



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Complexity of Games on Graphs

by

Václav Blažej

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Dissertation degree study programme: Informatics
Department of Theoretical Computer Science

Prague, June 2022

Supervisor:

doc. RNDr. Tomáš Valla, Ph.D.
Department of Theoretical Computer Science
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Copyright © 2022 Václav Blažej

Abstract

In this dissertation thesis, we study the complexity of games on graphs. The main goal is to establish which settings of the problems are tractable and which settings are hard. To reach these goals, we use algorithmization together with hardness reductions; and structural and parameterized analysis.

First, we focus on a game variant of the domination number called the ETERNAL DOMINATION NUMBER. We provide several new techniques that allow us to tackle the problem. Using those we obtain an algorithm for determining the exact value of the ETERNAL DOMINATION NUMBER for the class of cactus graphs.

We also study the HAT CHROMATIC NUMBER problem which is related to graph coloring. We introduce its generalization and show its connection to the independence polynomial on graphs. This connection then allows us to obtain a polynomial algorithm to solve the problem on chordal graphs, give bounds based on maximal vertex degree, and solve complete graphs, paths, and cycles.

Further, we investigate a game variant of Ramsey numbers called the ONLINE RAMSEY NUMBERS. The classical Ramsey number gives a lower bound on the size of the graph such that it is guaranteed to contain some homogeneous structure. We show that the Online Ramsey game gives an asymptotic advantage compared to the classical problem variant. We also initiate the study of obtaining induced subgraphs within the game setting and show solutions for paths, cycles, and several tree families.

Last, we show a generalization of the GROUP IDENTIFICATION PROBLEM where a process over a graph models spreading of secrets. The model of a single process is known to be tractable, however, the generalization with several processes becomes NP-hard. We provide a complete parameterized complexity analysis of four variants of this problem showing P, FPT, and XP algorithms or W[1] and NP-hardness results via standard means. We characterize which secondary measures of the input make the problem hard and which make it tractable.

Keywords: combinatorial game theory, complexity theory, graph, domination number, hat chromatic number, online Ramsey theory, group identification, parameterized complexity, algorithmic game theory

Abstrakt

V této disertační práci se zabýváme složitostí her na grafech. Hlavním cílem je zjistit, kdy jsou problémy řešitelné a kdy jsou těžké. K dosažení těchto cílů používáme algoritmicizaci společně s těžkostními redukcemi; a strukturální i parametrizovanou analýzu.

Nejprve se zaměříme na herní variantu dominujícího čísla zvané ETERNAL DOMINATION. Poskytujeme několik nových technik, které nám umožňují problém řešit. Pomocí těch získáme algoritmus pro určení přesné hodnoty ETERNAL DOMINATION pro třídu kaktusových grafů.

Studujeme také problém BEARS WITH HATS, který souvisí s barvením grafů. Zavádíme jeho zobecnění a ukazujeme jeho spojení s grafovým independence polynomem. Toto spojení nám pak umožňuje získat polynomiální algoritmus pro chordální grafy. Také díky němu získáme meze na základě maximálního stupně grafu a zcela vyřešíme úplné grafy, cesty a cykly.

Dále zkoumáme herní variantu Ramseyho čísel, která se nazývá ONLINE RAMSEY NUMBERS. Klasické Ramseyho číslo udává spodní hranici velikosti grafu takovou, aby bylo zaručeno, že obsahuje nějakou homogenní strukturu. Ukazujeme, že Online Ramsey hra poskytuje asymptotickou výhodu v porovnání s klasickou variantou problému. Také zahajujeme studii získávání indukovaných podgrafů v rámci herního prostředí a ukazujeme řešení cest, cyklů a několika rodin stromů.

Nakonec si ukážeme zobecnění GROUP IDENTIFICATION PROBLEM, kdy proces přes graf modeluje šíření tajemství. Je známo, že model jednoho procesu je řešitelný, nicméně při zobecnění na více procesů se problém stává NP-těžkým. Poskytujeme kompletní parametrizovanou analýzu složitosti čtyř variant tohoto problému pomocí P, FPT a XP algoritmů a $W[1]$ a NP-těžkosti zapomocí standardních nástrojů. Charakterizujeme, které vlastnosti vstupu dělají tento problém těžký a díky kterým je řešitelný.

Klíčová slova: kombinatorická teorie her, teorie složitosti, grafy, dominující číslo, hat barvnost, online Ramseyho teorie, určování skupiny, parametrizovaná složitost, algoritmická teorie her

Acknowledgements

I thank my supervisor Tomáš Valla and my coauthors Dušan Knop, Ondřej Suchý, Jan Matyáš Křišťan, Šimon Schierreich, Pavel Dvořák, Michal Opler, Pratibha Choudhary, Matas Šileikis, Jiří Fiala, Giuseppe Liotta, and Pavel Valtr for valuable lessons and collaboration throughout the last 5 years. I also value conversations about life, research, and board games with Štěpán Plachý, Jakub Jirůtka, Josef Malík, David Bernhauer, Michal Dvořák, Jan Pokorný, Tung Anh Vu, Tomáš Brunclík, Jiří Chvojka, Tomáš Šidlík, Milošlav Brožek, Kristina Roučová, Tomáš Pavlík, Monika Pavlíková, Sanjukta Roy, Jocelyn Thiebaut, Radovan Červený, Pavel Veselý, Karel Král, Tomáš Masařík, Jana Novotná, Aneta Pokorná, Jana Syrovátková, Martin Koutecký, Miloš Chromý, Adam Kabela, Mara Nehring, Pankaj Kumar, Denys Bulavka, Tuan Tran, Eoin Long, Cory Palmer, Marcin Witkowski, Michał Dębski, Grzegorz Adamski, Małgorzata Bednarska-Bzdega, Francesco Dolce, Giuseppe Italiano, Johanness Zink, Felix Klesen, Ignaz Rutter, Alexander Wolff, Ludwig Kampel, and Julian Nickerl. I value friendships made along the way and the people that push me out of my comfort-zone to benefit my future.

I am grateful to my mom, dad, and sister for supporting my studies while encouraging me to pursue what I do.

I acknowledge the support of my research by the Grant Agency of the Czech Technical University in Prague, grant No. SGS20/208/OHK3/3T/18, by OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics”, and by the Department of Theoretical Computer Science.

Contents

Notation	xiii
1 Introduction	1
1.1 Complexity analysis	2
1.2 Graph Theory	3
1.3 Goals and Problems	3
1.4 Structure of the Dissertation Thesis	7
2 m-eternal Domination Number of Cactus Graphs	9
2.1 Introduction	9
2.1.1 Original Results	10
2.1.2 Preliminaries	10
2.2 High-level Overview of the Proof	11
2.3 Reducing Trees	12
2.4 The m-eternal domination Toolbox	16
2.4.1 Lower Bounds	17
2.4.2 Upper Bounds	20
2.4.3 Tools for Altering Strategies	26
2.5 Reducing Cactus Graphs	29
2.5.1 Technique and Overview	31
2.5.2 Properties of Cycle Edges	33
2.5.3 Cycle Reductions	35
2.5.4 Constant Component Reductions	43
2.6 Future Work	51
2.7 Additional observations	51
2.7.1 Complete Strategies	51
2.7.2 Non-complete Strategy	52
3 Bears with Hats	53

3.1	Introduction	53
3.1.1	Related and Follow-up Works	55
3.2	Preliminaries	55
3.3	Fractional Hat Chromatic Number	57
3.4	Basic Blocks	58
3.4.1	Graph Products	60
3.5	Independence Polynomial	61
3.6	Applications	68
3.6.1	Fractional Hat Chromatic Number is Almost Linear in the Maximum Degree	68
3.6.2	Paths and Cycles	69
4	Online Ramsey Numbers	73
4.1	Introduction	73
4.2	Induced Paths	76
4.3	Cycles and Induced Cycles	77
4.4	Tight Bounds for a Family of Trees	80
4.5	Family of Induced Trees with an Asymptotic Gap	82
5	Group Identification	87
5.1	Introduction	87
5.1.1	Contribution and Organization	88
5.2	Preliminaries	89
5.2.1	Computational Complexity	90
5.2.2	Parameterized Complexity	91
5.3	Node and Composition Notation	92
5.4	Liberal Starting Rule	93
5.4.1	Easy variants	94
5.4.2	Agent deletion: Constructive-Destructive	95
5.4.3	Agent addition: Constructive-Destructive	102
5.4.4	Agent deletion: Destructive-Destructive	105
5.4.5	Agent addition: Constructive-Constructive	110
5.5	Future work	113
6	Conclusions	115
6.1	Contributions of the Dissertation Thesis	115
6.2	Future Work	116
	Bibliography	119
	Reviewed Publications of the Author Relevant to the Thesis	129
	Remaining Publications of the Author	131

List of Figures

1.1	Graph classes overview	2
2.1	Example of reducing tree graphs	14
2.2	Overview diagram of Section 2.4	17
2.3	Example application of Definition 2.4.5	19
2.4	Graph interface and transitions	21
2.5	Example (partial) labelled strategy, cutting, and composing	23
2.6	Cartesian product over subset	27
2.7	Block-cut tree substructures	30
2.8	Example leaf cycle and vertex colors	31
2.9	Overview of Section 2.5	33
2.10	Part of the lower bound proof for Reduction c_6	39
2.11	Example part of an RW-cycle strategy	40
2.12	Movement through the interface in RW-cycles	42
2.13	Part of the proper edge states proof	42
2.14	Case analysis of leaf cycle reductions	44
2.15	Strategy for $(\widehat{X}, \widehat{2}, \widehat{2}, \widehat{X})$	45
2.16	Loop and multiedge reduction	45
2.17	Cases of multiedge reduction	46
2.18	Strategy for $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X})$ leaf cycle	49
2.19	Strategy for $(\widehat{X}, \widehat{2}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X})$ leaf cycle	50
2.20	Strategy on a 5×5 grid	52
3.1	Clique join of graphs	60
3.2	Example of application of the clique join lemma	61
3.3	Example application of clique join theorem for chordal graphs	66
3.4	A construction of the hat chromatic game on paths	70
4.1	Various notions of C_4 being a copy in G	75
4.2	One step in creating an induced monochromatic P_n	77

4.3	Creation of $\rho_{n/2}$ for $n = 18$.	79
4.4	Building an induced C_9 .	79
4.5	More efficient construction for even non-induced cycles.	80
4.6	Building one red leg of a spider $\sigma_{4,5}$.	81
4.7	One step of building a $S_{k,\ell}$ where $k = 3$.	84
5.1	Serial composition	93
5.2	Hasse diagram for results in the DLCD	96
5.3	Serial composition and NP-hardness reduction for DLCD	97
5.4	W[1]-hardness reduction for DLCD	100
5.5	GRID TILING WITH \leq reduction gadget	101
5.6	Hasse diagram for results in the ALCD	103
5.7	Hasse diagram for results in the DLDD	106
5.8	NP-hardness reduction for DLDD	107
5.9	W[1]-hardness proof for DLDD	109
5.10	Hasse diagram for results in the ALCC	111
5.11	W[1]-hardness reduction for ALCC	112

List of Tables

2.1	Leaf reductions	13
2.2	List of cycle reductions	36
2.3	List of constant component reductions	47
5.1	Results summary for GROUP IDENTIFICATION PROBLEM.	94
5.2	Results summary for the DLCD	95
5.3	Results for ALCD	103
5.4	Results summary for the DLDD	106
5.5	Results for ALCC	110

Notation

Math

$\lceil x \rceil, \lfloor x \rfloor, \text{round}(x)$	Ceil, floor, and round to the closest integer
\mathbb{N}	Positive integers
$[h]$	Set of numbers $\{0, 1, \dots, h - 1\}$
$\mathbf{h}, \mathbf{g}, \mathbf{w}, \mathbf{r}$	Vectors
sup	Supremum
∞	Infinity
$[a, b)$	Closed-open interval
F_i	Fibonacci numbers
$P_G(\mathbf{x})$	Independence polynomial
$A \times B$	Cartesian product
GCD	Greatest common divisor
LCM	Least common multiple

Complexity

$\mathcal{O}(f), \Omega(f), o(f), \omega(f), \Theta(f)$	Bachmann–Landau notation
P	Polynomial $O(\text{poly}(n))$
NP	Non-deterministic polynomial
PSPACE	Polynomial space
EXPTIME	Exponential time $O(2^{\text{poly}(n)})$
FPT	Fixed-parameter tractable $f(k) \cdot \text{poly}(n)$
XP	Slicewise polynomial $n^{f(k)}$
W[1]-hard	Parameterized hardness (refutes FPT)
ETH	Exponential Time Hypothesis

Graphs

G	Graph
$E(G)$	Graph edges
$V(G)$	Graph vertices
n	Number of graph vertices
m	Number of graph edges
$G[X]$	Induced subgraph
$N(v), N[v]$	Open and closed neighborhood
$\deg(v)$	Degree of a vertex
$\deg_b(v)$	Degree in only blue-colored edges
$\overline{\deg}(v)$	Degree over subgraph
$\Delta(G)$	Maximum vertex degree in G
$\delta(G)$	Minimum vertex degree in G
$G_A \square G_B$	Graph Cartesian product

Standard graphs

K_n	Complete graph on n vertices
C_n	Cycle on n vertices
P_n	Path on n vertices or n edges (this is usually stated explicitly)

Graph classes

Chordal	There is no induced cycle on more than 3 vertices.
Forest	Contains no cycles.
Tree	Is connected and contains no cycles.
Cactus	No edge lies on more than one cycle.
Christmas cactus	Cactus where no vertex lies on more than two cycles.

Graph parameters

$\chi(G)$	Chromatic number
$\chi'(G)$	Edge chromatic number
$\mu(G)$	Hat chromatic number
$\hat{\mu}(G)$	Fractional hat chromatic number
$VC(G)$	Vertex cover
$r(G)$	Ramsey number
$\bar{r}(G), \bar{r}_{\text{ind}}(G)$	Size-Ramsey number, Induced size-Ramsey number
$\tilde{r}(G), \tilde{r}_{\text{ind}}(G)$	Online Ramsey number, Induced online Ramsey number

Introduction

Computational problems are classically shown to be tractable or intractable in the following way. On the one hand, the tractability of a problem is usually shown by implementing an algorithm that solves the problem in polynomial time (computational class P). On the other hand, the NP-hardness of a problem is shown by designing a reduction that reduces any instance of an already known-to-be NP-complete problem to an instance of our problem. Using these approaches, we may decide whether a problem is either in P or is NP-hard.

The problem of deciding whether P equals NP or not is at the center of classical complexity theory. As no one was able to resolve this problem for over 50 years, it became standard practice to obtain results that are based on the assumption that $P \neq NP$. Moreover, several even stricter assumptions were developed to get conditional results.

Modern approaches to obtaining results for NP-hard problems are via parameterized complexity, structural restrictions, and approximation algorithms. The parameterized analysis considers the problem's complexity with respect to not only the input size but also the secondary measure – parameter. Such a finer-grained analysis can distinguish tractable and intractable variants of the problem assuming bounded parameter value. We can also restrict the input structurally so that the NP-hard problem becomes tractable, e.g., by restricting the input graph to be acyclic. Approximation algorithms are another approach that gives us solutions that may be slightly worse than optimal, however, their computational complexity is reasonable. The next tool may be shifting the attention to a generalized problem to come up with novel solutions. These tools allow us to refine the boundary between the tractable and intractable cases of given problems.

The game theory contains problems with complexity which usually far exceeds the mentioned complexity classes used for classical decision problems. In games, we aim to show that a problem is solvable in polynomial space (class $PSPACE$) or at least with the use of (only) exponential time (class $EXPTIME$). When we want to obtain polynomial results in this field we often have to restrict the problem.

1.1 Complexity analysis

In complexity analysis of classical graph problems we aim to establish tractability and hardness results for the best possible classes of graphs. “Best” in this context is relative – we want to push tractability results to a class which encompasses as many graphs as possible; and we want hardness results even in the most restricted cases. Restricting graph structure can be done by focusing on a graph class, such as interval, chordal, cactus, series-parallel, and sparse graphs.

Also, the structure may be restricted by graph parameters which include e.g. vertex cover, tree-width, clique-width, tree-depth, and feedback vertex set; but also the novel twin-width introduced in 2020 by Bonnet et al. [14]. There is a hierarchy of graph parameters (see [103]) and graph classes (see ISGC¹) which allow us to refine the boundary of complexity for a given problem. See Figure 1.1 for classes relevant to this thesis.

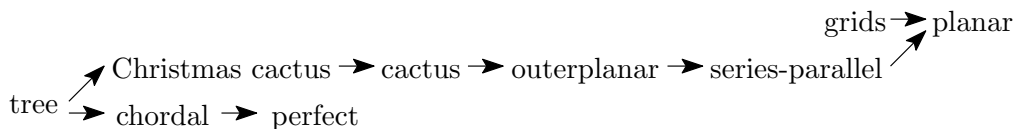


Figure 1.1: Graph classes mentioned in this thesis, their inclusions, and context within well-known graph classes.

For graphs, there are many classical combinatorial problems. NP-hardness of many of these problems was established by Karp [72] in 1972.

The complexity analysis experienced a fast development in the last three decades after Downey and Fellows [39] established parameterized complexity in 1992. This analysis looks at the problem from a finer-grained perspective than just P versus NP-hard. It moves some of the secondary measures of the input to *parameters*. Then, we ask if the complexity is polynomial for the input while it can be non-polynomial in the parameters. Such problems are called *fixed-parameter tractable* (FPT) and their complexity is of the form $f(t) \cdot n^{\mathcal{O}(1)}$ where f is a computable function, n is the size of the input, and t is the parameter.

To complement FPT, the field of parameterized complexity gives us tools to show that a problem is W[1]-hard, which conditionally refutes the existence of an FPT algorithm; and to develop OR-cross composition which refutes the existence of a polynomial kernel.

The Exponential Time Hypothesis (ETH) [68] poses that (in short) SAT cannot be solved in subexponential time. Combined with W[1]-hardness reductions, we may obtain conditional parameterized lower bounds on the time complexity of algorithms for NP-hard problems. It is common to show that an elementary exponential algorithm is optimal with respect to this conditional lower bound. It is widely believed that this hypothesis could be true; it was not refuted in over 20 years. However, proving ETH is also not expected as it would imply $P \neq NP$.

For more details on parameterized complexity including kernelization, W[1]-hardness, ETH, and graph parameters see Cygan et al. [35].

¹<https://www.graphclasses.org/>

1.2 Graph Theory

We use standard graph theory notation, for more we refer to the book by Diestel [36]. By a graph $G = (V, E)$ we mean an ordered pair of vertices and edges, where edges are unordered pairs of vertices, i.e., $V(G)$ is a set of elements, and $E(G) \subseteq \binom{V}{2}$. We may refer to $V(G)$ and $E(G)$ by V and E if the meaning is clear from the context. A vertex u and an edge e are incident if and only if $u \in e$. Two vertices u and v are adjacent if and only if $\{u, v\} \in E(G)$. Neighbors (or neighborhood) $N(u)$ of u is a set of all adjacent vertices $N(u) = \{v \mid v \text{ is adjacent to } u\}$. Degree of a vertex $v \in V(G)$ is denoted by $\deg(v) = |N(v)|$. A closed neighborhood is $N[u] = N(u) \cup \{u\}$. Graph has a maximum degree $\Delta(G) = \max_u \deg(u)$ and a minimum degree $\delta(G) = \min_u \deg(u)$. An induced subgraph $G[A] = (A, E(G) \cap \binom{A}{2})$. By $G \square H$ we denote a graph Cartesian product, i.e.,

$$G \square H = (V(G) \times V(H), \{ \{(u_1, v), (u_2, v)\} \mid (u_1, u_2) \in E(G) \} \cup \{ \{(u, v_1), (u, v_2)\} \mid (v_1, v_2) \in E(H) \}).$$

We use standard graphs: a complete graph $K_n = ([n], \binom{[n]}{2})$, a cycle $C_n = ([n], \{i, i + 1 \bmod n\}_{i=0}^{n-1})$, and a path on n vertices $P_n = ([n], \{i, (i + 1)\}_{i=0}^{n-2})$. A path on n edges is also commonly used, so which one is means is usually explicitly stated to avoid confusion.

