DIMACS-DIMATIA International REU
Research Experience for Undergraduates 2015

May 31 – August 1 2015,

Piscataway, NJ, USA and Prague, Czech Republic

Tomáš Masařík (ed.)

DIMACS/DIMATIA International REU 2015

Created by IUUK–ITI series 2015-630.
Centre of Excellence – Institute for Theoretical Computer Science,
Faculty of Mathematics and Physics, Charles University
Malostranské náměstí 25, 118 00, Praha 1, Czech Republic

Published by MatfyzPress,
Publishing House of the Faculty of Mathematics and Physics,
Charles University,
Sokolovská 83, 186 75 Prague 8, Czech Republic
as the 509th publication

Publishing House MatfyzPress is not responsible for quality or correctness of this text.

Prague 2015

ISBN 978-80-7378-315-0

Preface

Charles University in Prague and particularly Department of Applied Mathematics (KAM), Computer Science Institute of Charles University (IUUK) and its international centre DIMATIA, are very proud that they are hosting one of the very few International REU programmes which are funded jointly by NSF and the Ministry of Education of Czech Republic (under the framework of Kontakt programmes ME 521, ME 886 and ME 09074). This programme is a star programme at both ends and it exists for more than a decade since 1999. Repeatedly, it has been awarded for its accomplishments and educational excellence. The program of Kontakt on the Czech side was not renewed for year 2014 and 2015 and thus the programme was financed jointly by Section Informatics of MFF and our grants CE-ITI P202/12/G061, ERCCZ LL1201 and SVV 202-09/260332 (Discrete Models and Algorithms) as well as Department of Applied Mathematics (KAM) and Computer science Institute of Charles University (IUUK). We thank all the contributors and hope that the future will bring us a more stable support. Nevrtheless all our institutions are proud sponsors of this unique activity.

This booklet reports just the programme in 2015. I thank to Tomáš Masařík, the Czech mentor of this year, for a very good work both during the programme itself and after.

Prague, October 25, 2015
Jaroslav Nešetřil

DIMACS/DIMATIA Research Experiences for Undergraduates (REU) is a joint program of the DIMATIA center, Charles University in Prague, The Czech Republic and DIMACS center, Rutgers University, The State University of New Jersey, NJ, USA. This year's participants from Charles University were students Adam Juraszek, Jitka Novotná, Martin Töpfer, Tomáš Toufar, Jan Voborník and Peter Zeman. I (Tomáš Masařík) was their graduate coordinator. I participated in the scientific work as well as I took care of organizing the DIMATIA's part of the program both at Rutgers and in Prague.

We spent our time in Piscataway plesantly together with more than thirty students from universities all over the United States. We participated in the first part of the program at Rutgers University from May 31th

to July 19th. This part of the program mainly consists of open mathematical problems being solved by students and led by their mentors. Students attended several lectures, workshops and tutorials. By the end they also participated in a Field Trip to Rutgers Cancer Institute of New Jersey.

In addition to the scientific program, an important part of the REU is an intercultural experience. At the beginning, a whole day was dedicated to presentations of the Czech Republic, its customs and culture as well as to demonstrations of different cultures of the American students. Moreover, the students together participated in many sport activities, several hiking and sightseeing trips.

Six American students were selected to join, together with their graduate coordinator, the Czech students in the second part of the REU which took place at Charles University in Prague from July 20st to August 1st. The US students were Hadley Black, Linda Cook, Asa Goodwillie, Rayanne Luke, Kevin Sun and Andrew Wells. Their graduate coordinator was Matt Charnley. In Prague, the students attended a series of lectures given by professors mainly from the Department of Applied Mathematics and the Computer Science Institute of Charles University. They also had the opportunity to attend the Midsummer Combinatorial Workshop XXI held from July 27th to July 31st.

All the students got an important experience with research and life abroad. For some of them, the program will certainly be an important milestone in their future scientific career. Many results and thoughts from the last summer are still being improved and some of them are going to be submitted to international conferences.

This booklet presents the results of the Czech students stemming from the REU programme and reports of the American students about their lectures at Prague.

For me it was a perfect experience and I am glad that I could have been a part of this program.

At the end, I would like to thank all the participating students, people at DIMACS and other organizers. They all were absolutely perfect. Also an important role played people from the both our departments at Prague. I thank them for many helpful comments, encouraging advices and an overal support.

Prague, Winter 2015
Tomáš Masařík,
Charles University in Prague

The DIMACS/DIMATIA Exchange program has been going on for many years and is a valuable experience for all involved. This year, our DIMATIA Program group consisted of Hadley Black (UC-Santa Cruz), Linda Cook (Rutgers), Asa Goodwillie (Amherst College), Rayanne Luke (SUNY-Geneseo), Kevin Sun (Rutgers), and Andrew Wells (Catholic University). I was the graduate coordinator from Rutgers this year, and it was a pleasure to help them with the program at Rutgers, as well as the trip to Charles University. Not only did they have a great time on the trip to Prague, they also learned a lot of interesting mathematics.

During the first week, we attended lectures from faculty and visitors of Charles University. These lectures were given by Jirka Fiala, John Gimbel, and Andrew Goodall. The American students have written up summaries and discussions about each of these talks, which can be seen on the following pages. These workshops were meant to cover interesting topics in discrete math and combinatorics and help prepare the students for the Midsummer Combinatorial Workshop the following week. I found the lectures very interesting, and feel like the students got a lot out of them as well.

The second week of the Prague trip consisted of the Midsummer Combinatorial Workshop. Even though most of the material was very advanced, I know it was a great experience for the participants. Having the opportunity to attend a professional math conference, especially as an undergraduate, is something that does not happen very often. With the caliber of guests and presenters at this conference, this really was a once-in-a-lifetime experience for all of us. Even though I am not looking to study combinatorics in my math career, it was a great experience for me as well.

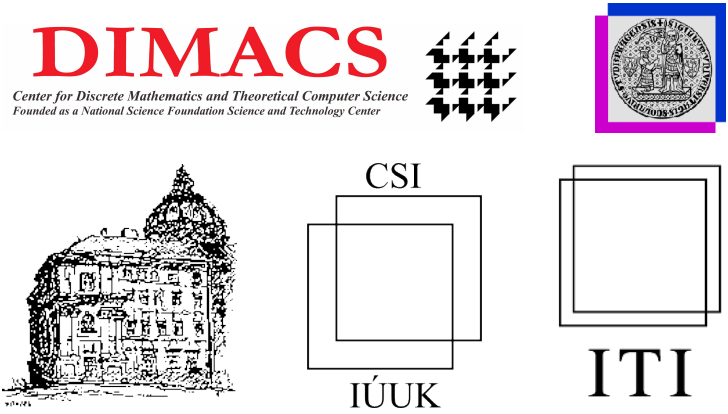
Thanks are in order for all of the people who made this possible. On the Rutgers end, the entire DIMACS staff worked tirelessly to make this trip happen. Gene Fiorini, the head of the REU Program, has been doing this for many years, and always keeps things running like they should, so they all deserve our thanks for that. At Charles University, the KAM MFF faculty and staff gave us all the assistance we needed for our stay in Prague. We would especially like to thank Jaroslav Nešetřil, Tomáš Masařík, and all of the other Czech REU students for being our welcoming committee and guides for our two weeks in Prague. Thank you very much. None of this would have been possible without all of your help and guidance.

Personally, I would also like to thank our REU participants from Rutgers. You all were a pleasure to work with for the 7 weeks in New Jersey as well as the two weeks in Prague. I hope you got as much out of the experience as I did, both mathematically and from the other activities of

the weeks. Best of luck with everything.

Matt Charnley
Rutgers University

List of sponsors



The participants of the Internation REU programme at Rutgers University.

The participants of the Prague part of the programme.

Midsummer Combinatorial Workshop XXI—conference group photos.

Contents

Parameterized complexity of fair deletion problems.

Tomáš Masařík, Tomáš Toufar

Abstract

Edge deletion problems are those where given a graph G and a graph property π, the goal is to find a subset of edges such that after its removal the graph G will satisfy the property π. Typically, we want to minimize the number of edges removed. In fair deletion problem we change the objective: we minimize the maximum number of edges incident to a single vertex.

We study the parameterized complexity of fair deletion problems with respect to the structural parameters of the tree-width, the path-width, the size of a minimum feedback vertex set, the neighborhood diversity, and the size of minimum vertex cover of graph G.

We prove the W[1]-hardness of the fair MSO edge-deletion with respect to the first three parameters combined. Moreover, we show that there is no algorithm for fair MSO edge-deletion running in time $n^{o(\sqrt{k})}$, where n is the size of the graph and k is the sum of the first three mentioned parameters, provided that the Exponential Time Hypothesis holds.

On the other hand, we provide an FPT algorithm for the fair MSO edge-deletion parameterized by the size of minimum vertex cover and an FPT algorithm for the fair MSO vertex-deletion parameterized by the neighborhood diversity.

1 Introduction

We study the computational complexity of *fair deletion problems*. Deletion problems are a standard reformulation of some classical problems in combinatorial optimization examined by Yannakakis [18]. For a graph property π we can formulate an *edge deletion problem*. That means, given a graph

$G = (V, E)$, find the minimum set of edges F that need to be deleted for graph $G' = (V, E \setminus F)$ to satisfy property π. A similar notion holds for the *vertex deletion problem*.

Many classical problems can be formulated in this way such as MINIMAL VERTEX COVER, MAXIMUM MATCHING, MAXIMAL EDGE CUT or MINIMAL FEEDBACK ARC SET. For example MINIMAL VERTEX COVER is formulated as a vertex deletion problem since we aim to find a minimum set of vertices such that the rest of the graph forms an independent set. An example of an edge deletion problem is PERFECT MATCHING: we would like to find a minimum edge set such that resulting graph has all vertices being of degree at most one. Many of such problems are NP-complete [17, 1, 12].

Fair deletion problems are such modifications where the cost of the solution should be split as evenly as possible between all participants. More formally, the FAIR EDGE DELETION PROBLEM for a given graph $G = (V, E)$ and a property π finds a set F which minimizes the maximum degree of graph $G^* = (V, F)$ where graph $G' = (V, E \setminus F)$ satisfies the property π.

We focus on fair deletion problems with properties definable in monadic second order (MSO) logic. Our work extends the result of Kolman et al. [11]. They showed an XP algorithm for slightly different version of fair deletion problems definable by MSO_2 formula on graphs of bounded tree-width. The difference is that the formula should be satisfied for the removed set F, not only for resulting graph G'. We give formal definitions of problems.

Definition 1.1 (FAIR MSO EDGE-DELETION)**.**

Input:	An undirected graph G, an MSO formula ψ with one free edge-set variable, and a positive integer k.
Question:	Is there a set $F \subseteq E(G)$ such that $G \models \psi(F)$ and for every vertex v of G, the number of edges in F incident with v is at most k?

This problem was introduced by Lin and Sahni in [14]. Similarly, FAIR VERTEX DELETION PROBLEM finds, for a given graph $G = (V, E)$ and a property π, the solution, which is the minimum of maximum degree of graph $G^* = (W, E)$ where graph $G = (V \setminus W, E)$ satisfy property π. Those problems are usually NP-complete as well [14].

Definition 1.2 (Fair MSO vertex-deletion)**.**

Input:	An undirected graph G, an MSO formula ψ with one free vertex-set variable, and a positive integer k.
Question:	Is there a set $W \subseteq V(G)$ such that $G \models \psi(W)$ and for every vertex v of G, it holds that $\lvert N(v) \cap W \rvert \leq k$?

Courcelle and Mosbah [5] introduced a semiring homomorphism framework that can be used to minimize various functions over all sets satisfying a given MSO formula. A natural question is whether this framework can be used to minimize the fair objective function. The answer is no, as we exclude the possibility of FPT algorithm under reasonable assumption. Note that there are semirings that capture the fair objective function, but their size is of order $n^{\mathrm{tw}\, G}$, so this approach will not lead to an FPT algorithm.

1.1 Our results

We prove that the XP algorithm given by Kolman et al. [11] is almost optimal under exponential time hypothesis (ETH) for both edge and vertex version.

Theorem 1.3. *If there is an FPT algorithm for* Fair MSO edge-deletion *parameterized by the size of the formula ψ, the pathwidth of G, and the size of smallest feedback vertex set of G combined, then* FPT = W[1]. *Moreover, let k denote* $\mathrm{pw}(G) + \mathrm{fvs}(G)$. *If there is an algorithm for* Fair MSO edge-deletion *with running time $f(|\psi|, k)n^{o(\sqrt{k})}$, then Exponential Time Hypothesis fails.*

Theorem 1.4. *If there is an FPT algorithm for* Fair MSO vertex-deletion *parameterized by the size of the formula ψ, the pathwidth of G, and the size of smallest feedback vertex set of G combined, then* FPT = W[1]. *Moreover, let k denote* $\mathrm{pw}(G) + \mathrm{fvs}(G)$. *If there is an algorithm for* Fair MSO edge-deletion *with running time $f(|\psi|, k)n^{o(\sqrt{k})}$, then Exponential Time Hypothesis fails.*

On the other hand we show some positive algorithmic results.

Theorem 1.5. Fair MSO_1 vertex-deletion *is in* FPT *with respect to the neighborhood diversity* $\mathrm{nd}(G)$.

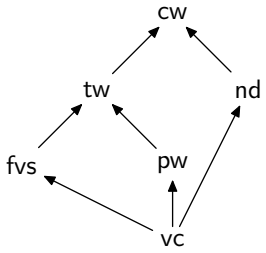


Figure I.1: Hierarchy of graph parameters. An arrow indicates that a graph parameter upper-bounds the other. So, hardness results are implied in direction of arrows and FPT algorithms are implied in the reverse direction.

Theorem 1.6. FAIR MSO_2 EDGE-DELETION *is in* FPT *with respect to the minimum size of vertex cover* $\mathrm{vc}(G)$.

2 Preliminaries

Throughout the paper we deal with simple undirected graphs. For further standard notation in graph theory, we refer to Diestel [6]. For terminology in parameterized computational complexity we refer to Downey and Fellows [7].

2.1 Graph parameters

We define several graph parameters being used throughout the paper. We start by definition of *vertex cover* being a set of vertices such that their neighborhood is an independent set. By $\mathrm{vc}\,(G)$ we denote the size of smallest such set. This is the strongest of considered parameters and it is not bounded for any natural graph class.

A *feedback vertex set* is a set of vertices whose removal leaves an acyclic graph. Again, by $\mathrm{fvs}\,(G)$ we denote the size of smallest such set.

Another famous graph parameter is *tree-width* introduced by Bertelé and Brioshi in [3].

