tuples of vertices in G which are connected with the l-tuple α by edges corresponding to those between $H \setminus H_{sub}$ and H_{sub}.

prime number. There is a randomized algorithm that detects if a graph on n vertices contains an induced subgraph isomorphic to a graph in $\mathcal{H}_k(H_{sub}, H) \setminus \mathcal{H}_k^p(H_{sub}, H)$, with one-sided error probability polynomially small in n, in $O(n^{l+1})$ time.

Proof: By Lemma 3.8, it is sufficient to show how to test if the polynomial $P(H_{sub}, H, G)$ is not identical to zero, with one-sided error probability polynomially small in n, in $O(n^{l+1})$ time.

Note that the polynomial $P(H_{sub}, H, G)$ is of degree k. We can use Lemma 3.9 with a field $F_{p^{c \log n}}$ of characteristic p to obtain a randomized test of the polynomial $P(H_{sub}, H, G)$ for not being identical to zero with one side errors of probability not larger than $\frac{k}{p^{c \log n}}$. For sufficiently large constant c, the error probability is not larger than $\frac{1}{n^{\alpha}}$, $\alpha > 1$.

By Theorem 3.10, the test can be performed in $O(n^{l+1})$ time. □

Theorem 3.12 *Let $H \in \mathcal{H}_k(l)$, let H_{sub} be an induced subgraph of H on l vertices such that the $k-l$ vertices in $H \setminus H_{sub}$ form an independent set, and let $F \in \mathcal{H}_k(H_{sub}, H)$, where $k = O(1)$. Suppose that there is a prime number p that is a factor of $B(H_{sub}, H, H')$ for all $H' \in \mathcal{H}_k(H_{sub}, H)$, except for $H' = F$. There is a randomized algorithm that detects if a graph on n vertices contains an induced sub-*

graph isomorphic to F, with one-sided error of probability polynomially small in n, in $O(n^{l+1})$ time.

Note that in Theorem 3.12, if $k - l = 2$ then $\mathcal{H}_k(H_{sub}, H)$ contains two graphs.

Theorem 3.13 *Let $H \in H_k(k-2)$, let H_{sub} be an induced subgraph of H on $k-2$ vertices such that the two vertices in $H \setminus H_{sub}$ form an independent set, and let $F \in \mathcal{H}_k(H_{sub}, H)$, where $k = O(1)$. Suppose that there is a prime number p that is a factor of the number of automorphisms of the other $H' \in \mathcal{H}_k(H_{sub}, H)$ and that is not a prime factor of the number of automorphisms of F. There is a randomized algorithm that detects if a graph on n vertices contains an induced subgraph isomorphic to F, with one-sided error of probability polynomially small in n, in $O(n^{k-1})$ time.*

Proof: Let $H' \in \mathcal{H}_k(H_{sub}, H)$, and let $\mathcal{F}$ be the set of all isomorphisms f between H_{sub} and an induced subgraph of F satisfying the requirements from the definition of $B(H_{sub}, H, H')$.

Consider an extension of $f \in \mathcal{F}$ to an isomorphism between H and the subgraph of H' composed of H'^{f}_{sub}, all edges of H' incident to H'^{f}_{sub}, and all other vertices of H'. If $H' = H$ then f' is an automorphism of H'. Otherwise, H' is the other member of $\mathcal{H}_k(H_{sub}, H)$ obtained by adding the edge between the two independent vertices of H outside H_{sub}. Then, f' is also an automorphism of H' since the only edge in H' not incident to H'^{f}_{sub} has to connect the images by f' of the aforementioned two independent vertices in H.

It follows that each $f \in \mathcal{F}$ can be identified with the class of all automorphisms of H' that are equal to each other on H_{sub}. Conversely, each such class yields a distinct member in $\mathcal{F}$.

We conclude that $B(H_{sub}, H, H')$ is equal to the number of automorphisms of H' divided by the number of automorphisms of H' that are identity on H_{sub}. It remains to observe that the latter number is the same for both members in $\mathcal{H}_k(H_{sub}, H)$. Simply, for each $H' \in \mathcal{H}_k(H_{sub}, H)$, the set of automorphisms of H' that are identity on H_{sub} contains either only the identity automorphism or also the automorphism that switches the two vertices of H' outside H_{sub} in case their H_{sub}-neighbourhoods types are equal. □

Efficient evaluation of $P(H_{sub}, H, G)$: We shall define a polynomial equivalent with $P(H_{sub}, H, G)$ (more precisely, a different decomposition of $P(H_{sub}, H, G)$) and show that it can be efficiently evaluated. To begin with, we need the following notation and lemma.

Let α be a fixed ordered l-tuple of vertices of the graph H that induces the subgraph H_{sub} and for any $b \in \{0,1\}^l$, let $\eta_{\alpha,H}(b)$ be the number of vertices of the α-neighbourhood type b in H. By the definition of $\mathcal{H}_k(H_{sub}, H)$, we obtain the following lemma.

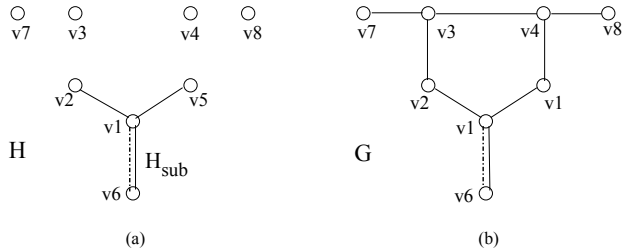

Figure 3.4: An example of a graph H isomorphic to $K_{1,3} + 4K_1$, its induced subgraph H_{sub} isomorphic to K_2, and a graph G containing three subgraphs isomorphic to H. Note that G has only two automorphisms.

Lemma 3.14 *Let f be an isomorphism between a graph $H' \in \mathcal{H}_k(H_{sub}, H)$ and a subgraph of G induced by a set S of k vertices in $V(G)$ and let $\alpha_f = (f(\alpha_1), ..., f(\alpha_l))$. The number of vertices of the α_f-neighbourhood type b in S equals $\eta_{\alpha,H}(b)$.*

For an ordered l-tuple γ of l different vertices in $V(G)$ and for $b \in \{0,1\}^l$, let $V(\gamma, b)$ be the set of all vertices of the γ-neighbourhood type b in $V(G) \setminus \gamma$. The polynomial $Q(\gamma, H, G)$ is defined by

$$\prod_{i=1}^{l} x_{\gamma_i} \prod_{b \in \{0,1\}^l \& \eta_{\alpha,H}(b) \neq 0} \left(\sum_{U \subseteq V(\gamma,b) \wedge |U| = \eta_{\gamma,H}(b)} \prod_{v \in U} x_v \right) .$$

Lemma 3.15 *Let γ be an ordered l-tuple of vertices in $V(G)$ inducing a subgraph of G isomorphic to H_{sub}, and let S be a set of k vertices in $V(G)$. The monomial $\prod_{v \in S} x_v$ occurs (exactly once) in $Q(\gamma, H, G)$ iff S includes the vertices of γ and there is an isomorphism between a graph in $\mathcal{H}_k(H_{sub}, H)$ and the subgraph of G induced by S that maps the i-th vertex of the l-tuple α on the i-th vertex of the l-tuple γ.*

Proof: To begin with, observe that each monomial of $Q(\gamma, H, G)$ is unique. If $\prod_{v \in S} x_v$ occurs in $Q(\gamma, H, G)$ then S has to include the vertices of γ by the definition of $Q(\gamma, H, G)$. Specify a mapping $g : V(H) \to S$ such that $g(\alpha_i) = \gamma_i$ for $i = 1, ..., l$, and for each $b \in \{0,1\}^l$, g maps the j-th vertex of the α-neighbourhood type b in $V(H)$ onto the j-th vertex of the γ-neighbourhood type b in S for $j = 1, .., \eta_{\alpha,H}(b)$ (for any orderings of the vertices of the respective type b). Now, observe that g defines an isomorphism between some graph in $\mathcal{H}_k(H_{sub}, H)$ and that induced by S in G.

Conversely, if there is an isomorphism f between some graph in $\mathcal{H}_k(H_{sub}, H)$ and the subgraph of G induced by S that maps α_i on γ_i for $i = 1, .., l$, then the l-tuple α_f in Lemma 3.14 equals γ and hence by this lemma and the definition of $Q(\gamma, H, G)$, $\prod_{v \in S} x_v$ occurs in $Q(\gamma, H, G)$. □

Let L be the set of all l-tuples γ of different l vertices such that there is an isomorphism between H_{sub} and the subgraph of G induced by γ that maps α_i on γ_i for $i = 1, ..., l$. Next, let $Q(H_{sub}, H, G) = \sum_{\gamma \in L} Q(\gamma, H, G)$.

The idea behind the following key lemma is as follows. Consider a monomial that corresponds to a subgraph G' of G isomorphic to $H' \in \mathcal{H}_k(H_{sub}, H)$ and occurs with the coefficient $B(H_{sub}, H, H')$ in $P(H_{sub}, H, G)$. It occurs once in each of the $Q(\beta, H, G)$ forming $Q(H_{sub}, H, G)$, where there is an extension of the function mapping the i-th vertex of α on the i-th vertex of β to a subgraph isomorphism between H and G', satisfying the requirements from the definition of $B(H_{sub}, H, H')$. Hence, the number of such β is $B(H_{sub}, H, H')$.

Lemma 3.16 *The equality $P(H_{sub}, H, G) = Q(H_{sub}, H, G)$ holds.*

Proof: By the definition of $P(H_{sub}, H, G)$, if $\prod_{v \in S} x_v$ is a monomial of this polynomial then there is an isomorphism f between a graph in $\mathcal{SH}_k(H_{sub}, H)$ and the subgraph of G induced by S. Let $\alpha_f = (f(\alpha_1), ..., f(\alpha_l))$. Then, $\prod_{v \in S} x_v$ is a monomial in $Q(\alpha_f, H, G)$ and hence also in $Q(H_{sub}, H, G)$.

Conversely, if $\prod_{v \in S} x_v$ is a monomial in $Q(H_{sub}, H, G)$, i.e., a monomial in $Q(\gamma, H, G)$ for some $\gamma \in L$, then it follows from Lemma 3.15 that the subgraph of G induced by S is isomorphic to a graph H' in $\mathcal{SH}_k(H_{sub}, H)$. Hence, it is also a monomial in $P(H_{sub}, H, G)$. To show the equality, it remains to show that the monomial occurs $B(H_{sub}, H, H')$ times in $Q(H_{sub}, H, G)$, i.e., that are $B(H_{sub}, H, H')$ l-tuples β such that $\prod_{v \in S} x_v$ is a monomial in $Q(\beta, H, G)$.