We shall introduce relevant notions and notation (mainly graph parameters) in respective chapters.

1.3 Goals and Problems

The combinatorial game theory is quite old. It developed through the 20th century. The desire to understand basic games, such as tic-tac-toe, drove scientists to develop the basics of what we now call combinatorial game theory. Gradually, a sophisticated theory to analyze the complexity of game trees, decision trees, and state spaces was created, see surveys [3, 11, 29]. Novel games are played over more complex combinatorial structures – these are the ones we are interested in. In particular, we aim to

- establish tractability and hardness bounds for game theoretic problems,
- identify which natural and structural parameters make the problems tractable,
- show in which settings games exhibit significantly different computational complexity compared to their graph-theoretic counterparts.

We now present a brief introduction into the problems we focused on in this thesis along with our contributions.

M-ETERNAL DOMINATION. A game variant of DOMINATING SET that was introduced by Goddard et al. [58]. In M-ETERNAL DOMINATION, the players are given a graph G and a number k . Defender (first player) chooses where to initially place k vertex guards. Each turn, Attacker (second player) chooses a vertex to attack. Now, Defender may move each guard by at most one edge. After that, the attacked vertex must be occupied by a guard. What is the minimum k such that G may be defended indefinitely? (Note the difference from ETERNAL DOMINATION introduced by Burger et al. [20] where at most one guard may be moved each turn.) At each point in time, the guards must constitute a dominating set, otherwise, Attacker wins in the next turn. This topic is quite active, with many recent papers getting bounds based on structural properties of the input graph [42, 70, 84, 100, 101] while others try to generalize the problem [34].

DOMINATING SET

Input: A graph G and a positive integer k .

Question: Is there a set $X \subseteq V(G)$ of size at most k such that for every $v \in V(G)$ either v is in X or it is adjacent to some vertex from X ?

Our contribution mainly consists obtaining auxiliary tools that can be used to devise lower and upper bounds. Using these tools, we obtained an algorithm for determining the M-ETERNAL DOMINATION NUMBER for the class of cactus graphs in polynomial time, which is summed up in the following theorem.

Theorem 2.1.1 (B., Křišťan, Valla). Let G be a cactus graph on n vertices. Then there exists a polynomial algorithm which computes $\gamma_m^\infty(G)$.

Preliminary version of these results was published in:

[A.1] Václav Blažej, Jan Matyáš Křišťan, and Tomáš Valla. On the m-eternal domination number of cactus graphs. *Reachability Problems - 13th International Conference, RP 2019*, volume 11674 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2019. doi:10.1007/978-3-030-30806-3_4.

HAT CHROMATIC NUMBER. In the HAT CHROMATIC NUMBER problem, the players are given a graph G and a number k . First player comes up with a function that maps any combination of possible colorings of its neighborhood to a “guessed” color, i.e., $f_v: [k]^{|N(v)|} \rightarrow [k]$ for each vertex $v \in V(G)$. Then, the second player chooses a coloring c of all vertices. First player wins if there is at least one vertex “guessed” correctly, i.e., $f_v(v) = c(v)$. This game is often introduced from a viewpoint of agents coming up with a collective strategy to subdue a common adversary. This game was shown to have connections to coding theory [71], network coding [55], auctions [2], finite dynamical systems [53], and circuits [108]. Recent publications focused on devising bounds or getting exact results on various classes of graphs and providing bounds on graphs that are constructed by combining (in various ways) smaller graphs [16, 63, 80]

CHROMATIC NUMBER

Input: A graph G and a positive integer k .

Question: Is there a function $\phi: V(G) \rightarrow [k]$ such that, if u and v are adjacent, then $\phi(u) \neq \phi(v)$?

We introduce a natural generalization of this problem and show its connection to the independence polynomial on graphs. This connection then allows us to obtain a polynomial algorithm to solve the problem on chordal graphs, give bounds based on the maximal vertex degree, and solve cliques, paths, and cycles exactly. We point out the following theorem.

Theorem 3.5.7 (B., Dvořák, Opler). There is an algorithm \mathcal{A} such that given a chordal graph G as an input, it approximates $\hat{\mu}(G)$ up to an additive error $1/2^k$. The running time of \mathcal{A} is $2k \cdot \text{poly}(n)$, where n is the number of vertices of G . Moreover, if $\hat{\mu}(G)$ is rational, then the algorithm \mathcal{A} outputs the exact value of $\hat{\mu}(G)$.

These results were published in:

[A.2] Václav Blažej, Pavel Dvořák, and Michal Opler. Bears with hats and independence polynomials. *Graph-Theoretic Concepts in Computer Science - 47th International Workshop, WG 2021*, volume 12911 of *Lecture Notes in Computer Science*, pages 283–295. Springer, 2021. doi:10.1007/978-3-030-86838-3_22.

ONLINE RAMSEY NUMBER. The SIZE-RAMSEY NUMBER was introduced by Erdős, Faudree, Rousseau, and Schelp [48] in 1978. Since then, it became a well-researched topic, see the survey by Conlon, Fox, and Sudakov [27]. The field is of interest to extremal graph theorists as it asks for a minimum size of a structure to guarantee the existence of a specific substructure. Recent activity by major researchers [28] resolved several open problems which were posed right at its creation. The ONLINE RAMSEY NUMBER looks at the problem from a game-theoretic point of view.

RAMSEY NUMBER

Input: A graph H and a positive integer k .

Question: Does any two-edge-coloring of K_k contain a monochromatic copy of H ?

The ONLINE RAMSEY NUMBER problem was introduced by Beck [8]. In this game, the players are given a graph H . The game is played on a graph G which has an infinite number of vertices. In each turn, Builder (first player) creates an edge in G , and Painter (second player) colors it either red or blue. ONLINE RAMSEY NUMBER is the minimum number of rounds such that Builder can force a monochromatic copy of H in G .

In this game, the built structure can change in an online manner which gives the Builder the ability to adapt. Hence, the ONLINE RAMSEY NUMBER is always at most the SIZE-RAMSEY NUMBER and by that, they bound each other. Natural generalizations of the

ONLINE RAMSEY NUMBER are investigated, with a focus on establishing bounds for graph classes, e.g. in [59, 60, 93]. If the structure of the built graph G has no other restrictions, then by Ramsey theorem the game always ends in the victory of Painter. This prompted investigations of graph classes to which one can restrict the graph G so that Builder is restricted in what edges he may create. This topic is quite active with several recent results [1, 19, 44].

We also study variant of the problem where the subgraph must be found induced (no extra edges are present). We show asymptotically optimal solutions for paths, cycles, and several tree families.

Theorem 4.1.1 (simplified) (B., Dvořák, Valla). $\tilde{r}_{\text{ind}}(P_n) = \mathcal{O}(n)$ and $\tilde{r}_{\text{ind}}(C_n) = \mathcal{O}(n)$.

One tree family in particular exhibits an asymptotic advantage compared to the classical problem variant as their size-Ramsey number $\bar{r}(S_{k,\ell}) = \Omega(k^2\ell)$.

Theorem 4.5.2 (B., Dvořák, Valla). $\tilde{r}_{\text{ind}}(S_{k,\ell}) \leq \mathcal{O}(k\ell)$.

These results were published in:

[A.3] Václav Blažej, Pavel Dvořák, and Tomáš Valla. On induced online Ramsey number of paths, cycles, and trees. *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019*, volume 11532 of *Lecture Notes in Computer Science*, pages 60–69. Springer, 2019. doi:10.1007/978-3-030-19955-5_6.

GROUP IDENTIFICATION. This problem is inspired by target set selection – a selection model from algorithmic game theory that was introduced by Kempe et al. [74]. It can be thought of as selecting starting vertices that spread an opinion throughout the entire graph. Many variants of this problem were studied, one of which is manipulation of the outcome to secure given targets [13, 47, 109]. While group identification problem for more than one secret was studied [23, 56, 79] the secrets usually interact.

We study the problem where we want to control the target when more secrets spread at the same time. Each secret spreads through its own graph, however, the graphs share the same set of vertices. Through controlling participation of vertices in the spreading process we want to achieve that certain set learns or does not learn the secret – these targets may be set for each secret separately. This setting was studied for single secret and were shown to be tractable [47, 109]. We show that all non-trivial combinations of two secrets are NP-complete, while also providing a complete parameterized analysis over secondary measures of the input: size of the solution, length of the process, and sizes of the target sets. For all the settings we have NP-complete, XP and W[1]-hard, FPT, or P.

We show several approaches that solve the problem efficiently (while parameterized). To complete the complexity picture we present NP-hardness or W[1]-hardness using reductions from INDEPENDENT SET, 3-SAT, PARTITIONED SUBGRAPH ISOMORPHISM, GRID

TILING, and GRID TILING WITH \leq . Where appropriate, the hardness results are accompanied with a conditional complexity lower bound derived from ETH. For a comprehensive list of our complexity results we refer the reader to Table 5.1 on page 94.

Student abstract of these results was accepted to:

[A.4] Václav Blažej, Dušan Knop, and Šimon Schierreich. Controlling the spread of two secrets in diverse social networks (student abstract). In *Computer Science - Theory and Applications - 36th Conference on Artificial Intelligence, AAI 2022, (to appear), 2022*.

1.4 Structure of the Dissertation Thesis

The chapters of this thesis are organized as follows.

- *Introduction* (Chapter 1). This chapter introduced the study field and the targets that are a common point of our research. It also presented short descriptions of the problems and our contribution to them.
- *Problems* (Chapters 2 to 5). Each problem of interest, their notation, and related previous work, together with our contribution is introduced in depth in its own chapter.
- *Summary* (Chapter 6). Reviews our contributions and suggests further research directions.

m-eternal Domination Number of Cactus Graphs

In m-eternal domination attacker and defender play on a graph. Initially, defender places guards on vertices. Each round the attacker chooses a vertex to attack. Then, defender can move each guard to a neighboring vertex, and must move a guard to the attacked vertex. The m-eternal domination number is the minimum number of guards such that the graph can be defended indefinitely.

In this chapter, we study the m-eternal domination number of cactus graphs. We consider two variants of the m-eternal domination number: one allows multiple guards to occupy a single vertex, second variant requires the guards to occupy distinct vertices. We develop several tools for obtaining lower and upper bounds on these problems and we use them to obtain an algorithm which computes the minimum number of required guards of cactus graphs for both variants of the problem.

2.1 Introduction

Consider the following game, played by an attacker and a defender on graph G . The defender controls a set of guards, which he initially places on the vertices of G . Each vertex can be occupied by at most one guard.

In each round, the attacker first chooses one vertex, which he *attacks*. The defender then must *defend* against the attack by moving some or all of his guards along their adjacent edges, so that one of the guards moves to the attacked vertex.

If the attacked vertex is not occupied by a guard after the attack, the attacker wins. The defender wins if he can defend indefinitely.

Defending a graph from attacks using guards for an infinite number of steps was introduced by Burger et al. [20]. In this chapter we study the concept of the m-eternal domination, which was introduced by Goddard et al. [58] (eternal domination was originally called eternal security). Here, the notion of the letter “m” emphasizes that multiple

guards may move during each round. There is also a variant of the problem studied by Goddard et al. [58] where only one guard may move during each round, which is not considered in this work.

The m-eternal domination number $\gamma_m^\infty(G)$ is the minimum number of guards which defend against all attacks indefinitely. Goddard et al. [58] established γ_m^∞ exactly for paths, cycles, complete graphs and complete bipartite graphs. Since then, several results have focused on finding bounds on γ_m^∞ under different conditions or graph classes. Among the studied graph classes are trees [76, 65, 78], grids [52, 106, 89, 70], and interval graphs [18]. For a good survey of other related results and topics, see Klostermeyer and Mynhardt [77].

Very little is known regarding the algorithmic aspects of m-eternal domination. The decision problem (asking if $\gamma_m^\infty(G) \leq k$) is NP-hard and belongs to EXPTIME, however, it is not known whether it lies in the class PSPACE [77].

2.1.1 Original Results

In this chapter, we focus on the class of cactus graphs (connected graphs where each edge lies in at most one cycle) and provide an algorithm for computing γ_m^∞ in cactus graphs. In Section 2.4, we provide a set of tools with more general applications to proving upper and lower bounds of γ_m^∞ . Those tools are then used in Section 2.5 to describe a set of reductions, which allow us to compute γ_m^∞ of cactus graphs. This is a significant expansion of a basic principles which were introduced by Klostermeyer and MacGillivray [75], in which they provide an algorithm for computing γ_m^∞ of trees.

Our main result is summarized in the following theorem.

Theorem 2.1.1. Let G be a cactus graph on n vertices. Then there exists a polynomial algorithm which computes $\gamma_m^\infty(G)$.

2.1.2 Preliminaries

Let us now review all the standard concepts formally. A graph is a *cactus* if its every edge lies on at most one cycle. For an undirected graph G let a *configuration* be a multiset of its vertices $C = \{c_1, \dots, c_n \mid c_i \in V(G)\}$. We will refer to the elements of configurations as *guards*. If a vertex is an element of a configuration, then it is *occupied* (by a guard). Two configurations C_1 and C_2 of G are mutually *traversable* if there is some set of pairs $\mathcal{T}(C_1, C_2) = \{(v_1, u_1), (v_2, u_2), \dots, (v_n, u_n)\}$ such that $C_1 = \{v_1, \dots, v_n\}$ and $C_2 = \{u_1, \dots, u_n\}$ and $\{v_i, u_i\} \in E(G)$ for all i from 1 to n . We perceive the guards as tokens which move through the graph. The elements of $\mathcal{T}(C_1, C_2)$ are called *movements* and a single ordered pair among them is a *move* of a guard. A guard that moves in $\mathcal{T}(C_1, C_2)$ to the same vertex where he started is called *stationary*. A *strategy* in G is a graph $S_G = (\mathbb{C}, \mathbb{F})$ where \mathbb{C} is a set of configurations over $V(G)$ such that all of the configurations have the same size and $\mathbb{F} \subseteq \mathbb{C}^2$ describe possible transitions between the configurations. The *order* of a strategy is the number of guards in each of its configuration. In papers on this topic it is often assumed that the strategy edges are given implicitly as

$\mathbb{F} = \{\{C_1, C_2\} \in \mathbb{C}^2 \mid C_1 \text{ and } C_2 \text{ are mutually traversable in } G\}$. For our purposes, we want to prescribe the strategy explicitly. We introduce the notions for exact strategy prescription in Section 2.4.2.

We call the strategy S_G to be *defending against vertex attacks* if for any $C \in \mathbb{C}$ the configuration C and its neighbors in S_G cover all vertices of G , i.e., when a vertex $v \in V(G)$ is “attacked” one can always respond by changing to a configuration which has a guard at the vertex v . Formally, $S_G = (\mathbb{C}, \mathbb{F})$ is defending if

$$(\forall C \in \mathbb{C}) (\forall v \in V(G)) (v \in C \vee (\exists C' \in \mathbb{C}) (\{C, C'\} \in \mathbb{F} \wedge v \in C')).$$

Note that every configuration in a strategy which defends against vertex attacks induces a dominating set in G as otherwise the attacker would win in the next round.

We investigate two variants of the game. The variants differ in whether they allow multiple guards to occupy the same vertex. Let an *m-Eternal Guard Strategy* in G be a strategy defending against vertex attacks in G .

M-ETERNAL DOMINATION

Input: An undirected graph $G = (V, E)$.

Question: What is the minimum number of guards γ_m^∞ such that there exists an m-Eternal Guard Strategy S_G where each vertex is occupied by at most one guard that defends against vertex attacks in G ?

M-ETERNAL GUARD CONFIGURATION

Input: An undirected graph $G = (V, E)$.

Question: What is the minimum number of guards Γ_m^∞ such that there exists an m-Eternal Guard Strategy S_G that defends against vertex attacks in G ?

The open neighborhood of u in G will be denoted as $N_G(u)$. By P_n we denote a path with n edges and $n + 1$ vertices. By $G[U]$ we denote the subgraph of G induced by the set of vertices $U \subseteq V(G)$.

2.2 High-level Overview of the Proof

In order to solve the M-ETERNAL DOMINATION and the M-ETERNAL GUARD CONFIGURATION on cactus graphs, we use induction on the number of vertices. Base cases will be presented in Definition 2.5.6. In the induction step, we show how to reduce cactus graph G to a smaller cactus graph G' while showing lower bound and upper bound in the following ways. Reduction from G to G' is done using Observations 2.4.3 and 2.4.8 to 2.4.10. These directly show a lower bound $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + K$ for some constant K . Then, we show an

expansion from G' to G . We assume that G' has an optimal defending strategy that holds several nice properties from the induction. We show that a part of the graph G' along with its strategy can be exchanged for a different one by showing that Definition 2.4.26 holds for them. Such parts are then exchanged using Definition 2.4.28 which expands G' into G while showing that an upper bound devised by Observation 2.4.30 applies. This gets us an upper bound $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + K$ (the same K as in the lower bound). Combining the lower and upper bound using Lemma 2.4.2 gets us the optimal number of guards for G .

The used reduction depends on a leaf component that the cactus graph contains by Observation 2.5.3. One case is that the subgraph is a tree and the second case is that there is a *leaf cycle* – a cycle with leaves which is connected to the rest of the graph via a single articulation. We split the reductions into three groups.

The first group called *leaf reductions* shown in Section 2.3 has a few simple reductions of leaves which are not incident to a leaf cycle. These were shown to be sufficient to determine the γ_m^∞ for any tree by Goddard, Hedetniemi, and Hedetniemi [58]. We reintroduce these reductions in our framework and show more general results so that the reductions can be used over tree subgraphs of non-tree graph classes. They also serve as an introductory example of how to use the tools from Section 2.4.

Further reductions are more involved and require non-trivial manipulation with strategies. It is beneficial to establish strategies with nice properties in the induction to allow stronger induction step. In Section 2.5.2, we show the properties which are used in the two other groups of reductions.

The last two groups called *cycle reductions* and *constant component reductions* are shown in Sections 2.5.3 and 2.5.4. Cycle reductions concern substructures that appear on leaf cycles. We fix a leaf cycle and use these reductions repeatedly on it. Each reduction shortens the leaf cycle. Eventually, the cycle is very short and is reduced by constant component reductions. After these reductions, the leaf cycle is removed entirely and only zero, one, or two leaves are left in its place. Such leaves are then processed either as tree leaves or leaf vertices adjacent to another leaf cycle.

2.3 Reducing Trees

In this section, we intuitively present tools to achieve lower and upper bounds and which will be formally introduced in Section 2.4. We focus on tree reductions, which were first described by Goddard, Hedetniemi, and Hedetniemi [58] as a part of the linear algorithm for computing the m-eternal domination number γ_m^∞ on trees. We now show this set of reductions along with the proofs of their correctness in Lemmas 2.3.1 to 2.3.3.

For a graph G , let us have a vertex $u \in V(G)$ which is adjacent to $\ell \geq 1$ leaves and has degree d . Let v be one of the leaves adjacent to u . We define three leaf reductions of G to G' as follows. See Table 2.1 for illustration of respective bound proofs.

Reduction 1. t_1 If $\ell = 1$ and $d \leq 2$, let $G' = G \setminus \{u, v\}$.

Reduction 2. t_2 If $\ell > 2$, let $G' = G \setminus \{v\}$.