Definition 2.1 (Tree decomposition)**.** A *tree decomposition* of a graph G is a pair (T, X), where $T = (I, F)$ is a tree, and $X = \{X_i \mid i \in I\}$ is a family of subsets of $V(G)$ such that:

- the union of all X_i, $i \in I$ equals V,
- for all edges $\{v, w\} \in E$, there exists $i \in I$, such that $v, w \in X_i$ and

- for all $v \in V$ the set of nodes $\{i \in I \mid v \in X_i\}$ forms a subtree of T.

The *width* of the tree decomposition is $\max(|X_i| - 1)$. The *tree-width* of a graph $\mathrm{tw}(G)$ is the minimum width over all possible tree decompositions of the graph G. The parameter of *path-width* (analogically $\mathrm{pw}(G)$) is almost the same except the decomposition need to form a path instead of a general tree.

A less known graph parameter is the *neighborhood diversity* introduced by Lampis [13].

Definition 2.2 (Neighborhood diversity)**.** The *neighborhood diversity* of a graph G is denoted by $\mathrm{nd}(G)$ and it is the minimum size of a partition of vertices into classes such that all vertices in the same class have the same neighborhood, i.e. $N(v) \setminus \{v'\} = N(v') \setminus \{v\}$, whenever v, v' are in the same class.

It can be easily verified that every class of neighborhood diversity is either a clique or an independent set. Moreover, for every two distinct classes C, C', either every vertex in C is adjacent to every vertex in C', or there is no edge between C and C'. If classes C and C' are connected by edges, we refer to such classes as *adjacent*.

2.2 Parameterized problem and Exponential Time Hypothesis

Definition 2.3 (Parameterized problem)**.** Let Σ be a finite alphabet. A parameterization of Σ^* (set of all words over the alphabet Σ) is a polynomial-time computable mapping $\kappa : \Sigma^* \to \mathbb{N}$. A parameterized language is a set of pairs $(x, \kappa(x))$ where x is a word and κ is its parameterization.

We now briefly introduce the Exponential Time Hypothesis (ETH for short). It is a complexity theoretic assumption introduced by Impagliazzo, Paturi and Zane [10]. It is useful for proving lower bounds on NP-hard combinatorial problems. We follow a survey on this topic by Fellows et al. [9], which contains more details on this topic.

The hypothesis states that there is no subexponential time algorithm for 3-SAT if we measure the time complexity by the number of variables in the input formula, denoted by n.

> **Exponential Time Hypothesis [10]** There is a positive real s such that 3-SAT with parameter n cannot be solved in time $2^{sn}(n+m)^{\mathrm{O}(1)}$.

Definition 2.4 (Standard parameterized reduction)**.** We say that parameterized language L reduces to parameterized language L' by the *standard parameterized reduction* if there are functions $f\colon \mathbb{N} \to \mathbb{N}$, $g\colon \mathbb{N} \to \mathbb{N}$ and $h\colon \Sigma^* \times \mathbb{N} \to \Sigma^*$ such that:

- function h is computable in time $g(k)|x|^c$ for a constant c,
- $(x, k) \in L$ if and only if $(h(x, k), f(k)) \in L'$.

For preserving bounds obtained from the ETH, the asymptotic growth of the function f need to be as slow as possible.

2.3 Logic systems

We will heavily use graph properties that can be expressed in certain types of logical systems. In the paper it is *Monadic second-order logic* (MSO) where monadic means that we allow quantification only over sets not over functions as it is in full second order logic.

We distinguish MSO_2 and MSO_1. In MSO_1 quantification only over sets of vertices is allowed and we can use the predicate of adjacency $\mathrm{adj}(u, v)$ returning true whenever there is an edge between vertices u and v. In MSO_2 we can additionally quantify over sets of edges and we can use the predicate of incidence $\mathrm{inc}(v, e)$ returning true whenever a vertex v belongs to an edge e.

2.4 Courcelle's theorem

The famous Courcelle's metatheorem [4] proves that there is an FPT algorithm deciding any property definable in MSO_2 on graphs of bounded tree-width.

Theorem 2.5 (Courcelle with free variables [2])**.** *For any* MSO *language formula ϕ with free set variables $A_1, A_2, \ldots, A_p$ and for any $k > 0$ there exists an FPT algorithm that given a graph G with $\mathrm{tw}(G) \leq k$ find sets $A_1, A_2, \ldots, A_p$ such that $G \models \phi(A_1, A_2, \ldots, A_p)$.*

3 Hardness results

To prove the hardness of fair deletion problems, we provide a reduction from a variant of MSO partitioning. The classical MSO partitioning was

introduced by Rao [16]. In our variant, we study equitable partitions instead of arbitrary partitions. We say that a partition is *equitable* if the sizes of any two classes differ by at most one. The equitable version of MSO partitioning is as follows:

Definition 3.1 (Equitable MSO partition)**.**

Input:	A graph G, an MSO formula ϕ with a free vertex-set variable, and a positive integer $r \geq 2$.
Question:	Is there an equitable partition of vertices into r sets such that each class of the partition satisfies ϕ?

Equitable MSO partition generalizes several problems already studied before. For example, if the formula $\phi(X)$ is "X is independent", then we get an instance of Equitable coloring [9]. If we set $\phi(X)$ to "X is connected", then we get an instance of Equitable connected partition [8].

We first prove the hardness of Equitable MSO partition with respect to $|\phi|, \mathrm{pw}(G), \mathrm{fvs}(G)$, and r combined using the result of Enciso et al. [8]. We then construct a parameterized reduction from Equitable MSO partition to Fair MSO edge-deletion to complete the proof of Theorem 1.3.

Theorem 3.2. Equitable MSO partition *is* W[1]*-hard with respect to* $|\phi|$, $\mathrm{pw}(G)$, $\mathrm{fvs}(G)$, *and* r *(the number of partitions) combined. Moreover, let* k *be* $r + \mathrm{pw}(G) + \mathrm{fvs}(G)$*. If there exist an algorithm running in time* $f(|\phi|, k)n^{o(\sqrt{k})}$*, then the Exponential Time Hypothesis fails.*

Proof. Since Equitable connected partition is just a special case of Equitable MSO partition, the first part follow directly from the result of Enciso et al. [8]. The second part, though not mentioned explicitly also follows from [8]. The proof is based on the reduction of Multicolored clique of size ℓ to Equitable connected partition with $\mathrm{fvs}, \mathrm{pw}, r$ of order $O(\ell^2)$. Therefore, an algorithm for Equitable connected partition with running time $f(k)n^{o(\sqrt{k})}$ would lead to an algorithm for Multicolored clique of size k with running time $f(k)n^{o(k)}$. It was proven by Lokshtanov, Marx, and Saurabh [15] that Multicolored clique of size k cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails. □

We now sketch the reduction from Equitable MSO partition to Fair MSO edge-deletion. Let us denote by n the number of vertices of G and

for simplicity assume that r divides n. We add r new vertices $s_1, \ldots, s_r$ called *selector* vertices, each corresponding to one class of partition. We connect each of these vertices with each vertex of G. Note that this can increase the path-width or the size of minimum feedback vertex set by at most r. The partition will be determined by deleted edges from $s_1, \ldots, s_r$ to V. If a vertex $v \in V(G)$ is incident with a deleted edge $\{v, s_i\}$, then v will belong to the class i. The formula needs to ensure several things; formal description will be shown later:

- no edges from the original graph are deleted,
- each vertex v of G is incident with exactly one deleted edge, and
- every class of the partition satisfies ϕ.

The equitability of the partition will be handled by the fair objective function. Note that we always delete n edges. Those n edges are incident with r vertices, so the best possible fair cost is n/r. A solution with fair cost n/r corresponds to an equitable partition.

However, we need to deal with two problems. First, we need to distinguish the added vertices from the vertices of the original graph. To accomplish that, we add more vertices than just $s_1, \ldots, s_r$. Next, we need to handle the case when r does not divide n, as the condition that solution has fair cost $\lceil n/r \rceil$ does not work. For example, partitioning the vertex set of size 7 into 3 sets of sizes $3, 3, 1$ has the fair cost $3 = \lceil 7/3 \rceil$, but the partition is not equitable.

Let us now describe the reduction formally.

of Theorem 1.3. Let $G = (V, E)$ be a graph with $|V| \geq 2$, let r be the desired number of classes in the partition, and finally let ϕ be the formula with one free vertex-set variable. Denote by n the number of vertices of G, and let r' be the smallest nonnegative integer such that r divides $n + r'$.

We add r vertices called *selector vertices* and connect each one of them to each vertex of G. Then, to every selector vertex we attach a vertex called a *marker vertex*. Finally, we choose r' selector vertices and subdivide the edges between those selector vertices and the marker vertex adjacent to it. The new vertices created in this way are called *padding vertices*. Denote the new graph by G'.

If an edge connects a selector vertex to a padding vertex, it is called *padding edge*. Edges between selector vertices and vertices of the original graph are called *selector edges* (see Fig. I.2).

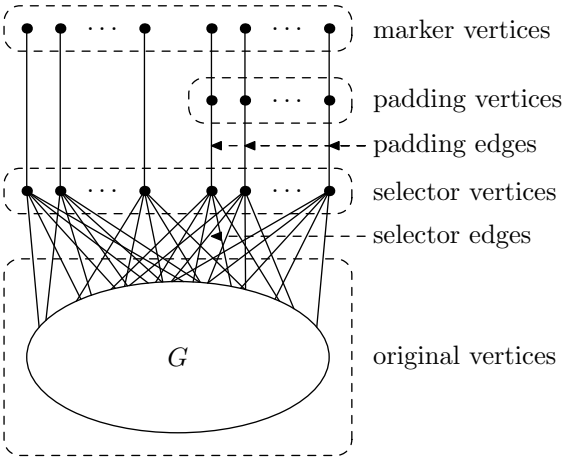


Figure I.2: Reduction

Consider a set $F \subseteq E(G')$ satisfying the following conditions:

(i) no edge other than a padding edge or a selector edge is in F,

(ii) all padding edges are in F, and

(iii) every vertex of G is incident to exactly one edge of F.

Such a set is called a *selector set*. Every selector set encodes a partition of $V(G)$ into r classes in the following way: if F is a selector set and v_i is a selector vertex, then a class induced by this vertex is the set $\{v \in V(G) \mid \{v, v_i\} \in F\}$. It follows from condition (iii) that the set of all such classes forms a partition of $V(G)$. We can also construct a selector set given a partition $V(G)$. Note that a selector set is uniquely determined by a partition up to a permutation of selector vertices.

In order to specify a ϕ-partition, we need to add the following condition:

(iv) every class induced by F satisfies ϕ.

We now describe a formula $\text{part}_\phi(F)$ with one free vertex set such that $G' \models \text{part}_\phi(F)$ if and only if F satisfies (i)-(iv). The building blocks for the formula $\text{part}_\phi(F)$ are as follows:

$$\text{marker}(v) \equiv (\deg(v) = 1)$$

$$
\begin{aligned}
\text{padding}(v) &\equiv (\exists w)\big(\text{marker}(w) \wedge \text{adj}(v,w) \wedge (\deg(v)=2)\big)\\
\text{selector}(v) &\equiv (\exists w)(\text{marker}(w) \wedge \text{adj}(v,w) \wedge \neg\text{padding}(v))\\
&\quad \vee (\exists w)(\text{padding}(w) \wedge \text{adj}(v,w) \wedge \neg\text{marker}(v))\\
\text{orig}(v) &\equiv \neg\text{marker}(v) \wedge \neg\text{padding}(v) \wedge \neg\text{selector}(v)\\
\text{padding\_edge}(e) &\equiv (\exists v,w)(\text{inc}(e,v) \wedge \text{inc}(e,w) \wedge \text{padding}(v) \wedge\\
&\quad \text{selector}(w))\\
\text{selector\_edge}(e) &\equiv (\text{inc}(e,v) \wedge \text{inc}(e,w) \wedge \text{selector}(v) \wedge \text{orig}(w))\\
\text{selector\_set}(F) &\equiv (\forall e \in F)(\text{padding\_edge}(e) \vee \text{selector\_edge}(e))\\
&\quad \wedge (\forall e)(\text{padding\_edge(e)} \rightarrow e \in F)\\
&\quad \wedge (\forall v)(\text{orig}(v) \rightarrow (\exists_{=1} e \in F)(\text{inc}(e,v))\\
\text{is\_in\_class}(w,u,F) &\equiv (\text{selector}(u) \wedge \text{orig}(w)) \wedge (\exists e \in F)(\text{inc}(u,e) \wedge\\
&\quad \text{inc}(w,e))\\
\text{class}(W,u,F) &\equiv (\forall w \in W)(\text{is\_in\_class}(w,u,F))\\
&\quad \wedge (\forall v)(\text{is\_in\_class}(v,u,F) \rightarrow v \in W)\\
\text{part}_\phi(F) &\equiv \text{selector\_set}(F)\\
&\quad \wedge (\forall v,W)(\text{selector}(v) \wedge \text{class}(W,v,F) \rightarrow \phi(W))
\end{aligned}
$$

Marker vertices are the only vertices of degree one: padding vertices and selector vertices have degree at least two, and all vertices in the original graph are adjacent to $r \geq 2$ selector vertices. *If a vertex is adjacent to a marker vertex and has degree two, it is a padding vertex:* Since we assumed $|V| \geq 2$, selector vertices have degree at least three. A vertex with degree two adjacent to a marker vertex is necessarily a padding vertex. A selector can be adjacent either to a marker vertex (the first part of the disjunction in the selector formula), or to a padding vertex (the second part of the disjunction in the formula). Clearly, all remaining vertices are vertices of the original graph. The fact that formulae for padding edges and selector edges match their definition is immediate. The formula selector_set(F) describes exactly conditions (i), (ii), and (iii) that define a selector set. The formula is_in_class(w,u,F) is true if and only if the vertex $w \in V(G)$ is in the class induced by the vertex u in the partition determined by F. The formula class(W,u,F) is true if and only if W is a class induced by the vertex u in the partition determined by F. Finally, the correctness of part$_\phi$ is clear from the previous observations.

The described reduction will map an instance (G,ϕ,r) of EQUITABLE

MSO PARTITION to instance $(G', \mathrm{part}_\phi, \lceil n/r \rceil)$ of FAIR MSO EDGE-DELETION (where G', part_ϕ, and n are defined as above).

We now prove that described reduction is indeed a valid parameterized reduction from EQUITABLE MSO PARTITION to FAIR MSO EDGE-DELETION. Given an equitable partition such that each class satisfies ϕ, we set F as a selector set that induces the given partition – we need to match the smaller classes with selector vertices adjacent to padding vertices. It is straightforward to check that F satisfies part_ϕ and the fair cost of F is $\lceil n/r \rceil$.