An occurrence of $\prod_{v \in S} x_v$ as a monomial in $Q(\beta, H, G)$ is in one-to-one correspondence with the class of all subgraph isomorphisms between H and the subgraph G' of G induced by S that map α_i on β_i for $i = 1, ..., l$. It follows from the definition of $B(H_{sub}, H, H')$ that the number of such l-tuples β is equal to $B(H_{sub}, H, H')$.

□

Proof of Theorem 3.10. **Proof:**

By Lemma 3.16, it is sufficient to show that $Q(H_{sub}, H, G)$ can be evaluated over the field in the claimed time.

The set L of all ordered l-tuples γ of different l vertices such that there is an isomorphism between H_{sub} and the subgraph of G induced by γ that maps α_i on γ_i for $i = 1, ..., l$, can be easily computed in $O(l^2 l! n^l) = O(n^l)$ time.

For all $\gamma \in L$, and all $b \in \{0, 1\}^l$, we can compute the sets $V(\gamma, b)$ of vertices $v \in V$ that have γ-neighbourhood of type b in $O(2^l l n^{l+1}) = O(n^{l+1})$ time in total.

By the definition of $Q(H_{sub}, H, G)$, it is sufficient to show that for an arbitrary $\gamma \in L$, the polynomial $Q(\gamma, H, G)$ can be evaluated in $O(n)$ time (recall that $k = O(1)$).

This in turn by $l = O(1)$ reduces to showing that for an arbitrary $b \in \{0, 1\}^l$, the polynomial $\prod_{i=1}^{l} x_{\gamma_i} \sum_{U \subseteq V(\gamma, b) \wedge |U| = \eta_{\alpha, H}(b)} \prod_{x_v \in U} x_v$, can be evaluated in $O(n)$ time.

For $i = 1, ..., n$, let X_i be the set of variables $x_1, ..., x_i$.

Next, for a positive integer q, let $C^q(X_i)$ denote the elementary symmetric polynomial of degree q, i.e., $\sum_{T \subseteq X_i \wedge |T|=q} \prod_{x_j \in T} x_j$. For convention, we let $C^0(X_i)$ to be 1 in the field. $C^q(X_n)$ can be evaluated for a given assignment of values over the field by the recurrence $C^q(X_{i+1}) = x_{i+1}C^{q-1}(X_i) + C^q(X_i)$.

For $q = O(1)$, we can evaluate all $C^q(X_i)$ using this recurrence by dynamic programming in lexicographic order of (q, i) in $O(n)$ time.

It follows from $\eta_{\alpha,H}(b) = O(1)$ that the polynomial $\prod_{i=1}^{l} x_{\gamma_i} \sum_{U \subseteq V(\gamma,b) \wedge |U|=\eta_{\alpha,H}(b)} \prod_{x_v \in U} x_v$, can be evaluated in $O(n)$ time. □

Applications to pattern graphs on at most six vertices.

Corollary 3.17 *The method of Theorem 3.13 can be used to detect* $K_2 + 3K_1$, $2K_2 + K_1$, $P_3 + 2K_1$, $K_3 + e + K_1$, $fork$, $cricket$, P_5, $(3,2) - lollipop$, $banner$, $W_5 - e$, $K_4 - e + K_1$, $dart$, $kite$, $houseX$, W_5, *and* $K_5 - e$ *as an induced subgraph, with one-sided error of probability polynomially small in* n, *in* $O(n^4)$ *time.*

With the exception of P_5 [35] and $K_5 - e$ [61], our upper bounds of $O(n^4)$ are new.

γ	γ-neighbourhood type 00	γ-neighbourhood type 10	$Q(\gamma, H, G)$
$(1,2)$	$4,7,8$	$5,6$	0
$(2,1)$	$4,7,8$	3	0
$(1,5)$	$3,7,8$	$2,6$	0
$(5,1)$	$3,7,8$	4	0
$(1,6)$	$3,4,7,8$	$2,5$	$x_1x_2...x_8$
$(6,1)$	$3,4,7,8$		0
$(2,3)$	$5,6,8$	1	0
$(3,2)$	$5,6,8$	$4,7$	0
$(3,4)$	$1,6$	$2,7$	0
$(4,3)$	$1,6$	$5,8$	0
$(4,5)$	$2,6,7$	$3,8$	0
$(5,4)$	$2,6,7$	1	0
$(3,7)$	$1,5,6,8$	$2,4$	$x_1x_2...x_8$
$(7,3)$	$1,5,6,8$		0
$(4,8)$	$1,2,6,7$	$3,5$	$x_1x_2...x_8$
$(8,4)$	$1,2,6,7$		0

Table 3.2: The monomials of the polynomial $Q(H_{sub}, H, G)$ for the graphs H_{sub}, H, G depicted in Fig. 2. The polynomial $P(H_{sub}, H, G)$ has only one monomial, $x_1x_2...x_8$, corresponding to $H' \in \mathcal{H}_k(H_{sub}, H)$ isomorphic to G. Since $B(H_{sub}, H, H') = 3$, $P(H_{sub}, H, G) = Q(H_{sub}, H, G)$ holds.

Example 1: We obtain the following pairs of graphs H, H' in H_5, sharing a common subgraph $H_{sub} \in \mathcal{H}_3$, such that $H \in \mathcal{H}_5(3)$, $H' \in \mathcal{H}_5(H_{sub}, H)$, and the number of automorphisms (in parentheses) for one of them has a prime factor that is not a prime factor of the other one.

- $5K_1(120)$, $K_2 + 3K_1(12)$
- $K_2 + 3K_1(12)$, $2K_2 + K_1(8)$
- $K_2 + 3K_1(12)$, $P_3 + 2K_1(4)$
- $K_{1,3} + K_1(6)$, $K_3 + e + K_1(2)$
- $K_{1,3} + K_1(6)$, $fork(2)$
- $K_{1,4}(24)$, $cricket(4)$
- $P_5(2)$, $C_5(10)$
- $K_3 + K_2(12)$, $(3,2) - lollipop(2)$
- $banner(2)$, $K_{2,3}(12)$
- $K_{2,3}(12)$, $W_5 - e(4)$
- $K_4 - e + K_1(4)$, $K_4 + K_1(24)$
- $dart(2)$, $(3,2) - fan(12)$
- $dart(2)$, $K_4 + e(6)$
- $kite(2)$, $K_4 + e(6)$
- $(3,2) - fan(12)$, $houseX(4)$
- $W_5(8)$, $K_5 - e(12)$
- $K_5 - e(12)$, $K_5(120)$

In all the cases where an upper bound better than that for K_4 is known except for the case of P_4 and $K_{1,3}$, our combinatorial method yields the upper bound of $O(n^3)$ that is also better than that known for K_4. Similarly, for pattern graphs on three vertices, our method yields the upper bound of $O(n^2)$ in all the cases for which an upper bound better than that for K_3 is known.

Example 2: We obtain the following polynomials $P(H_{sub}, H)$ (for brevity, we skip the parameter G), where H_{sub} is either $2K_1$ or K_2, and $H \in H_4(2)$

1) $P(2K_1, 4K_1) = 24PI(4K_1) + 2PI(K_2 + 2K_1)$

2) $P(K_2, K_2 + 2K_1) = 2PI(K_2 + 2K_1) + 4PI(2K_2)$

3) $P(2K_1, K_{1,2} + K_1) = 2PI(K_{1,2} + K_1) + 6PI(K_{1,3})$

4) $P(2K_1, 2K_2) = 8PI(2K_2) + 2PI(P_4)$

5) $P(K_2, K_3 + K_1) = 6PI(K_3 + K_1) + 2PI(K_3 + e)$

6) $P(K_2, P_4) = 2PI(P_4) + 8PI(C_4)$

7) $P(K_2, K_{1,3}) = 3PI(K_{1,3}) + PI(K_3 + e)$

8) $P(2K_1, C_4) = 4PI(C_4) + 2PI(K_4 - e)$

9) $P(K_2, K_3 + e) = 2PI(K_3 + e) + 4PI(K_4 - e)$

10) $P(K_2, K_4 - e) = 2PI(K_4 - e) + 12PI(K_4)$

In Example 2, several polynomials $P(H_{sub}, H, G)$ with explicit coefficients $B(H_{sub}, H, H')$ are listed for graphs H on four vertices. By applying Theorem 3.12 to these polynomials, we obtain the following corollary.

Corollary 3.18 *The method of Theorem 3.12 can be used to detect $K_3 + e$, $K_4 - e$ and their complements as an induced subgraph, with one-sided error of probability polynomially small in n, in $O(n^3)$ time.*

We obtain also the following corollary from Theorem 3.13 confirming in part the upper bounds from [35, 61].

Corollary 3.19 *The method of Theorem 3.13 can be used to detect P_k if k has a prime factor larger than 2, and $K_k - e$ if k or $k - 1$ is a prime number larger than 2, as an induced subgraph, with one-sided error of probability polynomially small in n, in $O(n^{k-1})$ time.*

Proof: P_k and C_k differ by a single edge. To use Theorem 3.13, set $H_{sub} = P_{k-2}$ with the endpoints of P_k outside H_{sub}. Let q be a prime factor of k larger than 2. The number of automorphisms of C_k is divisible by q while that of P_k is not divisible by q.

Similarly, $K_k - e$ and K_k differ by a single edge. To use Theorem 3.13, set $H_{sub} = K_{k-2}$. The number of automorphisms of K_k is divisible by k and $k - 1$ while that of $K_k - e$ is not divisible by k or $k - 1$ if k or $k - 1$ respectively is a prime number. □

Note that Corollary 3.19 yields in particular the upper bound of $O(n^2)$ on the detection of P_3, or equivalently $K_3 - e$, and the complement graph as induced subgraphs,

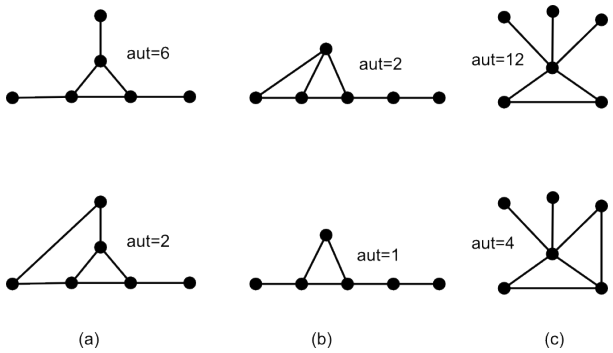

Figure 3.5: Pairs of pattern graphs on six vertices forming a family $\mathcal{H}_6(H_{sub}, H)$, where the common subgraph H_{sub} has three vertices and the corresponding numbers of automorphisms have different prime factors. Each of the graphs in the bottom row can be detected as an induced subgraph by our method in $O(n^5)$ time

confirming all the known sub-$O(n^\omega)$ time upper bounds on detection of pattern graphs on three vertices [22, 42, 61].