Reduction 3. t_3 If $\ell = 2$ and $d = 3$, let $G' = G \setminus \{\text{all leaves adjacent to } u\}$.

Reductions t_2 and t_3 can be joined to a single reduction which removes all leaves of a vertex with $\ell \geq 2$ and $d = \ell + 1$ (used in [58]). However, Reduction t_2 may be used in wider range of scenarios as it does not require a specific value of d .

Assume now, that we know the optimum number of guards for G' (for both Γ_m^∞ and γ_m^∞). Our goal is to show two things. By showing that G always uses at least K more guards than G' we get a lower bound on the number of guards necessary for G . By showing that there is a strategy for G which uses at most K more guards than an optimum strategy on G' we get an upper bound on the number of guards on G . Together, these bounds give us an optimum number of guards for G . This concept is formally introduced in Lemma 2.4.2.

Table 2.1: Leaf reductions; Lower bound side depicts clique reductions (removal of marked vertices and joining its neighborhood with a clique); Upper bound side labels vertices with Greek letters of states where they belong, and arrows show how one state transitions to another. The marked groups of vertices are created with Definition 2.4.31.

Reduction	Lower bound	Upper bound
t_1		
t_2		
t_3		

Having the reductions in hand see Figure 2.1 for an example of how the reductions are used to construct a strategy for a tree.

We now proceed to show the bounds obtained from these reductions. Generally, the proofs contain lower bound and upper bound portion, see Table 2.1 for accompanying illustrations. Lower bounds can be shown quite easily – delimit a connected part of a graph which is guaranteed to contain K guards, remove it, and join its neighborhood with a clique. Upper bounds are more tricky – we assume some optimal strategy on G' , which has nice properties, then we expand it to G while preserving the properties. The notation used in the following proofs is defined in Section 2.4.

We say that graph G is *defended* with k guards if $k = \gamma_m^\infty(G) = \Gamma_m^\infty(G)$ and the strategy using k guards is *proper* in the sense of Property 2.5.11, which is defined in Section 2.5. This allows us to state the lemmas concisely. Let us now see the proofs for the three leaf reductions.

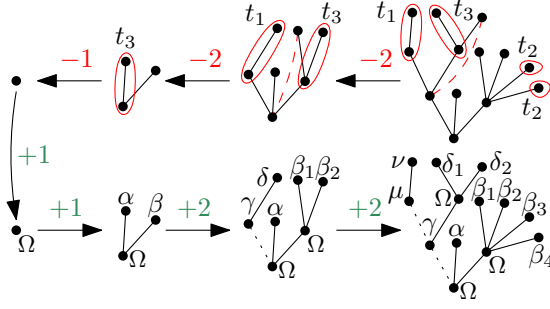


Figure 2.1: An example application of leaf reductions on a tree graph. Dotted lines signify that the strategy does not use that edge, and the strategies on subtrees are independent, which is caused by Reduction t_1 . Note how Reduction t_3 can be used even when there is no vertex a .

Lemma 2.3.1. Let G' be G after application of Reduction t_1 . G is defended with 1 more guard than G' .

Proof. As $\{u, v\}$ is a leaf and its neighbor, there is always at least one guard so we may apply Observation 2.4.8 on $\{u, v\}$ to get a lower bound of $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 1$. To get an upper bound $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 1$, we dedicate one new guard to defend $\{u, v\}$ independently on the rest of the strategy. Putting the lower bound and upper bound together using Lemma 2.4.2 we get that G is defended with 1 more guard than G' . \square

Note, that the final strategy graph after Reduction t_1 is a Cartesian product of the strategy graph on G' and a graph with single edge. Cartesian product is a basis for Definition 2.4.31 where we introduce an operation which joins strategies even if the strategies are not entirely independent. We shall use this operation along with a property shown in Lemma 2.4.38 – that a strategy can be altered so that a vertex adjacent to at multiple leaves is always occupied.

To ease notation, we shall reserve the prime symbol ($'$) to denote structures of the reduced instance such as the graph G' , defending strategy \mathcal{B}' , strategy graph $S'_{G'}$, its states (vertices) Ω' and transitions (edges) \mathbb{F}' , etc.

Lemma 2.3.2. Let G' be G after application of Reduction t_2 . G is defended with the same number of guards as G' .

Proof. Lower bound of 0 is obtained by using Observation 2.4.3 to identify v with u so $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G')$.

For upper bound, from induction we have a defending labelled strategy \mathcal{B}' of G' . We wish to alter it so it defends vertex v as well. We apply Lemma 2.4.38 to alter \mathcal{B}' so that u is occupied in each state of Ω' . Let w be a leaf adjacent to u distinct from v . We partition all states (vertices) of the strategy $S'_{G'}$ as follows. A state α' belongs to $\mathcal{S}'(w)$ if in α' vertex w is occupied.

Now we perform graph Cartesian product of $S'_{G'}$ with a single edge $\{\alpha, \beta\}$ over subset $\mathcal{S}'(w)$ (Definition 2.4.31). Written in short as $S_{G'} = S'_{G'} \square_{\mathcal{S}'(w)} \{\alpha, \beta\}$. This splits all

vertices of the strategy where w is occupied into two. We denote the new sets as α and β . This got us a new strategy graph $S_{G'}$ over the reduced graph G' . Now we expand from G' to G while altering the strategy slightly. In β we substitute the guard on w with a guard on v . The guards shall transition between states of α and β as $\mathcal{T}(\alpha, \beta) = \{(w, u), (u, v)\}$ while the rest of them shall not move. As w and v are siblings it follows that we can transition from any $\gamma \in \Omega$ to w the same way as to v if they were swapped. This remains defended by Lemma 2.4.33. Hence, $\gamma_m^\infty(G) \leq \gamma_m^\infty(G')$ and by Lemma 2.4.2 we get that G is defended with the same number of guards as G' . \square

The shown strategy basically defends v in the “same way” it defends w . We can do this when one can transition from one to the other in a single step while the remaining guards remain stationary. For a detailed explanation see Definition 2.4.31 and its lemmas that show its properties.

The previous reduction bounds were proven with extensive explanation. In the following we just use the tools to arrive at the result directly. Note that the very similar argument could be used to obtain an arbitrary number of leaves.

Lemma 2.3.3. Let G' be G after application of Reduction t_3 . G is defended with 1 more guard than G' .

Proof. Lower bound of 1 is obtained by using Observation 2.4.8 on vertices $\{u, v\}$, which results in a graph isomorphic to one that is created by removing all leaves adjacent to u which gets us $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 1$. For upper bound, we apply Lemma 2.4.34 which adds the two leaves to u using one extra guard which directly results in $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 1$. Using Lemma 2.4.2 we get that G is defended with 1 more guard than G' . \square

Note that the bounds devised for Reductions t_1 , t_2 , and t_3 do not require the graph to be a tree. We may use these reductions in any graph class. Hence, we may reduce any leaves in subtrees which appear as parts of other graphs. In particular, Reduction t_2 may be also used to reduce the number of leaves adjacent to any vertex to 2 because connections of u to other vertices do not interfere with the reduction. Note that in that case, we obtain lower bounds for the M-ETERNAL GUARD CONFIGURATION and upper bounds for the M-ETERNAL DOMINATION.

It was previously shown by Goddard, Hedetniemi, and Hedetniemi [58] that these reductions (originally given in slightly different form) are sufficient to solve any tree graph. Note that this can be shown by rooting the tree and repeatedly applying Reductions t_1 , t_2 , and t_3 on the parent of the deepest leaf.

Also note, that the reductions t_1 and t_3 do not require the rest of the graph (signified by vertex a) to be there at all, hence, these solve base-cases where only a single edge or a star remain.

When the reductions are used on a tree we get a partitioning of vertices into subtrees which are defended independently. These components constitute a *neo-colonization*, a notion introduced by Goddard et al. [58] and often used in contemporary papers.

2.4 The m-eternal domination Toolbox

This section gives tools to show lower and upper bounds for the m-eternal domination problem. Before we present the approach in detail we show several key ideas and a detailed structure for the rest of this section. Throughout this chapter, we reserve *prime* (e.g. G' and α') to denote structures of the reduced instance.

Observation 2.4.1. $\gamma(G) \leq \Gamma_m^\infty(G) \leq \gamma_m^\infty(G) \leq 2 \cdot \gamma(G)$ for any graph G .

Proof. Every m-Eternal Domination strategy can be applied as an m-Eternal Guard Configuration strategy so $\gamma_m^\infty \geq \Gamma_m^\infty$. Every configuration in each of these strategies must induce a dominating set. Therefore, they are all lower bound by the domination number γ .

It is also known that an m-Eternal Domination strategy can be constructed by defending neighborhood of each vertex in the dominating set independently of each other (with a simple strategy for stars) that uses at most $2 \cdot \gamma(G)$ guards as shown by Klostermeyer and Mynhardt. [77]. \square

We now show the lemma which sums up how the bounds on the optimal strategies are obtained.

Lemma 2.4.2. Let us assume that for graphs G, G' , and an integer constant k

$$\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + k, \quad (2.1)$$

$$\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + k, \quad (2.2)$$

$$\gamma_m^\infty(G') = \Gamma_m^\infty(G'). \quad (2.3)$$

Then $\gamma_m^\infty(G) = \Gamma_m^\infty(G) = \gamma_m^\infty(G') + k = \Gamma_m^\infty(G') + k$.

Proof. Given the assumptions, we have

$$\gamma_m^\infty(G) \stackrel{(2.1)}{\leq} \gamma_m^\infty(G') + k \stackrel{(2.3)}{=} \Gamma_m^\infty(G') + k \stackrel{(2.2)}{\leq} \Gamma_m^\infty(G) \stackrel{\text{Obs. 2.4.1}}{\leq} \gamma_m^\infty(G).$$

As the first and the last term is identical all these values are equal. \square

Hence, it suffices to prove that for G and its reduction G' we have $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + k$ and $\Gamma_m^\infty(G') \leq \Gamma_m^\infty(G) - k$. If we already have the optimal strategy for G' , then our constructive upper bounds together with Lemma 2.4.2 give us an optimal strategy for G .

We present the tools for obtaining lower bounds in Section 2.4.1, terminology and new concepts for upper bounds in Section 2.4.2, and tools which use the new concepts to obtain upper bounds in Section 2.4.3. See Figure 2.2 for a detailed section overview.

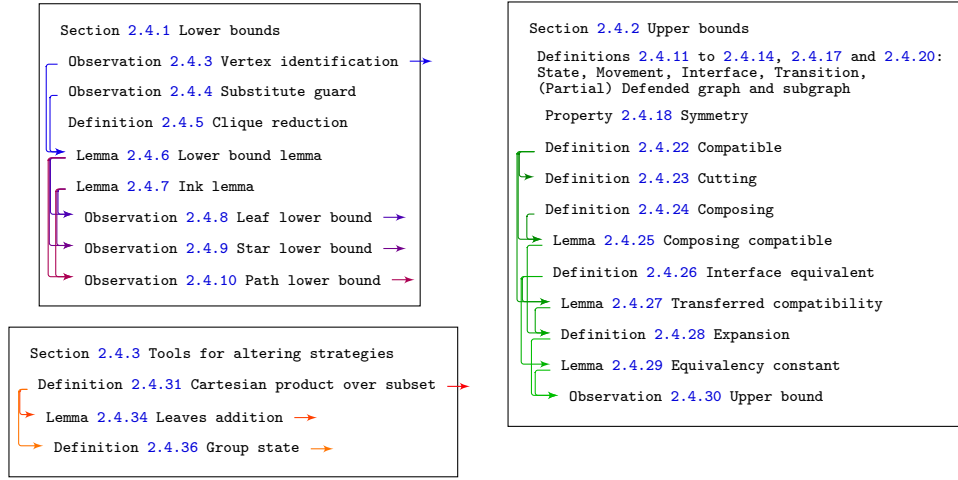


Figure 2.2: Overview of Section 2.4. Boxes represent respective subsections; Arrows on the left side show which notions are used to prove other notions; Right arrows show which notions are frequently used in Section 2.5 to obtain results for cactus graphs.

2.4.1 Lower Bounds

We start this section with a few elementary observations about strategies. Then, we show a pair of lemmas which are the main tools in obtaining lower bounds. Last, using these lemmas, we obtain three lower bound observations which we use frequently in Section 2.5.

We say that G' is a result of *identifying* u with v in G if it is a result of removing u while adding the edges so that $N_{G'}(v) = N_G(u) \cup N_G(v)$.

Observation 2.4.3 (Vertex identification). Let G be a graph and u and v be its two distinct vertices. Then for a graph G' , which is a result of identifying u with v in G , $\Gamma_m^\infty(G') \leq \Gamma_m^\infty(G)$.

Proof. Let $v' \in V(G')$ be the vertex created by identifying u with v in G . Let S_G be an optimal strategy of G . Let $S_{G'}$ be a strategy on G' which is the same as S_G except that in every configuration each u and v is substituted by v' .

Any pair of traversable configurations in S_G is still traversable in $S_{G'}$ as in every movement u and v can be replaced by v' . Any attack on $V(G') \setminus \{v'\}$ is defended by a configuration in $S_{G'}$ which was created from a respective configuration of S_G , and v' is defended by configuration which defended u in G . \square

Observation 2.4.4. If a graph has a clique on distinct vertices u, v, w , then guard movements (u, v) and (v, w) can be substituted with movement (u, w) and a stationary guard on v .

Along with vertex identification the following reduction is the main tool for obtaining lower bounds.

Definition 2.4.5 (Clique reduction). Let G be a graph and H be its non-empty induced connected subgraph. By *clique reduction* of H in G we mean creation of a new graph G' that is the result of removing H from G and mutually connecting all neighbors of H in $G \setminus H$ by an edge.

See Figure 2.3 for an illustration of a clique reduction. Using Observations 2.4.3 and 2.4.4 we now show that the clique reduction implies a lower bound on G which can be later used to show tight strategy lower bounds.

Lemma 2.4.6 (Lower bound lemma). Let G be a graph and H be its non-empty induced connected subgraph such that in at least one optimal m-eternal guard strategy, there are always at least k guards present on H . Let G' be the result of a clique reduction of H in G . Then

$$\Gamma_m^\infty(G') \leq \Gamma_m^\infty(G) - k.$$

Proof. If there is no neighbor of H in G , then clique reduction removes H and adds no edges. We can remove all the guards which were standing on H so G' is clearly defended by $\Gamma_m^\infty(G) - k$ guards.

Otherwise, there is a neighbor of H in $G \setminus H$, say v . Let $S_G = (\mathbb{C}, \mathbb{F})$ be an optimal strategy on G . We use Observation 2.4.3 to identify all vertices $V(H)$ with v in G to obtain a subgraph of G' along with a strategy $S_{G'}$. Note that in G' each configuration of $S_{G'}$ has at least k guards on v because before identification H always contained k guards. Also, configurations which defend v in $S_{G'}$ have at least $k + 1$ guards on v by the same argument.

Let $S_{G'}^-$ be a strategy which is the same as $S_{G'}$, except it has k less guard on v in each configuration. We see that each configuration which defended v in S_G had at least $k + 1$ guards on v so it defends v in $S_{G'}^-$. Guards which defended $V(G) \setminus (V(H) \cup \{v\})$ remain unchanged. It remains to check whether configurations which were traversable in S_G remain traversable in $S_{G'}^-$.

Our goal is to show that there exists a set of movements between each configurations of $S_{G'}$ which have k stationary guards on v which are not needed for defense of G' . Such guards then may be removed to obtain $S_{G'}^-$ and the remaining movements show that the respective configurations are traversable.

Each movement (u, h) and (h, w) such that $h \in V(H)$ in S_G has its respective pair of movements $(u, v), (v, w)$ in $S_{G'}$. As u and w are neighbors of $V(H)$ then there is an edge $\{u, w\} \in E(G')$ added by the construction of G' . By Observation 2.4.4 we may substitute movements $(u, v), (v, w)$ with (u, w) and a stationary guard on v .

Assume there are less than k stationary guards on v in $S_{G'}$ after applying the substitution exhaustively. Then there must be at most $k - 1$ stationary guards and at least one guard which leaves v or at least one guard which arrives to v , but there may not be both (one leaving and one arriving) as they would form $(u, v), (v, w)$ pair and the substitution could be applied. When the guard is leaving or arriving there are at most $k - 1$ guards in the final or starting configuration, respectively, which is a contradiction because there are at least k guards on v .

Removing k guards from v in $S_{G'}$ yields $S_{G'}^-$ where each configuration pair remains traversable, which concludes the proof. \square

To use Lemma 2.4.6 we need to show that an induced subgraph H of G is always occupied by at least k guards. To do that we have the following lemma.

Lemma 2.4.7 (Ink lemma). Let H be an induced subgraph of G . Let (v_1, v_2, \dots, v_k) be a sequence of vertices in H such that it holds $d(u, v_i) > i$ for every i and for every $u \in V(G) \setminus V(H)$, and also for every $j < i$ it holds that $d(v_j, v_i) > i - j$. Then there are at least k guards on H in every defending m-eternal guard configuration.

Proof. Let C be any fixed configuration of a defending strategy. We show that C contains at least k guards on H .

Assume that the attacker performed a sequence of attacks (v_1, v_2, \dots, v_k) one by one. At the i -th step of the attack sequence the following is true. Guards who were standing on $V(G) \setminus V(H)$ at the beginning of the attack sequence are more than i edges far from v_i so they cannot reach v_i in time to defend it. Similarly, any guard that defended v_j with $j < i$ can not defend the attack on v_i as their distance from v_i is more than $i - j$ at the time they defended v_j . Therefore, none of the guards can reach v_i in time and we need an additional guard placed on H .

In total, we need k guards on H in a configuration to be able to defend the attack sequence. This is true for any configuration so every defending strategy must have k guards on H in every configuration. \square

The operation in Lemma 2.4.6 together with the lower bound obtained from Lemma 2.4.7 allows us to make a graph smaller while showing that the removed part required some minimum number of guards. See an example usage of Lemmas 2.4.6 and 2.4.7 in Figure 2.3.

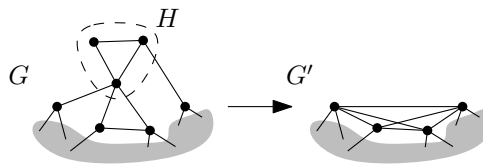


Figure 2.3: A graph G with an induced subgraph H . Graph G' is obtained by a clique reduction from Definition 2.4.5. By Lemma 2.4.7 we have that every configuration contains at least 1 guard on H . Hence, by Lemma 2.4.6 we have $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 1$.

Observation 2.4.8. In graph G , let v be a leaf vertex and let u be its neighbor, and let $H = G[\{u, v\}]$. By Lemma 2.4.7 with a sequence (v) we obtain that 1 guard is on H . In other words, the closed neighborhood of every leaf must contain at least one guard otherwise an attack on the leaf could not be defended. Let G' be a graph obtained by using operation of Lemma 2.4.6 on H , this gives us $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 1$.

Observation 2.4.9. In graph G , let u be a vertex which is adjacent to at least two leaves $\{v_1, v_2, \dots\}$, and let $H = N[u]$ denote the closed neighborhood of u . By Lemma 2.4.7 with a sequence (v_1, v_2) we obtain that 2 guards are on H . In other words, the closed neighborhood of u must contain at least 2 guards otherwise two consecutive attacks on different leaves adjacent to u could not be defended. Let G' be a graph obtained by using operation of Lemma 2.4.6 on H , this gives us $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 2$.

Observation 2.4.10. Let us have a graph G and its induced subgraph H that is isomorphic to a path on three vertices. We label these three vertices of H as u_1, u_2, u_3 (in order). By Lemma 2.4.7 with a sequence (u_2) we have lower bound of 1 on the number of guards on H . Let G' be a graph obtained by using operation of Lemma 2.4.6 on $\{u_1, u_2, u_3\}$. This gives us $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 1$.