For the other direction, let F be a set satisfying part_ϕ with fair cost $\lceil n/r \rceil$. The number of deleted edges is always $n + r'$, since we have r' padding edges that have to be deleted, plus we have to delete exactly one edge incident to every original vertex. All those edges are incident to r selector vertices, the best achievable fair cost is therefore $(n+r')/r = \lceil n/r \rceil$. In an optimal solution, every selector has exactly $\lceil n/r \rceil$ incident edges. This means that every selector vertex adjacent to a padding vertex induces a class with $\lceil n/r \rceil - 1$ vertices and every other selector vertex induces a class with $\lceil n/r \rceil$ vertices. Hence, the partition is equitable. By the construction of part_ϕ, every class in the partition has to satisfy ϕ.

Let us now discuss the parameters. If G has a feedback vertex set S of size k, then the union of S with all selector vertices of G' is a feedback vertex set of G'. Therefore, $\mathrm{fvs}(G') \leq \mathrm{fvs}(G) + r$. Since we add at most $3r - 1$ vertices, we have $\mathrm{pw}(G') \leq \mathrm{pw}(G) + 3r - 1$. Finally, the size of the formula part_ϕ can be bounded in terms of size of the formula ϕ. The whole construction can be clearly carried out in polynomial time. □

As we have $\mathrm{tw}(G) \leq \mathrm{pw}(G)$ and $\mathrm{tw}(G) \leq \mathrm{fvs}(G) + 1$ for every graph G, we immediately get the following corollary:

Corollary 3.3. *If there is an FPT algorithm for* FAIR MSO EDGE-DELETION *parameterized by the size of the formula ψ and the tree-width of G combined, then* FPT = W[1]. *Furthermore, if there is an algorithm for* FAIR MSO EDGE-DELETION *with running time $f(|\psi|, \mathrm{tw}(G))n^{o(\sqrt{\mathrm{tw}(G)})}$, then the Exponential Time Hypothesis fails.*

We now sketch the proof of Theorem 1.4:

of Theorem 1.4. The idea of reduction is essentially the same. Now we need to encode the partition by deleting vertices. We subdivide every padding edge and every selector edge (note that this does not increase the size of

smallest feedback vertex set and can increase the path-width by at most one) Then, instead of deleting padding or selector edges, we delete the corresponding subdividing vertices. To achieve this, we need to update the formula accordingly; the details are left to the reader. □

As before, we obtain the following corollary for parameterization by tree-width.

Corollary 3.4. *If there is an FPT algorithm for* Fair MSO vertex-deletion *parameterized by the size of the formula ψ and the tree-width of G combined, then* FPT = W[1]. *Furthermore, if there is an algorithm for* Fair MSO vertex-deletion *with running time $f(|\psi|, \mathrm{tw}(G))n^{o(\sqrt{\mathrm{tw}(G)})}$, then the Exponential Time Hypothesis fails.*

4 FPT algorithms

We now turn our attention to FPT algorithms for fair deletion problems.

4.1 FPT algorithm for parameterization by neighborhood diversity

Definition 4.1. Let $G = (V, E)$ be a graph of neighborhood diversity k and let $N_1, \ldots, N_k$ denote its classes of neighborhood diversity. A *shape of a set $X \subseteq V$ in G* is a k-tuple $s = (s_1, \ldots, s_k)$, where $s_k = |X \cap N_i|$.

We denote by $\overline{s}$ the *complementary shape to s*, which is defined as the shape of $V \setminus X$, i.e. $\overline{s} = (|N_1| - s_1, \ldots, |N_k| - s_k)$.

Proposition 4.2. *Let $G = (V, E)$ be a graph, π a property of a set of vertices, and let $X, Y \subseteq V$ be two sets of the same shape in G. Then X satisfies π if and only if Y satisfies π.*

Proof. Clearly, we can construct an automorphism of G that maps X to Y. □

Definition 4.3. Let r be a non-negative integer and $(s_1, \ldots, s_k)$, $(t_1, \ldots, t_k)$ be two shapes. Theuuu shapes are *r-equivalent*, if for every i:

- $s_i = t_i$, or
- both s_i, t_i are strictly greater than r,

and the same condition hold for the complementary shapes $\overline{s}$, $\overline{t}$.

The following proposition gives a bound on the number of r-nonequivalent shapes.

Proposition 4.4. *For any graph G of neighborhood diversity k, the number of r-nonequivalent shapes is at most $(2r+3)^k$.*

Proof. We show that for every i, there are at most $(2r+3)$ choices of s_i. This holds trivially if $|N_i| \leq 2r+3$. Otherwise we have following $2r+3$ choices:

- $s_i = k$ and $\overline{s_i} > r$ for $k = 0, 1, \ldots, r$, or
- both $s_i, \overline{s_i} > r$, or
- $s_i > r$ and $\overline{s_i} = k$ for $k = 0, 1, \ldots, r$.

□

The next lemma states that the fair cost of a set can be computed from its shape in a straightforward manner. Before we state it, let us introduce some auxiliary notation.

If a graph G of neighborhood diversity k has classes of neighborhood diversity $N_1, N_2, \ldots, N_k$, we write $i \sim j$ if the classes N_i and N_j are adjacent. If the class N_i is a clique, we set $i \sim i$. Moreover, we set $\eta_i = 1$ if the class N_i is a clique and $\eta_i = 0$ if it is an independent set. The classes of size one are treated as cliques for this purpose.

Lemma 4.5. *Let $G = (V, E)$ be a graph of neighborhood diversity k and let N_i be its classes of neighborhood diversity. Moreover, let $X \subseteq V$ be a set of shape s. Then the fair vertex cost of X is*

$$\max_i \sum_{j:i\sim j} s_j - \eta_i.$$

Proof. It is quite straightforward to check that vertex $v \in N_i$ has exactly $\sum_{j:i\sim j} s_j - \eta_i$ neighbors in X. □

Our main tool is a reformulation of Lemma 5 from [13]:

Lemma 4.6. *Let ψ be an MSO_1 formula with one free edge-set variable, q_E vertex element quantifiers, and q_S vertex set quantifiers. Let $r = 2^{q_S} q_E$. If $G = (V, E)$ is a graph of neighborhood diversity k and $X, Y \subseteq V$ are two sets such that their shapes are r-equivalent, then $G \models \psi(X)$ if and only if $G \models \psi(Y)$.*

The last result required is the MSO_1 model checking for graphs of bounded neighborhood diversity [13]:

Theorem 4.7. *Let ψ be an MSO_1 formula with one free vertex-set variable. There exists an FPT algorithm that given a graph $G = (V, E)$ of neighborhood diversity k and a set $X \subseteq V$ decides whether $G \models \psi(X)$. The running time of the algorithm is $f(k, |\psi|)n^{\mathrm{O}(1)}$.*

We now have all the tools required to prove Theorem 1.5.

of Theorem 1.5. Let ψ be an MSO_1 formula in the input of Fair MSO_1 vertex-deletion. Denote by q_S the number of vertex-set quantifiers in ψ, by q_E the number of vertex-element quantifiers in ψ, and set $r = 2^{q_S} q_E$.

By Proposition 4.2, the validity of $\psi(X)$ depends only on the shape of X. Let us abuse notation slightly and write $G \models \psi(s)$ when "X has shape s" implies $G \models \psi(X)$. Similarly, Lemma 4.5 allows us to refer to the fair cost of a shape s.

From Lemma 4.6 it follows that the validity of $\psi(s)$ does not depend on the choice of an r-equivalence class representative. The fair cost is not same for all r-equivalent shapes, but since the fair cost is monotone in s, we can easily find the representative of the minimal fair cost.

Suppose we have to decide if there is a set of a fair cost at most ℓ. The algorithm will proceed as follows: For each class of r-equivalent shapes, pick a shape s of the minimal cost, if the fair cost is at most ℓ and $G \models \psi(s)$, output `true`, if no such shape is found throughout the run, output `false`.

By the previous claims, the algorithm is correct. Let us turn our attention to the running time. The number of shapes is at most $(2r + 3)^k$ by Proposition 4.4, and so it is bounded by $f(|\psi|, k)$ for some function f. The MSO_1 model checking runs in time $f'(|\psi|, k)n^{\mathrm{O}(1)}$ by Theorem 4.7, so the total running time is $f(|\psi|, k)f'(|\psi|, k)n^{\mathrm{O}(1)}$, so the described algorithm is in FPT.

□

4.2 FPT algorithm for parameterization by vertex cover

The FPT algorithm for parameterization by the size of minimum vertex cover uses the same idea. We use the fact that every MSO_2 formula can be translated to MSO_1 formula — roughly speaking, every edge-set variable is replaced by $\mathrm{vc}(G)$ vertex-set variables.

We only sketch translation from MSO_2 to MSO_1, for the proof we refer the reader to Lemma 6 in [13]. Let $G = (V, E)$ be a graph with vertex cover $C = \{v_1, \ldots, v_k\}$ and $F \subseteq E$ a set of edges. We construct vertex set $U_1, \ldots, U_k$ in the following way: if w is a vertex such that an edge in F connects w with v_i, we put w into V_i. It is easy to see that the sets $U_1, \ldots, U_k$ together with the vertex cover $v_1, \ldots, v_k$ describe the set F.

The translation from set of edges into k sets of vertices is captured by the following definition.

Definition 4.8. Let $G = (V, E)$ be a graph with vertex cover $v_1, \ldots, v_k$. For a set $F \subseteq E$, we define *the signature of F with respect to $v_1, \ldots, v_k$* as the k-tuple $\mathcal{U} = (U_1, \ldots, U_k)$, where $U_i = \{w \in V \mid \{w, v_i\} \in E\}$. If the vertex cover is clear from the context, we refer to it as *the signature* of F and denote it by $S(F)$.

In the original problem, we had an MSO_2 formula ψ_2 with one free edge-set variable. By the translation, we obtain an MSO_1 formula ψ with k free vertex-set variables and k free vertex-element variables (the vertex-element variables will describe the vertex cover; the formula need to have access to a vertex cover and it will be useful to fix one throughout the whole run of the algorithm).

We start by finding a vertex cover $v_1, \ldots, v_k$ (this can be done in FPT [7]). Now, we want to find the sets $U_1, \ldots, U_k$ such that:

$$G \models \psi(v_1, \ldots, v_k, U_1, \ldots, U_k).$$

To find such k-tuple of sets, we need to extend the notion of shapes to signatures.

Definition 4.9. Let $G = (V, E)$ be a graph with vertex cover $v_1, \ldots, v_k$, and let $\mathcal{U} = (U_1, \ldots, U_k)$ be a collection of k subsets of V. Denote by $N_1, \ldots, N_\ell$ the classes of neighborhood diversity of G. For $j \in \{1, \ldots, \ell\}$ and $I \subseteq \{1 \ldots k\}$, denote by $\overline{I}$ the set $\{1, \ldots, k\} \setminus I$. Furthermore, we define

$S_{\mathcal{U}}(j, I)$ as

$$S_{\mathcal{U}}(j, I) = \left| N_j \cap \bigcap_{i \in I} U_i \cap \bigcap_{i \in \overline{I}} (V \setminus U_i) \right|.$$

The mapping $S_{\mathcal{U}}$ is called *the shape of a signature* $\mathcal{U}$.

The shapes defined in this way have properties similar to those defined for neighborhood diversity; we only state those properties without proofs.

Definition 4.10. Two shapes S, S' are called *r-equivalent* if for every $j \in \{1, \ldots, k\}$, $I \subseteq \{1, \ldots, k\}$ it holds that

- $S(j, I) = S'(j, I)$, or
- both $S(j, I)$, $S'(j, I)$ are strictly greater than r.

As in the neighborhood diversity case, the number of r-nonequivalent shapes is bounded by a function of r and k.

Proposition 4.11. *Let $G = (V, E)$ be a graph with vertex cover $v_1, \ldots, v_k$ and denote by ℓ the neighborhood diversity of G. The number of r-nonequivalent shapes is at most $(2r + 3)^{\ell 2^k}$.*

We now state corresponding variants of Lemma 4.5 and Lemma 4.6.

Lemma 4.12. *Let $G = (V, E)$ be a graph with a vertex cover $v_1, \ldots, v_k$ and let F be a subset of E.*

The number of edges in F incident to v_i is $|U_i|$. If w is a vertex different from $v_1, \ldots, v_k$, then the number of edges in F incident to w is $|\{i \mid w \in U_i\}|$.

Those quantities (and therefore the fair cost of F) can be determined from the shape of $S(F)$.

Lemma 4.13. *Let $G = (V, E)$ be a graph with a vertex cover $v_1, \ldots, v_k$, let ψ be an MSO_1 formula with k free vertex-element variables and k free vertex-set variables, and let $\mathcal{U} = (U_1, \ldots, U_k)$, $\mathcal{W} = (W_1, \ldots, W_k)$ be two signatures. If the shapes of $\mathcal{U}$ and $\mathcal{W}$ are r-equivalent, then:*

$$G \models \psi(v_1, \ldots, v_k, U_1, \ldots, U_k)$$

if and only if

$$G \models \psi(v_1, \ldots, v_k, W_1, \ldots, W_k).$$

of Theorem 1.6. The algorithm goes as follows:

- we translate the MSO_2 formula ψ_2 with one free edge-set variable to the MSO_1 formula ψ with k vertex-set variables and k edge-set variables.
- We find a vertex cover $c_1, \ldots, c_k$.
- For each class of r-equivalent shapes, we pick the one achieving the minimal fair cost, determine the signature $U_1, \ldots, U_k$ and check whether $G \models \psi(c_1, \ldots, c_k, U_1, \ldots, U_k)$.

Similarly to Theorem 1.5, the algorithm is correct. Moreover, we do only bounded number (Proposition 4.11) of MSO_1 model checking, so the whole algorithm runs in FPT time. □

5 Open problems

The main open problem is whether the bound in Theorems 1.3 and 1.4 can be improved to $f(|\psi|, k)n^{o(k/\log k)}$ or even to $f(|\psi|, k)n^{o(k)}$.

It could also be useful to have a similar bound or an FPT algorithm for the classical version of fair deletion problems i.e. when an MSO formula is checked only for a graph after the removal of edges.

Acknowledgment

The authors would like to thank Martin Koutecký for helpful discussions.

References

[1] Tadashi Ae, Toshimasa Watanabe, and Akira Nakamura. On the NP-hardness of edge-deletion and -contraction problems. *Discrete Applied Mathematics*, 6(1):63–78, 1983.

[2] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[3] U. Bertelè and F. Brioschi. *Nonserial Dynamic Programming*. Mathematics in science and engineering. Academic Press, 1972.

[4] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, March 1990.

[5] Bruno Courcelle and Mohamed Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comput. Sci.*, 109(1&2):49–82, 1993.

[6] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[7] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[8] Rosa Enciso, Michael R. Fellows, Jiong Guo, Iyad A. Kanj, Frances A. Rosamond, and Ondrej Suchý. What Makes Equitable Connected Partition Easy. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 122–133. Springer, 2009.