We can also apply the method of Theorem 3.13 to several pattern graphs on more than five vertices. Although we cannot improve the triangle based upper bounds [22] as for the asymptotic time complexity for so large pattern graphs, our method not relying on fast matrix multiplication may be still more practical. For example, see the three pairs of pattern graphs on six vertices in Fig. 3.2.3 It follows from Theorem 3.12 that each of the graphs in the bottom row can be detected as an induced subgraph by our method in $O(n^5)$ time.

3.2.4 Counting subgraph isomorphisms

Our first method for counting subgraph isomorphisms relies on Fact 1, see Prel.

Theorem 3.20 *The number of subgraph isomorphisms between a fixed graph with k vertices and with an independent set on s vertices and a host graph on n vertices can be computed in time $O(n^{\omega(\lceil (k-s)/2 \rceil, 1, \lfloor (k-s)/2 \rfloor)})$.*

Proof: sketch. Let H be the pattern graph on k vertices with an independent set I on s vertices. Next, let H_{sub} be the subgraph of H induced by all its $l = k - s$ vertices outside I.

For a vertex $v \in I$, let $b(v)$ denote the H_{sub}-neighbourhood type of v. For $w \in \{0,1\}^l$, let t_w be the number of vertices v in I for which $b(v) = w$.

Let $G = (V, E)$ be the host graph on n vertices. Consider an ordered l-tuple α of vertices in G such that the function f_α mapping the i-th vertex of H_{sub} onto the i-th

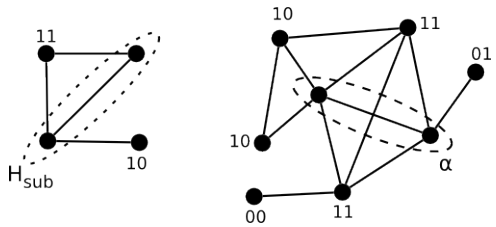

Figure 3.6: (left) An example of the pattern graph H and its subgraph H_{sub} within the dotted circle. The vertices of the independent set outside H_{sub} are labelled with their respective H_{sub}-neighbourhood type. (right) An example of the host graph G and an (encircled) ordered l-tuple α of its vertices. The vertices of G outside α are labelled with their α-neighbourhood types respectively.

vertex of α is a subgraph isomorphism between H_{sub} and G.

We shall count the number of different subgraph isomorphisms between H and G that are extensions of the subgraph isomorphism f_α between H_{sub} and G.

For this purpose, we denote the α-neighbourhood type of a vertex u in G by $b_\alpha(u)$. For $w \in \{0,1\}^l$, let $n_w(\alpha)$ be the number of vertices u in G outside α for which $b_\alpha(u) = w$. Also, for two vectors $c,\ d \in \{0,1\}^l$, we say that d *dominates* c if for $i = 1, ..., l$, $c_i \leq d_i$. For a vector $c \in \{0,1\}^l$, the vector directly following c in the lexicographic order (if any) is denoted by $suc(c)$.

Observe that by our definitions, the number of extensions of the subgraph isomorphism f_α between H_{sub} and G to a subgraph isomorphism between H and G is equal to the number of ways we can choose for each $w \in \{0,1\}^l$ an ordered tuple of t_w vertices u in G such that $b_\alpha(u)$ dominates w. It is sufficient to consider solely those w for which $t_w \neq 0$, i.e., there are vertices in $H \setminus H_{sub}$ whose H_{sub}-neighbourhood type is w.

The extensions can be counted by a straightforward recursive algorithm provided that the numbers t_w and $n_b(\alpha)$ are known. If H has $O(1)$ vertices then the algorithm runs in $O(1)$ time. Also, the numbers t_w can be computed in $O(1)$ time if H has $O(1)$ vertices.

Note that for two such distinct l-tuples α, the subgraph isomorphisms f_α between H_{sub} and G are different and thus their extensions to a subgraph isomorphism between H and G have to be different too. Thus, it is sufficient to sum the numbers of extensions over such l-tuples α.

We can list all such l-tuples α where the function f_α mapping the i-th vertex of H_{sub} on the i-th vertex of α is a subgraph isomorphism between H_{sub} and G in $O(n^l)$ time if $l = O(1)$. Finally, all the numbers $n_w(\alpha)$ can be obtained by solving the l-neighbourhood problem in time $O(2^l n^{\omega(\lceil l/2 \rceil, 1, \lfloor l/2 \rfloor)})$ by Fact 1. □

Alternatively, we could formulate the problem of counting the extensions as follows. We are given a bipartite graph with vertices in one-to-one correspondence with buckets.

In the first set of buckets (vertices), we have initially empty buckets $Hbuc_w$, $w \in \{0,\ 1\}^l$, of capacity t_w. In the other set, we have buckets $Gbuc(\alpha)_b$, $b \in \{0,1\}^l$, filled respectively with $n(\alpha)_b$ distinct balls. There is an edge between $Hbuc_w$ and $Gbuc(\alpha)_b$ iff b dominates w. The task is to count the number of ways in which we can fill simultaneously each $Hbuc_w$ with an ordered sequence of exactly t_w balls taken from possibly different $Gbuc(\alpha)_b$, where there is an edge between b and w.

The following algorithm, involving the recursive procedure $COUNT$, counts the extensions.

Algorithm 1
input for all $w \in \{0,1\}^l$, the numbers t_w and $n_w(\alpha)$.

1. **for** $w \in \{0,1\}^l$ **do**
   $n^*_w \leftarrow n_w(\alpha)$;
2. **return** $COUNT(0^l, \{n^*_q\}_{q\in\{0,1\}^l})$

The procedure $COUNT$ is defined as follows.

procedure $COUNT(w, \{n^*_q\}_{q\in\{0,1\}^l})$
input $w \in \{0,1\}^l$ and a sequence $\{n^*_q\}_{q\in\{0,1\}^l}$ of nonnegative integers.

1. **if** $t_w = 0$ **then if** $w \neq 1^l$ **then return** $COUNT(suc(w), \{n^*_q\}_{q\in\{0,1\}^l})$ **else return** 1;
2. $sum \leftarrow 0$;
3. **for** all decompositions of t_w into the sum of nonnegative integers n^w_q, where $q \in \{0,1\}^l$ dominates w and $n^w_q \leq n^*_q$ **do**
   (a) $ways \leftarrow (t_w)! \prod_{q\ dominates\ w} \binom{n^*_q}{n^w_q}$
   (b) **for** all $q \in \{0,1\}^l$ that dominate w **do**
   if $w \neq 1^l$ **then** $sum \leftarrow sum + ways \times COUNT(suc(w), \{n^*_q - n^w_q\}_{q\in\{0,1\}^l})$
   else $sum \leftarrow sum + ways$
4. **return** sum

Suppose that the edges and/or vertices of the host graph G have some real weights. For the problem of counting lightest subgraph isomorphisms between the pattern graph H and G, i.e., the isomorphisms between H and lightest subgraphs isomorphic to H, we obtain the following theorem.

Theorem 3.21 *Let H be a pattern graph on k vertices with an independent set on s vertices, and let G be a host graph on n vertices with vertex and/or edge real weights. If $k = O(1)$ then the number of lightest subgraph isomorphisms between H and G can be computed by a combinatorial algorithm in $O(n^{k-s+1} \log n)$ time.*

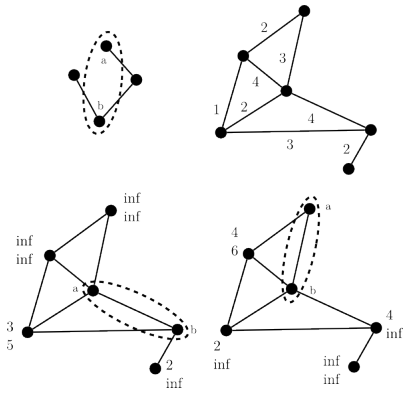

Figure 3.7: (up left) An example of the pattern graph H and its subgraph H_{sub} within the dotted circle. (up right) An example of an edge weighted host graph G. (down) Two examples of an ordered l-tuple within G marked with a dotted circle. The vertices of G outside the l-tuple are labelled with the total weight of the edges that connect them with the l tuple and are relevant if the first or the second vertex of H outside H_{sub}, respectively, is mapped on them under a subgraph isomorphism. If such a mapping is not possible, the corresponding label is "inf".

Proof:

Consider first the problem of finding a lightest subgraph of G isomorphic to H.

Let α be an ordered l-tuple of vertices in G such that the mapping f_α of the i-th vertex of H_{sub} on the i-th vertex of α for $i = 1, ..., l$, is a subgraph isomorphism between H_{sub} and G. For each $0-1$ vector b of length l and any vertex v of G outside α, whose α-neighbourhood type dominates b, define the combined weight $w_b(v)$ as follows. The weight $w_b(v)$ is the sum of the weight of v (if any) and the weights of the edges connecting v with the vertices in α whose numbers j in α correspond to 1's in b, i.e., satisfy $b_j = 1$. Let $L_b(\alpha)$ be a list of all such vertices v sorted by the weight $w_b(v)$ in non-decreasing order.

All the sorted lists $L_b(\alpha)$ and corresponding ranks of vertices can be easily computed by at most 2^l examinations and sorting of vertices in G outside α in $O(2^l ln \log n)$ time.

Now the following observation is crucial: *to find a lightest extension of the subgraph isomorphism f_α between H_{sub} and G to a subgraph isomorphism between H and G it is sufficient to consider solely vertices of G that have rank at most $s = k - l$ on any of the lists $L_b(\alpha)$*. The observation follows immediately from the fact that we need to define the extension for only the $s = k - l$ independent vertices in $H \setminus H_{sub}$. See Fig. 6.

Thus, the subgraph $G^*(\alpha)$ of G induced by α and all the aforementioned at most $2^l s$ vertices includes a lightest extension of the subgraph-isomorphism between H_{sub}

and α. Consequently, a lightest extension of the subgraph-isomorphism can be found easily by enumerating all possible mappings between the vertices in $H \setminus H_{sub}$ and those in $G^*(\alpha)$ outside α in time $O(s!\binom{2^l s}{s}) = O(1)$.