2.4.2 Upper Bounds

This section introduces notation to describe strategies which are used to achieve upper bounds for γ_m^∞ . We assume that we have a graph G and its reduced copy G' . The main idea is that a strategy for G' can be locally changed to obtain a strategy for G . To accommodate this local change, we show how to cut and compose parts of the graph while preserving its strategy. At the end of this section, we present a set of sufficient rules that allow such a local change. Then, in Section 2.4.3, we present tools which we use to obtain upper bounds.

In our constructions, we need to have control over movements of the guards. We also need a way to represent only part of the strategy over an induced subgraph of G . To do so, we introduce states (labelled configurations) and labelled strategy that prescribes the guard movements on state transition.

Definition 2.4.11 (States). Let *states* be a set of labels Ω and let *state vertex mapping* P of Ω to $V(G)$ be $P: \Omega \rightarrow 2^{V(G)}$, i.e., a state $\alpha \in \Omega$ represents a subset of vertices $P(\alpha) \subseteq V(G)$ (also called guards) of a graph G . Let $\mathcal{S}(v) = \{\beta \mid \beta \in \Omega, v \in P(\beta)\}$ (states that contain v) for every $v \in V(G)$.

We will use Greek letters such as $\alpha, \beta, \gamma, \delta, \varphi$ to signify states or sets of states. *Move* of a guard is still an ordered pair of vertices (u, v) such that $\{u, v\} \in E(G)$ or $u = v$ (*stationary* guard).

Building towards a comprehensive definition of a labelled strategy, we first build a more general concept – partial labelled strategy. This will allow us to do cutting and composing with a well defined strategy over a subgraph.

Definition 2.4.12 (Interface). Let an *interface* R of a graph G with respect to its supergraph H be a subset of vertices such that

$$R = \{u \mid (\exists v) u, v \in V(H), \{u, v\} \in E(H), u \in V(G), v \notin V(G)\},$$

i.e., those vertices of G which have a neighbor in $V(H) \setminus V(G)$ in H .

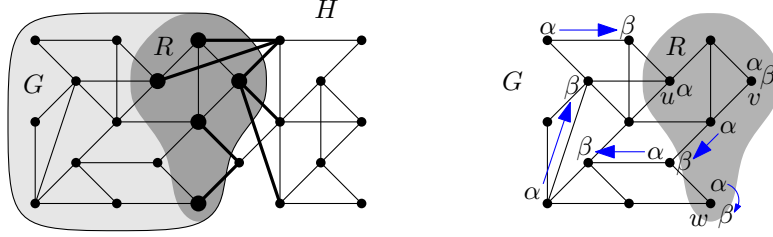


Figure 2.4: Left: An interface R of G with respect to H . Bold edges signify the cut between $V(G)$ and $V(H) \setminus V(G)$ that is responsible for the vertices in R ; Right: Transition $\mathcal{T}(\alpha, \beta)$ from α to β . Arrows signify movements of the transition. All vertices in states α and β must be paired up with a movement if they are not in the interface R . In the interface, a guard moves from u to the rest of the graph outside of G so a movement is missing for u . Note the difference between v and w : in w the same guard stays on the vertex, in v (as there is no (v, v) movement) the guard on v moves out and a different one moves to v .

See Figure 2.4 for an example of an interface. The interface marks the vertices where the strategy may be incomplete. The transitions between states incorporate the interface by allowing the moves to be incomplete in the following way.

Definition 2.4.13 (Transition). For states α and β and a graph G with an interface R , let a *transition* (from α to β) denoted by $\mathcal{T}(\alpha, \beta)$ be a set of moves such that

- $\mathcal{T}(\alpha, \beta) \subseteq \{(u, v) \mid u \in P(\alpha), v \in P(\beta), \{u, v\} \in E(G) \vee u = v\}$,
- for each $u \in P(\alpha) \setminus R$ there exists exactly one $(u, v) \in \mathcal{T}(\alpha, \beta)$,
- for each $v \in P(\beta) \setminus R$ there exists exactly one $(u, v) \in \mathcal{T}(\alpha, \beta)$,
- for each $u \in P(\alpha) \cap R$ there exists at most one $(u, v) \in \mathcal{T}(\alpha, \beta)$, and
- for each $v \in P(\beta) \cap R$ there exists at most one $(u, v) \in \mathcal{T}(\alpha, \beta)$.

See Figure 2.4 for an example of a transition and how it interacts with an interface. Note that if the interface R is empty, then the transition yields a bijection between guards of the states, which gives an exact prescription on how they move between the two states.

Transition gives us that each guard can be in relation with at most one other guard. In our case, there is at most one guard on each vertex. Hence, we may use the standard relation terminology for the set of pairs defined by a transition.

Definition 2.4.14 (Partial labelled strategy). A *partial labelled strategy* is $(G, S_G, P, \mathcal{T}, R)$ where G is a graph, $S_G = (\Omega, \mathbb{F})$ is a *strategy graph* such that Ω is a set of vertices (states) and \mathbb{F} is a set of edges, P is a state vertex mapping of Ω to $V(G)$, $R \subseteq V(G)$ is an interface of G , and \mathcal{T} maps orientations (α, β) and (β, α) of every edge $\{\alpha, \beta\} \in \mathbb{F}$ to transitions $\mathcal{T}(\alpha, \beta)$ and $\mathcal{T}(\beta, \alpha)$, respectively.

For various purposes, it may be beneficial to think of the strategy graph S_G as an oriented graph (allowing non-symmetric transitions, see Property 2.4.18), or even multigraph (allowing multiple different transitions between the same set of states).

The labelled strategy may defend attacks indefinitely if it is in accordance with the following definition.

Definition 2.4.15 (Defending). A partial labelled strategy $(G, (\Omega, \mathbb{F}), P, \mathcal{T}, R)$ is *defending* G if for every state $\alpha \in \Omega$ and each vertex $v \in V(G)$ there is $\beta \in \Omega$ such that $\{\alpha, \beta\} \in \mathbb{F}$ and $v \in P(\beta)$.

In other words, Definition 2.4.15 says that for each vertex of the graph every state is either occupying it or a state which occupies it is reachable with only one transition. This directly leads to the following observation. Recall that $\mathcal{S}(u)$ denotes set of states that contain u .

Observation 2.4.16. A strategy is defending a graph if for every $u \in V(G)$ set $\mathcal{S}(u)$ is a dominating set of the strategy graph S_G .

Definition 2.4.17 (Labelled strategy). A *labelled strategy* is $\mathcal{D} = (G, S_G, P, \mathcal{T})$ such that $(G, S_G, P, \mathcal{T}, \emptyset)$ is a partial labelled strategy.

Note, that all the states in the labelled strategy must contain the same number of guards because the transitions are bijections. When the strategy is optimal the number of guards corresponds to γ_m^∞ .

Partial labelled strategies can have several nice properties, which we present now. When the strategy graph is unoriented it is natural to require symmetry of transitions.

Property 2.4.18 (Symmetry). A partial labelled strategy $\mathcal{B} = (G, (\Omega, \mathbb{F}), P, \mathcal{T}, R)$ is *symmetrical* if and only if $\mathcal{T}(\alpha, \beta)$ is a converse relation to $\mathcal{T}(\beta, \alpha)$ (i.e., $\mathcal{T}(\alpha, \beta) = \{(a, b) \mid (b, a) \in \mathcal{T}(\beta, \alpha)\}$) for every $\{\alpha, \beta\} \in \mathbb{F}$.

Lemma 2.4.19. For each partial labelled strategy $\mathcal{B} = (G, (\Omega, \mathbb{F}), P, \mathcal{T}, R)$ there exists a symmetrical partial labelled strategy $\mathcal{B}' = (G, (\Omega, \mathbb{F}), P, \mathcal{T}', R)$.

Proof. For each pair of states $\{\alpha, \beta\} \in \mathbb{F}$ fix an arbitrary orientation (α, β) and take the $\mathcal{T}(\alpha, \beta)$ with an interface R which gives us $\mathcal{T}(\alpha, \beta)$. Note that by swapping u with v and α with β in Definition 2.4.13 we obtain the same definition but for $\mathcal{T}(\beta, \alpha)$. We substitute the transition $\mathcal{T}(\beta, \alpha)$ for this newly found transition. Performing this substitution for every pair of states in \mathbb{F} gives us the desired \mathcal{T}' . \square

By Lemma 2.4.19, we will always assume that the partial labelled strategy is symmetrical. We also use this property to infer transitions. We show only one direction of the transition mapping and let the other direction be the converse transition given by symmetry.

It is not easy to grasp the labelled strategy description only from the formal notation so we shall draw many auxiliary pictures. Vertices and edges shall be depicted by small circles

(or squares) and line segments, respectively; vertices may be labelled by their letter name; by Greek letters we signify the states which contain respective vertices; guard moves in transitions are depicted by differently styled arrows on edges which point between the state labels (Note that the arrows are always shown only in one direction because we assume Property 2.4.18.); the interface vertices are marked by gray-filled areas; see Figure 2.5 for an example of a labelled strategy.

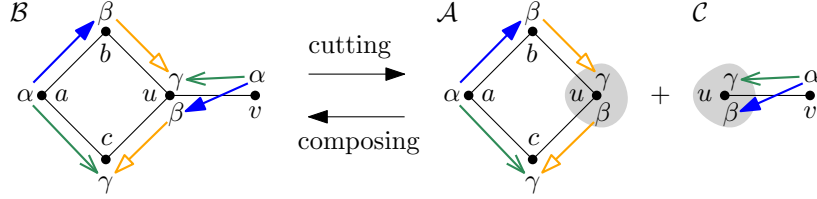


Figure 2.5: Example of a labelled strategy \mathcal{B} and two partial labelled strategies \mathcal{A} and \mathcal{C} ; the full formal description of the labelled strategy $\mathcal{B} = (G, (\Omega, \mathbb{F}), P, \mathcal{T})$ is $G = (V, E)$, $V = \{a, b, c, u, v\}$, $E = \{\{a, b\}, \{b, u\}, \{u, v\}, \{a, c\}, \{c, u\}\}$, $\Omega = \{\alpha, \beta, \gamma\}$, $\mathbb{F} = \{\{\alpha, \beta\}, \{\alpha, \gamma\}, \{\beta, \gamma\}\}$, $P(\alpha) = \{v, a\}$, $P(\beta) = \{u, b\}$, $P(\gamma) = \{u, c\}$, ($R = \emptyset$), $\mathcal{T}(\alpha, \beta) = \{(a, b), (v, u)\}$, $\mathcal{T}(\alpha, \gamma) = \{(a, c), (v, u)\}$, $\mathcal{T}(\beta, \gamma) = \{(b, u), (u, c)\}$; the formal descriptions of partial labelled strategies \mathcal{A} and \mathcal{C} are similar while restricted to their subgraph and they contain an interface $R = \{u\}$.

We will need to cut part of the labelled strategy and put something slightly different in its place. To tackle that we put forward the following notions.

Definition 2.4.20 (Partial labelled substrategy). The *partial labelled substrategy* \mathcal{B}' of a labelled strategy $\mathcal{B} = (G, (\Omega, \mathbb{F}), P, \mathcal{T})$ for some induced subgraph G' of G is a partial labelled strategy $\mathcal{B}' = (G', (\Omega, \mathbb{F}), P', \mathcal{T}', R)$ where R is an interface of G' with respect to G , for all $\alpha \in \Omega$ it holds $P'(\alpha) = P(\alpha) \cap V(G')$, and for all $\beta, \gamma \in \Omega$ it holds $\mathcal{T}'(\beta, \gamma) = \{(a, b) \mid (a, b) \in \mathcal{T}(\beta, \gamma) \wedge a, b \in V(G')\}$.

It is not immediately obvious that the Definition 2.4.20 made a partial labelled substrategy in a way that it constitutes a partial labelled strategy; so we show that next.

Observation 2.4.21. A partial labelled substrategy $\mathcal{B}' = (G', (\Omega, \mathbb{F}), P', \mathcal{T}', R)$ of a labelled strategy $\mathcal{B} = (G, (\Omega, \mathbb{F}), P, \mathcal{T})$ is a partial labelled strategy.

Proof. G' is a graph, Ω is a set of labels, and R is subset of $V(G')$ which is in accordance to Definition 2.4.14. P' was created by restricting P to the vertices of $V(G')$. We only removed some guards from the mapping so this is okay by Definition 2.4.14. Last, the only guards which are not included in \mathcal{T}' are those whose moves in \mathcal{T} went outside of G' . Assume such guard on vertex u with a move (u, v) where $v \notin G'$, hence, $u \in R$ by Definition 2.4.12. As stated in Definition 2.4.13 any guard in R does not have to be included in a move so any partial labelled substrategy is a partial labelled strategy. \square

Now we present conditions which are necessary to be able to combine two partial labelled strategies into one labelled strategy. Based on that, we show how to split a labelled strategy into two partial labelled strategies. Figure 2.5 shows an example of the following operations.

Definition 2.4.22 (Compatible). Two partial labelled strategies \mathcal{B}_1 and \mathcal{B}_2 (denoted as $\mathcal{B}_i = (G_i, (\Omega_i, \mathbb{F}_i), P_i, M_i, R_i)$) are called *compatible* if the following conditions hold true.

- $R_1 = R_2 = V(G_1) \cap V(G_2)$, i.e., their graphs overlap exactly in the interface,
- $(\Omega_1, \mathbb{F}_1) = (\Omega_2, \mathbb{F}_2)$, i.e., the strategy graphs are the same,
- $M_1(\alpha, \beta) \cup M_2(\alpha, \beta)$ is a bijection between $P_1(\alpha) \cup P_2(\alpha)$ and $P_1(\beta) \cup P_2(\beta)$ for every $\alpha, \beta \in \Omega_1$.

The conditions for compatible partial labelled strategies ensure that the interfaces overlap in a way that a composed function will be a bijection which allows us to cut and compose them in the following way.

Definition 2.4.23 (Cut). Let us have a labelled strategy \mathcal{B} and a vertex cut R which partitions the vertices of $G(\mathcal{B})$ into R , A and C in such a way that there are no edges between A and C . We say \mathcal{B} is *cut* along R into two partial labelled substrategies \mathcal{A} and \mathcal{C} where \mathcal{A} is a partial labelled substrategy induced by $V(G(\mathcal{A})) = R \cup A$ and \mathcal{C} is a partial labelled substrategy induced by $V(G(\mathcal{C})) = R \cup C$ such that \mathcal{A} and \mathcal{C} are compatible.

Definition 2.4.24 (Composing). By *composing* two partial labelled strategies \mathcal{B}_1 and \mathcal{B}_2 ($\mathcal{B}_i = (G_i, (\Omega_i, \mathbb{F}_i), P_i, M_i, R_i)$) we mean getting $(G^*, (\Omega, \mathbb{F}), P^*, \mathcal{T}^*)$ where $G^* = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$, $\Omega = \Omega_1 \cup \Omega_2$, $\mathbb{F} = \mathbb{F}_1 \cup \mathbb{F}_2$, $\forall \gamma \in \Omega$ we have $P^*(\gamma) = P_1(\gamma) \cup P_2(\gamma)$, and $\mathcal{T}^*(\alpha, \beta) = M_1(\alpha, \beta) \cup M_2(\alpha, \beta)$ for every $\alpha, \beta \in \Omega$.

Lemma 2.4.25. Composing two compatible partial labelled strategies yields a labelled strategy.

Proof. Let us use the notation of Definitions 2.4.22 and 2.4.24. We need to check whether $(G^*, (\Omega, \mathbb{F}), P^*, \mathcal{T}^*, \emptyset)$ is a partial labelled strategy. First, G^* is a graph where we unite vertices and edges, while only the interface vertices are overlapping; this constitutes a well defined graph without multiedges and loops. The states Ω_1 are the same for the compatible \mathcal{B}_1 and \mathcal{B}_2 . Next, mapping of the states to vertices is done by uniting the individual sets $P_1(\gamma) \cup P_2(\gamma)$ for each $\gamma \in \Omega_1$. Each vertex is now guarded in the union of states it was guarded before. Last, we check whether the union of M_1 and M_2 always maps to a well defined transition, however, this is ensured by compatibility conditions over P_i and M_i in Definition 2.4.22. \square

We define an equivalency relation (reflexive, symmetric, and transitive) with respect to the interfaces as follows.

Definition 2.4.26 (Interface equivalent). Two partial labelled strategies \mathcal{B}_1 and \mathcal{B}_2 ($\mathcal{B}_i = (G_i, (\Omega_i, \mathbb{F}_i), P_i, M_i, R_i)$) are *interface equivalent* if $G[R_1] = G[R_2]$, $\Omega_1 = \Omega_2$, $\mathbb{F}_1 = \mathbb{F}_2$, for all $\alpha \in \Omega_1$ we have $P_1(\alpha) \cap R_1 = P_2(\alpha) \cap R_2$, and we have $(a, b) \in M_1(\beta, \gamma) \Leftrightarrow (a, b) \in M_2(\beta, \gamma)$ for all u such that $a = u \vee b = u$ for all $\beta, \gamma \in \Omega_1$.

Interface equivalent partial labelled strategies have the same states with respect to the interface. This allows us to infer compatibility as stated in the following lemma.

Lemma 2.4.27. For three partial labelled strategies \mathcal{B}_1 , \mathcal{B}_2 , and \mathcal{B}_3 if \mathcal{B}_1 is compatible with \mathcal{B}_2 , \mathcal{B}_2 is interface equivalent with \mathcal{B}_3 , and $V(G(\mathcal{B}_1)) \cap V(G(\mathcal{B}_3)) = R(\mathcal{B}_3)$, then \mathcal{B}_1 is compatible with \mathcal{B}_3 .

Proof. Let $\mathcal{B}_i = (G_i, (\Omega_i, \mathbb{F}_i), P_i, M_i, R_i)$. We will check the conditions stated in Definition 2.4.22. As $V(G_1) \cap V(G_3) = R_3$ and $R_3 = R_2$ by interface equivalency, and $R_2 = R_1$ by compatibility, the first condition holds. As $G_3[R_3] = G_2[R_2]$ and $V(G_1) \cap V(G_3) = R_3$ there are no possible edges which would be shared by G_1 and G_3 outside of R_3 . $\Omega_1 = \Omega_2$ by their compatibility, $\Omega_2 = \Omega_3$ by interface equivalency, so $\Omega_1 = \Omega_3$. The \mathcal{B}_3 is a partial labelled strategy so each guard on vertex in $V(G_3) \setminus R_3$ is covered by M_3 exactly once. The guards on R_3 are covered exactly when they were covered on R_2 . As \mathcal{B}_1 and \mathcal{B}_2 are compatible the guards on R_2 were covered by M_1 exactly when they were not covered by M_2 and vice-versa. Hence, this property still holds for \mathcal{B}_1 and \mathcal{B}_3 . \square

The culmination of the previous notions and lemmas is the following procedure which we use as one major part for proving upper bounds.

Definition 2.4.28 (Expansion). Let us have a labelled strategy \mathcal{B} with a partial labelled substrategy \mathcal{C} . Let us also have a partial labelled strategy \mathcal{C}' which is interface equivalent with \mathcal{C} . An *expansion* of \mathcal{B} from \mathcal{C} to \mathcal{C}' is the following sequence of operations.

- Cutting \mathcal{B} along $R(\mathcal{C})$ into \mathcal{C} and \mathcal{D} (see Definition 2.4.23),
- composing \mathcal{D} with \mathcal{C}' into a labelled strategy \mathcal{R} (see Definition 2.4.24).

Partial labelled strategies \mathcal{D} and \mathcal{C}' are compatible due to Lemma 2.4.27. The result \mathcal{R} is a labelled strategy due to Lemma 2.4.25.