[9] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. In Andreas W. M. Dress, Yinfeng Xu, and Binhai Zhu, editors, *Combinatorial Optimization and Applications, First International Conference, COCOA 2007, Xi'an, China, August 14-16, 2007, Proceedings*, volume 4616 of *Lecture Notes in Computer Science*, pages 366–377. Springer, 2007.

[10] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal Comput. Syst. Sci.*, 63(4):512–530, 2001.

[11] Petr Kolman, Bernard Lidicky, and Jean-Sébastien Sereni. On fair edge deletion problems, 2009.

[12] M. S. Krishnamoorthy and Narsingh Deo. Node-deletion np-complete problems. *SIAM Journal on Computing*, 8(4):619–625, 1979.

[13] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2011.

[14] L. Lin and S. Sahni. Fair edge deletion problems. *IEEE Trans. Comput.*, 38(5):756–761, May 1989.

[15] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

[16] Michaël Rao. MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theor. Comput. Sci.*, 377(1-3):260–267, 2007.

[17] Mihalis Yannakakis. Node- and edge-deletion NP-complete problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 253–264, 1978.

[18] Mihalis Yannakakis. Edge-deletion problems. *SIAM J. Comput.*, 10(2):297–309, 1981.

Dominating Sets on Colored Tournaments

Adam Juraszek, Jitka Novotná, Martin Töpfer

Our team decided to study the problem of dominating sets on colored graph tournaments, which was offered to us by Hans Raj Tiwary. Although related problems with various modifications, such as allowing non-transitive colorings, were studied, we could find only one article on our topic ([1]). Its author András Gyárfás conjectured that there always exists a dominating set of size of at most three for three colors.

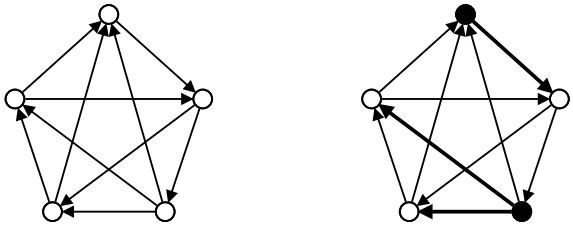
Our aim was either to prove his conjecture, or to reject it by finding a counter-example. In order to do so, we were using techniques of constraint satisfaction and procedural programming for generating hypotheses and proving them by methods of graph theory. Although we didn't solve the conjecture, we were able to create several programs and generate many tournaments satisfying various properties.

1 Problem Description

Definition 1.1. A *tournament* is a complete graph with oriented edges.

Definition 1.2. A vertex v is *dominated by* vertex u if there is an edge uv. Vertex u is then called the *dominator* of v. A *dominating set* is a set of vertices such that every vertex of the graph is a dominator or is dominated by at least one. A dominating set is said to be *minimum* if its size is the smallest possible.

An example of a tournament and one of its minimum dominating sets is shown on Figure II.1.



(a) An example of a tournament on 5 vertices

(b) A marked domination and a dominating set thereof

Figure II.1: A tournament and its dominating set

Definition 1.3. A *k-colored tournament* is a tournament with each edge having assigned one of k *colors*. A *transitively k-colored tournament* is a k-colored tournament which satisfies an additional property on its edges between every three vertices:

$$c = color(uv) = color(vw) \implies uw \in E(G) \wedge c = color(uw)$$

Conjecture 1.4. *For every k, there exists $p(k)$ such that every transitively k-colored tournament contains a dominating set of size of at most $p(k)$.*

The conjecture holds trivially for $k = 1$ because transitively colored tournaments form a linear order. In the case of $k = 2$, the tournament may not contain an oriented cycle, therefore it also forms a linear order, resulting in $p(2) = 1$.

We restrict ourselves to the problem of three colors, for which the conjecture is that $p(k) = 3$.

1.1 Example

The smallest possible transitively 3-colored tournament with minimum dominating set of size three has 7 vertices. This graph is an example of a Paley tournament. By using our constraint model we proved that there exists only one such tournament, up to isomorphism. There is a noticeable symmetry in the subgraphs induced by each color on Figure II.2.

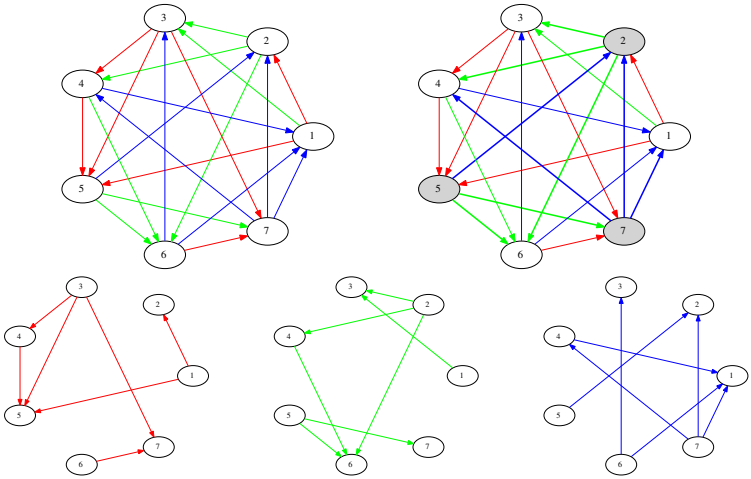


Figure II.2: A Paley tournament on 7 vertices with its transitive 3-colorings and its color-induced subgraphs

2 Simple Arguments

As we were describing the problem as a constraint model, we made a few observations.

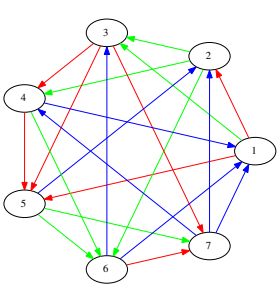
Definition 2.1. A *rainbow triangle* is an oriented cycle of length three; we also call it K_3. A rainbow triangle must be colored by three distinct colors in order to satisfy the condition of transitive coloring.

Observation 2.2. *Every transitively 3-colored tournament is either isomorphic to a linear order, or it contains a rainbow triangle.*

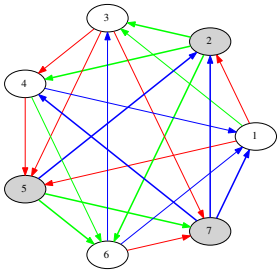
Observation 2.3. *Every transitively 3-colored tournament with a minimum dominating set of size of at least three has two edge-disjoint rainbow triangles.*

Using counting arguments we came up with the following lemma.

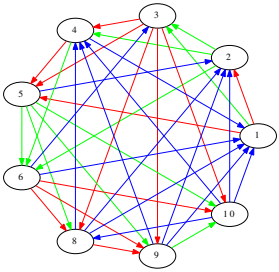
Lemma 2.4. *A toumament must have at least $2^D - 1$ vertices in order to support a minimum dominating set of size D.*



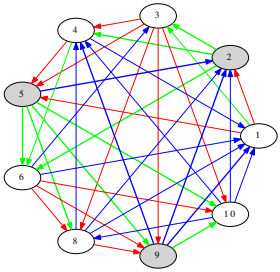
(a) $Paley_7$



(b) $Paley_7$ with a marked dominating set



(c) Substitution of 7 for K_3 in $Paley_7$



(d) $Paley_{7_{7=K_3}}$ with a marked dominating set

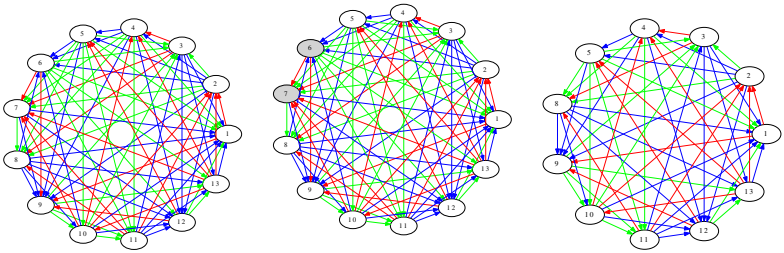
Figure II.3: Substitution of K_3 into $Paley_7$

Proof. In every tournament on N vertices there exists a vertex which dominates at least $\lceil \frac{N}{2} \rceil$ vertices. A induced subgraph of a tournament is a tournament. □

3 Operations on Tournaments

While we were looking at the structure of tournaments, we described several useful operations on them.

Definition 3.1. For the transitively k-colored tournaments G and H, a *substitution* of H into G at vertex $v \in G$ is a tournament $G_{v=H}$ which contains $|H|$ copies of vertex v connected to the rest of the graph, while all the copies of v form the graph isomorphic to H. The new graph $G_{v=H}$ is



(a) A tournament on 13 vertices

(b) ... with marked vertices which are to be removed

(c) The reduced tournament on 11 vertices

Figure II.4: Reduction of graph by removal of vertices

also a transitively k-colored tournament.

An example of substitution of K_3 into $Paley_7$ is shown on Figure II.3.

Substitution allows construction of larger graphs, but it cannot introduce new dominators, except for trivial cases of substitution of K_3 into K_3. It cannot be used for disproving the conjecture by finding a tournament with a minimum dominating set of size four.

Definition 3.2. A *reduction* of a tournament is an operation which removes some or all vertices which are not in any minimum dominating set.

Reduction does not change dominating sets in a way which we are interested in. We used reduction to look for minimal transitively k-colored tournaments with large minimum dominating sets. An example of reduction is in Figure II.4.

Lemma 3.3. *Every minimal transitively k-colored tournament is strongly connected.*

Proof. Every graph can be divided into strongly connected components which form an acyclic graph. In the case of tournaments, this graph is a path. All minimum dominating sets are in the first strongly connected component. □

Only strongly connected tournaments are interesting in our study. The following observation was important for the performance of our constraint model.

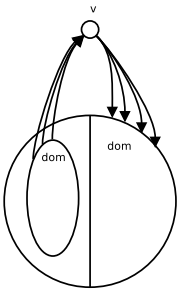


Figure II.5: Illustration of inductive construction

Observation 3.4. *A strongly connected tournament has a Hamiltonian cycle.*

4 Inductive Construction

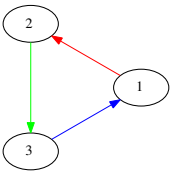
Every tournament can be constructed by the introduction of a new vertex and orientation/coloring of edges leading to all other vertices. The addition of the new vertex can decrease the number of strongly connected components and the size of the minimum dominating set arbitrarily and increase them by one at most.

Lemma 4.1. *The addition of vertex v increases the size of the minimum dominating set if v dominates all vertices in the existing minimum dominating sets and is dominated by a dominating set of the same size.*

Proof. For construction of graphs with larger minimum dominating sets $(D + 1)$, it is sufficient to study edges incident to the first strongly connected component. Vertex v must have both incoming and outgoing edges; otherwise it would become a single dominator or would be dominated by the existing dominating sets.

Following this idea we obtain the following: If v did not dominate a vertex in some minimum dominating set S, then S would still be a minimum dominating set of the graph.

The subtournament induced by vertices which were not in any minimum dominating set must have a minimum dominating set of the size of at least D. If it was smaller, then the minimum dominating set and vertex v would



$1 > 2 > 3$

$2 > 3 > 1$

$3 > 1 > 2$

Figure II.6: An example of a 2-majority graph and its defining linear orders

dominate the whole tournament, forming a dominating set of size of at most D.

The proof is illustrated in Figure II.5. □

For the purposes of obtaining a tournament with a minimum dominating set of size of at least 4, equality holds in the last part of the lemma.

5 k-majority Graphs

Definition 5.1. A *k-majority graph* is a graph induced by $2k - 1$ linear orders voting for orientations of edges.

An example illustrating the definition is shown on Figure II.6.

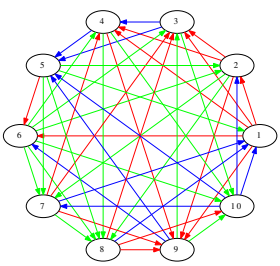
Lemma 5.2. *2-majority graphs have a minimum dominating set of size of at most three. If they do not have a single dominator, then they have a dominating set of size three on a rainbow triangle.*

Proof can be found in [2].

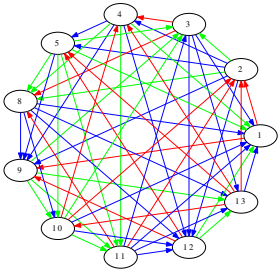
Lemma 5.3. *If all subgraphs induced by two of the three colors of a tournament are acyclic, the tournament is a 2-majority graph.*

Proof. An acyclic graph represents a partially ordered set which can be extended into a linear order. Each edge of the tournament is in two of the three subgraphs, therefore at least two subgraphs agree on the orientation of the edge. The three linear orders derived from the subgraphs are those needed by the definition of a 2-majority graph. □

Examples of tournaments with a minimum dominating set of size three which are or are not 2-majority graphs are in Figure II.7.



(a) The color-induced subgraphs are acyclic, therefore it is a 2-majority graph



(b) A color-induced subgraph contains cycle, therefore it is not a 2-majority graph

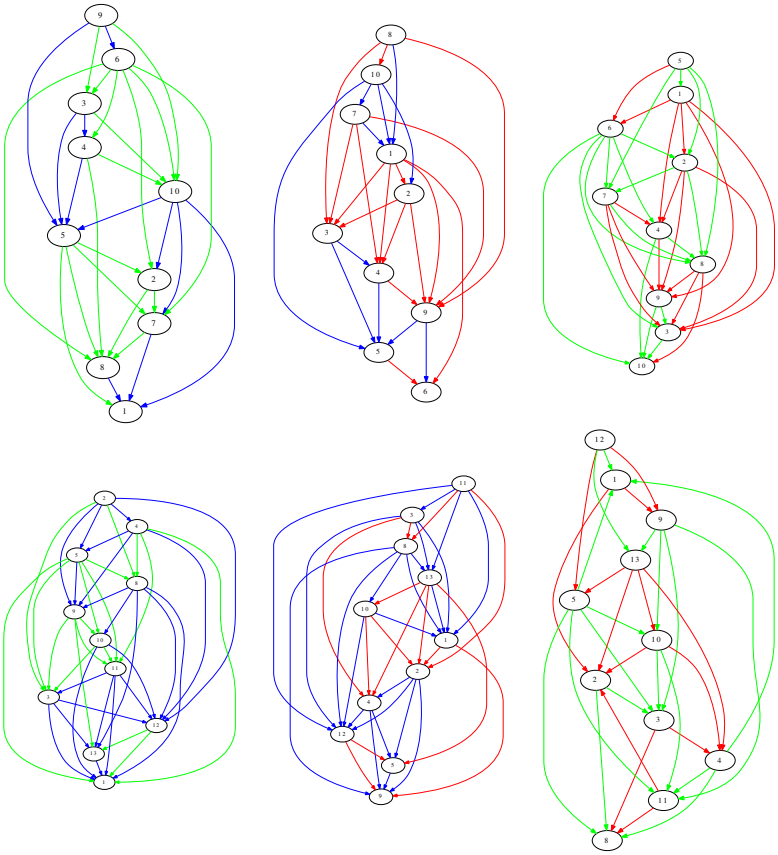


Figure II.7: Examples of tournaments in respect to 2-majority

6 Conclusion

We tried to exploit algorithmic generation of transitively 3-colored tournaments which helped us to reprove parts of [1] and to find examples of graphs supporting our proofs. However, we didn't prove nor disprove the main conjecture.