Finally, we can list all such α l-tuples, whose number is $O(n^l)$, in $O(l^2 n^l)$ time. Hence, we conclude that one can find a lightest subgraph isomorphism between H and G in $O(n^{l+1}]$ time when $k = O(1)$. □

We can easily extend our method of finding a lightest subgraph isomorphism between H and G to include counting lightest subgraph isomorphisms between H and G as follows.

First, we find all lightest subgraph isomorphisms f between H and G that are extensions of the mapping f_α of the i-th vertex of H_{sub} on the i-th vertex of an ordered l-tuple α, and that are constrained to the finite induced subgraph $G^*(\alpha)$. This can be done by an immediate adaptation of our method for finding a lightest subgraph isomorphisms between H and G in $O(n^{l+1})$ overall time.

The only case when we could miss some lightest extensions in the aforementioned enumeration is as follows.

Suppose that for some of the found lightest extensions f there is $b \in \{0,1\}^l$ such that f maps some vertices v of $H \setminus H_{sub}$ whose H_{sub}-neighbourhood type $b(v)$ equals b onto vertices $f(v)$ of G whose α-neighbourhood type dominates b and whose weight $w_b(f(v))$ is equal that of the vertex of rank s on $L_b(\alpha)$. Next, suppose that there are some other vertices u of higher ranks on $L_b(\alpha)$ such that $w_b(f(v)) = w_b(u)$. Then, we can obtain other lightest extensions by partially using the vertices u instead.

We can identify such cases. On the other hand, it would be to costly to include the aforementioned missing lightest extensions in the enumeration. Instead, we propose a method to just count the number of missing lightest extensions (if any).

We shall call any of the enumerated lightest extensions f *canonical* iff for all $b \in \{0,1\}^l$, f satisfies the following condition:
if f maps two vertices $v_1,\ v_2$ in $H \setminus H_{sub}$ whose H_{sub}-neighbourhood type is b on two vertices in G whose α-neighbourhood type dominates b, and $w_b(f(v_1))$ and $w_b(f(v_2))$ are equal to the corresponding weight of the vertex of rank s on $L_b(\alpha)$, then whenever v_1 has a smaller number than that of v_2 in H, $f(v_1)$ has a smaller rank than that of $f(v_2)$ in $L_b(\alpha)$.

Now for each found canonical lightest extension f, we count the number of lightest extensions that can be obtained as follows. We replace for each $b \in \{0,1\}^l$, some vertices v of $G^*(\alpha)$ that are images of vertices with the H_{sub}-neighbourhood type equal to b and that have rank at most $s = k - l$ on $L_b(\alpha)$, and whose weight $w_b(v)$ equals that of the vertex of rank s in $L_b(\alpha)$ by some other vertices u in $L_b(\alpha)$ whose rank is not necessarily smaller than s but still $w_b(u) = w_b(v)$.

While producing the lists $L_b(\alpha)$, for $b \in \{0,1\}^l$, we can precompute for all $S \subset \{0,1\}^l$, the number $n_S(\alpha)$ of all vertices u that occur on the lists $L_b(\alpha)$, for all $b \in S$,

and that have the weights $w_b()$ equal to that of the vertex of rank s on L_b for $b \in S$, and that do not have these properties for any $b \notin S$. The precomputation takes time proportional to that required by sorting under assumption that $l = O(1)$.

Next, suppose that for $b \in \{0,1\}^l$, there are r_b vertices of G that under the canonical extension f are images of vertices in $H \setminus H_{sub}$ whose H_{sub}-neighbourhood type is b and whose weight $w_b()$ is equal to that of the vertex of rank s on $L_b(\alpha)$.

Knowing the numbers $n_S(\alpha)$ and r_b, we can compute the number of the aforementioned possible replacements in the canonical lightest extension f by running the following algorithm.

Algorithm 2

1. **for** $S \subseteq \{0,1\}^l$ **do**
   $n^*_S \leftarrow n_S(\alpha)$;
2. **return** $REPLACE(0^l, \{n^*_S\}_{S \subseteq \{0,1\}^l})$

Algorithm 2 involves the procedure $REPLACE$ similar to the $COUNT$ procedure from Algorithm 1.

procedure $REPLACE(b, \{n^*_S\}_{S \subseteq \{0,1\}^l})$
input $b \in \{0,1\}^l$ and a sequence $\{n^*_S\}_{S \subseteq \{0,1\}^l}$ of nonnegative integers.

1. **if** $r_b(\alpha) = 0$ **then if** $b \neq 1^l$ **then return return** $REPLACE(b, \{n^*_S\}_{S \subseteq \{0,1\}^l})$ **else return** 1
2. $sum \leftarrow 0$;
3. **for** all decompositions of $r_b(\alpha)$ into the sum of nonnegative integers n^b_S, where $b \in S$ and $n^b_S \leq n^*_S$ **do**
   (a) $ways \leftarrow (r_b(\alpha))! \prod_{b \in S} \binom{n^*_S}{n^b_S}$
   (b) **for** all $S \subseteq \{0,1\}^l$ where $b \in S$ **do**
      i. **if** $b \neq 1^l$ **then** $sum \leftarrow sum + ways \times REPLACE(suc(b), \{n^*_S - n^b_S\}_{q \in \{0,1\}^l})$ **else** $sum \leftarrow sum + ways$
4. **return** sum

By taking the sum of the number of replacements over the lightest found canonical extensions, we obtain the total number of lightest subgraph isomorphisms.

Chapter 4

3D Rectangulations and Matrix Products

4.1 Introduction

This paper considers two intriguing and at a first glance unrelated problems.

The first problem lies at the heart of three-dimensional computational geometry. It belongs to the class of *polyhedron decomposition* problems, whose applications range from data compression and database systems to pattern recognition, image processing, and computer graphics [40, 57]. The problem is to partition a given rectilinear polyhedron into a minimum number of 3D rectangles. Dielissen and Kaldewai have shown this problem to be NP-hard [19]. In contrast, the problem of partitioning a rectilinear (planar) polygonal region into a minimum number of 2D rectangles admits a polynomial-time solution [40, 49]. Formally, the NP-hardness proof by [19] is for polyhedra with holes, but the authors remark that the proof should also work for simple polyhedra. To the best of our knowledge, no non-trivial approximation factors for minimum rectangular partition of simple rectilinear polyhedra are known, even in restricted non-trivial cases such as that of a 3D histogram (a natural generalization of a planar histogram, see Section 4.2).

The second problem we consider is that of multiplying two $n \times n$ matrices. There exist fast algorithms that do so in substantially subcubic time, e.g., a recent one due to Le Gall runs in $O(n^{2.3728639})$ time [46], but they suffer from very large overheads. On the positive side, input matrices in real world applications often belong to quite restricted matrix classes, so a natural approach is to design faster algorithms for such special cases. Indeed, efficient algorithms for *sparse* matrix multiplication have been known for long time. In the Boolean case, despite considerable efforts by the algorithms

community, the fastest known combinatorial algorithms for Boolean $n \times n$ matrix multiplication barely run in subcubic time (in $O(n^3(\log\log n)^2/(\log n)^{9/4)})$ time [5], to be precise), but much faster algorithms for Boolean matrix product for restricted classes of Boolean matrices have been developed [8, 31, 48]. For example, when at least one of the input Boolean matrices admits an exact covering of its ones by a relatively small number of rectangular submatrices, the Boolean matrix product can be computed efficiently [48]; similarly, if the rows of the first input Boolean matrix or the columns of the second input Boolean matrix can be represented by a relatively cheap minimum cost spanning tree in the Hamming metric (or its generalization to include blocks of zeros or ones) then the Boolean matrix product can be computed efficiently by a randomized combinatorial algorithm [8, 31].

Our first contribution is an $O(m \log m)$-time, 4-approximation algorithm for computing a minimum 3D rectangular partition of an input 3D histogram with m corners. It works by projecting the input histogram onto the base plane, partitioning the resulting planar straight-line graph into a number of 2D rectangles not exceeding its number of vertices, and transforming the resulting 2D rectangles into 3D rectangles of appropriate height. Importantly, the known algorithms for minimum partition of a rectilinear polygon with holes into 2D rectangles [40, 49] do not yield the aforementioned upper bound on the number of rectangles in the more general case of planar straight-line graphs.

Our second contribution is a new technique for multiplying two matrices with nonnegative integer entries. We interpret the matrices as 3D histograms and decompose them into blocks that can be efficiently manipulated in a pairwise manner using the interval tree data structure. Let A and B be two $n \times n$ matrices with nonnegative integer entries, and let r_A and r_B denote the minimum number of 3D rectangles into which the 3D histograms induced by A and B can be partitioned.

By applying our 4-approximation algorithm above, we can compute

$A \times B$ in $\tilde{O}(n^2 + r_A r_B)$ time, where $\tilde{O}$ suppresses polylogarithmic (in n) factors. Furthermore, by using another idea of slicing the histogram of A (or B) into parts corresponding to rows of A (or columns of B) and measuring the cost of transforming a slice into a consecutive one, we obtain an upper bound of $\tilde{O}(n^2 + n \min\{r_A,\ r_B\})$.

We also give a generalization of the latter upper bound in terms of the minimum cost of a spanning tree of the slices, where the distance between a pair of slices corresponds to the cost of transforming one slice into the other.

Organization: Section 4.2 presents our 4-approximation algorithm for a partition of a 3D histogram into a minimum number of 3D rectangles. Section 4.3 presents our algorithms for the arithmetic matrix product. Section 4.4 concludes with some final remarks.

4.2 3D Histograms and Their Rectangular Partitions

A 2D histogram is a polygon with an edge e, which we call the *base* of the histogram, having the following property: for every point p in the interior of histogram, there is a (unique) line segment perpendicular to e, connecting p to e and lying totally in the interior of the histogram. In this paper, we consider orthogonal histograms only. For simplicity, we consider the base of a histogram as being horizontal, and all other edges of the histogram lying above the base. In this way, a 2D histogram can also be thought of as the union of rectangles standing on the base of the histogram.

A *3D histogram* is a natural generalization of a 2D histogram. To define a 3D histogram, we need the concept of the "base plane", which for simplicity we define as the horizontal plane containing two of the axes in the Euclidean space. A 3D histogram can then be thought of as the union of rectilinear 3D rectangles, standing on the base plane. The *base of the histogram* is the union of the lower faces (also called *bases*) of all these rectangles.

Definition 4.1 A 3D histogram is a union of a finite set C of rectilinear 3D rectangles such that: (i) each element in C has a face on the horizontal base plane; and (ii) all elements in C are located above the base plane.