To establish the difference in the number of guards used to defend \mathcal{B} and \mathcal{R} we have the following lemma.

Lemma 2.4.29. For two interface equivalent partial labelled strategy \mathcal{B}_1 and \mathcal{B}_2 (as in Definition 2.4.26) there is some constant $K(P_1, P_2) \in \mathbb{Z}$ such that for all $\alpha \in \Omega_1$ we have

$$K(P_1, P_2) = |P_2(\alpha)| - |P_1(\alpha)|.$$

Proof. Suppose we have arbitrary states $\alpha, \beta \in \Omega_1$ and let $K_\alpha = |P_2(\alpha)| - |P_1(\alpha)|$ and $K_\beta = |P_2(\beta)| - |P_1(\beta)|$. Let $M_1(\alpha, \beta)$ be part of \mathcal{B}_1 and $M_2(\alpha, \beta)$ part of \mathcal{B}_2 . Each defines a pairing of guards in respective states. However, the guards on the interface are not

required to participate in the pairing. So we have $|P_1(\alpha)| + g_1(\alpha, \beta) = |P_1(\beta)| + g_1(\beta, \alpha)$ where g_i is the number of guards that do not participate in the pairing of respective M_i (we assume symmetric moves). Similarly for \mathcal{B}_2 we have $|P_2(\alpha)| + g_2(\alpha, \beta) = |P_2(\beta)| + g_2(\beta, \alpha)$.

As the partial labelled strategies are interface equivalent, the sets of guards which do not participate in the pairings is the same, so $g_1(\gamma, \delta) = g_2(\gamma, \delta)$ for all $\gamma, \delta \in \Omega_1$. We get

$$\begin{aligned} K_\alpha &= |P_2(\alpha)| - |P_1(\alpha)| \\ &= |P_2(\beta)| + g_1(\beta, \alpha) - g_1(\alpha, \beta) - (|P_1(\beta)| + g_2(\beta, \alpha) - g_2(\alpha, \beta)) \\ &= |P_2(\beta)| - |P_1(\beta)| + (g_1(\beta, \alpha) - g_2(\beta, \alpha)) + (g_2(\alpha, \beta) - g_1(\alpha, \beta)) \\ &= |P_2(\beta)| - |P_1(\beta)| = K_\beta. \end{aligned}$$

We set $K(P_1, P_2) = K_\alpha$ as we showed that this value is the same irrespective of the chosen α . \square

To be able to use an expansion we need to select a partial labelled substrategy \mathcal{C} of \mathcal{B} and then show that \mathcal{C} is interface equivalent with \mathcal{C}' . The expansion then proceeds as in Definition 2.4.28 and an upper bound is obtained from the following observation.

Observation 2.4.30. Let us have an expansion of \mathcal{B} from \mathcal{C} to \mathcal{C}' (Definition 2.4.28) which results in a labelled strategy \mathcal{R} . The expansion increases the number of used guards by $K(P(\mathcal{C}), P(\mathcal{C}'))$ due to Lemma 2.4.29. Assuming that \mathcal{B} is an optimal strategy we obtain $\gamma_m^\infty(G(\mathcal{R})) \leq \gamma_m^\infty(G(\mathcal{B})) + K(P(\mathcal{C}), P(\mathcal{C}'))$.

We showed a way to describe a labelled strategy and how we can exchange the underlying defended graph. However, to be able to do this we need the strategy to be the same for the original and expanded graph. So before we start expansion we alter the strategy on the original graph. This is discussed in the following section.

2.4.3 Tools for Altering Strategies

In this section, we introduce further notions useful for working with strategies when building upper bound constructions. The typical upper bound proof uses tools introduced in this section to alter the strategy and then applies expansion (Definition 2.4.28) which gives the upper bound by Observation 2.4.30.

First, let us note that all the notions can be thought of as “up to isomorphism” because we can relabel graph or strategy vertices and relabeling does not fundamentally change them. We skipped this in definitions for the sake of readability. Let us also set from now on $\mathcal{B} = (G, S_G, P, \mathcal{T}, R)$ and $S_G = (\Omega, \mathbb{F})$, and similarly for \mathcal{B}' and \mathcal{B}^* have respective graphs G' and G^* , strategies, mappings, etc.

Now we present the main operation for altering strategy graphs.

Definition 2.4.31 (Graph Cartesian product over subset). Let us have graph G_1 and G_2 while $A \subseteq V(G_1)$. The *graph Cartesian product over subset A* denoted as $G_1 \square_A G_2$ is a

graph H such that

$$\begin{aligned} V(H) &= \{(u, \emptyset) \mid u \in V(G_1) \setminus A\} \cup \{(u, v) \mid u \in A, v \in V(G_2)\}, \\ \{(a, b), (c, d)\} \in E(H) &\Leftrightarrow ((a = c) \wedge (a \in A) \wedge (\{b, d\} \in E(G_2))) \vee \\ &\vee (\{a, c\} \in E(G_1) \wedge ((b = d) \vee (b = \emptyset) \vee (d = \emptyset))). \end{aligned}$$

The operation in Definition 2.4.31 can be thought of as a Cartesian product where the sets of vertices created from G_1 which are not present in A are identified to a single vertex. Equivalently, H can be constructed by taking the graph Cartesian product of $G[A]$ and H , adding $G[V(G) \setminus A]$, relabeling each new vertex u as (u, \emptyset) and connecting each such $(u, \emptyset) \in V(G) \setminus A$ to all $(v, x) \in A \times V(H)$ such that $v \in N_{G_1}(u)$. This operations will prove very useful when altering strategy graphs – we will see it used soon in Lemma 2.4.38 and many times in Section 2.5. The aim of this operation is to defend parts of the graph almost independently. The edges created from G_1 represent changes of guard positions within one part of the graph and edges from G_2 represent changes in another part. While guards move within one part of the graph then the guards in the other part will remain stationary. Necessity of the set A comes from the fact that the strategy in one part assumes that a guard occupies vertex (e.g. u) so then the altered part is restricted to vertices where the guard is present on the vertex ($A = \mathcal{S}(u)$). See Figure 2.6 for an example application of the Cartesian product over subset.

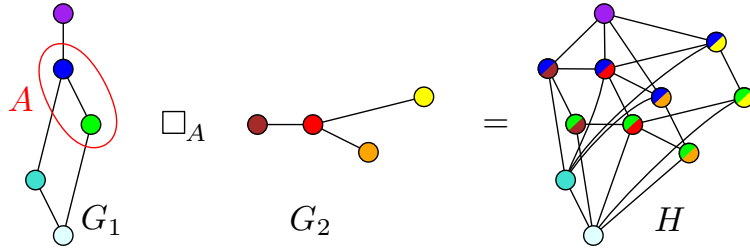


Figure 2.6: Example of a graph Cartesian product of G_1 and G_2 over subset A .

We shall use the Cartesian product of G' and complete graph over subset very often so we will use short notation that allows us to focus on what happens in the created strategy.

Definition 2.4.32 (Short notation). Let $G \square_A \{\alpha_1, \alpha_2, \dots, \alpha_n\} = G \square_A K_n$ where $V(K_n) = \{\beta_1, \dots, \beta_n\}$ and α_i denotes sets of states created from β_i , i.e., $\alpha_i = \{(a, \beta_i) \mid a \in A\}$.

The Cartesian product over subset will be used to first alter the strategy graph. The multiplied states shall defend the same set of vertices as before. Then, during expansion, the guards shall be moved in order to defend new parts of the graph. There, we need to ensure that the strategy remains defending. For this, we have the following lemma that tackles the unchanged and changed states in separate cases.

Lemma 2.4.33. Let us have $H = G_1 \square_A G_2$ with vertices labelled as in Definition 2.4.31.

1. If C is a dominating set of G_1 , then $\{(c, b) \mid (c, b) \in V(H), c \in C, b = \emptyset \vee b \in A\}$ is a dominating set of H .
2. If A is a dominating set of G_1 and B is a dominating set of G_2 , then $A \times B$ (i.e., $\{(a, b) \mid a \in A, b \in B\}$) is a dominating set of H .

Proof. Let us have a vertex (x, y) in $V(H)$.

In Case 1, there is a $c \in C$ that dominates x in G_1 . Therefore, (c, y') with $y' = \emptyset$ if $c \notin A$ or with $y' = y$ if $c \in A$ dominates (x, y) .

In Case 2, if $x \notin A$ (so $y = \emptyset$), then there is some $a \in A$ that dominates x in G_1 . As $|B| \geq 1$ there is $(a, b) \in A \times B$ that dominates (x, y) in H . Otherwise $x \in A$ so there is $b \in B$ that dominates y in G_2 . Hence, (x, y) is dominated by $(x, b) \in A \times B$ in H . \square

When changing the strategy, we want to keep the properties of Lemma 2.4.33 to ensure that labelled strategy is defending.

The following lemma shows the second major operation for changing strategies. It allows us to add leaves to arbitrary vertex and defend the new graph with one more guard.

Lemma 2.4.34 (Leaves addition). Let us have a graph G and let $u \in V(G)$ such that it has $\ell \geq 1$ adjacent leaf vertices v_1, \dots, v_ℓ . Let G' be a graph G with vertices v_1, \dots, v_ℓ removed. For any defending labelled strategy \mathcal{B}' there is a defending labelled strategy \mathcal{B} with strategy graph $S_G = S'_{G'} \sqcup_{S(u)} K_\ell$ that uses one more guard than \mathcal{B}' .

Proof. First, let $S_{G'} = S'_{G'} \sqcup_{S(u)} \{\alpha_1, \dots, \alpha_\ell\}$ (see short notation Definition 2.4.32). As u must be defended $S(u) \neq \emptyset$. By its construction, all guards of strategy $S_{G'}$ are stationary on $\mathcal{T}(\alpha_i, \alpha_j)$. Let $\delta' = \Omega' \setminus \{\alpha_1, \dots, \alpha_\ell\}$. We expand the strategy over $S_{G'}$ to G by adding u to δ (i.e., $P(\delta) = P'(\delta') \cup \{u\}$) and adding v_i to α_i . We set $\mathcal{T}(\alpha_i, \alpha_j) = \{(v_i, u), (u, v_j)\}$ and we extend $\mathcal{T}(\delta, \alpha_i)$ with (u, v_i) .

As α_i dominates the clique and $S'(u)$ dominates $S'_{G'}$, we have by Lemma 2.4.33 that \mathcal{B} is a defending labelled strategy for G . \square

Observation 2.4.35. In Lemma 2.4.34 the construction works the same for any number of leaves, hence, we may add additional leaves after its use retroactively.

We shall build strategies where vast majority of leaves are defended with Lemma 2.4.34. This gives a merit to treat all such states in the same way as their transitions with respect to the rest of the graph are isomorphic. To do this we put forward the following notion.

Definition 2.4.36 (Group state). Let a *group defense* be a set of states which were created by Cartesian product of G and a clique K_n over a subset.

We shall use group defense only to describe groups of leaves. By Observation 2.4.35 we will be able to add new leaves to such group at any point of the construction.

In group states, but also in general strategies we investigated, it seems that vertices which are adjacent to multiple leaves are often permanently occupied. To get a concrete result from this observation let us show how to alter an m-Eternal Domination strategy such that such vertices are permanently occupied.

Definition 2.4.37 (Permanently defended). A vertex u is *permanently defended* (permanently occupied) in \mathcal{B} if $\mathcal{S}(u) = \Omega$, i.e., $u \in P(\alpha)$ for every $\alpha \in \Omega$.

Lemma 2.4.38. For a graph G and its arbitrary defending labelled strategy \mathcal{B}' we may create a defending labelled strategy \mathcal{B} which uses the same number of guards and where each vertex adjacent to at least 2 leaves is permanently defended.

Proof. Let vertex u be a vertex with at least 2 adjacent leaves such that u is not permanently occupied in strategy \mathcal{B}' . Let α' be a state where no guard occupies u . In $P'(\alpha')$, there must be a guard on each leaf adjacent to u . Let v and w be two of the leaves adjacent to u in G' . Let $S_{G'} = S'_{G'} \sqcup_{\Omega' \setminus \mathcal{S}'(u)} \{\alpha_v, \alpha_w\}$ (see short notation Definition 2.4.32). Let $P'(\alpha_v) = P(\alpha') \cup \{u\} \setminus \{w\}$ and $P'(\alpha_w) = P(\alpha') \cup \{u\} \setminus \{v\}$, so $P(\alpha_v)$ occupies v and $P(\alpha_w)$ occupies w .

To create transitions \mathcal{T} , we keep all transitions between states within $\mathcal{S}'(u)$ the same. For all $\beta \in \Omega \setminus \mathcal{S}'(u)$, we set $\mathcal{T}(\alpha_v, \beta)$ as $\mathcal{T}(\alpha', \beta)$ with w in each movement substituted by u . Such substitution still constitutes movements as $N[w] \subseteq N[u]$. Similarly, we create the transitions for $\mathcal{T}(\alpha_w, \beta)$. Furthermore, we set $\mathcal{T}(\alpha_w, \alpha_v) = \{(v, u), (u, w)\}$. Note, that the transitions in reverse direction are derived from symmetry. This shows that there are valid transitions for all edges of the created strategy graph S_G .

In the obtained strategy \mathcal{B} vertex u is permanently defended. As we use Cartesian product with a complete graph over a dominating subset it follows from Lemma 2.4.33 that the strategy is still defending.

We repeat the above procedure for each vertex adjacent to at least 2 leaves until all such vertices are permanently defended. \square

2.5 Reducing Cactus Graphs

In this section, we prove that $\Gamma_m^\infty(G) = \gamma_m^\infty(G)$ for cactus graphs by showing optimal strategies and unconditional lower bounds. The main idea is to repeatedly use reductions on the cactus graph G to produce smaller cactus graph G' . Then we prove that a strategy for G uses a fixed number of guards more than an optimal strategy for G' . Respective lower bound then shows that the strategy for G is indeed also optimal. We will describe precise way we get such results in Section 2.5.1 but before that, we show the overall structure of the proof.

The proof uses an induction on the number of vertices. The base case is a small graph (1 or 2 vertices) where the optimal strategy is elementary (see Definition 2.5.6). The induction step is described in detail later. Now we show several structural properties of cactus graphs which allow us to do the induction.

Definition 2.5.1 (Leaf cycle). *Leaf cycle* is a cycle which has at most one vertex (called *connecting vertex*) which has neighbor such that it is not a vertex of the cycle nor a leaf.

See a leaf cycle on Figure 2.8.

Definition 2.5.2 (Leaf component). By a *leaf component* we mean either a leaf cycle or a leaf vertex which is not adjacent to a leaf cycle.

Observation 2.5.3. Every cactus graph with at least 3 vertices contains a leaf component.

Proof. Let us obtain the block-cut tree \mathcal{T} representation of the cactus graph and root it in an arbitrary block node (see [61, block-cutpoint trees, page 36]). Each block node of \mathcal{T} either represents a single edge or a cycle. We observe the deepest nodes of \mathcal{T} to get the following three cases, see Figure 2.7.

- A There is a deepest node which represents a cycle.
- B A deepest node's grandparent block is a single edge block.
- C A deepest node's grandparent block is a cycle block.
- D No deepest node has a grandparent.

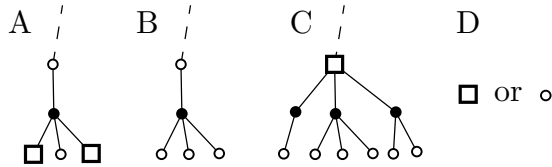


Figure 2.7: Example subtrees for structures which always appear in the block-cut tree of a cactus graph. Big squares represent cycle nodes, small full circles represent articulations, and small empty circles represent single edge nodes.

Note that in cases A and C the graph contains a leaf cycle, in case B it contains a leaf vertex which is not adjacent to a leaf cycle. The case D is trivial and the graph is either a cycle or a single edge. Hence, a cactus graph always contains a leaf component. \square

Definition 2.5.4 (Vertex colors). A non-connecting vertex v of a leaf cycle C is labeled with a color $\text{col}(v)$ which depends on the number of adjacent leaves in the following way.

$$\text{col}(v) \text{ is } \begin{cases} \widehat{0} & \text{if } v \text{ is adjacent to 0 leaves} \\ \widehat{1} & \text{if } v \text{ is adjacent to 1 leaf} \\ \widehat{2} & \text{if } v \text{ is adjacent to at least 2 leaves} \end{cases}$$

We shall label v as $\text{col}(v) = \widehat{X}$ if v is the connecting vertex of C . When a vertex can have different colors (to cover several cases at once) we list them by set of colors. For that purpose, we may write $\widehat{0}$ instead of $\{\widehat{0}\}$, and similarly for $\widehat{1}$, $\widehat{2}$, and \widehat{X} . Also, we use a shortcut to denote all colors $\widehat{_} = \{\widehat{0}, \widehat{1}, \widehat{2}, \widehat{X}\}$.

See Figure 2.8 for an example of $\widehat{0}$, $\widehat{1}$, and $\widehat{2}$ vertices.

To describe reductions over leaf components we will use a concise notation for the leaf cycles which just lists the colors of consecutive vertices of the cycle as follows. See Figure 2.8 for an example.

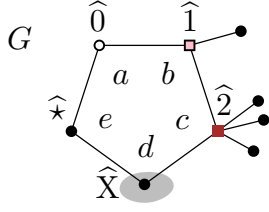


Figure 2.8: A leaf cycle of a graph G with a partial labelled strategy \mathcal{B} containing vertices $a, b, c, d,$ and e with colors $\hat{0}, \hat{1}, \hat{2}, \hat{X},$ and $\hat{_}$, respectively. The original graph was bigger and its rest was connected to d . To look at the leaf cycle in isolation, the graph was cut in vertex d that now constitutes interface of \mathcal{B} . This cycle would be denoted by a leaf sequence $(\hat{X}, \hat{_}, \hat{0}, \hat{1}, \hat{2}, \hat{X})$ (or reversed).

Definition 2.5.5 (Leaf sequence). Let (v_1, \dots, v_n) be n consecutive vertices of a leaf cycle. The *leaf sequence* of vertices (v_1, \dots, v_n) is $(\text{col}(v_1), \dots, \text{col}(v_n))$ where $\text{col}(v_i) \subseteq \{\hat{0}, \hat{1}, \hat{2}, \hat{X}\}$. Moreover, given two leaf sequences A and B and a graph G which contains a leaf cycle with a leaf sequence A , let $A \rightarrow B$ denote a reduction of subgraph with leaf sequence A to one with leaf sequence B in G to obtain G' .

Note that if the leaf sequence starts and ends with a connecting vertex and contains no $\hat{_}$, then it describes the whole cycle because colors correspond to the number of leaves and there is only one connecting vertex in a leaf cycle.

Now, we show the base case and the overview of the induction step.

Definition 2.5.6 (Base cases). Let the *base cases* be the following graphs along with their optimal defending labelled strategies.

- A single isolated vertex with no edges defended by labelled strategy

$$((\{u\}, \emptyset), (\{\alpha\}, \emptyset), \{\alpha \rightarrow \{u\}\}, \emptyset).$$

- A single isolated edge is defended by labelled strategy

$$\left((\{u, v\}, \{\{u, v\}\}), (\{\alpha, \beta\}, \{\{\alpha, \beta\}\}), \{\alpha \rightarrow \{u\}, \beta \rightarrow \{v\}\}, \{(\alpha, \beta) \rightarrow \{(u, v)\}\} \right).$$

2.5.1 Technique and Overview

For the induction step, every reduction takes the cactus graph G and changes it to G' which has smaller number of vertices. Reductions will be performed on a leaf component which by Observation 2.5.3 is always present in a cactus graph on at least three vertices. The two cactus graphs which have at most two vertices are covered by base cases from Definition 2.5.6.