There is more to study about generalizations of 2-majority graphs as transitively 3-colored tournaments seem to be their approximation. It could also be beneficial to describe other operations on tournaments and their connection to dominating sets.

References

[1] Pálvölgyi, Dömötör and Gyárfás, András, Domination in transitive colorings of tournaments, Journal of Combinatorial Theory, Series B, 2014.

[2] Alon, Noga and Brightwell, Graham and Kierstead, Hal A and Kostochka, Alexandr V and Winkler, Peter, Dominating sets in k-majority tournaments, Journal of Combinatorial Theory, Series B, 2006.

On H-Topological Intersection Representations of Graphs

Martin Töpfer, Jan Voborník, Peter Zeman

Abstract

A lot is known about interval and chordal graphs. Chordal graph generalize interval graphs since they can be represented as intersection graphs of subtrees of some tree T. Many computational problems turn out to be solvable in polynomial time on interval graphs, but NP-complete on chordal graphs. For example, interval graphs can be recognized in linear time, however it is NP-complete to decide whether a graph can be represented as an intersection graph of subtrees of a given tree. In this paper, we investigate what happens when we fix the tree T. We show that the problem of recognition and the problem of finding a minimum dominating set can be solved in polynomial time, for $T = S_d$.

1 Introduction

The study of geometric representations of graphs is motivated by finding ways to visualize some given data efficiently. Graph drawing and visualization are important topics in graph theory. One of the most famous problems in this area is to draw a graph in the plane while minimizing the number of edge crossings.

In this paper, we study intersection representations of graphs. An intersection representation of a graph assigns a set to each vertex and intersections of those sets encode its edges. More formally, an *intersection representation* $\mathcal{R}$ of a graph G is a collection of sets $\{R_v : v \in V(G)\}$ such that $R_u \cap R_v \neq \emptyset$ if and only if $uv \in E(G)$. Many important classes of graphs are obtained by restricting the sets R_v to some specific geometrical objects.

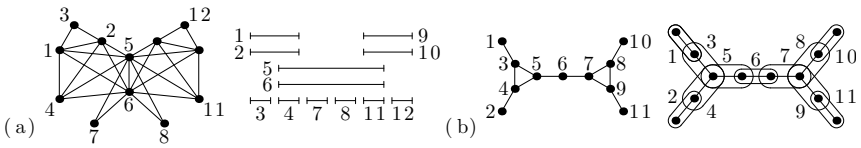


Figure III.1: (a) An interval graph and its interval representation. (b) A chordal graph and its representation as an intersection graph of subtrees of a tree.

Interval Graphs. The most studied are *interval graphs*, defined by Hajós [9]. In an *interval representation* of a graph, each set R_v is a closed interval of the real line. A graph is an interval graph if it has an interval representation; see Fig. III.1a. The set of all interval graphs will be denoted by INT.

Interval graphs have various interesting mathematical characterizations and many important computational turn out to be solvable in polynomial time on interval graphs. The recognition of interval graphs in linear time was a long-standing open problem solved by Booth and Lueker using PQ-trees [2] which can be used to describe the structure of all possible representations of an interval graph. Moreover, other important problems are solvable in linear time on intervla graphs. These include, for example, the problem of finding a minimum dominating set [4], and the graph isomorphism problem [11].

Circular-Arc Graphs. These are a natural generalization of interval graphs. Here, each set R_v corresponds to an *arc of a circle*; Fig III.1c. They are denoted by CARC.

Chordal Graphs. There are many equivalent definitions of *chordal graphs*. The most known states that a chordal graph is graph with no induced cycles of length at least 4. For our purposes we use another definition which is due to Gavril [7]. It states that a graph is chordal if and only if it is an *intersection graph of subtrees* of some tree T; see Fig III.1c. The set of all chordal graphs will be deoted by CHOR.

Compared to interval graphs, some computational problems seem to be much harder on chordal graphs. The recognition problem can be solved easily in linear time for chordal graphs using perfect elimination scheme. However, the problem of deciding, given a graph G and a tree T, whether G can be represented as an intersection graph of subtrees of T is NP-complete [10]. Finding a minimum dominating set is NP-complete [3] on chordal graphs. The graph isomorphism problem is GI-complete on chordal graphs [11], i.e.,

it is as hard as the general graph isomorphism problem.

H-graphs. In [1] Biró et al. introduced the concept of an *H-graph*. Let H be fixed graph. A graph G is an *intersection graph of* H if it is an interection graph of connected subgraphs of H, i.e., for $u, v \in V(G)$, the assigned subgraphs H_v and H_u of H share a vertex if and only if u and v are adjacent.

A subdivision H' of H is obtained by replacing each edge in H by a path of length at least one. A graph G is a *topological intersection graph* of a graph H if G is an intersection graph of some subdivision of H. We say that G is an H-graph. We denote the set of all H-graphs by H-GRAPH.

Notice that we have the following relations:

$$\mathsf{INT} = K_2\text{-GRAPH}, \ \mathsf{CARC} = K_3\text{-GRAPH}, \text{ and } \mathsf{CHOR} = \bigcup_{\text{Tree } T} T\text{-GRAPH}.$$

An open question asked by Biró et al. [1] is the complexity of the recognition problem for H-graphs.

An Infinite Hierarchy. Clearly, we have $\mathsf{INT} \subsetneq \mathsf{CHOR}$. For a tree T, we have T-GRAPH $\subsetneq$ CHOR. Therefore, we have an infinite hierarchy of graph classes between interval and chordal graphs. Since many computational problems are polynomial on interval graphs and hard on chordal graphs, an interseting question is the complexity of those on T-graphs, for a fixed tree T.

Our Results. A lot is known about interval and chordal graphs. As mentioned earlier, many important computational problems that are polynomial on interval graphs turn out to be harder on chordal graphs. On the other hand, not much is known about the classes T-GRAPH. In this paper, we investigate the parametrized complexity of recognition, representation extension, graph isomorphism, and domination on T-GRAPH with respect to the size of the tree T, denoted by $\|T\|$.

We show that the recognition of T-GRAPH, for $T = S_d$, can be solved in polynomial time, depending only on n, where n is the number of vertices of the input graph.

Theorem 1.1. *The problem* RECOGNITION(S_d-*GRAPH*) *can be solved in* $\mathcal{O}(n^4)$.

Generalizing our approach for $T = S_d$, we can get an $\mathcal{O}(n^{\|T\|})$ recognition algorithm, for a general T.

Chaplick and Stacho [5] showed that for chordal graphs of leafage at most ℓ, the representation can be constructed in time $n^{\mathcal{O}(\ell)}$. The *leafage* of a chordal graph G is the smallest number k such that there is a representation of G in which every subtree of G has at most k leaves.

Further, we give an FPT-time algorithm for the problem of finding a minimum dominating set on S_d-GRAPH.

Theorem 1.2. *For $G \in S_d$-GRAPH, the minimum dominating set of G can be found in $\mathcal{O}(n \cdot f(d))$.*

2 Recognition and Helly H-graphs

In this section, we partially answer the question asked by Biró et al. [1], i.e., the complexity of recognition of H-graphs. We show that S_d-graphs can be recognized in polynomial time, where S_d is the star of degree d, i.e., the complete bipartite graph $K_{1,d}$. Notice that S_d-graphs are *generalized split-graphs*.

A graph G is called a *Helly* H-graph when G is a topological intersection graph of H, and the collection $\mathcal{S} = \{V(H_v) : v \in V(G)\}$ satisfies the *Helly property*, i.e., for each sub-collection of $\mathcal{S}$ whose sets pairwise intersect, their common intersection is non-empty.

Suppose that G is a T-graph and let T' a subdivision of T such that G is an intersection graph of T'. Then every vertex of T' that corresponds to some vertex in T is a *branching vertex*. Let $P_{[x,y]}$ be the path from x to y. Further, we define the path $P_{(x,y]} = P_{[x,y]} - x$. Paths $P_{[x,y)}$ and $P_{(x,y)}$ are defined analogously.

The following lemma states that there exists a representation of G such that every branching point is contained in some maximal clique of G.

Lemma 2.1. *Let G be a T-graph, where T is a general tree. Every representation of G can be modified such that every branching point is contained in an intersection of the form $\bigcap_{v \in C} V(T_v)$, for a maximal clique C of G.*

Proof. Let b be a branching point, let $V_b = \{u \in V(G) : b \in V(T_u)\}$, and suppose that V_b is not a maximal clique. Let C be a maximal clique of G such that $V_b \subseteq C$ and the distance between b and the set $T_C = \bigcap\{V(T_v) : v \in C\}$ is minimal. Notice that T_C is non-empty since the Helly property is satisfied. Since the distance between b and T_C is minimal, no vertex of a path from b to T_C is contained in $T_{C'}$, for some other maximal clique C'.

Therefore, every T_v, for $v \in C \setminus V_b$, can be extended by the shortest path from b to T_C, thus obtaining a correct representation of G with $b \in T_C$. We repeat this process until there exists a branching point that is not contained in a maximal clique. □

2.1 Recognition of S_d-graphs

In this section, we deal with the case when $H = S_d$. We describe a polynomial-time algorithm for recognition of S_d-graphs.

Suppose that G is an S_d-graph and let that S'_d be a sufficiently large subdivision of S_d such that G is an intersection graph of S_d. Later, we show that subdivision that is linear in $|V(G)|$ suffices. Let b be the *central branching vertex* of S'_d, i.e., the only vertex that has degree d. Let $b_1, \ldots, b_d$ denote the remaining *leaf branching vertices*.

It is a well-known fact that chordal graphs, and therefore also S_d-graphs, have at most n maximal cliques and that they can be found in linear time. The idea of the recognition algorithm for S_d-graphs is to test, for every maximal clique C of G, if there is a representation of G with $b \in \bigcap\{T_v : v \in C\}$. According to Lemma 2.1, if G is an S_d-graph, then such representation exists.

Suppose now that G has a representation such that $b \in \bigcap\{T_v : v \in C\}$, then the connected components of $G - C$ are interval graphs and each connected component can be represented on one of the paths $P_{(b,b_i]}$, called *branches*; see Fig. III.2a and III.2c. Our algorithm first finds interval representations of the connected components of $G - C$, then every component is placed on a some branch, and finally the representations of the vertices in C are found.

Not every two components H and H' of $G - C$ can be placed on the same branch $P_{(b,b_i]}$. To see this, suppose that we want find a representation of the induced subgraph $G[C, V(H), V(H')]$. We define

$$N_C(H) = \{x \in C : xu \in E(G) \text{ for some } u \in V(H)\}$$

to be *the set of neighbors of H in C*. Clearly, the necessary condition for H and H' to be placed on the same branch, with H closer to b, is that every vertex $x \in N_C(H')$ is adjacent to every vertex of H; see Fig. III.2. We use this necessary condition to define a relation R on connected components of $G - C$:

$$(H, H') \in R \iff \forall x \in N_C(H')\ \forall v \in V(H) \colon x \in N(v). \qquad (1)$$

The goal is to define a partial ordering $\triangleright$ on the connected components of $G - C$ such that $H \triangleright H'$ if and only if H and H' can be placed on the same branch, with H closer the branching vertex b.

Let H and H' be two components such that $N_C(H) = N_C(H')$ and for every $v \in V(H)$ and $u \in V(H')$, we have $N_C(v) = N_C(H) = N_C(H') = N_C(u)$. Then $(H, H') \in R$ and also $(H', H) \in R$. Therefore, the relation R is not necessarily a partial ordering. Such components are *equivalent with respect to* C and we werite $H \sim H'$. Clearly, the relation $\sim$ is an equivalence relation.

We factorize the set of all connected components of $G - C$ by $\sim$ and obtain a set of non-equivalent connected components of $G - C$, denoted by $\mathcal{H}$. Now we use the condition (1) to define a partial ordering $\triangleright$ on $\mathcal{H}$.

The set $\mathcal{H}$ contains a representative from every equivalence class of $\sim$. First, we find a correct placement for every component $H \in \mathcal{H}$. Notice that the whole equivalence class $[H] \in \sim$ can be placed on the same branch next to H since they are all equivalent with respect to C.

Lemma 2.2. *The relation $\triangleright$ is a strict partial ordering on $\mathcal{H}$.*

Proof. Suppose that $H \triangleright H'$ and $H' \triangleright H$. Then for every $x \in N_C(H')$ and $v \in V(H)$ we have $x \in N_C(v)$. Therefore, $N_C(H') \subseteq N_C(v) \subseteq N_C(H)$. Similarly, for every $y \in N_C(H)$ and $u \in V(H')$ we have $N_C(H) \subseteq N_C(u) \subseteq N_C(H')$. We get that $N_C(H) = N_C(H')$ and also that $N_C(v) = N_C(H)$ and $N_C(u) = N_C(H')$. Thus, $H = H'$ and $\triangleright$ is antisymmetric. Clearly, $H \triangleright H'$ and $H' \triangleright H''$ implies $H \triangleright H''$, so $\triangleright$ is also transitive. □

The next lemma shows in what conditions we can place some connected components in $\mathcal{H}$ on one branch.