(In the literature, what we call a 3D histogram is sometimes termed a 2D histogram or a 1D histogram when used to summarize 2D or 1D data, respectively [52].)

By a *rectangular partition* of 3D histogram P, we mean a rectilinear partition of P into 3D rectangles.

In Section 4.2.2 below, we consider the problem of finding a rectangular partition of a given 3D histogram P into as few 3D rectangles as possible. We present a 4-approximation algorithm for this problem with time complexity $O(m \log m)$, where m denotes the number of vertices in P. The algorithm partitions P into less than m' 3D rectangles, where m' is the number of vertices in the vertical projection of P (i.e., $m' < m$), by applying a subroutine described in Section 4.2.1 that partitions any rectilinear planar straight-line graph (PSLG) with m' vertices into less than m' 2D rectangles. Finally, the approximation factor is derived by observing that any rectangular partition of P must contain at least $m'/4$ 3D rectangles.

4.2.1 Partitioning a Rectilinear PSLG into 2D Rectangles

The problem of partitioning a rectilinear polygon into rectangles in two dimensions has been well studied in the literature [40, 49]. An optimal solution for this problem can be computed in polynomial time [40, 49]. However, to use the result in 3D, we need a bound on the number of produced rectangles, expressed in terms of the number of vertices. Therefore, it is not so crucial for our purposes to compute an optimal solution

for the 2-dimensional problem, but instead, we need to partition planar straight-line graphs (PSLGs) into at most m' rectangles, where m' denotes the number of vertices in the input PSLG. We will show that a simple algorithm suffices to obtain this bound.

Since this subsection considers 2D only, we use the term "horizontal" for line segments parallel to the X-axis. By "vertical" lines, we mean lines or line segments parallel to the Y-axis. Each vertex in the planar graphs in our application has degree 2, 3, or 4.

Definition 4.2 A planar straight-line graph (PSLG) $PG = (V, E)$, as used in this paper, is a planar graph where every vertex has an x- and a y-coordinate. Each edge is drawn as a straight line segment, all edges meet at right angles, and each vertex has degree 2, 3, or 4. A *rectangular partition* of PG is a partition $R = (V \cup V_R, E \cup E_R)$ that adds edges and vertices to PG so that R is still a PSLG while every face in R is a rectangle.

Given a PSLG PG, we denote $m' = |V|$. We say that a vertex v of PG is *concave* if it has degree 2, its two adjacent edges are perpendicular to each other, and the corner at v which is of 270 degrees does not lie in the outer, infinite face of PG. Any vertex which is not concave is called *convex*.

We use a sweep line approach to generate a partition into less than m' rectangles. We perform a horizontal sweep with a vertical sweep line [6], using the vertices of PG as event points. Whenever the sweep line reaches a concave vertex v, we insert into the graph PG a vertical line segment s connecting v to the closest edge of PSLG upwards or downwards, thus cancelling the concavity at v and transforming v into a convex vertex of degree 3. Hence, if there was already an edge of PG below v, then the new segment s is inserted above v, otherwise it is inserted below v. To preserve the property that the resulting graph is still a PSLG, the other endpoint of s may have to become a new vertex of the PSLG. This is a standard procedure for trapezoidation; see, e.g., [6] for more details. After the sweep is complete, all concave vertices have been eliminated. (Remark: In a special case it may happen that two concave vertices with the same x-coordinate are connected by a single vertical segment that is disjoint from the rest of the input PSLG. In this case, the plane sweep algorithm will produce this segment. Thus, no two segments produced by the algorithm overlap or touch each other.)

The correctness of the algorithm is easy to see: it eliminates all concave corners of PG by adding vertical line segments. Hence, in the resulting PSLG, each face, except for the outer face, is a rectangle. The running time of this algorithm is dominated by the cost of the plane sweep, which is $O(m' \log m')$ according to well-known methods in computational geometry; see, e.g., [6].

We need to relate the number of vertices in the input PSLG to the number of 2D rectangles. This is done in the following lemma:

Lemma 4.3 *Any PSLG $PG = (V, E)$ with $|V| = m'$ and minimum vertex degree 2 can be partitioned into b rectangles with $b < m'$ using $O(m' \log m')$ time.*

Proof: Let R denote the set of rectangles in the rectangular partition produced by the plane sweep algorithm described above. We use a "charging scheme" to prove the stated inequality. The charging scheme starts by giving each vertex $v \in V$ four tokens; thus, a total of $4m'$ tokens are used. Each vertex v then distributes its tokens in a certain way to the rectangles in R that are adjacent to v. We will show that every rectangle in R receives at least four tokens. Since we started by giving a total of $4m'$ tokens to the vertices, this will prove that there exist at most m' rectangles, and thus $b \leq m'$. Moreover, vertices adjacent to the outer face do not give away more than three tokens. We will thus obtain the strict inequality $b < m'$.

Now, we describe the details of the charging scheme. (More explanations and illustrating figures are included in the full version.) Let v be any vertex of V. The vertex v gives one token to each rectangle r in R which in any way is adjacent to it, with one exception. The exception occurs when v is a concave vertex; then, v is partitioned by a vertical segment e_r added by the algorithm. This segment partitions the three quadrants at the concave corner around the vertex so that one rectangle occupies one quadrant and one occupies the two others. Then v distributes two tokens to the new rectangle occupying only one quadrant, which therefore has a corner at v, and only one token to each one of the other rectangles of R adjacent to v.

We now show that each rectangle receives at least four tokens. Let r be any rectangle in R. First note that each vertical segment added by the algorithm has at least one endpoint at a vertex in V. Moreover, for any rectangle r in R, each of the vertical sides of r includes at least one vertex of V. Therefore, each rectangle is adjacent to at least two vertices of V. We distinguish three cases, depending on the number of vertices of V adjacent to r. Observe that the adjacencies are not necessarily at the corners of r.

- Case 1: r is adjacent to at least four vertices of V. Since r will receive at least one token from each of them we are done.

- Case 2: r is adjacent to precisely three vertices of V. Then at one of the vertical sides of r there is only one vertex of V. Moreover, this vertex v must be at a corner of r and fulfils the criteria for giving two tokens to r. The remaining two adjacent vertices of V give at least one token each, so we are done.

- Case 3: r is adjacent to precisely two vertices of V. This must mean that both vertical sides of r are segments added by the algorithm, and that one of the endpoints of each of these sides is a vertex of V at a corner of r. This corresponds to the condition for receiving two tokens mentioned earlier. So in total, r receives four tokens from the two corners, and we are done.

□

4.2.2 Partitioning a 3D Histogram into 3D Rectangles

We now explain how to obtain the projected PSLG from the 3D histogram P and how to use the rectangular partition of this PSLG to yield a good partition into 3D rectangles.

Definition 4.4 The planar projection PP is an orthogonal projection of the input 3D histogram P along the "down" direction onto the base plane in Definition 4.1.

We can interpret PP as a PSLG where each corner and each subdividing point on a line segment corresponds to a vertex. The edges naturally correlate to the connecting line segments between vertices. Each vertex in PP is the vertical projection of at least two vertices of P. Two edges of the 3D histogram may partially overlap in the 2D projection, but the edges in the 2D projection are considered as non-overlapping. Thus, an edge of the 3D histogram may split into several edges in the 2D projection, since vertices should only appear as endpoints of edges.

Remark Every vertex in PP must have at least two neighbours. This follows from the fact that each vertex of P (and of any orthogonal polyhedron) has at least two incident horizontal edges. It may happen that some vertex of PP is the vertical projection of up to four vertices of P, so those four vertices of P may have a total of eight neighbours in P. But since PP is an orthogonal PSLG, no vertex of PP has more than four neighbours.

Now we are ready to show the main theorem of this section.

Theorem 4.5 *For any 3D histogram P with m corners, a 4-approximation R of a partition of P into as few 3D rectangles as possible can be computed in $O(m \log m)$ time.*

Proof: We use the projection in Definition 4.4, let $PG = PP$, and apply Lemma 4.3 to compute a planar partition R'. The final 3D partition R is obtained from R' by reversing the projection so that each 2D rectangle corresponds to the top of a 3D rectangle in R.

To analyse the approximation factor, denote the number of 3D rectangles in an optimal solution R^* by OPT and the number of 3D rectangles produced by the algorithm described above by b. We denote by m' the number of vertices in PP. By Lemma 4.3, we have $b < m'$ since each 2D rectangle corresponds to one 3D rectangle. Every vertex of P must be adjacent to at least one vertical edge of a 3D rectangle in R^*. Hence, each vertex in PP has to be at a corner of the vertical projection of at least one 3D rectangle in R^* onto the base plane. Since each 3D rectangle in R^* only has 4 vertical edges, its vertical projection can be adjacent to at most 4 vertices of PP. It follows that $m' \leq 4OPT$ and $b < m' \leq 4OPT$.

Since the projection can be obtained by contracting each corner in P and all of its vertical neighbours into one vertex, the projection can be implemented in $O(m)$ time. Thus, the $O(m \log m)$-term from Lemma 4.3 will dominate the time complexity. □

4.3 Geometric Algorithms for Arithmetic Matrix Product

4.3.1 Geometric Data Structures and Notation

Our algorithms for arithmetic matrix multiplication use some data structures for interval and rectangle intersection. An *interval tree* is a leaf-oriented binary search tree that supports intersection queries for a set Q of closed intervals on the real line as follows:

Fact 1 [51]. *Suppose that the left endpoints of the intervals in a set Q belong to a subset $\mathcal{U}$ of real numbers of size l and $|Q| = q$. An interval tree T of depth $O(\log l)$ for Q can be constructed in $O(l + q \log lq)$ time using $O(l + q)$ space. The insertion or deletion of an interval with left endpoint in $\mathcal{U}$ into T takes $O(\log l + \log q)$ time. The intersection query is supported by T in $O(\log l + r)$ time, where r is the number of reported intervals.*

Remark The interval tree of Fact 1 ([51]) can easily be generalized to the weighted case, where with an interval to insert or delete an integer weight is associated. It can be done by maintaining in each node of the interval tree the sum of weights of intervals whose fragments it represents. In effect, the generalized interval insertions or deletions as well the intersection query have the same time complexity as those in Fact 1. Moreover, the generalized interval tree supports a weight intersection query asking for the total weight of the intervals containing the query point in $O(\log l + \log q)$ time.