More precisely, every reduction shows lower bound and upper bound. Lower bound is shown for the M-ETERNAL GUARD CONFIGURATION and involves using Observations 2.4.3 and 2.4.8 to 2.4.10. These tools make the graph smaller and show that in any defending strategy the removed parts required some minimum number of guards; they give us lower bound $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + K$ for some constant K .

Upper bounds are shown for the M-ETERNAL DOMINATION and usually involve two separate steps. First step takes the reduced graph G' and its optimal strategy $S'_{G'}$ and shows how to alter the strategy by tools shown in Section 2.4.3. This does not change the number of guards, but only structure of the defense. Second step uses the framework shown in Section 2.4.2. It takes part of the graph we intend to expand (Definition 2.4.28), cuts it, and replaces with an interface equivalent (Definition 2.4.26) partial labelled strategy, as described in Observation 2.4.30. During the expansion, the strategy graph S_G does not change (so $S_G = S_{G'}$), however the graph and mapping does change. The labelled strategy now maps strategy graph states so that there are new guards and some states have guards moved to other vertices of the graph. We also show how the transitions change between states that were altered. When the defense of the graph is managed with K additional guards, this gives us an upper bound $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + K$ (the same K as in the lower bound).

Combining the lower and upper bound using Lemma 2.4.2 results in an optimal number of guards for G for M-ETERNAL DOMINATION and M-ETERNAL GUARD CONFIGURATION.

The used reduction depends on a leaf component that the cactus graph contains by Observation 2.5.3. If the deepest node is not adjacent to a leaf cycle, then we use leaf reductions shown in Section 2.3. Using these reduction exhaustively results in having a leaf cycle (or a base case) – we will show this soon in Lemma 2.5.7.

To reduce leaf cycles we will need additional properties on edges of the leaf cycle. This involves being able to forbid movement along an edge, and forcing move along an edge. We achieve this by partitioning all states of the strategy graph into tree groups which ensure these properties. The properties are established in Section 2.5.2.

Having the properties we take the leaf cycle and look at its vertex colors. If a color pattern is listed among reductions then we have a way to remove it. The reductions are split into two groups. First recognizes just a small part of the cycle, making it shorter – these are called cycle reductions Section 2.5.3. The second recognizes the whole cycle and removes it entirely, leaving just a few leaves in its place – these are constant component reductions shown in Section 2.5.4. We show that one of these reductions may always be used by exhaustive search of all possibilities in depicted in Figure 2.14; and doubling this function, we show a slightly different proof in Lemma 2.5.19.

We end this section with the aforementioned proof of the cactus graph structure after application of leaf reductions in Lemma 2.5.7 and a diagram overview of the remaining sections in Figure 2.9.

Lemma 2.5.7. Exhaustive application of Reductions t_1 , t_2 , and t_3 on a cactus graph G results in reaching the base case or it results in a cactus graph with a leaf cycle.

Proof. We saw in Observation 2.5.3 that in every cactus there either is a leaf cycle or there is a set of $\ell \geq 1$ leaves with a common parent which is connected to the rest of the graph with a single edge. The number ℓ directly implies which tree reduction may be applied. If $\ell = 1$, then we may use Reduction t_1 ; if $\ell = 2$, then we use Reduction t_3 ; and last, if $\ell > 2$, then we use Reduction t_2 . After exhaustive application we either reach the base case or the other case applies – we have a leaf cycle. \square

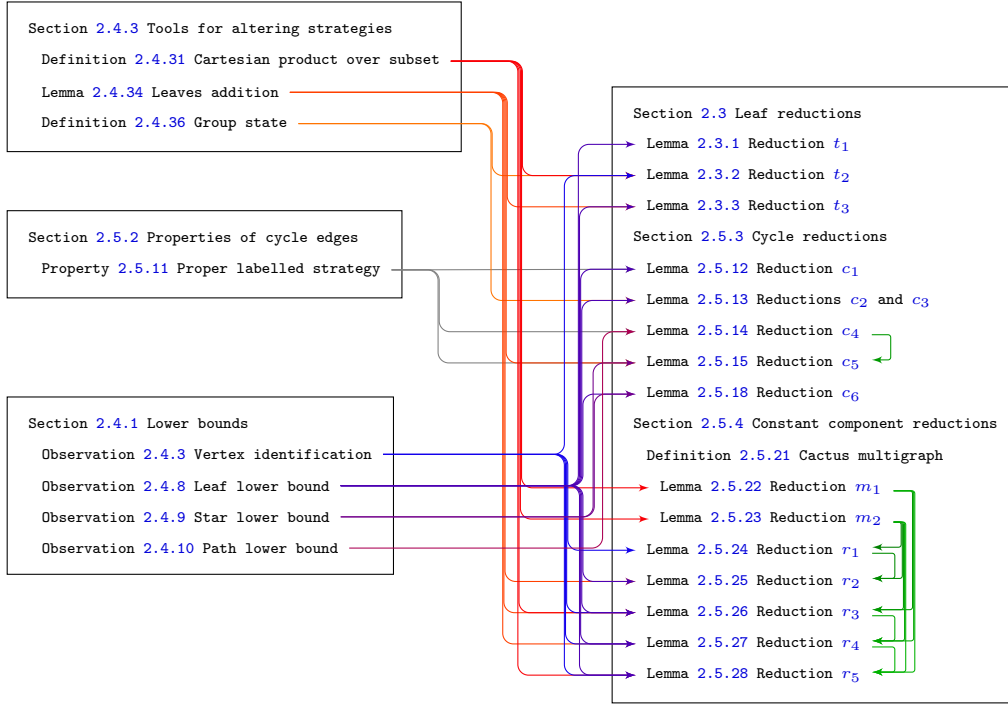


Figure 2.9: Overview of Section 2.5. Left boxes represent tools obtained in Section 2.4 (see Figure 2.2) and properties we introduce in Section 2.5.2; Right box shows structure of Section 2.5; Left-to-right arrows show which tools are used for which results. Right-to-right (green) arrows show that the reduction is partially based on or uses another reduction.

2.5.2 Properties of Cycle Edges

We shall assume that the built strategy over the graph holds some properties which allow us to make stronger induction step. More precisely, these properties shall be necessary to show Reductions c_1 , c_4 , and c_5 .

Definition 2.5.8 (Edge states). By *edge states of (u, v) in Ω* (where $\{u, v\} \in E(G)$) we mean creating sets $L_{u,v}$, $R_{u,v}$, and $N_{u,v}$ such that

$$\begin{aligned} \alpha \in L_{u,v} & \text{ if } \exists \beta \in \Omega, (u, v) \in \mathcal{T}(\alpha, \beta), \\ \alpha \in R_{u,v} & \text{ if } \exists \beta \in \Omega, (v, u) \in \mathcal{T}(\alpha, \beta), \\ N_{u,v} & = \Omega \setminus (L_{u,v} \cup R_{u,v}) \end{aligned}$$

Note that because the orientation of the edge plays a role in these definitions, we have $L_{a,b} = R_{b,a}$, $R_{a,b} = L_{b,a}$, and $N_{a,b} = N_{b,a}$. The names of the sets reflect from which side a guard can traverse the edge. Also note, that if we assume symmetry then when moving to and from $N_{u,v}$ the edge $\{u, v\}$ cannot be traversed. We propose the following edge property which is somewhat similar to Observation 2.4.16.

Property 2.5.9 (Proper edge states). For a strategy \mathcal{B} over graph G an edge $\{u, v\}$ holds *Property 2.5.9* if and only if its edge states $L_{u,v}$, $R_{u,v}$, and $N_{u,v}$ are all non-empty, $L_{u,v} \cap R_{u,v} = \emptyset$, and each of them is a dominating set over S_G .

There are several ramifications of an edge having *Property 2.5.9*. Because of $L_{u,v} \cap R_{u,v} = \emptyset$ there is no state where we may choose to move over (u, v) or (v, u) , i.e., at most one of these movements is available. At the same time, as each of these sets is dominating S_G , it follows that we may get into any of these sets in one transition. Last, as each set is non-empty we may force the strategy to forbid to move over $\{u, v\}$ in the current and one future transition by moving to $N_{u,v}$ at any point. Additionally, we may force a movement over (u, v) by moving first to some $\alpha \in L_{u,v}$ and then to $\beta \in R_{u,v}$ such that $(u, v) \in \mathcal{T}(\alpha, \beta)$ as per *Definition 2.5.8*.

All the properties that proper edge states additionally have compared to non-proper edge states are true irrespective of permutations of $L_{u,v}$, $R_{u,v}$, and $N_{u,v}$. Hence, we may use the same sets on different edges by permuting them and checking that they constitute edge states of the new edge.

Observation 2.5.10. For edges $\{u, v\}$ and $\{a, b\}$ if we map proper edge states $L_{u,v}$, $R_{u,v}$, and $N_{u,v}$ to new sets $L_{a,b}$, $R_{a,b}$, and $N_{a,b}$ (with possibly permuting them) then these constitute proper edge states of $\{a, b\}$ if and only if they constitute edge states of $\{a, b\}$.

Proof. If the new edge states $L_{a,b}$, $R_{a,b}$, and $N_{a,b}$ do not constitute edge states then they trivially cannot be proper edge states. When $L_{u,v}$, $R_{u,v}$, and $N_{u,v}$ are proper edge states then they are disjoint and nonempty. These properties do not depend on their order so as long as the new states are edge states they will be proper. \square

Our goal will be to have *Property 2.5.9* on all edges that lie on a leaf cycle that are incident to at least one $\widehat{0}$ or \widehat{X} vertex. We shall also show that it holds in some special cases to make several reductions easier.

In reductions, we will check that an edge has *Property 2.5.9*, however, the intuition about it is as follows. We need to check whether each cycle edge is traversed at least once and whether it is not traversed at all by at least one state. Also, it is usually trivial, but we should check that the edge cannot be traversed in both directions from some state.

Property 2.5.11 (Proper labelled strategy). A partial labelled strategy \mathcal{B} over a cactus graph G has *Property 2.5.11* if and only if *Property 2.5.9* holds for each edge that lie on a cycle and

- is incident to a $\widehat{0}$ or a \widehat{X} vertex,
- or is on a leaf cycle $(\widehat{X}, \widehat{2}, \widehat{2}, \widehat{X})$,
- or is incident to a \widehat{X} vertex while not being a edge which lies between \widehat{X} and a $\widehat{2}$ vertex on leaf cycle $(\widehat{X}, \widehat{2}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X})$ or $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X})$.

Our goal is to keep our cactus graph proper (as per Property 2.5.11) in all steps of reducing. For simplicity, we shall work with reductions as if all edges on cycles which are incident to $\widehat{0}$ or \widehat{X} vertex have Property 2.5.9 and we shall tackle the exceptions to this rule separately in Observation 2.5.20 and Lemma 2.5.29.

2.5.3 Cycle Reductions

Due to Lemma 2.5.7 we know that applying tree reductions may result in either solving the instance entirely or we obtain a leaf cycle. In this section, we will tackle leaf cycles with cycle reductions which results in a leaf cycle of constant size. Constant-sized leaf cycles are then resolved in Section 2.5.4.

Let C denote a leaf cycle where vertices are labeled with colors according to Definition 2.5.4. Cycle reductions consist of the following reductions (see notation in Definition 2.5.5). E.g., Reduction c_1 describes that a graph G with a leaf cycle that contains consecutive vertices U with colors $(\widehat{_}, \widehat{1}, \widehat{_})$ may be changed to G' by substituting U with a vertices of colors $(\widehat{_}, \widehat{_})$ (so just $\widehat{1}$ was removed). At the same time, it claims that $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 1$ and $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 1$. All of this is concisely written as $(\widehat{_}, \widehat{1}, \widehat{_}) \rightarrow (\widehat{_}, \widehat{_}) + 1$.

Reduction 4. c_1 $(\widehat{_}, \widehat{1}, \widehat{_}) \rightarrow (\widehat{_}, \widehat{_}) + 1$ where $(\widehat{_}, \widehat{_})$ has Property 2.5.9.

Reduction 5. c_2 $(\widehat{2}, \widehat{1}, \widehat{_}) \rightarrow (\widehat{2}, \widehat{_}) + 1$

Reduction 6. c_3 $(\widehat{2}, \widehat{2}, \widehat{_}) \rightarrow (\widehat{2}, \widehat{_}) + 1$

Reduction 7. c_4 $(\widehat{_}, \widehat{0}, \widehat{0}, \widehat{0}, \widehat{_}) \rightarrow (\widehat{_}, \widehat{_}) + 1$ where $(\widehat{_}, \widehat{_})$ has Property 2.5.9.

Reduction 8. c_5 $(\widehat{_}, \widehat{0}, \widehat{2}, \widehat{0}, \widehat{_}) \rightarrow (\widehat{_}, \widehat{_}) + 2$ where $(\widehat{_}, \widehat{_})$ has Property 2.5.9.

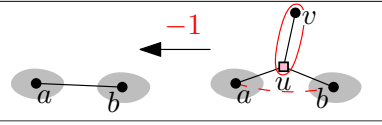
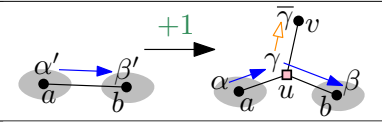
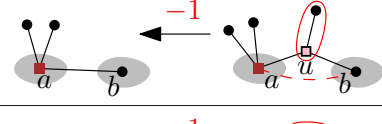
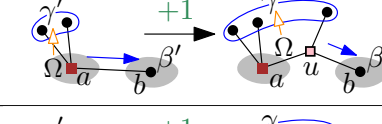
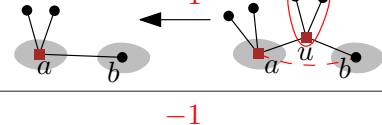
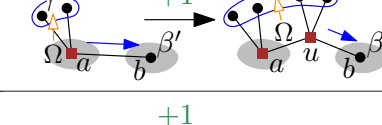
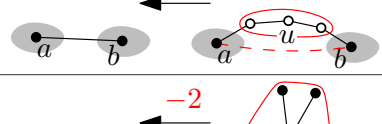
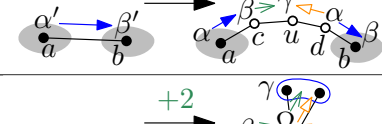
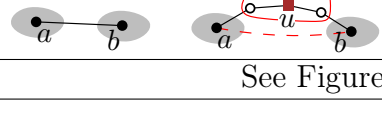
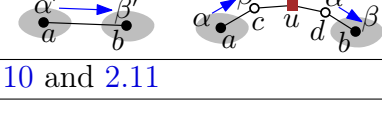
Reduction 9. c_6 $(\widehat{X}, \widehat{2}, [\widehat{0}, \widehat{2}]^{2k}, \widehat{X}) \rightarrow (\widehat{1}) + 3k + 1$ and $(\widehat{X}, \widehat{2}, [\widehat{0}, \widehat{2}]^{2k+1}, \widehat{X}) \rightarrow (\widehat{2}) + 3k + 2$

Let a and b be the first and the last vertex of the leaf cycle in G' , respectively, that are described by the reduction. It is clear that these reductions may be used in cases where a and b are non-connected disjoint vertices. We note that the reductions will be used when $\{a, b\} \in E(G')$ though the result contains a pair of multiedges between a and b in G . Moreover, these reductions may be used even in case where $a = b$. Applying the reduction in such a case results in a loop in a within G' . Though loops and multiedges may be created by the process they will be immediately removed. These cases will be addressed in Section 2.5.4.1.

Reductions c_1 , c_4 , and c_5 require the edge that is being expanded (edge $\{a, b\}$ in G') holds Property 2.5.9. We shall ensure this by keeping Property 2.5.11 for G' while ensuring that during every expansion this property is preserved.

Lemma 2.5.12. Let G' be G after application of Reduction c_1 . G is defended with 1 more guard than G' .

Table 2.2: List of the cycle reductions; notation is the same as in Table 2.1

Reduction	Lower bound	Upper bound
c_1		
c_2		
c_3		
c_4		
c_5		
c_6	See Figures 2.10 and 2.11	

Proof. Let us label the vertices of colors $(\hat{\square}, \hat{1}, \hat{\square})$ by a, u, b , respectively. Let v be the leaf adjacent to u . By using Observation 2.4.8 on vertices $\{u, v\}$ we get lower bound $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 1$.

For upper bound, let $L_{a,b}$, $R_{a,b}$, and $N_{a,b}$ be edge states of the edge a, b in the strategy of G' obtained as stated in Definition 2.5.8. We extend all states of $L_{a,b}$ and $R_{a,b}$ by adding u to them, and we add v to $N_{a,b}$. We substitute movements (a, b) with $\{(a, u), (u, b)\}$ in $\mathcal{T}(L_{a,b}, R_{a,b})$ and we add (u, v) to $\mathcal{T}(L_{a,b} \cup R_{a,b}, N_{a,b})$. The new vertices are defended as $L_{a,b}$ and $N_{a,b}$ are dominating the strategy graph because Property 2.5.9 holds for $\{a, b\}$ in G' . The edge states for the new edges $\{a, u\}$ and $\{u, b\}$ in G remain the same as for $\{a, b\}$ in G' . Therefore, these edges now hold Property 2.5.9 in G . By extending all states with one guard we got a defending labelled strategy, so $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 1$. By Lemma 2.4.2 we get that G is defended with one more guard than G' . \square

Reductions c_2 and c_3 merge a group of consecutive red and pink vertices and defend leaves adjacent to them by a group state (Definition 2.4.36).

Lemma 2.5.13. Let G' be G after application of Reduction c_2 or c_3 . G is defended with 1 more guard than G' .

Proof. The reductions are separate for the sake of future argument but they are proven in the same way. Let us label the vertices of colors $(\hat{2}, \{\hat{1}, \hat{2}\}, \hat{\square})$ by a, u, b , respectively. Let R_1 denote all leaves adjacent to u . By applying Observation 2.4.8 on $R_1 \cup \{u\}$ we get lower bound $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 1$.

For upper bound, let γ' be the group state for leaves adjacent to a . We add to leaves R_1 two new vertices which are defended by γ' as pointed out in Lemma 2.4.34. Now we split u from a , taking its leaves with it that we now label by R_2 . Transitions between leaves is extended to $\mathcal{T}(R_1, R_2) = \{(R_1, a), (a, u), (u, R_2)\}$ and similarly, we extend all transitions which used a . The transitions that interacted with a and b are preserved, so the reduction expands interface equivalent partial labelled strategies. Though we did not need Property 2.5.9 the graph still has Property 2.5.11 because the new edge $\{u, b\}$ takes on exact transitions that $\{a, b\}$ had. So if $\{a, b\}$ held the property in G' , then $\{b, u\}$ holds it in G .

We added one guard so $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 1$ and by Lemma 2.4.2 we get that G is defended with one more guard than G' . \square

Lemma 2.5.14. Let G' be G after application of Reduction c_4 . G is defended with 1 more guard than G' .

Proof. Let us label the vertices of colors $(\hat{_}, \hat{0}, \hat{0}, \hat{0}, \hat{_})$ by a, c, u, d, b , respectively. Using Observation 2.4.10 on $\{c, u, d\}$ we get lower bound $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 1$.