Lemma 2.3. *Let $H_1, \ldots, H_k \in \mathcal{H}$. Then the graph $G[C, V(H_1), \ldots, V(H_k)]$ can be represented on a branch $P_{(b,b_i]}$ if and only if $H_1, \ldots, H_k$ form a chain in $\triangleright$ and each $G[C, H_i]$ has an interval representation with C being the leftmost clique.*

Proof. Since H_i are connected components of $G - C$, their representations have to be placed on non-overlapping parts of $P_{(b,b_i]}$. Suppose that we have a representation of $G[C, V(H_1), \ldots, V(H_k)]$ on $P_{(b,b_i]}$. Assume that the components $H_1, \ldots, H_k$ are ordered such that $i < j$ if and only if H_i is closer to b than H_j. For vertex in $x \in N_C(H_i)$, the representation of v has to overlap at least a part of the representation of H_i, and therefore, it is adjacent to all vertices in H_j, for all $j < i$. Therefore, the components $H_1, \ldots, H_k$ form

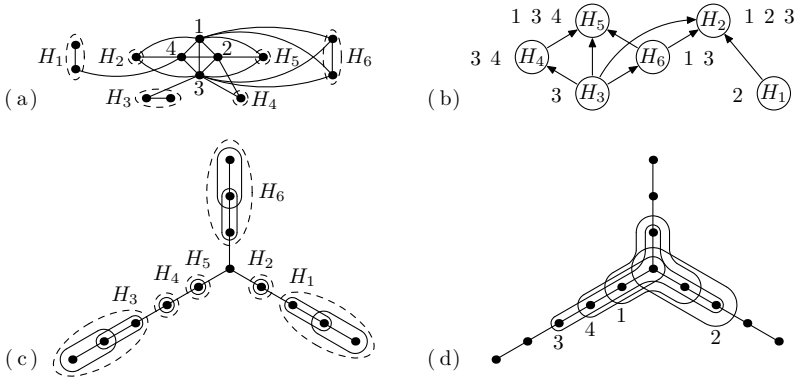


Figure III.2: An example of a construction of a representation.

a chain in $\triangleright$. Clearly, every $G[C, H_i]$ has an interval representation with C being the leftmost clique.

For the converse, assume that $H_1, \ldots, H_k$ form a chain in $\triangleright$ and every $G[C, H_i]$ has an interval representation with C being the leftmost clique. Since $H_j \triangleright H_i$, for $j < i$, all vertices in $N_C(H_i)$ are adjacent to every vertex in H_j. To obtain a representation of $G[C, V(H_1), \ldots, V(H_k)]$, we first place the represenations of all H_i on $P_{(b,b_i]}$ according to $\triangleright$, with H_1 being closest to b. All $G[C, V(H_i)]$ have an interval representation. We first place the representations of the vertices in $N_C(H_k)$. From the definition of $\triangleright$ we have that all those vertices are adjecent to all vertices in $H_1, \ldots, H_{k-1}$. For $i = k-1, \ldots, 1$, we place the representations of the vertices in $N_C(H_i) \setminus N_C(H_{i+1}) \cup \cdots \cup N_C(H_k)$. Again, from the definition of $\triangleright$ we have that those vertices are adjacent to all vertices in $H_1, \ldots, H_{i-1}$. Finally, we obtain a correct representation of $G[C, V(H_1), \ldots, V(H_k)]$. □

The following theorem generalizes the characterization of interval graphs which is due to Fulkerson and Gross [6].

Theorem 2.4. *Let C a maximal clique of G, and let $\mathcal{H}$ be the set of connected components of $G - C$ that are non-equivalent with respect to C. Then G is an S_d-graph with $b \in \bigcap\{T_v : v \in C\}$ if and only if the following hold:*

(i) For every $H \in \mathcal{H}$, the induced subgraph $G[C, H]$ has an interval representation with C being the leftmost clique.

(ii) The partial order $\triangleright$ on $\mathcal{H}$ has a chain cover of size d.

Proof. Suppose that G is an S_d-graph such that $b \in \bigcap\{T_v : v \in C\}$. The representation of a connected component $H \in \mathcal{H}$ can not pass through the branching point b since otherwise C would not be a maximal clique. Clearly, the condition (i) is satisfied. Moreover, the representations of every two components in $\mathcal{H}$ have to be placed on non-overlapping parts of S_d. By Lemma 2.3 we have that the components placed on some branch of S_d form a chain in $\triangleright$. Therefore, the partial order $\triangleright$ has a chain cover of size d and the condition (ii) is satisfied; see Fig. III.2b.

For the converse, suppose that the conditions (i) and (ii) are satisfied. We place the components in $\mathcal{H}$ on the $P_{(b,b_1]}, \ldots, P_{(b,b_d]}$ according to the chain cover of $\triangleright$ which has size d. By Lemma 2.3, for every chain $H_1, \ldots, H_k$ in $\triangleright$, we can find an interval representation of the graph $G[C, V(H_1), \ldots, V(H_k)]$. We combine the representations of the d chains. Further, the connected components that are equivalent with respect to C, i.e., belong to the same equivalence class in $\sim$, can be easily placed next to its representative by subdividing the branches. Thus, we obtain an representation of G. $\square$

Based on the previous discussion, we give an algorithm for recognizing S_d-graphs; see Algorithm 1. It remains to show that its running time is polynomial and that it does not depend on the parameter d.

Algorithm 1: RECOGNITION(S_d-GRAPH)

Input: A graph G and a tree T.

Output: An intersection representation of G on a subdivision of T if it exists.

1: Find all maximal cliques $\mathcal{C}$ of G.
2: If G is an S_d-graph, then by Lemma 2.1 there exists a representation $\mathcal{R}$ of G with $b \in \bigcap\{R_v : v \in C\}$, for some $C \in \mathcal{C}$.
3: **for** a maximal clique $C \in \mathcal{C}$ **do**
4: Try to find a representation with C in the branching point:
5: Find the equivalence classes of $\sim$ and pick a representative for each equivalence class. Let $\mathcal{H}$ be the set of these representatives.
6: **for** a connected component $H \in \mathcal{H}$ **do**
7: Find an interval representation of $G[C, H]$.
8: If there is a chain cover of $\triangleright$ of size at most d, then construct a representation as desribed in the proof of Theorem 2.4.

Theorem 1.1. Every chordal graph has at most n maximal cliques, where n is the number of vertices. All maximal cliques of a chordal graph can be found in linear time. For every clique, our algorithm tries to find an S_d-representation with this clique placed on the central branching point. Construction of such representation takes $\mathcal{O}(n^3)$ steps since interval graph recognition can be done in linear time and clique-cover can be found in $\mathcal{O}(n^3)$ time for comparability graphs [8]. Therefore, the overall complexity is $\mathcal{O}(n^4)$. □

The following proposition justifies our approach for recognizing S_d-graphs. It says that every partially ordered set with chain cover of size d can be encoded in an S_d-graph. Therefore, to recognize S_d-graphs, one somehow needs to solve the problem of finding a chain cover of a given size in a partially ordered set. Here, we omit the proof.

Proposition 2.5. *For every partially ordered set $(X, \leq)$ that has a chain cover of size d there exists an S_d-graph G and a maximal clique C of G such that the partial order $\triangleright$ defined on the components of $G \setminus C$ is isomorphic to $\leq$.*

3 Dominating Set

In this section, we discuss the problem of finding minimum dominating set. First, we recall the algorithm for finding a minimum dominating set in an interval graph. Then we give an FPT-time algorithm for finding a minimum dominating set in an S_d-graph.

3.1 Minimum Dominating Set for Interval Graphs

Suppose that we have an interval representation of a graph G and let $C_1, \ldots, C_k$ be the left-to-right ordering of the maximal cliques. For every vertex $v \in C_1$ there exists ℓ such that $v \in C_\ell$, but $v \notin C_{\ell+1}$ We pick the vertex $v \in C_1$ such that ℓ is maximal. We add v to the dominating set and continue smilarly for $C_{\ell+1}$; see Algorithm 2.

Lemma 3.1. *The described algorithm finds a minimum dominating set of an interval graph G.*

Algorithm 2: DominatingSet(INT)

Input: An interval representation $\mathcal{R}$ of a graph G.
Output: A minimum dominating set of G.

```
1: D ← ∅, i ← 1.
2: Let C_1, ..., C_k be the left-to-right ordering of the maximal cliques in R.
3: while i ≤ k do
4:     Choose v ∈ C_i with maximal ℓ such that v ∈ C_ℓ, but v ∉ C_{ℓ+1}.
5:     D ← D ∪ {v}, i ← ℓ + 1.
6: return D
```

Proof. Suppose that at some step of the algorithm we choose $v \in C_i$ with maximal ℓ such that $v \in C_\ell$, but $v \notin C_{\ell+1}$. Let D' be a minimum dominating set for the induced subgraph $G[C_{\ell+1}, \ldots, C_k]$. The set D' does not dominate the maximal clique C_i since then ℓ would not be maximal and this would contradict the choice of v. Therefore, the minimum dominating set of $G[C_i, \ldots, C_k]$ has size at least $|D'| + 1$. It follows that $D' \cup \{v\}$ is a minimum dominating set of $G[C_i, \ldots, C_k]$ with size $|D'| + 1$. We have shown that for an arbitrary minimum dominating set of $G[C_{\ell+1}, \ldots, C_k]$, the vertex v is an optimal choice. □

Lemma 3.2. *The problem* DominatingSet(INT) *can be solved in linear time.*

Proof. An interval representation of an interval graph can be found in linear time [2]. The number of maximal cliques in an interval graph is at most n. In the procedure, described in Algorithm 2, every maximal clique in the left-to-right ordering $C_1, \ldots, C_k$ is checked at most constant number of times. □

3.2 Minimum Dominating Set for S_d-graphs

Here, we prove Theorem 1.2. First, we prove the following lemma.

Lemma 3.3. *Let $G = (V, E)$ be an interval graph and let $C_1, \ldots, C_k$ be the left-to-right ordering of the maximal cliques in an interval representation of G. For every $x \in C_1$, a minimum dominating set of G containing x can be found in linear time.*

Proof. We construct a new graph $G' = (V', E')$ where $V' = V \cup \{u\}$ and $E' = E \cup \{u, x\}$. Clearly, G' is an interval graph and there is an interval representation with $\{u, x\} = C_0, C_1, \ldots, C_k$ being the left-to-right ordering of the maximamal cliques in this representation. The procedure in Algorithm 2 always picks the vertex $x \in C_0$ and finds a minimum dominating set D of G'. The set D is a minimum dominating set of G containing the vertex x. □

Theorem 1.2. Let G be an S_d-graph and let S be a subdivision of the star S_d such that G is an intersection graph of S. Let b be the central branching vertex of S and let $b_1, \ldots, b_d$ be the remaining branching vertices. Suppose that we have an intersection representation $\mathcal{R}$ of G on S such that $b \in \bigcap\{R_v : v \in C\}$, for some maximal clique C of G (by Lemma 2.1 we know such representation exists). Let $C_{i,1}, \ldots, C_{i,k_i}$ be the maximal cliques of G as they appear on the branch $P_{(b,b_i]}$, for $i = 1, \ldots, d$.

For every $G_i = G[C, C_{i,1}, \ldots, C_{i,k_i}]$, we use the procedure described in Algorithm 2 to find the size d_i of the minimum dominating set in G_i. Let B_i be the set of vertices of C that can appear in a minimum dominating set of G_i. By Lemma 3.3, a minimum dominating set D_i^x containing a vertex $x \in C$ can be found in linear time. We have $x \in B_i$ if and only if $|D_i^x| = d_i$. Therefore, every B_i can be found in polynomial time. Let $\mathcal{B} = \{B_1, \ldots, B_d\}$.

If B_i is empty, then every minimum dominating set of G_i does not contain a vertex from C. For G_i, we can choose an arbitrary dominating set D_i and this choice is optimal. If every B_i is empty, then the minimum dominating set of G is the union $D_1 \cup \cdots \cup D_d \cup \{x\}$, for an arbitrary $x \in C$.

Let us assume now that the B_i's are nonempty (every branch with an empty B_i can be simply ingored). Let H be a subset of C such that $H \cap B_i$ is not empty, for every $i = 1, \ldots, d$, and $|H|$ is smallest possible. For every branch $P_{(b,b_i]}$, we pick a minimum dominating set D_i of G_i containing an arbitrary vertex $x_i \in H \cap B_i$. The minimum dominating of G is then the union $D_1 \cup \cdots \cup D_d$. It remains to prove that we can find the set H in time that only depends on the parameter d.

For every $x \in C$, there are at most 2^d possibilities to which sets from $\mathcal{B}$ it can belong. We consider a subset A of $\bigcup \mathcal{B} \subseteq C$ such that $|A| \leq 2^d$ and no two vertices in A belong to exactly the same B_i's. The set A, can be clearly found in polynomial time. To find H, it suffices to check for every subset of A whether it intersects the B_i's and pick the smallest one. The number of subsets of A is at most 2^{2^d}. □

References

[1] Miklós Biro, Mihály Hujter, and Zs Tuza. Precoloring extension. i. interval graphs. *Discrete Mathematics*, 100(1):267–279, 1992.

[2] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms. *J. Comput. System Sci.*, 13:335–379, 1976.

[3] Kellogg S Booth and J Howard Johnson. Dominating sets in chordal graphs. *SIAM Journal on Computing*, 11(1):191–199, 1982.

[4] Maw-Shang Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM Journal on computing*, 27(6):1671–1694, 1998.

[5] Steven Chaplick and Juraj Stacho. The vertex leafage of chordal graphs. *Discrete Applied Mathematics*, 168:14–25, 2014.

[6] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pac. J. Math.*, 15:835–855, 1965.

[7] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

[8] Martin Charles Golumbic. The complexity of comparability graph recognition and coloring. *Computing*, 18(3):199–208, 1977.

[9] G. Hajós. Über eine Art von Graphen. *Internationale Mathematische Nachrichten*, 11:65, 1957.

[10] P. Klavík, J. Kratochvíl, Y. Otachi, and T. Saitoh. Extending partial representations of subclasses of chordal graphs. *Theoretical Computer Science*, 576:85–101, 2015.

[11] G. S. Lueker and K. S. Booth. A linear time algorithm for deciding interval graph isomorphism. *Journal of the ACM (JACM)*, 26(2):183–195, 1979.

Packing chromatic numbers

Transcribed by Hadley Black and Asa Goodwillie
from a lecture by Dr. Jirka Fiala

1 Introduction

The notions of coloring and of the chromatic number are well studied graph-theoretic concepts. The definition of packing coloring (and the associated packing chromatic number of a graph) extend these concepts by modifying the label restrictions to depend on the specific label used. Packing colorings arose originally through applications to television broadcast networks, where broadcast locations for different frequencies must be chosen to avoid interference.

2 Definitions

Definition 2.1. A *coloring* or *labeling* of a graph $G = (V, E)$ is a function $f : V \to S$ for some finite set S.

A coloring is called a *proper coloring* if it satisfies the following restriction: $(u, v) \in E \implies f(u) \neq f(v)$. This is equivalent to the following: for distinct vertices u and v, $f(u) = f(v) \implies \text{dist}(u, v) > 1$, where $\text{dist} : V \times V \to \mathbb{N}$ is the usual notion of graph distance, denoting the length of the shortest path between two vertices. We define a slightly more restrictive version of this second formulation as follows.

Definition 2.2. A *packing coloring* is a coloring $f : V \to S$ for some finite subset $S \subset \mathbb{N}$ such that for distinct vertices u and v, $f(u) = f(v) = i$ implies $\text{dist}(u, v) > i$. The *packing chromatic number* of a graph G, $\chi_p(G)$, is the minimal number of colors needed for a packing coloring of G (i.e., the smallest possible value of $|S|$).