We use the following data structure, easily obtained by computing all prefix sums:

Fact 2. *For a sequence of integers a_1, a_2,...,a_n, one can construct a data structure that supports a query asking for reporting the sum $\sum_{k=i}^{j} a_k$ for $1 \leq i \leq j \leq n$ in $O(1)$ time. The construction takes $O(n)$ time.*

In the rest of the paper, A and B denote two $n \times n$ matrices with nonnegative integer entries, and C stands for their matrix product. We also need the following concepts:

1. For an $n \times n$ matrix D with nonnegative integer entries, consider the $[0, n] \times [0, n]$ integer grid whose unit cells are in one-to-one correspondence with the entries of$D$. The grid cell between the horizontal lines $i - 1$ and i (counting from the top) and vertical lines $j - 1$ and j (counting from the left) corresponds to $D_{i,j}$ (see Fig. 4.1). Then, $his(D)$ stands for the 3D histogram whose base consists of all unit cells of the $[0, n] \times [0, n]$ integer grid corresponding to positive entries of D and whose height over the cell corresponding to $D_{i,j}$ is the value of $D_{i,j}$ (see Fig. 4.2).

2. For the $n \times n$ matrix D, nonnegative integers $1 \leq i_1 \leq i_2 \leq n$, $1 \leq k_1 \leq k_2 \leq n$, and h_1, h_2, where $h_1 < h_2 \leq D_{i,j}$ for $i_1 \leq i \leq i_2$ and $j_1 \leq j \leq j_2$,

Figure 4.1:

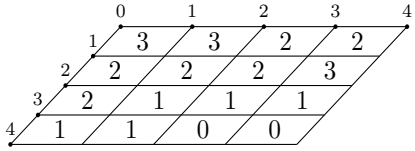

Figure 4.2:

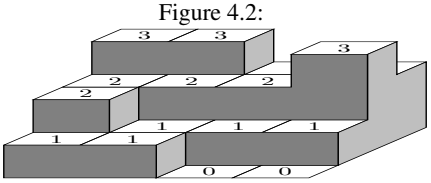

Figure 4.3: (a) A matrix D on a grid, and (b) its corresponding histogram $his(D)$.

$rec_D(i_1, i_2, k_1, k_2, h_1, h_2)$ is the 3D rectangle with the corners (i_1-1, k_1-1, h_l), (i_1-1, k_2, h_l), (i_2, k_1-1, h_l), (i_2, k_2, h_l), where $l = 1, 2$, lying within $his(D)$.

3. For the matrix D, r_D is the minimum number of 3D rectangles $rec_D(i_1, i_2, k_1, k_2, h_1, h_2)$ which form a partition of $his(D)$. Note that $r_D \leq n^2$.

4.3.2 Algorithms

Our first geometric algorithm for nonnegative integer matrix multiplication relies on the following key lemma.

Lemma 4.6 *Let P_A be a partition of the matrix A into 3D rectangles $rec_A(i_1, i_2, k_1, k_2, h_1, h_2)$, and let P_B be a partition of the matrix B into 3D rectangles $rec_B(k'_1, k'_2, j_1, j_2, h'_1, h'_2)$. For any $1 \leq i \leq n$, $1 \leq j \leq n$, the entry $C_{i,j}$ of the matrix product C of A and B is equal to the sum of $(h_2 - h_1)(h'_2 - h'_1) \times \#[k_1, k_2] \cap [k'_1, k'_2]$. over rectangle pairs $rec_A(i_1, i_2, k_1, k_2, h_1, h_2) \in P_A$, $rec_B(k'_1, k'_2, j_1, j_2, h'_1, h'_2) \in P_B$ satisfying $i \in [i_1, i_2]$ and $j \in [j_1, j_2]$.*

Proof: For $1 \leq l_1 < l_2 \leq n$ and $1 \leq m_1 < m_2 \leq n$, let $I(l_1, l_2, m_1, m_2)$ be the $n \times n$ $0-1$ matrix where $I(l_1, l_2, m_1, m_2)_{i,k} = 1$ iff $l_1 \leq i \leq l_2$ and $m_1 \leq k \leq m_2$.

Clearly, we have $A = \sum_{rec_A(i_1,i_2,k_1,k_2,h_1,h_2) \in P_A} (h_2 - h_1) I(i_1, i_2, k_1, k_2)$. Similarly, we have $B = \sum_{rec_B(k'_1,k'_2,j_1,j_2,h'_1,h'_2) \in P_B} (h'_2 - h'_1) I(k'_1,' k_2, j_1, j_2)$.

It follows that $C = A \times B$ is the sum over pairs $rec_A(i_1, i_2, k_1, k_2, h_1, h_2) \in P_A$, $rec_B(k'_1, k'_2, j_1, j_2, h'_1, h'_2) \in P_B$ of $(h_2-h_1)(h'_1-h'_2) I(i_1, i_2, k_1, k_2) \times I(k'_1, k'_2, j_1, j_2)$. It remains to observe that $(I(i_1, i_2, k_1+1, k_2) \times I(k'_1, k'_2, j_1+1, j_2))_{i,j} = \#[k_1, k_2] \cap [k'_1, k'_2]$ if $i_1 < i \leq i_2$ and $j_1 < j \leq j_2$ and it is equal to zero otherwise. □

Algorithm 1
Input: Two $n \times n$ matrices $A,\ B$ with nonnegative integer entries.
Output: The arithmetic matrix product C of A and B.

1. Find a partition P_A of $his(A)$ into 3D rectangles $rec_A(i_1, i_2, k_1, k_2, h_1, h_2)$ whose number is within $O(1)$ of the minimum.

2. Find a partition P_B of $his(B)$ into 3D rectangles $rec_B(k'_1, k'_2, j_1, j_2, h'_1, h'_2)$ whose number is within $O(1)$ of the minimum.

3. Initialize an interval tree S on the k-coordinates of the rectangles in P_A and P_B. For each 3D rectangle $rec_A(i_1, i_2, k_1, k_2, h_1, h_2) \in P_A$ insert $[k_1, k_2]$, with a pointer to $A(i_1, i_2, k_1, k_2, h_1, h_2)$, into S.

4. Initialize interval lists $Start_j$, End_j, for $j = 1, \ldots, n$. For each rectangle $rec_B(k'_1, k'_2, j_1, j_2, h'_1, h'_2) \in P_B$ report all intervals $[k_1, k_2]$ in S that intersect $[k'_1, k'_2]$. For each such interval $[k_1, k_2]$, with pointer to $rec_A(i_1, i_2, k_1, k_2, h_1, h_2)$, insert the interval $[i_1, i_2]$ with the weight $(h_2 - h_1) \times (h'_2 - h'_1) \times \#[k_1, k_2] \cap [k'_1, k'_2]$ into the lists $Start_{j_1}$ and End_{j_2}.

5. Initialize a weighted interval tree U on endpoints $1, \ldots, n$. For $j = 1, \ldots, n$, iterate the following steps. For $j > 1$, remove all weighted intervals $[i_1, i_2]$ on the list End_{j-1} from U. Insert all weighted intervals $[i_1, i_2]$ on the list $Start_j$ into U. For $i = 1, \ldots, n$, set $C_{i,j}$ to the value returned by U in response to the weight query at i.

Lemma 4.7 *Let $int(P_A, P_B)$ stand for the number of pairs $rec_A(i_1, i_2, k_1, k_2, h_1, h_2) \in P_A$, $rec_B(k'_1, k'_2, j_1, j_2, h'_1, h'_2) \in P_B$, for which $[k_1, k_2] \cap [k'_1, k'_2] \neq \emptyset$. Algorithm 1 runs in time $\tilde{O}(n^2 + int(P_A, P_B)) = \tilde{O}(n^2 + r_A r_B)$.*

Proof: To implement steps 1 and 2 in $\tilde{O}(n^2)$ time, use the algorithm from the preceding section (Theorem 4.5). Step 3 can be implemented in $\tilde{O}(n + r_A + r_B) = O(n^2)$ time by Fact 1. In Step 4, the queries to S take $\tilde{O}(int(P_A, P_B))$ time by Fact 1.

In Step 5, the initialization of the data structure U takes $\tilde{O}(n)$ time by Lemma 4.6. Next, the updates of the data structure U take $\tilde{O}(int(P_A, P_B))$ time by Lemma 4.6, while computing all columns of C takes $\tilde{O}(n^2)$ time by Remark 4.3.1. □

Theorem 4.8 *The matrix product of two $n \times n$ matrices $A,\ B$ with nonnegative integer entries can be computed in $\tilde{O}(n^2 + r_A r_B)$.*

Proof: Algorithm 1 yields the theorem. Its correctness follows from Lemma 4.6 that basically says that for each pair of 3D rectangles, $rec_A(i_1, i_2, k_1, k_2, h_1, h_2) \in P_A$ and $rec_B(k'_1, k'_2, j_1, j_2, h'_1, h'_2) \in P_B$, $C_{i,j}$ should be increased by $(h_2 - h_1) \times (h'_2 - h'_1) \times$

$\#[k_1, k_2] \cap [k'_1, k'_2]$ for $i \in [i_1, i_2]$ and $j \in [j_1, j_2]$. In Step 4, two identical intervals $[i_1, i_2]$ corresponding to the left and right edge of the submatrix of C whose entries should be increased by the aforementioned value are inserted in the lists $Start_{j_1}$ and End_{j_2}, respectively. In both cases, they are weighted by the aforementioned value. In Step 5, in iteration j_1, the weighted interval $[i_1, i_2]$ from $Start_{j_1}$ is inserted into the weighted interval tree U, and in iteration $(j_2 + 1)$, it is removed from U as its copy is in End_{j_2}. In the iterations $j = j_1, \ldots, j_2$ in Step 5, when the interval $[i_1, i_2]$ is kept in the weighted interval tree, U and the entries of the submatrix $C_{i,j}$, $i_1 \leq i \leq i_2$, $j_1 \leq j \leq j_2$, are evaluated, the weight of the interval contributes to their value. The upper time bound follows from Lemma 4.7. □

When only one of the matrices A and B admits a partition of its 3D histogram into relatively few 3D rectangles and we have to assume the trivial partition of the other one into n^2 3D rectangles, the upper bound of Theorem 4.8 in terms of r_A, r_B and n seems too weak. In this case, an upper bound in terms of $int(P_A, P_B)$ and n in Lemma 4.7 may be much better. To derive a better upper bound in terms of just $\min\{r_A, r_B\}$ and n, we shall design another algorithm based on the slicing of the 3D histogram admitting a partition into relatively few 3D rectangles.