For upper bound, let $L'_{a,b}$, $R'_{a,b}$, and $N'_{a,b}$ be edge states of the edge a, b in the strategy of G' obtained from Definition 2.5.8. These are proper edge states as $\{a, b\}$ holds Property 2.5.9 in G' . We extend the states by adding d to all states of $L'_{a,b}$, c to $R'_{a,b}$, and u to $N'_{a,b}$; this creates sets $L_{a,b}$, $R_{a,b}$, and $N_{a,b}$. We substitute movements along (a, b) with $\{(a, c), (d, b)\}$ in $\mathcal{T}(L_{a,b}, R_{a,b})$, hence, the exchanged parts of the graph are interface equivalent. We add (c, u) to $\mathcal{T}(R_{a,b}, N_{a,b})$ and (d, u) to $\mathcal{T}(L_{a,b}, N_{a,b})$. The new $\{c, u, d\}$ vertices are defended by the nonempty sets $R_{a,b}$, $N_{a,b}$, and $L_{a,b}$, respectively. The edge states for the new edges are as follows.

$$\begin{aligned} L_{a,b} &= L_{a,c} = L_{d,b} = N_{c,u} = L_{d,u} \\ R_{a,b} &= R_{a,c} = R_{d,b} = L_{c,u} = N_{d,u} \\ N_{a,b} &= N_{a,c} = N_{d,b} = R_{c,u} = R_{d,u} \end{aligned} \tag{2.4}$$

As these are only permutations of the edge sets by Observation 2.5.10 they hold Property 2.5.9. We get $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 1$. By Lemma 2.4.2 we get that G is defended with one more guard than G' . \square

Lemma 2.5.15. Let G' be G after application of Reduction c_5 . G is defended with 2 more guards than G' .

Proof. The proof goes very similarly as the proof of Lemma 2.5.14, but all states $N_{a,b}$ shall group defend leaves adjacent to u while u will be permanently occupied.

We label the vertices of colors $(\hat{_}, \hat{0}, \hat{2}, \hat{0}, \hat{_})$ by a, c, u, d, b , respectively. Let R be the leaves neighboring u . Using Observation 2.4.9 on u and its neighborhood we get lower bound $\Gamma_m^\infty(G) \geq \Gamma_m^\infty(G') + 2$.

Repeat the same sequence of steps as in the proof of Lemma 2.5.14 which uses one guard and then add leaves adjacent to u by Lemma 2.4.34 using one extra guard. This

does not change transitions over the edges which are not incident to the leaves so by Observation 2.5.10 they still hold Property 2.5.9. We get $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 2$. By Lemma 2.4.2 we get that G is defended with two more guards than G' . \square

We remark that using reductions t_1, t_2, t_3, c_1, c_4 , and a small set of constant component reductions is sufficient to solve so-called Christmas cactus graphs (graphs where each edge is in at most one cycle and each vertex is in at most two 2-connected components) for which the optimal strategy we presented in [?]. The remaining reductions tackle vertices of color $\widehat{2}$, which are not present in the class of Christmas cactus graphs.

The last cycle reduction is a curious special case, let us recall it first.

Reduction 9. $c_6 \ (\widehat{X}, \widehat{2}, [\widehat{0}, \widehat{2}]^{2k}, \widehat{X}) \rightarrow (\widehat{1}) + 3k + 1$ and $(\widehat{X}, \widehat{2}, [\widehat{0}, \widehat{2}]^{2k+1}, \widehat{X}) \rightarrow (\widehat{2}) + 3k + 2$

We shall use all the other cycle reductions first and if none of them can be used, then we use Reduction c_6 . This allows us to assume a particular structure which we define and prove now. The reason behind this structure may also be well understood from decision diagram of reduction application in Figure 2.14.

Definition 2.5.16 (RW-cycle). A leaf cycle C is a *RW-cycle* if it consists of vertices with alternating $\widehat{2}$ and $\widehat{0}$ colors such that the first and last is $\widehat{2}$, i.e., $C = (\widehat{X}, \widehat{2}, \widehat{0}, \widehat{2}, \widehat{0}, \dots, \widehat{2}, \widehat{0}, \widehat{2}, \widehat{X})$.

Lemma 2.5.17. Assume a leaf cycle C where Reductions c_1, c_2, c_3, c_4 , and c_5 cannot be applied anywhere. Then, $|C| \leq 6$ or C is a *RW-cycle*.

Proof. First, note that if the leaf cycle contains $\widehat{1}$ vertices, then there always is a $\widehat{1}$ vertex which neighbors $\widehat{0}$ or \widehat{X} , in that case we use Reduction c_1 , or it neighbors $\widehat{2}$, we use Reduction c_2 . Hence, all $\widehat{1}$ vertices are removed if Reductions c_1 and c_2 were exhaustively used.

Next, as only $\widehat{2}, \widehat{0}$, and \widehat{X} vertices remain, exhaustively using Reduction c_3 ensures that there are no two adjacent $\widehat{2}$ vertices. (Note that if only red vertices remained, than we end up with $(\widehat{X}, \widehat{2}, \widehat{X})$ which removes the multiedge by Reduction m_2 and then uses Reduction t_3 .)

Last, if there is a $(\widehat{2}, \widehat{0}, \widehat{0})$ part of a leaf cycle then the $\widehat{2}$ vertex is either also adjacent to \widehat{X} or Reduction c_5 can be used as would $(\widehat{X}, \widehat{0}, \widehat{2}, \widehat{0}, \widehat{0})$ necessarily occur. Exhaustively using Reduction c_4 ensures that such cases do not occur. So whenever there are two $\widehat{0}$ adjacent vertices the \widehat{X} vertex is at distance at most 2 from them. This means that either the cycle has at most 5 vertices or the vertices of colors $\widehat{2}$ and $\widehat{0}$ alternate and constitute a RW-cycle. \square

The lower bound for an RW-cycle will not be much harder than for other reductions, however, for the upper bound we will need a strategy made just right for such a cycle.

Lemma 2.5.18. Reduction c_6 is correct.

Proof. Let us label the vertices along the cycle as u_1, u_2, \dots, u_n with u_1 being the connecting vertex (so that $\widehat{2}$ vertices are even). Let $u_{n+1} = u_1$. Let R_2, R_4 , etc. be the leaves adjacent to the red vertices u_2, u_4 , and so on.

First, we use Observation 2.4.8 on $\{u_6\} \cup R_6$; second, we use Observation 2.4.9 on $N[u_4]$. We shortened the cycle by 4 and got lower bound of 3. By repeating the argument k times we end up with a cycle G' of constant size 2 or 4. The cycle of size 2 gets reduced by m_2 and then t_3 which results in a lower bound of 1. The RW-cycle of size 4 has form $(\widehat{X}, \widehat{2}, \widehat{0}, \widehat{2}, \widehat{X})$ and its lower bound is shown in Reduction r_4 to be 2. Putting the $3k$ for every 4 vertices together with 1 and 2 lower bound for respective sizes of the cycle, we get the desired lower bounds. See Figure 2.10 for an illustration of shortening the cycle by 4.

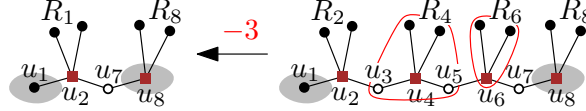


Figure 2.10: Part of the lower bound proof for Reduction c_6

Strategy for the upper bound is quite tricky to describe so let us define a few new notions just for its description. Let *red-parity* of an even number i be the parity of $i/2$. This divides red vertices of the RW-cycle into red-odd and red-even, based on the red-parity. Let *reverse labeling* be the labeling of the RW-cycle in opposite ordering, i.e., if $u'_1 = u_{n+1}$, $u'_2 = u_n$, $u'_3 = u_{n-1}$, \dots , $u'_n = u_2$, and $u'_{n+1} = u_1$, then u'_1, \dots, u'_n is the reverse labeling with respect to labeling u_1, \dots, u_n .

For the upper bound, we distinguish two cases depending on the size of the RW-cycle. First case is that the size is $n = 4k + 2$, the second has $n = 4k$.

We now focus on the first case, where the RW-cycle has size $n = 4k + 2$. Note that in RW-cycle of this size reverse labeling does not change the red-parity of red vertices. We alter the strategy by gradually expanding the states as follows.

$$S_{G'}^* = S'_{G'} \square_{S'(u_1)} \{\alpha_1, \alpha_2, \beta_4, \beta_8, \dots, \beta_{4k}\} \quad \text{and} \quad S_{G'} = S_{G'}^* \square_{\Omega \setminus S'(u_1)} \{\beta_2, \beta_6, \dots, \beta_{4k+2}\}$$

Now we perform expansion from G' to G and set the states Ω of S_G as follows.

$$\begin{aligned} P(\alpha_1) &= \bigcup_{x=0}^k \{u_{4x+3}\} & P(\alpha_2) &= \bigcup_{x=0}^k \{u_{4x+1}\} \\ P(\beta_{4x}) &= ((P(\alpha_2) \cap \{u_j\}_{j < i}) \cup (P(\alpha_1) \cap \{u_j\}_{j > i})) \cup \{R_{4x}\} \\ P(\beta_{4x+2}) &= ((P(\alpha_1) \cap \{u_j\}_{j < i}) \cup (P(\alpha_2) \cap \{u_j\}_{j > i})) \cup \{R_{4x+2}\} \setminus \{u_1\} \end{aligned} \quad (2.5)$$

Notice that $u_1 \in P(\alpha_2)$ as $u_{4x+1} = u_1$ for $x = 0$ and $u_1 \in P(\alpha_1)$ as $u_{4x+3} = u_{4k+3} = u_{n+1} = u_1$ for $x = k$. For $P(\beta_{4x+2})$ the intersections imply that u_1 is not contained, but we mention it explicitly for clarity (as we do not consider u_{n+1} to be u_j for $j < i$ even though it equals u_1). The state β_{2i} group defends leaves adjacent to u_{2i} . Note that they behave differently based on their their red-parity.

Now for the transitions. To make the notation concise let us shorten consecutive movements through red vertices as $F_i = \{(u_i, u_{i+1}), (u_{i+1}, u_{i+2})\}$ and $B_i = \{(u_i, u_{i-1}), (u_{i-1}, u_{i-2})\}$ (as forward and backward). Note that we use F_i and B_i only for odd values of i .

2. M-ETERNAL DOMINATION NUMBER OF CACTUS GRAPHS

Let us have integers x and y and assume, without loss of generality, that $x \leq y$. We set the movements of transitions as follows. If $2x$ and $2y$ have the same red-parity, then

$$\mathcal{T}(\beta_{2x}, \beta_{2y}) = \{(R_{2x}, u_{2x}), (u_{2x}, u_{2x+1})\} \cup \bigcup_{i=x}^{y-4} F_{2i+3} \cup \{(u_{2y-1}, u_{2y}), (u_{2y}, R_{2y})\}. \quad (2.6)$$

Otherwise, $2x$ and $2y$ have different red-parity. If x is odd (so y is even), then their transition is defined as follows.

$$\mathcal{T}(\beta_{2x}, \beta_{2y}) = \{(R_{2x}, u_{2x}), (u_{2x}, u_{2x-1})\} \cup \bigcup_{i=2}^x B_{2i-3} \cup \bigcup_{i=y+1}^{2k-1} B_{2i+3} \cup \{(u_{2y+1}, u_{2y}), (u_{2y}, R_{2y})\}$$

If the red-parity is different and x is even, then in reverse labeling and swapping x with y we end up in the case where the red-parity is still different, but x is odd. This case was already solved. The other direction of these transitions is filled in by symmetry (Property 2.4.18).

It remains to describe transitions with α_1 and α_2 .

$$\mathcal{T}(\alpha_1, \alpha_2) = \{(u_1, u_1)\} \cup \bigcup_{i=0}^{k-1} F_{4i+3}$$

$$\mathcal{T}(\beta_{4x}, \alpha_1) = \{(R_{4x}, u_{4x}), (u_{4x}, u_{4x-1})\} \cup \bigcup_{i=1}^{x-1} B_{4i+1} \quad (2.7)$$

$$\mathcal{T}(\beta_{4x+2}, \alpha_2) = \{(R_{4x+2}, u_{4x+2}), (u_{4x+2}, u_{4x+1})\} \cup \bigcup_{i=1}^x B_{4i-1} \quad (2.8)$$

Note that α_1 is α_2 in reverse labeling. Hence, the case $\mathcal{T}(\beta_{4x}, \alpha_2)$ is equivalent to $\mathcal{T}(\beta_{4x}, \alpha_1)$ in reverse labeling. Similarly, the case $\mathcal{T}(\beta_{4x+2}, \alpha_1)$ is equivalent to $\mathcal{T}(\beta_{4x+2}, \alpha_2)$ in reverse labeling. See a part of this strategy on Figure 2.11.

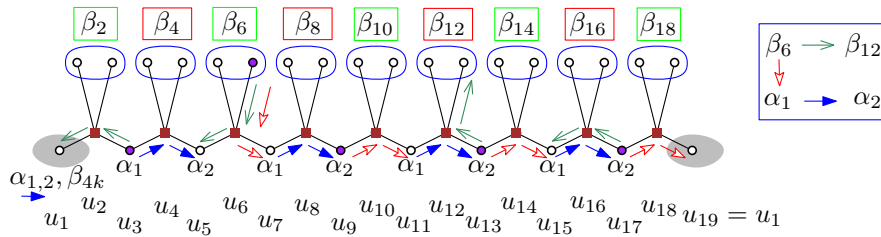


Figure 2.11: Part of a strategy on a RW-cycle of red-odd size 18 with guards (shown purple) placed on $P(\beta_6)$. A few selected transitions are shown as an example.

For the interface equivalency, note that in α_1 , α_2 , and β_{4k} occupy u_1 and states β_{4k+2} do not occupy u_1 . We showed how to transition between every pair of states, so the strategies are interface equivalent with a single pink vertex with expanded states, as in $S_{C'}$.

Now for the second case, where the RW-cycle has size $n = 4k$. Note that in RW-cycle of this size reverse labeling changes the red-parity of red vertices, which was not true in the

first case. The difference in the construction of the strategy is that now we expand from a red vertex. Let u'_1 and u'_2 be the two leaves of the red vertex. Let $\delta' = \Omega' \setminus (\mathcal{S}(u'_1) \cup \mathcal{S}(u'_2))$. We gradually alter the strategy in the following way.

$$\begin{aligned} S_{G'}^1 &= S'_{G'} \sqcap_{\mathcal{S}(u'_1)} \{\gamma, \beta_4, \beta_8, \beta_{12}, \dots, \beta_n\} \\ S_{G'}^2 &= S'_{G'} \sqcap_{\mathcal{S}(u'_2)} \{\alpha_1, \beta_2, \beta_6, \dots, \beta_{n-2}\} \\ S_{G'} &= S_{G'}^2 \sqcap_{\delta'} \{\alpha_2\} \end{aligned}$$

We perform the expansion to get S_G such that all the states have exactly the same definitions as in the first case, see Equation (2.5). We note a major difference: in the second case, u_1 is not an element of $P(\alpha_1)$. We added one extra state γ which has $P(\gamma) = P(\alpha_1)$. There will be a major significance for this state when proving edge properties.

Now we describe the transitions for the the strategy on G . For $2x$ and $2y$ of the same red-parity, the transition Equation (2.6) still holds. In case $2x$ and $2y$ (with $x < y$) have different red-parity, then we consider two separate cases based on red-parity of $2x$.

$$\begin{aligned} \mathcal{T}(\beta_{4x}, \beta_{4y+2}) &= \{(R_{4x}, u_{4x}), (u_{4x}, u_{4x-1})\} \cup \bigcup_{i=1}^{x-1} B_{4i+1} \cup \bigcup_{i=y+2}^k B_{4i-1} \cup \\ &\quad \cup \{(u_{4y+1}, y_{4y+2}, (u_{4y+2}, R_{4y+2}))\} \\ \mathcal{T}(\beta_{4x+2}, \beta_{4y}) &= \{(R_{4x+2}, u_{4x+2}), (u_{4x+2}, u_{4x+1})\} \cup \bigcup_{i=0}^{x-1} B_{4i+3} \cup \bigcup_{i=y+2}^{k+1} B_{4i-3} \cup \\ &\quad \cup \{(u_{4y-1}, y_{4y}, (u_{4y}, R_{4y}))\} \end{aligned}$$

Notice the difference in u_1 - transition $\mathcal{T}(\beta_{4x}, \beta_{4y+2})$ does not move through u_1 so there u_1 is stationary during it; in $\mathcal{T}(\beta_{4x+2}, \beta_{4y})$ movements $\{(u_2, u_1), (u_1, u_n)\}$ happen. To fill all possibilities of mutual transitions among β_{2x} we add transitions obtained by reversed labeling and symmetry.

Now we show the transitions with α_1 and α_2 . Note that reversed labeling does not change these two states. For β_{4x+2} we can apply Equation (2.8) to get $\mathcal{T}(\beta_{4x+2}, \alpha_2)$, and by reversing the labeling this gives us also $\mathcal{T}(\beta_{4x}, \alpha_2)$. Note that after this transition there is one less guard on G as it leaves through the interface $\{u_1\}$. In particular, $\mathcal{T}(\beta_{2x}, \alpha_2)$ moves to u_1 via (u_2, u_1) if $2x$ is red-odd, and via (u_n, u_{n+1}) if $2x$ is red-even.

Similarly, for β_{4x} we can apply Equation (2.7) to get $\mathcal{T}(\beta_{4x}, \alpha_1)$, and by reversing the labeling we get $\mathcal{T}(\beta_{4x+2}, \alpha_1)$. This transition did not interact with the interface. The transition among the two states is as follows.

$$\mathcal{T}(\alpha_1, \alpha_2) = \bigcup_{i=0}^{k-1} F_{4i+3}$$

Note that this again results in a move (u_n, u_{n+1}) .

Last, we introduce the new state γ which has the same guard configuration as α_1 , but differs in one transition. So $\mathcal{T}(\gamma, \beta_{2x}) = \mathcal{T}(\alpha_1, \beta_{2x})$, and $\mathcal{T}(\gamma, \alpha_2) = \emptyset$ (all guards are stationary), but $\mathcal{T}(\gamma, \alpha_2)$ shall be $\mathcal{T}(\alpha_1, \alpha_2)$ in reverse labeling. More precisely,

$$\mathcal{T}(\gamma, \alpha_2) = \bigcup_{i=0}^{k-1} B_{4i+3}.$$

This contains a move (u_2, u_1) . See how movements interact with the interface in Figure 2.12.

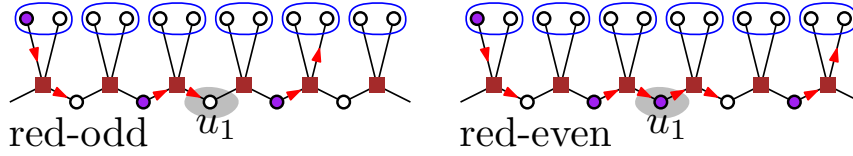


Figure 2.12: Movements through the \widehat{X} vertex in red-odd and red-even RW-cycle.

We discussed the interface impact of all transitions and note that they are equivalent to those in $S_{G'}$, hence, the exchanged strategy is interface equivalent.

It remains to show that S_G is a proper strategy in both cases. The strategy started $S'_{G'}$ was a clique and by Cartesian product over single vertices it remained a clique. Thus, it suffices to say that there is at least one state in $L_{u_i, u_{i+1}}$, $R_{u_i, u_{i+1}}$, and $N_{u_i, u_{i+1}}$, and that $L_{u_i, u_{i+1}} \cap R_{u_i, u_{i+1}} = \emptyset$ for every $i \in \{1, \dots, n\}$, as any non-empty subset of vertices of the clique is dominating.

Now we show the partitioning of the states into $L_{u_x, u_{x+1}}$, $R_{u_x, u_{x+1}}$, and $N_{u_x, u_{x+1}}$ for each $x \in \{1, \dots, n\}$, see Figure 2.13 for an illustration. First, observe that all closed neighborhoods of u_{2x} for $4 \leq 2x \leq n-2$ contain exactly 2 guards in all the states we defined for this strategy. Let x be an even integer such that $4 \leq x \leq n-2$. Let $e = \{u_{x+1}, u_{x+2}\}$.