We want to find an upper bound on the packing chromatic numbers for various classes of graphs.

3 Some simple proofs

Proposition 3.1. *Every path admits a packing coloring using three colors.*

Proof. A repeating labeling $(1, 2, 1, 3)$ suffices, since the vertices labeled 1 are separated by the others, and the vertices labeled 2 (resp. 3) are four vertices from each other. □

It is clear that for every path of length at least four, two colors are not sufficient, so the packing chromatic number of a path of length at least four is exactly three.

Proposition 3.2. *Every cycle admits a packing coloring using four colors.*

Proof. Let $G = (V, E)$ be an n-cycle (i.e, let $V = \{1, \ldots, n\}$ and let $E = \{(1, 2), (2, 3), \ldots, (n-1, n), (n, 1)\}$). Let $n = |V|$. We have four cases, depending on the remainder of n modulo 4.

- If $n \equiv 0 \pmod 4$, then we may simply use the path coloring

$$f(v) = \begin{cases} 1 & \text{if } v \equiv 0 \pmod 2 \\ 2 & \text{if } v \equiv 1 \pmod 4 \\ 3 & \text{if } v \equiv 3 \pmod 4. \end{cases}$$

- If $n \equiv 1 \pmod 4$, then we use the same coloring as above for all vertices but the last (nth) vertex, and set $f(n) = 4$.
- If $n \equiv 2 \pmod 4$, we use the above (repeating) coloring except for the last two vertices, and set $f(n-1) = 1$ and $f(n) = 4$.
- Finally, if $n \equiv 3 \pmod 4$, we use the above coloring for all vertices but the last three, and set $f(n-2) = 1, f(n-1) = 2$, and $f(n) = 4$.

□

It is clear that for cycles of lengths congruent to 0 modulo 4, three colors are both necessary and sufficient. For a cycle of length $n \equiv 1, 2,$ or $3 \pmod 4$, four colors are necessary as long as n is greater than four.

4 Known results

The following result on the chromatic numbers of 4-regular trees – that is, infinite trees in which every vertex has degree exactly four – is due to Christian Sloper.

Theorem 4.1. *There is no upper bound on the packing chromatic numbers of 4-regular trees.*

The next result was obtained with the aid of computer search by Danilo Korže and Aleksander Vesel.

Theorem 4.2. *Every hexagonal grid graph has a packing chromatic number of no more than 7.*

Theorem 4.3. *The set of packing chromatic numbers of triangular grid graphs is unbounded.*

The proof of this last result is by a density argument, i.e, by considering upper bounds on the densities of nodes labeled 1, 2, etc., in the graph. The proof is complicated, so we will not attempt to reproduce it here. Similarly intricate density arguments also provide a lower bound of 10 on the upper bound of the set of packing chromatic numbers of square grid graphs. For square grid graphs, an upper bound of 25 is also known. More recently, further computer-aided results have reduced this gap to 15 and 17.

5 Open questions

It is unknown whether or not the packing chromatic numbers of planar graphs of maximum degree 3 are bounded. This is an important open question in the study of packing colorings.

Definition 5.1. A graph $G = (V, E)$ is called *outer planar* if it is possible to embed G in the plane such that all vertices of G are incident on the outer (unbounded) face.

A two-connected outer planar graph is simply a cycle with some chords. In general, an outer planar graph is essentially a tree-like structure where each "node" of the tree is a two-connected outer planar graph. Dr. Fiala gave us the following problem as a potentially interesting and viable open

question: Are the packing chromatic numbers of two-connected outer planar graphs of maximum degree 3 bounded?

A two-connected outer planar graph is a cycle with some chords, so a two-connected outer planar graph of maximum degree 3 is just a cycle with chords where no two chords share an endpoint. Since we have already know that the chromatic number of a cycle (without chords) is at most 4, we decided to begin by considering the "next case up", i.e., a cycle with a single chord. It turns out that four colors suffice here as well, although the proof is somewhat longer (though not fundamentally any more difficult).

Theorem 5.2. *A cycle with a single chord has packing chromatic number less than or equal to 4.*

Proof. Let u and v be the vertices of the cycle connected by the chord. Consider the two paths P and Q along the cycle from u to v. Let n_1 be the number of vertices on one of them, and let n_2 be the number of vertices on the other. We have ten cases (up to switching P and Q and consequently n_1 and n_2), depending on the remainders of n_1 and n_2 modulo 4. The packing colorings for these ten cases are illustrated at the end of this report. Note that in every case the entirety of the cycle is colored with the repeating $(1, 2, 1, 3)$ pattern, with the exception of a small neighborhood around v.

Case 1: If $n_1 \equiv n_2 \equiv 0 \pmod 4$, we set $f(u) = 1$ and $f(v) = 4$; along P, we repeat the sequence of labels $(3, 1, 2, 1)$; and along Q, we repeat the sequence of labels $(2, 1, 3, 1)$.

Case 2: If $n_1 \equiv n_2 \equiv 1 \pmod 4$, we set $f(u) = 2$ and $f(v) = 4$, and along both P and Q, we repeat the sequence of labels $(1, 3, 1, 2)$, followed by a trailing 1.

Case 3: If $n_1 \equiv n_2 \equiv 2 \pmod 4$, we set $f(u) = 1$ and $f(v) = 4$; along P, we repeat the sequence of labels $(3, 1, 2, 1)$, followed by a trailing $(3, 1)$; and along Q, we repeat the sequence of labels $(2, 1, 3, 1)$, followed by a trailing $(2, 1)$.

Case 4: If $n_1 \equiv n_2 \equiv 3 \pmod 4$, we set $f(u) = 3$ and $f(v) = 4$, and along both P and Q, we repeat the sequence of labels $(1, 2, 1, 3)$, followed by a trailing $(1, 2, 1)$.

Case 5: If $n_1 \equiv 1$ and $n_2 \equiv 2 \pmod 4$, we set $f(u) = 1$ and $f(v) = 4$; along P, we repeat the sequence of labels $(2, 1, 3, 1)$, followed by a trailing 2; and along Q, we repeat the sequence of labels $(3, 1, 2, 1)$, followed by a trailing $(3, 1)$.

Case 6: If $n_1 \equiv 1$ and $n_2 \equiv 3 \pmod 4$, we set $f(u) = 1$ and $f(v) = 4$; along P, we repeat the sequence of labels $(2, 1, 3, 1)$, followed by a trailing

2; and along Q, we repeat the sequence of labels $(3, 1, 2, 1)$, followed by a trailing $(3, 2, 1)$.

Case 7: If $n_1 \equiv 1$ and $n_2 \equiv 0 \pmod 4$, we set $f(u) = 1$ and $f(v) = 4$; along P, we repeat the sequence of labels $(2, 1, 3, 1)$, followed by a trailing 2; and along Q, we repeat the sequence of labels $(3, 1, 2, 1)$.

Case 8: If $n_1 \equiv 2$ and $n_2 \equiv 3 \pmod 4$, we set $f(u) = 1$ and $f(v) = 4$; along P, we repeat the sequence of labels $(2, 1, 3, 1)$, followed by a trailing $(2, 1)$; and along Q, we repeat the sequence of labels $(3, 1, 2, 1)$, followed by a trailing $(3, 2, 1)$.

Case 9: If $n_1 \equiv 2$ and $n_2 \equiv 0 \pmod 4$, we set $f(u) = 1$ and $f(v) = 4$; along P, we repeat the sequence of labels $(3, 1, 2, 1)$, followed by a trailing $(3, 1)$; and along Q, we repeat the sequence of labels $(2, 1, 3, 1)$.

Case 10: If $n_1 \equiv 3$ and $n_2 \equiv 0 \pmod 4$, we set $f(u) = 1$ and $f(v) = 4$; along P, we repeat the sequence of labels $(3, 1, 2, 1)$; and along Q, we repeat the sequence of labels $(2, 1, 3, 1)$. □

Note that, though we do not give the details here, it is possible to extend this to a four packing coloring on a cycle with any number of non-intersecting chords, provided the endpoints of any two chords are sufficiently far apart. Next, one might wonder about the chromatic number of outer planar, max degree 3 graphs with higher density. One such example is a ladder graph L_n.

Theorem 5.3. $\chi_p(L_n) \leq 5$.

Proof. It is sufficient to label the top path with the repeating pattern $(1, 2, 1, 3, 1, 5)$ and the bottom with $(3, 1, 4, 1, 2, 1)$. □

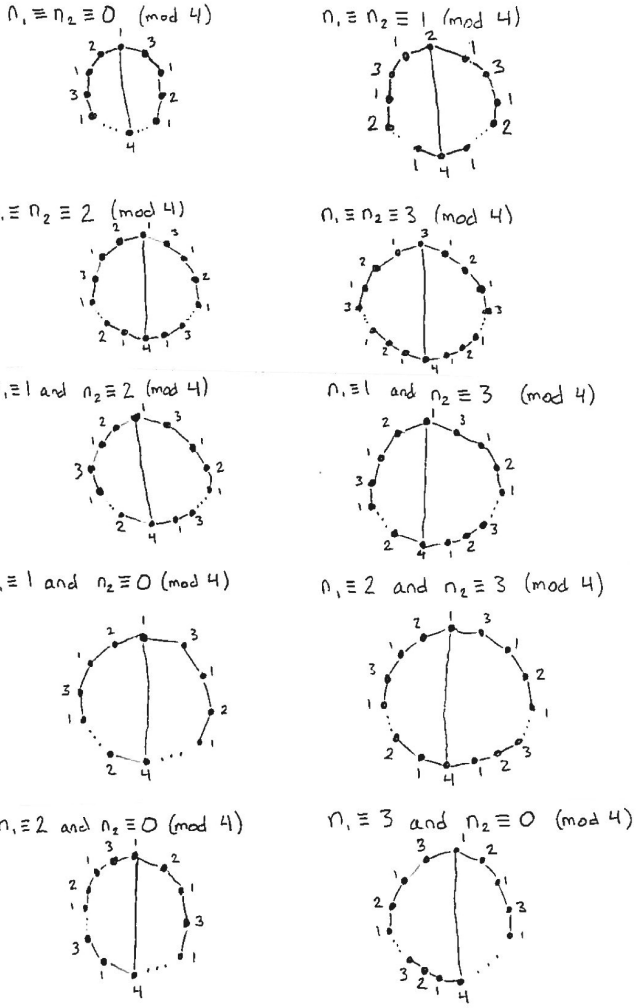
The ladder graph has a large number of edges compared to other outer planar, maximum degree 3 graphs. Thus Theorem 5 gives some direction towards a tight upper bound on the packing chromatic number of this family of graphs.

References

[1] Brešar, Klavžar Rall, On the Packing Chromatic Number of Cartesian Products, Hexagonal Lattice and Trees. 2007.

[2] Wayne Goddard, Sandra M. Hedetniemi, Stephen T. Hedetniemi. Broadcast Chromatic Numbers of Graphs. Clemson University. 2008.

[3] Sloper. Broadcast-Coloring of Trees. 2002.



Two Examples of Combinatorial Reciprocity

Transcribed by Kevin Sun and Andrew Wells
from a lecture by Dr. Andrew Goodall

1 Introduction

The theory of combinatorial reciprocity refers to the study of the relationship between combinatorial classes that share a counting function. In this article, we describe two illustrative examples of combinatorial reciprocity and provide some known results.

The two counting functions considered in this paper are those associated with the binomial coefficients, and the chromatic polynomial of graphs. At first glance, each of these functions seems to be only defined for nonnegative integers, but we shall see that evaluating these functions on negative integers yields interesting, meaningful results.

2 Binomial Coefficients

2.1 The Standard Interpretation

The binomial coefficient $\binom{n}{k}$ is usually defined as the number of ways to select k objects from a collection of n objects without order and without repetition. Therefore, the inputs n and k are typically positive integers, with $k \leq n$. We may compute various terms of the sequence by using the following well-known recurrence relation, known as Pascal's rule:

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}.$$

Furthermore, we know that

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k!},$$

so we may also interpret $\binom{n}{k}$ as a polynomial in n of degree k. From this perspective, n no longer seems to be restricted to the positive integers. A quick computation allows us to see that

$$\begin{aligned}\binom{-n}{k} &= \frac{(-n)(-n-1)\cdots(-n-k+1)}{k!} \\ &= \frac{(-1)^k n(n+1)\cdots(n+k-1)}{k!}.\end{aligned}$$

However, confusion can arise when one considers the definition given above. How can 2 elements be selected from a set of size -5, as seen in $\binom{-5}{2}$?

2.2 Choosing from a Set with Negative Size

Without an abundance of motivation, we examine the formula for counting the number of multisets of size k from a set of size n, that is, subsets of the original set, with repetition permitted. We denote this number by $\left(\!\binom{n}{k}\!\right)$. Using the classical "stars and bars" argument, we can conclude that $\left(\!\binom{n}{k}\!\right) = \binom{n+k-1}{k}$. Surprisingly, this is equal to $\binom{-n}{k}$, up to a difference in sign. Therefore, up to a difference in sign, $\binom{-n}{k}$ counts the number of multisets of size k that can be made from a set of size n.

3 The Chromatic Polynomial

3.1 The Standard Interpretation

We now continue with our second example, which concerns the chromatic polynomial of graphs. Before we give a definition of the chromatic polynomial, we must define a *k-coloring* of a graph.

Definition 3.1. A (proper) *k-coloring* of a graph G is a function $f : V(G) \to \{1, 2, \ldots, k\}$ such that for each edge $\{u, v\}$ of G, $f(u) \neq f(v)$.

Definition 3.2. For a graph G and an integer k, the *chromatic polynomial* $P(G, k)$ counts the number of proper k-colorings of G.

It is not too difficult to show that the chromatic polynomial of a graph is indeed a polynomial.

From its definition, the chromatic polynomial $P(G, k)$ of a graph G for some integer k seems to permit only nonnegative integers k. We recall that $\binom{-n}{k}$ counts the number of multisets of size k from a set of size n. So naturally, we might ask what $P(G, -k)$ counts. However, before we begin, we shall introduce the method of deletion-contraction.

3.2 The Method of Deletion-Contraction

Definition 3.3. The *deletion* of an edge e in a graph G produces the graph G' with the same vertex set as G and edge set $E(G)\backslash\{e\}$. In this case, we write $G' = G\backslash e$.

Definition 3.4. The *contraction* of an edge $\{u, v\}$ in a graph G produces the graph G'' with vertex set $V(G)\backslash\{u, v\}$ along with a new vertex w, which is adjacent to all the vertices of G adjacent to either u or v (or both). In this case, we write $G'' = G/e$.