For an $n \times n$ matrix D with nonnegative integer entries and $i = 1, \ldots, n$, let $slice_i(D)$ stand for the part of $his(D)$ between the two planes perpendicular to the Y axis whose intersection with the XY plane are the horizontal lines $i - 1$ and i on the $[0, n] \times [0, n]$ grid. In other words, $slice_i(D)$ is a 3D histogram for the i-th row. Also note that a $slice_i(D)$ can be identified with a rectilinear 2D histogram; see Fig. 4.6 for an example. We define a *geometric distance* between two rectilinear 2D histograms H_1 and H_2 with a common base as the number of maximal vertical strips s such that:

1. for $i = 1, 2$, s contains exactly one maximal subsegment e_i of an edge of H_i different from and parallel to the base of the histograms, and
2. the subsegments e_1 and e_2 do not overlap.

See Fig. 4.6. We shall call such strips *differentiating strips*. For $slice_i(D)$ and $slice_k(D)$, we define the geometric distance $gd(slice_i(D), slice_k(D))$ as that for the corresponding rectilinear 2D histograms.

Lemma 4.9 *For an $n \times n$ matrix D with nonnegative integer entries, $\sum_{i=1}^{n-1} gd(slice_i(D), slice_{i+1}(D)) = O(r_D)$ holds.*

Proof: Each differentiating strip contributes, possibly jointly with one or two neighbouring differentiating strips, to two vertices in the projected planar graph considered in the proof of Theorem 4.5. Thus, it contributes to the parameter m' in the aforementioned proof with at least 1. It follows $\sum_{i=1}^{n-1} gd(slice_i(D), slice_{i+1}(D)) \leq m'$. Hence, the inequality $m' \leq 4OPT$ established in the proof of Theorem 4.5 yields the thesis. □

Figure 4.4:

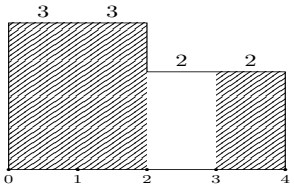

Figure 4.5:

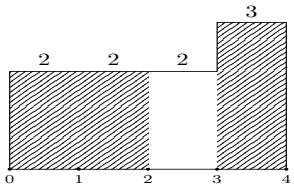

Figure 4.6: Let $slice_1(D)$ be the 2D histogram on the left and $slice_2(D)$ the 2D histogram on the right. Differentiating strips are shaded. Here, $gd(slice_1(D), slice_2(D)) = 2$.

Algorithm 2
Input: Two $n \times n$ matrices A and B with nonnegative integer entries.
Output: The matrix product C of A and B.

1. For $i = 1, \ldots, n-1$, find the differentiating strips for $slice_i(A)$ and $slice_{i+1}(A)$ and for each such strip s the indices $k_1(s)$ and $k_2(s)$ of the interval of entries $A_{i,k_1(s)}, \ldots, A_{i,k_2(s)}$ in the i-th row of A corresponding to it, as well as the difference $h(s)$ between the common value of each entry in $A_{i,k_1(s)}, \ldots, A_{i,k_2(s)}$ and the common value of each entry in $A_{i+1,k_1(s)}, \ldots, A_{i+1,k_2(s)}$.

2. For $j = 1, \ldots, n$, iterate the following steps:

   (a) Initialize a data structure T_j for counting partial sums of continuous fragments of the j-th column of the matrix B.

   (b) Compute $C_{1,j}$.

   (c) For $i = 1, \ldots, n-1$, iterate the following steps:

      i. Set $C_{i+1,j}$ to $C_{i,j}$.

      ii. For each differentiating strip s for $slice_i(A)$ and $slice_{i+1}(A)$, compute $\sum_{k=k_1(s)}^{k_2(s)} B_{k,j}$ using T_j and set $C_{i+1,j}$ to $C_{i+1,j} + h(s) \sum_{k=k_1(s)}^{k_2(s)} B_{k,j}$.

Lemma 4.10 *Algorithm 2 runs in $\tilde{O}(n(n + r_A))$ time.*

Proof: Step 1 can be easily implemented in $O(n^2)$ time. Step 2 (a) takes $\tilde{O}(n)$ time according to Fact 2 while Step 2 (b) can be trivially implemented in $O(n)$ time. Finally, based on Step 1, Step 2 (c) (ii) takes $\tilde{O}(gd(slice_i(D), slice_{i+1}(D))$ time. It follows that Step 2 (c) can be implemented in $\tilde{O}(\sum_{i=1}^{n-1} gd(slice_i(A), slice_{i+1}(A)))$ time, i.e., in $\tilde{O}(r_A)$ time by Lemma 4.9. Consequently, Step 2 takes $\tilde{O}(n(n + r_A))$ time. □

Theorem 4.11 *The arithmetic matrix product of two $n \times n$ matrices $A,\ B$ with non-negative integer entries can be computed in $\tilde{O}(n(n + \min\{r_A,\ r_B\}))$ time.*

Proof: The correctness of Algorithm 2 follows from the observation that a differentiating strip s for $slice_i(A)$ and $slice_{i+1}(A)$ yields the difference $h(s)\sum_{k=k_1(s)}^{k_2(s)} B_{k,j}$ between $C_{i+1,j}$ and $C_{i,j}$ just on the fragment corresponding to $A_{i,k_1(s)}, \ldots, A_{i,k_2(s)}$ and $A_{i+1,k_1(s)}, \ldots, A_{i+1,k_2(s)}$, respectively. Lemma 4.10 yields the upper bound $\tilde{O}(n(n + r_A))$. The symmetric one $\tilde{O}(n(n + r_B))$ follows from the equalities $AB = (B^T A^T)^T$, $his(B) \equiv his(B^T)$, and consequently $r_B = r_{B^T}$. □

In Algorithm 2, the linear order in which the $C_{i,j}$ are updated to $C_{i+1,j}$ for $i = 1, \ldots, n-1$, along the row order of the matrix A is not necessarily optimal. Following the Boolean case [8, 31], it may be more efficient to update $C_{i,j}$ while traversing a minimum spanning tree for the slices of $his(A)$ under the geometric distance. Here, however, we encounter the difficulty of constructing such an optimal spanning tree or a close approximation in substantially subcubic time. The next lemma will be useful.

Lemma 4.12 *Consider the family of rectilinear planar histograms with the base $[0, n]$, $n \geq 2$ and integer coordinates of its vertices in $[0, 2^M - 2]$, $M = O(\log n)$. There is a simple $O(n)$-time transformation of any histogram H in the family into an $0-1$ string $t(H)$, such that for any H_1 and H_2 in the family $gd(H_1, H_2) \leq ch(t(H_1), t(H_2)) \leq Mgd(H_1, H_2)$, where $ch(\ ,\ )$ stands for the Hamming distance.*

Proof: Any histogram H in the family is uniquely represented by the vector $(H[1], \ldots, H[n]) \in \{1, \ldots, 2^M - 1\}^n$, where $H[1], \ldots, H[n]$ are the values of Y coordinates of the points on the "roof" of H increased by one with X coordinates $0.5, 1.5, \ldots, n - 0.5$ respectively.

For any $y \in \{0, \ldots, 2^M - 1\}$ denote its binary representation of length exactly M (padded with leading zeros if necessary) as $\mathrm{bin}(y)$.

$$\text{Let } f(H, i) = \begin{cases} \mathrm{bin}(H[i]), & i = 1 \vee i > 1 \wedge H[i] \neq H[i-1] \\ \mathrm{bin}(0), & \text{otherwise.} \end{cases}$$

The transformation t is then defined as $t(H) = f(H,1)\dots f(H,n)$. We have $ch(t(H_1),t(H_2)) = \sum_{i=1}^{n} ch(f(H_1,i),f(H_2,i))$ and

$$gd(H_1,H_2) = \begin{cases} 1, & H_1[1] \neq H_2[1] \\ 0, & \text{otherwise} \end{cases} + \\ + \sum_{i=2}^{n} \begin{cases} 1, & (H_1[i] \neq H_1[i-1] \vee H_2[i] \neq H_2[i-1]) \wedge (H_1[i] \neq H_2[i]) \\ 0, & \text{otherwise.} \end{cases}$$

Consider all possibilities that contribute exactly one to $gd(H_1,H_2)$:

1. $H_1[1] \neq H_2[1]$. In this case $f(H_1,1) = \text{bin}(H_1[1])$, $f(H_2,1) = \text{bin}(H_2[1])$ and $0 \leq ch(\text{bin}(H_1[1]),\text{bin}(H_2[1])) \leq M$.

2. $2 \leq i \leq n \wedge H_1[i] \neq H_1[i-1] \wedge H_2[i] = H_2[i-1] \wedge H_1[i] \neq H_2[i]$. In this case $f(H_1,i) = \text{bin}(H_1[i])$, $f(H_2,i) = \text{bin}(0)$ and $1 \leq ch(\text{bin}(H_1[i]),\text{bin}(0)) \leq M$.

3. $2 \leq i \leq n \wedge H_1[i] = H_1[i-1] \wedge H_2[i] \neq H_2[i-1] \wedge H_1[i] \neq H_2[i]$. See case 2.

4. $2 \leq i \leq n \wedge H_1[i] \neq H_1[i-1] \wedge H_2[i] \neq H_2[i-1] \wedge H_1[i] \neq H_2[i]$. See case 1.

To complete the proof, observe that in all other cases $ch(f(H_1,i),f(H_2,i)) = 0$. □

Fact 3 [37]. *For $\epsilon > 0$, a $(1+\epsilon)$-approximation minimum spanning tree for a set of n points in R^d with integer coordinates in $O(1)$ under the L_1 or L_2 metric can be computed by a Monte Carlo algorithm in $O(dn^{1+1/(1+\epsilon)})$ time.*

By combining the transformation of Lemma 4.12 with Fact 3 applied to the L_1 metric in $\{0,1\}^n$ and selecting $\epsilon = \log n$, we obtain a Monte Carlo $O(\log^2 n)$-approximation algorithm for the minimum spanning tree of the slices of $his(A)$ under the geometric distance, which runs in $\tilde{O}(n^2)$ time. This yields a generalization of Algorithm 2 to Algorithm 3, described in the full version of our paper. By an analysis of Algorithm 3 analogous to that of Algorithm 2 and a proof analogous to that of Theorem 4.11, we obtain a randomized generalization of Theorem 4.11:

Theorem 4.13 *Let A, B be two $n \times n$ matrices A, B with nonnegative integer entries in $[0, n^{O(1)}]$. Next, for $D \in \{A, B^T\}$, let M_D be the minimum cost of a spanning tree of $slice_i(D)$ for $i = 1,\dots,n$. The arithmetic matrix product of A and B can be computed by a randomized algorithm in $\tilde{O}(n(n + \min\{M_A,\ M_{B^T}\}))$ time with high probability.*

4.4 Final Remarks

A natural question is: Would it help to apply an algorithm that optimally rectangulates the 2D projection in Section 4.2.2? Although it would yield improved results in certain cases, it would not give a better approximation factor than 4 in general for the minimum rectangular 3D partition. An example of this is when the optimal 3D partition consists of k cubes lying on top of each other. Then the 2D projection is k concentric squares of different sizes and an optimal rectangulation of the corresponding 2D projection consists of $4k - 3$ rectangles. Hence, for large k, the approximation factor tends to 4.

The 4-approximation algorithm for minimum rectangular partition of a 3D histogram in case the histogram is $his(D)$ for an input $n \times n$ matrix D with nonnegative integer entries can easily be implemented in $O(n^2)$ time. Also note that the resulting partition of $his(D)$ can be used to form a compressed representation of D requiring solely $\tilde{O}(r_D)$ bits if the values of the entries in D are $n^{O(1)}$-bounded.

Our geometric algorithms for integer matrix multiplication can also be applied to derive faster $(1 + \epsilon)$-approximation algorithms for integer matrix multiplication; if the range of an input matrix D is $[0, n^{O(1)}]$, then round each entry to the smallest integer power of $(1 + \epsilon)$ that is not less than the entry. The resulting matrix D' has only a logarithmic number of different entry values and hence $r_{D'}$ may be much less than r_D.

Our algorithms and upper time bounds for integer $n \times n$ matrix multiplication can easily be extended to include integer rectangular matrix multiplication.

Bibliography

[1] N. Alon, P. Dao, I. Hajirasouliha, F. Hormozdiari, and S. Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):i241–i249, 2008.

[2] E. Anshelevich, D. Chakrabarty, A. Hate, and C. Swamy. Approximation algorithms for the firefighter problem: Cuts over time and submodularity. In *Algorithms and Computation*, volume 5878 of *Lecture Notes in Computer Science*, pages 974–983. Springer Berlin Heidelberg, 2009.

[3] C. Bachmaier, W. Brunner, and C. König. Cyclic level planarity testing and embedding. In *Graph Drawing (GD'07)*, volume 4875 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 2008.

[4] M. Badent, E. Di Giacomo, and G. Liotta. Drawing colored graphs on colored points. *Theoretical Computer Science*, 408(2-3):129 – 142, 2008.

[5] N. Bansal and R. Williams. Regularity lemmas and combinatorial algorithms. *Theory of Computing*, 8(4):69–94, 2012.

[6] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

[7] A. Björklund. Determinant sums for undirected hamiltonicity. In *Proc. of FOCS 10, 2010*, pages 173–182.

[8] A. Björklund and A. Lingas. Fast boolean matrix multiplication for highly clustered data. In *Algorithms and Data Structures*, volume 2125 of *Lecture Notes in Computer Science*, pages 258–263. Springer Berlin Heidelberg, 2001.

[9] L. Cai, E. Verbin, and L. Yang. Firefighting on trees: (1-1/e)-approximation, fixed parameter tractability and a subexponential algorithm. In *Algorithms and Computation*, volume 5369 of *Lecture Notes in Computer Science*, pages 258–269. Springer Berlin Heidelberg, 2008.

[10] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 3122 of *Lecture Notes in Computer Science*, pages 72–83. Springer Berlin Heidelberg, 2004.

[11] Y. Chen and J. Flum. On parameterized path and chordless path problems. In *Computational Complexity, 2007. CCC '07. Twenty-Second Annual IEEE Conference on*, pages 250–263, June 2007.

[12] N. Chiba and T. Nishizeki. The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *Journal of Algorithms*, 10(2):187 – 211, 1989.

[13] D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Complexity*, 13(1):42 – 49, 1997.

[14] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251 – 280, 1990. Computational algebraic complexity editorial.

[15] D. Corneil, Y. Perl, and L. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[16] A. Czumaj and A. Lingas. Finding a heaviest vertex-weighted triangle is not harder than matrix multiplication. *SIAM Journal on Computing*, 39(2):431–444, 2009.

[17] R. A. Demillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett*, 1978.

[18] E. Di Giacomo, W. Didimo, G. Liotta, H. Meijer, and S. Wismath. Constrained point-set embeddability of planar graphs. In *Graph Drawing*, volume 5417 of *Lecture Notes in Computer Science*, pages 360–371. Springer, 2009.

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

[20] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999. 530 pp.

[21] J. Edmonds. Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards Sect. B*, 71B:241–245, 1967.

[22] F. Eisenbrand and F. Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57 – 67, 2004.

[23] E. M. Eschen, C. T. Hoang, J. P. Spinrad, and R. Sritharan. On graphs without a c_4 or a diamond. *Discrete Applied Mathematics*, 159(7):581 – 587, 2011. Graphs, Algorithms, and Their Applications - in Honor of Martin Charles Golumbic on the Occasion of His 60th Birthday.

[24] I. Fary. On straight line representation of planar graphs. *Szeged. Sect. Sci. Math*, 11:229–233, 1948.

[25] S. Finbow, A. King, G. MacGillivray, and R. Rizzi. The firefighter problem for graphs of maximum degree three. *Discrete Mathematics*, 307(16):2094 – 2105, 2007. EuroComb '03 - Graphs and Algorithms EuroComb '03 - Graphs and Agorithms.

[26] P. Floderus, M. Kowaluk, A. Lingas, and E.-M. Lundell. Induced subgraph isomorphism: Are some patterns substantially easier than others? In *COCOON*, volume 7434 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2012.

[27] F. V. Fomin, D. Lokshtanov, V. Raman, S. Saurabh, and B.V. R. Rao. Faster algorithms for finding and counting subgraphs. *J. Comput. Syst. Sci.*, 78(3):698–706, May 2012.

[28] H. Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

[29] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[30] A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.*, 31(2):601–625, 2001.

[31] L. Gąsieniec and A. Lingas. An improved bound on boolean matrix multiplication for highly clustered data. In *Algorithms and Data Structures*, volume 2748, pages 329–339. Springer Berlin Heidelberg, 2003.

[32] B. Gelbord. Graphical techniques in intrusion detection systems. In *Information Networking, 2001. Proceedings. 15th International Conference on*, pages 253–258, 2001.

[33] B. Hartnell. Firefighter! an application of domination. In *Proc. of the 24th Manitoba Conference on Combinatorial Mathematics and Computing, University of Manitoba in Winnipeg, Canada*, 1995.

[34] B. Hartnell and Q. Li. Firefighting on trees: how bad is the greedy algorithm? In *Graph Theory and Computing, Congressus Numerantum 145*, pages 187–192.

[35] C. T. Hoang, M. Kaminski, J. Sawada, and R. Sritharan. Finding and listing induced paths and cycles. *Discrete Applied Mathematics*, 161(4-5):633–641, 2013. Seventh International Conference on Graphs and Optimization 2010.

[36] X. Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *Journal of Complexity*, 14(2):257 – 299, 1998.

[37] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[38] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, pages 1–10, New York, NY, USA, 1977. ACM.

[39] M. Kaufmann and R. Wiese. Embedding vertices at points: Few bends suffice for planar graphs. In *Graph Drawing (GD'99)*, volume 1731 of *Lecture Notes in Computer Science*, pages 165–174. Springer Verlag, 1999.

[40] J. Mark Keil. Chapter 11 - polygon decomposition. In *Handbook of Computational Geometry*, pages 491 – 518. North-Holland, Amsterdam, 2000.

[41] S. Khuller, A. Moss, and J. (Seffi) Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39 – 45, 1999.

[42] T. Kloks, D. Kratsch, and H. Müller. Finding and counting small induced subgraphs efficiently. In *Graph-Theoretic Concepts in Computer Science*, volume 1017 of *Lecture Notes in Computer Science*, pages 14–23. Springer Berlin Heidelberg, 1995.

[43] I. Koutis and R. Williams. Limits and applications of group algebras for parameterized problems. In *Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 653–664. Springer Berlin Heidelberg, 2009.

[44] M. Kowaluk, A. Lingas, and E.-M. Lundell. Counting and detecting small subgraphs via equations and matrix multiplication. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1468–1476. SIAM, 2011.

[45] F. Le Gall. Faster algorithms for rectangular matrix multiplication. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 514–523, Oct 2012.

[46] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM.

[47] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Teubner/Wiley & Sons, Stuttgart/Chicester, 1990.

[48] A. Lingas. A geometric approach to boolean matrix multiplication. In *Algorithms and Computation*, volume 2518 of *Lecture Notes in Computer Science*, pages 501–510. Springer Berlin Heidelberg, 2002.

[49] W. Lipski. Finding a manhattan path and related problems. *Networks*, 13(3):399–409, 1983.

[50] R. J. Lipton and R. E. Tarjan. A Separator Theorem for Planar Graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[51] K. Mehlhorn. *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry*. Springer-Verlag New York, Inc., New York, NY, USA, 1984.

[52] S. Muthukrishnan, V. Poosala, and T. Suel. On rectangular partitionings in two dimensions: Algorithms, complexity and applications. In *Database Theory - ICDT 99*, volume 1540 of *Lecture Notes in Computer Science*, pages 236–256. Springer Berlin Heidelberg, 1999.

[53] J. Nesetril and S. Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 026(2):415–419, 1985.

[54] S. Olariu. Paw-free graphs. *Information Processing Letters*, 28(1):53 – 54, 1988.

[55] J. Pach and R. Wenger. Embedding planar graphs at fixed vertex locations. In *Graph Drawing (GD'98)*, volume 1547 of *Lecture Notes in Computer Science*, pages 263–274. Springer, 1998.

[56] C. H. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the v-c dimension. *Journal of Computer and System Sciences*, 53(2):161 – 170, 1996.

[57] J.-R. Sack and J. Urrutia. *Handbook of Computational Geometry*. North-Holland, Amsterdam, 2000.

[58] T. Schank and D. Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *Proceedings of the 4th International Conference on Experimental and Efficient Algorithms*, WEA'05, pages 606–609, Berlin, Heidelberg, 2005. Springer-Verlag.

[59] W. Schnyder. Embedding planar graphs on the grid. In *ACM-SIAM Symposium on Discrete Algorithms (SODA '90)*, pages 138–148, 1990.

[60] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980.

[61] V. Vassilevska. Efficient algorithms for path problems in weighted graphs, 2008. PhD thesis.

[62] V. Vassilevska and R. Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM Journal on Computing*, 42(3):831–854, 2013.

[63] K. Wagner. Bemerkungen zum Vierfarben Problem. *Jahresbericht Deutsch. Math.*, 46:26–32, 1936.

[64] V. V. Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898, New York, NY, USA, 2012. ACM.