An important recurrence relation, analogous to Pascal's rule, is the following:

$$P(G, k) = P(G\backslash e, k) - P(G/e, k),$$

which holds for any edge e of the graph G. This can be seen by considering all proper k-colorings of, for a given graph G, the graph $G\backslash e$ for some edge $e = \{u, v\}$ of G. A proper coloring of $G\backslash e$ that assigns u and v to different colors is also a proper coloring for G. However, if u and v are assigned the same color in $G\backslash e$, then we would not have a proper coloring for G; as a result, we identify u and v as one vertex, forming G/e, and subtract the number of proper k-colorings of G/e from $P(G\backslash e, k)$.

Using this recurrence relation to compute the chromatic polynomial of a graph is known as the method of deletion-contraction. At this point, we can consider the resulting polynomial when a negative value of k is plugged into $P(G, k)$ for some graph G.

3.3 Coloring with a Negative Number of Colors

Let P_n denote the graph that is a path on n vertices and C_n denote the graph that is a cycle on n vertices.

The chromatic polynomial of P_n and C_n can be computed using the deletion-contraction method, result in the following:

$$P(P_n, k) = k(k-1)^{n-1}$$

and

$$P(C_n, k) = (k-1)^n + (-1)^n(k-1).$$

Letting $k = -1$ yields

$$\begin{aligned} P(P_n, -1) &= (-1)(-2)^{n-1} \\ &= (-1)^n 2^n \end{aligned}$$

and

$$\begin{aligned} P(C_n, -1) &= (-2)^n + (-1)^n(-2) \\ &= (-1)^n(2^n - 2). \end{aligned}$$

Ignoring the sign (as we did for binomial coefficients), the obvious interpretation of this result is that it counts the number of acyclic orientations of the edges. A path has no cycles, so any one of the 2^n orientations is valid. A cycle has exactly one cycle, which can be traversed 2 ways: clockwise or counter-clockwise, and hence the subtraction of 2 from 2^n.

Before we generalize this observation, we define what it means for an orientation ρ and a k-coloring c of G to be *compatible*.

Definition 3.5. The pair (ρ, c) is a *compatible coloring* if and only if for every directed edge $u \to v$ in G, we have $c(u) \geq c(v)$.

Now we can provide the following theorem:

Theorem 3.6. *Let G be a finite graph on n nodes and $P(G, k)$ be its chromatic polynomial. Then $(-1)^n P(G, -k)$ equals the number of compatible pairs (ρ, c) where c is a k-coloring and ρ is an acyclic orientation. In particular, $(-1)^n P(G, -1)$ equals the number of acylic orientations of G.*[1]

4 Conclusion

The theory of combinatorial reciprocity enables us to extend functions beyond their initially-conceivable domain. Our two examples show that this can yield interesting and combinatorially interpretable results.

[1] Matthias Beck and Raman Sanyal. *Combinatorial Reciprocity Theorems: An Invitation to Enumerative Geometric Combinatorics*. July 12, 2015. math.sfsu.edu/beck/crt.html. pg. 6

Graph Coloring

Transcribed by Linda Cook and Rayanne Luke
from a lecture by prof. John Gimbel

1 Graph Coloring

Colorings are defined on graphs in various ways to break a vertex set up so that each subset is simple and easy to grasp. Although there are many other possible ways of coloring, we often examine a traditional coloring where no adjacent vertices are assigned the same color. Thus, each color class induces an empty graph. More generally, a coloring is a partition of the vertex set.

Definition 1.1. The **chromatic number** of a graph G is the smallest number of colors needed to assign every adjacent vertex a different color. The chromatic number of G is denoted by $\chi(G)$.

The chromatic number of a complete graph with n vertices is $\chi(K_n) = n$. Unless otherwise stated, graphs are assumed to be simple.

Example 1.2. ***The Hadwiger-Nelson Problem (1950, unsolved).*** *Consider an infinite graph G where the vertices are all the points in the Cartesian plane and two vertices are considered adjacent if their distance is exactly one. What is the chromatic number of this graph?*

Theorem 1.3. *The chromatic number in the Hadwiger-Nelson Problem is between 4 and 7 inclusive.*

Proof. We will show that the graph cannot be colored with 3 or less colors. The upper bound of 7 was demonstrated through the discovery of a number of possible colorings using only 7 colors.

We provide a counter example to a coloring of the Hadwidger-Nelson problem with three colors (Figure VI.1). Let ABC and BCD be equilateral triangles with sides of length 1. Clearly A, B, and C must be pairwise

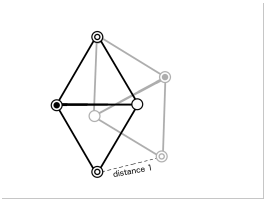


Figure VI.1: Counter example for a coloring of the Hadwidger-Nelson problem with 3 colors. All edges are of length 1.

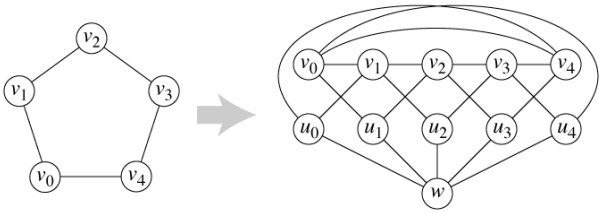


Figure VI.2: In this example of a Mycielskian construction the Grötzsch graph is created from a 5-cycle.[1]

different colors and B, C, D must be different as well. So A and D must be the same color if we can only use 3 colors. Now let AB'C' and B'C'D be a copy of the original figure that shares vertex A. By the same argument as before A and D' must be the same color, so D and D' are the same color. Now rotate the copied figure until D and D' are at distance 1 and we have a contradiction.

□

Chromatic Number of Triangle-Free Graphs

Theorem 1.4. *For all $n \in \mathbb{N}$, there exists a triangle-free graph G that has chromatic number n.*

Proof. The chromatic number of a triangle-free graph G can be made to be arbitrarily large. Given a triangle free graph $G = (V, E)$ with chromatic

[1] D. Eppstein. Grötzsch graph as a Mycielskian. Self-made; originally uploaded as en:Image:Groetzsch-as-Mycielski.png to English Wikipedia. Licensed under Public Domain via Wikimedia Commons.

number n, we can create a triangle-free graph G' with chromatic number $n+1$ through a Mycielskian construction (Figure VI.2). The process is as follows: Copy V to create U and add edges between every $u \in U$ and the neighbors of the corresponding vertex $v \in V$. Finally add a vertex w and connect it to every vertex in U [5].

It is easy to see that if G was triangle-free the new graph G' will also be triangle-free. We can color the graph G' with $n+1$ colors by copying the coloring of V to U and adding a new color for vertex w. To prove that we need $n+1$ colors to color G', we will assume, for the sake of contradiction, that we can use n colors.

Suppose vertex w is colored with color n. Then, U may only be colored by the colors $\{1, \ldots, n-1\}$ since each vertex in U is connected to w. Our goal now is to create a new proper coloring of G' such that the vertices in V contain no color n. We will then have a coloring of V, the same as G, with $n-1$ colors, which will produce a contradiction.

Copy the coloring of those vertices in V that correspond to a $v \in V$ that are not colored with color n to their corresponding twins in U. By construction, this gives another proper coloring of the graph G'. Now recolor every vertex in V that is colored n with the color of the corresponding vertex in U. Again, this is a proper coloring of G'.

Consider a $v \in V$ that was colored n. Let K be the neighborhood of v in the set V and K' be the set of corresponding vertices to K. Note the neighborhood of v in G' is $K \cup K'$ none of which are colored n. By our first recoloring, the corresponding vertices in K and K' have the same colors. The vertex u in U that corresponds to v has $K \cup \{w\}$ as its neighborhood. Since the colors used for the vertices in K are the same as those used for K', any color that does not appear in $K \cup \{w\}$ does not appear in $K \cup K'$. So, we may color v with u's color while preserving proper coloring. Therefore this step creates a new proper coloring of G'. This coloring uses n colors but the color n only appears at vertex w. Therefore V which is the original graph G is only colored with $n-1$ colors, which by definition can be colored with no less than n colors. $\rightarrow\leftarrow$

Thus, the chromatic number of G' is $n+1$, and thus the Mycielskian process can create graphs of arbitrarily large chromatic number. $\square$

Theorem 1.5. *(Robertson, Sanders, Seymour, Thomas.)* ***Four Color Theorem.*** *For a planar graph G, $\chi(G) \leq 4$.*

In 1879, Alfred Kempe provided a proof for the Four Color Theorem that was later shown to be incorrect. In 1976, Appel and Haken [1] announced a

proof verifying the theorem with the aid of a computer, but it was broken up into over 1900 cases and had many holes that needed to be filled. A systematic, computerized, and more concise proof emerged in 1997 by a team of mathematicians consisting of Neil Robertson, Daniel P. Sanders, Paul Seymour, and Robin Thomas [6]. Martin Loebl at Charles University however has shown that the chromatic number of all planar triangulations is 4 without the use of a computer and has provided a polynomial time algorithm to find such a coloring [4].

Ramsey Theory

Questions in the study of Ramsey Theory often ask for the number of elements necessary for a certain condition to emerge. We look specifically at Ramsey numbers for a certain coloring in this section.

Definition 1.6. The **Ramsey number**, denoted $R(n, m)$, is the smallest integer N such that any graph on N vertices contains either an independent set of n vertices or a clique of m vertices.

1.1 Bounds on Ramsey Numbers

The Ramsey number exists for any $n, m \in \mathbb{N}$, and a bound on the Ramsey number is given by

$$R(n, m) \leq 2^{n+m},$$

from which it follows that

$$R(n, n) \leq 4^n.$$

We will show that a lower bound also exists, making the inequality

$$\sqrt{2}^n \leq R(n, n) \leq 4^n.$$

Definition 1.7. The **random graph** on n vertices, sometimes denoted R_n, is defined as follows. For every two vertices, flip a coin and if the coin is heads, create an edge between the two vertices.

We now prove the lower bound for $R(k,k)$ using the definition of a random graph.

Theorem 1.8. *A lower bound for $R(k,k)$ is $\sqrt{2}^k$.*

Proof. Let $k \in \mathbb{N}$ and set $n = \lfloor \sqrt{2}^k \rfloor$. Let $X(G)$ count the number of independent sets on k vertices of a graph G.

We want to find the expected value of X where G varies over all random graphs on n vertices. Given k vertices, we can find the probability they are an independent set. There are $\binom{k}{2}$ possible edges and the probability of an edge existing is $\frac{1}{2}$. So the probability is $(1/2)^{\binom{k}{2}}$ and so $E(X) = \binom{n}{k}\left(\frac{1}{2}\right)^{\binom{k}{2}}$.

We can show that for large values, $E(X) < \frac{1}{2}$. We prove this inequality by first using an upper bound on $\binom{n}{k}$, and the rest follows from algebraic manipulation.

$$\begin{aligned} E(X) &= \binom{n}{k}\left(\frac{1}{2}\right)^{\binom{k}{2}} \\ &\leq \left(\frac{ne}{k}\right)^k \left(\frac{1}{2^{(k-1)/2}}\right)^k \\ &\leq \left(\frac{2^{k/2}e}{k}\right)^k \left(\frac{1}{2^{(k-1)/2}}\right)^k \\ &= \left(\frac{2^{1/2}e}{k}\right)^k < \frac{1}{2}, \end{aligned}$$

for k sufficiently large.

This computation can be repeated for cliques. If $Y(G)$ counts the number of cliques of size k, we can use the exact same calculation to see that $E(Y) < \frac{1}{2}$ for k sufficiently large. Thus, the expected value of a random graph containing either an independent set or a clique is $E(X) + E(Y) < \frac{1}{2} + \frac{1}{2} = 1$.

Thus, there exists a graph for which there is no independent set or clique on m vertices.

Thus, $R(k,k) > \sqrt{2}^k$.

□

2 Other Colorings

Traditional coloring is only one of many ways to color. A number are described in *Graph Coloring Problems* by Jensen and Toft [3]. Some examples will be discussed below.

Definition 2.1. A **k-defective coloring** of a graph allows each vertex to have at most k neighbors of the same color. Note that traditional coloring is 0-defective coloring.

One could also define a coloring by only allowing partitions where each part induces either a complete or an empty subgraph. The cardinality of the smallest such partition is called the **cochromatic number** and is denoted as $Z(G)$.

2.0.1 Some Properties of Cochromatic Numbers

Theorem 2.2. *For all $m \in \mathbb{N}$, there exists a graph G that has cochromatic number m.*

Proof. For some m, let $G = K_m \cup K_m \cup K_m \ldots \cup K_m$ so that there are m copies. The cochromatic number of G is trivially at most m. The chromatic number must be greater than $m-1$ via the pigeonhole principle. We know that G has m^2 vertices, so if $Z(G) \leq m-1$ we would need at least one either independent set or clique of size at least $\frac{m^2}{m-1}$. By construction G only has independent sets or cliques of size $m < \frac{m^2}{m-1}$, but thus $Z(G)$ needs to be at least m. □

Theorem 2.3. *Given a graph G on n vertices, the cochromatic number will be at most $\sqrt{n} + \frac{n}{\frac{1}{2}\log_4 n}$.*

Proof. Since $n > \sqrt{n}$, then by the previous Ramsey result there is a clique or an independent set of at least size $\log_4 \sqrt{n}$. Take out such a set and and assign it a color. Repeat this process until the number of remaining vertices is no longer bigger than $\sqrt{n}$. Color the remaining vertices all different colors.

Since each independent set or clique was at least of size $\log_4 \sqrt{n}$ by the Ramsey result, we used at most $\frac{n}{\frac{1}{2}\log_4 n}$ colors in the first step. There are at most $\sqrt{n}$ vertices left over afterwards. Thus $\sqrt{n} + \frac{n}{\frac{1}{2}\log_4 n}$ is an upper bound on the cochromatic number. □

References

[1] K. Appel, W. Haken, Every Planar Map is Four Colorable, Illinois Journal of Mathematics 21 (1977) 429–567.

[2] M. Gardner, Mathematical Games, Scientific American, 180 (1960) 203-204.

[3] T.R. Jensen, B. Toft, Graph coloring problems, John Wiley & Sons 39 (2011).

[4] M. Loebl. Deciding 4-colorability of planar triangulations. arXiv preprint arXiv:1505.03962 (2015).

[5] J. Mycielski, Sur le coloriage des graphes, Colloq. Math. 3 (1955) 161–162.

[6] N. Robertson, D.P. Sanders, P. Seymour, R. Thomas, A new proof of the four-colour theorem, Electron. Res. Announc. Amer. Math. Soc. 2 (1996) 17-25.

[7] A. Soifer, The Mathematical Coloring Book: Mathematics of Coloring and the Colorful Life of its Creators, New York: Springer, (2008) ISBN 978-0-387-74640-1.

The participants on the trip to Harriman State Park.