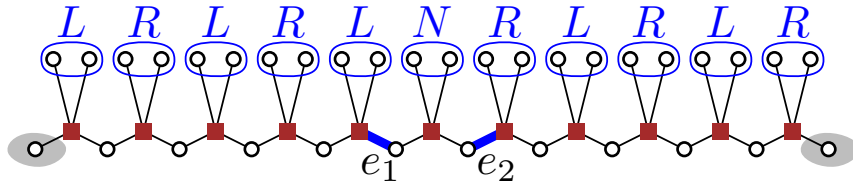


Figure 2.13: The states of β_{2x} that belong to L_e, R_e, N_e for e equal to the edges e_1 and e_2 .

We show that $\beta_x \in N_e$ by a contradiction. Assume that in some transition from β_x a guard moved through e . As in β_x vertex u_{x+1} is not occupied the guard must have moved from u_{x+1} . However, then $N[u_x]$ would have 3 guards after the transition which cannot happen as we observed; a contradiction.

For edges that could not be addressed in the argument because they are too close to the \widehat{X} vertex – $e_1 = \{u_1, u_2\}$ and $e_2 = \{u_3, u_4\}$. We observe that for red-even RW-cycles $\alpha_1 \in N_{e_1}$ and $\gamma \in N_{e_2}$. For red-odd RW-cycles $\beta_2 \in N_{e_2}$ and $\beta_n \in N_{e_1}$.

Now we claim that for any even x such that $2 \leq x \leq n$, $e = \{u_{x+1}, u_{x+2}\}$, the states β_y where $y \neq x$ are in L_e if and only if $u_{x+1} \in P(\beta_y)$, and they are in R_e otherwise. We shall prove this more intuitively, as otherwise the claim can be proved by exhaustively listing all edges in all the transitions. First notice, that all movements from β_y which are not incident to leaves are performed over a continuous part of the cycle which starts in u_y , and that they move “away” from u_y towards the other end of the part. The movements always move an occupied $\widehat{0}$ to $\widehat{2}$ and if the part continues then moves the $\widehat{2}$ to the adjacent $\widehat{0}$ (this is true even when moving through the \widehat{X} vertex). Hence, if e (that is not incident to \widehat{X}) is included in the part of the movement, then we move (u_{x+1}, u_{x+2}) if and only if u_{x+1} is occupied, and we move (u_{x+2}, u_{x+1}) if and only if u_{x+1} was unoccupied. This proves the claim.

Remainder of the edges which start at even positions and their N , L , and R sets can be obtained by the same argument on reversed labelling. □

2.5.4 Constant Component Reductions

The following lemma shows that considering the constant component cases completes the list of all necessary reductions.

Lemma 2.5.19. Let us have a cactus graph G . After an exhaustive application of leaf and cycle reductions the reduced cactus graph G' is either a base case or it contains a leaf cycle of constant size.

Proof. First, by Observation 2.5.3 the cactus always contains a leaf component. If we exhaustively apply tree reductions, then by Lemma 2.5.7 we are either done or there is a leaf cycle C . In 2.5.17 we saw that an exhaustive application of the cycle rules results either in a base case or a cycle with alternating $\widehat{2}$ and $\widehat{0}$ vertices, which gets tackled by Reduction c_6 . The cases that remain are cycles of constant sizes where none of the reductions may be applied. □

We obtain the list of constant leaf cycles by the following procedure. First, we apply Reductions c_1 and c_2 exhaustively. This removes all pink vertices from the leaf cycle. Now, let us scan over the vertices of the leaf cycle in a linear order of vertices along the cycle, starting from the connecting vertex. On the one hand, whenever there is a cycle reduction applicable on the vertices which were scanned so far, then we can apply it. Hence, such a leaf cycle does not belong to constant leaf cycle cases. On the other hand, when the cycle returns back to the connecting vertex and still no cycle reduction may be used, then this cycle constitutes a constant leaf cycle. We present a full search diagram in Figure 2.14.

Again, we shall denote the reductions concisely as defined by Definition 2.5.5. However, in constant component reductions the leaf sequence describes the whole leaf cycle and the connecting vertex is listed as the first and the last vertex. The vertices of the leaf cycle will be denoted by $u, u_1, u_2, \dots, u_{n-1}, u$ where u is the connecting vertex. Let R_1, \dots, R_{n-1} denote sets of all leaves adjacent to vertices u_1, \dots, u_{n-1} , respectively. Note that the size

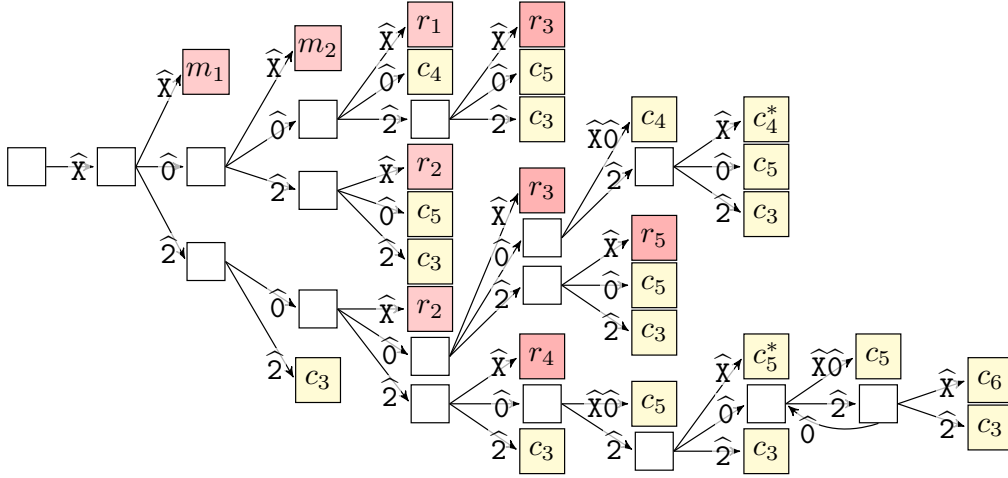


Figure 2.14: Case analysis of applied reductions on a leaf cycle. Vertices $\hat{1}$ were removed first by exhaustively applying their reductions. Scanning over vertices of a leaf cycle in order from the connecting vertex we identify these cases. The leaves show which reduction should be used for the scanned leaf cycle. Labels c_1 up to c_5 (yellow leaves) signify cycle reductions; labels m_i and r_i (red leaves) signify constant component reductions; nodes with a star * require Observation 2.5.20. We can check that all the cases are covered by seeing that all inner (empty) nodes have outgoing edges labelled $\hat{0}$, $\hat{2}$, and \hat{X} .

$0 \leq |R_i| \leq 2$ and directly coincides with color of respective vertex u_i . See Figure 2.15 for an example of a leaf sequence of constant leaf cycle and notation of its vertices.

Recall that the cycle reductions may be used even when the result does not create a simple graph, which is resolved in Section 2.5.4.1.

Observation 2.5.20. A strategy for a leaf cycle (u, u_1, u_2, u) with colors $(\hat{X}, \hat{2}, \hat{2}, \hat{X})$ is built in such a way that the edge (u_1, u_2) holds Property 2.5.9 (even though it is not incident to a $\hat{0}$ vertex) which makes an expansion of Reductions c_4 or c_5 over this edge possible.

Proof. This leaf cycle gets reduced by Reduction c_3 , then m_2 , and last with tree reduction t_3 . We show that in the strategy resulting for expansions holds Property 2.5.9 on edges $\{u, u_1\}$ and $\{u, u_2\}$. See Figure 2.15 for an illustration. Checking the exact movements of this strategy, we have that

$$(u_1, u_2) \notin \mathcal{T}(L_{u, u_1}, R_{u, u_1}), (u_1, u_2) \notin \mathcal{T}(L_{u, u_1}, N_{u, u_1}), (u_1, u_2) \in \mathcal{T}(R_{u, u_1}, N_{u, u_1}).$$

In particular, we may set $L_{u, u_1} = N_{u_1, u_2}$, $R_{u, u_1} = L_{u_1, u_2}$, and $N_{u, u_1} = R_{u_1, u_2}$. As the edge move sets for $\{u, u_1\}$ holds the properties which require all these sets to be non-empty, we have that they hold for $\{u_1, u_2\}$ as well. \square

From Observation 2.5.20 we know that the cases $(\hat{X}, \hat{2}, \hat{0}, \hat{0}, \hat{0}, \hat{2}, \hat{X})$ and $(\hat{X}, \hat{2}, \hat{0}, \hat{2}, \hat{0}, \hat{2}, \hat{X})$ can be reduced by Reductions c_4 and c_5 , respectively. See these cases in Figure 2.14.

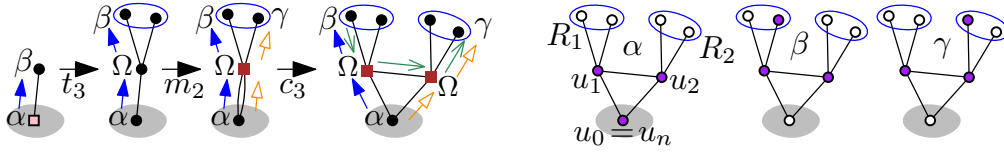


Figure 2.15: **Left:** Building the strategy for a $(\widehat{X}, \widehat{2}, \widehat{2}, \widehat{X})$ leaf cycle. The states α , β , and γ are representants of sets $L_{u,a}$, $R_{u,a}$, and $N_{u,a}$, respectively. **Right:** Example guard configurations for states α , β , and γ .

2.5.4.1 Loops and Multiedges

Similarly to Observation 2.5.20, for the constant cases where we need to show that the properties hold. By allowing cycle reductions to apply in cases where the vertices a and b are adjacent, or even identical, we allowed the result of the reduction to contain multiedges or loops. This intermediate form of the graph can be thought of as a generalized cactus graph.

Definition 2.5.21 (Cactus multigraph). Let the *cactus multigraph* be a multigraph (possibly with loops) that is connected and its every edge lies on at most one cycle.

A cactus multigraph differs from a cactus graph by allowing loops on arbitrary vertices (cycles of size 1) and allowing 2 multiedges between some vertices (cycles of size 2). The cactus multigraph may be changed to a cactus graph by removing multiedges and loops. The following two reductions take care of that.

Reduction 10. m_1 Let G' be G with one loop removed.

Reduction 11. m_2 Let G' be G with a multiedge $\{u, v\}$ (2 edges) where v has degree 2 (1 neighbor) changed to a single edge.



Figure 2.16: **Left:** loop reduction m_1 ; **Right:** multiedge reduction m_2

Observe these reductions on Figure 2.16. We now prove that they do not need any additional guards.

Lemma 2.5.22. Let G' be G after application of Reduction m_1 . G is defended with the same number of guards as G' .

Proof. The strategy on G can be easily adapted to G' by replacing any guard movement along the loop of u by not moving the guard on u , thus $\Gamma_m^\infty(G') \leq \Gamma_m^\infty(G)$. At the same time, any strategy on G' is applicable on G , so $\gamma_m^\infty(G) \leq \gamma_m^\infty(G')$. The equality follows from Lemma 2.4.2. However, we would like the loop in u to have Property 2.5.9.

Intuitively, to keep the properties, we could say that at any configuration where u is occupied the guard can be moved along the loop in any direction or to be forbidden from moving along it while the configuration stays the same. Formally, we can achieve the same by setting $S_G = S'_{G'} \sqcap_{S'u} \{\alpha, \beta, \gamma\}$. We now set that $\mathcal{T}(\alpha, \beta) = \{(u, u)\}$. This creates $L_{u,u} = \alpha$, $R_{u,u} = \beta$, and $Q_{u,u} = \Omega \setminus \{\alpha, \beta\}$. This altered strategy holds Property 2.5.9 for the loop of u as the sets $L_{u,u}$, $R_{u,u}$, and $N_{u,u}$ are non-empty and dominating S_G because $\mathcal{S}'(u)$ dominates $S'_{G'}$. \square

In our case, Reduction m_1 gets used after Reduction c_4 is used on $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{0}, \widehat{X})$ or after Reduction c_5 is used on $(\widehat{X}, \widehat{0}, \widehat{2}, \widehat{0}, \widehat{X})$. It could also be used on $(\widehat{X}, \widehat{1}, \widehat{X})$ after Reduction c_1 ; but in that case we can remove the multiedge first.

Lemma 2.5.23. Let G' be G after application of Reduction m_2 . G is defended with the same number of guards as G' .

Proof. Let e_1, e_2 be the two different edges $\{u, v\}$ oriented as (u, v) in G . We assume that G' is G with e_2 removed. Lower and upper bound are clear as every move along e_2 can be changed to a move along e_1 and the strategy on G' is applicable to G without change. The challenge is, again, to show that Property 2.5.9 holds for e_1 and e_2 in G .

Let $\beta' = \mathcal{S}'(v)$ and $\alpha' = \Omega' \setminus \beta$. To prove the property on e_1 and e_2 , we will modify the strategy on G' in the following way. If $\beta' \neq \Omega'$, then there is a move along e_1 in G' . In that case, we set $S_{G'} = S'_{G'} \sqcap_{\beta'} \{\beta, \gamma\}$ while we alter the movements $\mathcal{T}(\alpha, \gamma)$ to move along e_2 instead of e_1 . The edge states have $\alpha \in L_{e_1}$, $\beta \in R_{e_1}$, and $\gamma \in N_{e_1}$, and similarly for e_2 (with swapped β and γ).

Second case is that $\beta' = \Omega'$ while $\alpha' \neq \Omega'$. Here, we alter the strategy such that for all states where u is not occupied, we move the guard from v to u . This makes it so that v is occupied in states α which we now split into α_1 and α_2 in the same way as in the previous case.

The last case is $\beta' = \Omega'$ while $\alpha' = \Omega'$. Here we set $S_{G'} = S'_{G'} \sqcap_{\Omega'} \{\alpha_1, \alpha_2, \alpha_3\}$ and setting $\mathcal{T}(\{\alpha_1, \alpha_2\}) = \{(u, v), (v, u)\}$, i.e., transitioning along e_1 and e_2 in opposite directions. Also $\mathcal{T}(\alpha_1, \alpha_3)$ and $\mathcal{T}(\alpha_2, \alpha_3)$ have all guards stationary. This makes edge states as $\alpha_1 \in L_{e_1}$, $\alpha_2 \in R_{e_1}$, and $\alpha_3 \in N_{e_1}$ while the exact same edge states work for e_2 .

In all the cases the edge states are non-empty, hence, Property 2.5.9 holds for e_1 and e_2 after Reduction m_2 . \square

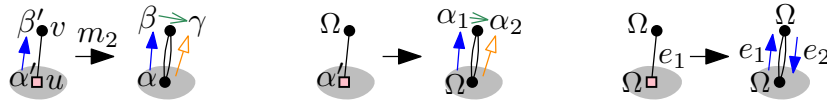


Figure 2.17: Cases of Reduction m_2 . **Left:** There is a movement along the edge. **Middle:** Leaf is permanently occupied. **Right:** Leaf and its neighbor are permanently occupied.

We note that in our strategy the case where v is permanently defended shall not occur.

If we did not use Reduction m_2 the number of constant size leaf cycle reductions would be significantly bigger. It gets used after reduction of $(\widehat{X}, \widehat{0}, \widehat{1}, \widehat{X})$ or $(\widehat{X}, \widehat{1}, \widehat{1}, \widehat{X})$ by c_1 , $(\widehat{X}, \widehat{2}, \widehat{1}, \widehat{X})$ by c_2 , $(\widehat{X}, \widehat{2}, \widehat{2}, \widehat{X})$ by c_3 , $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{0}, \widehat{0}, \widehat{X})$ or $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X})$ by c_4 , $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{0}, \widehat{X})$ or $(\widehat{X}, \widehat{0}, \widehat{2}, \widehat{0}, \widehat{2}, \widehat{X})$ by c_5 . Without Reduction m_2 each of these cases would have to be analyzed separately.

2.5.4.2 Constant Size Leaf Cycle Reductions

By Lemma 2.5.19 the last cases that have to be resolved are covered by the following reductions. See Table 2.3 for accompanying lower bound and upper bound proof illustrations. Also see Figure 2.9 for diagram of notions used within proofs of these reductions.

Reduction 12. r_1 $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{X}) \rightarrow (\widehat{1}) + 0$

Reduction 13. r_2 $(\widehat{X}, \widehat{0}, \widehat{2}, \widehat{X}) \rightarrow (\widehat{1}) + 1$

Reduction 14. r_3 $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X}) \rightarrow (\widehat{2}) + 1$

Reduction 15. r_4 $(\widehat{X}, \widehat{2}, \widehat{0}, \widehat{2}, \widehat{X}) \rightarrow (\widehat{2}) + 2$

Reduction 16. r_5 $(\widehat{X}, \widehat{2}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X}) \rightarrow (\widehat{2}) + 2$

Table 2.3: List of constant component reductions; thick red edges do not hold Property 2.5.9.

Reduction	Lower bound	Upper bound
r_1		
r_2		
r_3		
r_4		
r_5		

Now we proceed to show correctness of these reductions. First group consists of reductions where a leaf cycle is reduced to $\widehat{1}$ vertex u and its leaf v . The vertices of the expanded leaf cycle are denoted by $u, u_1, u_2, \dots, u_{n-1}, u$.

Lemma 2.5.24. Let G' be G after application of Reduction r_1 . G is defended with the same number of guards as G' .

Proof. Using Observation 2.4.3 to identify u_2 with u_1 then using Reduction m_2 results in lower bound of 0.

For the upper bound, we first expand $\{u, v\}$ to multiedges e_1 and e_2 as per Reduction m_2 . Then we take G' and change it to G by splitting v into two vertices u_1 and u_2 . We create β by substituting all occurrences of v in $P(\beta')$ with u_1 , and create γ by substituting all occurrences of v in $P(\gamma')$ with u_2 . The transition between them becomes $\mathcal{T}(\beta, \gamma) = \{(u_1, u_2)\}$. The strategy is interface equivalent as the strategy did not change states or transitions of the interface.

We set $L_{u_1, u_2} = N_{u, u_1}$, $R_{u_1, u_2} = N_{u, u_1}$, and $N_{u_1, u_2} = L_{u, u_1}$ so the new edge $\{u_1, u_2\}$ holds Property 2.5.9 and the strategy for G holds Property 2.5.11.

No guard was added so $\gamma_m^\infty(G) \leq \gamma_m^\infty(G')$ and by Lemma 2.4.2 we get that G is defended with the same number of guards as G' . \square

We recall that by R_i we denote all leaves adjacent to u_i .

Lemma 2.5.25. Let G' be G after application of Reduction r_2 . G is defended with 1 more guard than G' .

Proof. Using Observation 2.4.8 on $\{u_2\} \cup R_2$ then using Reduction m_2 results in lower bound of 1.

We do the same expansion as in the proof of Lemma 2.5.24. After that, we use Lemma 2.4.34 to add leaves R_2 to u_2 while using one extra guard to defend it. Graph holds Property 2.5.11 by the same argument as in the proof of Lemma 2.5.24. We added one extra guard which results in $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 1$ and by Lemma 2.4.2 we get that G is defended with one more guard than G' . \square

We now prove correctness of the other three cases. The reduced graph G' now consists of a single $\widehat{2}$ vertex u (and its leaves). The partial labelled strategy on G' has states α' and β' that defend the two leaves adjacent to u . Also, let $\delta' = \Omega' \setminus (\alpha' \cup \beta')$, which may be an empty set.

Lemma 2.5.26. Let G' be G after application of Reduction r_3 . G is defended with 1 more guard than G' .

Proof. Using Observation 2.4.8 on u_3 and one of its leaves, identifying u_2 with u using Observation 2.4.3, and using Reductions m_1 and m_2 to remove loops and multiedges results in lower bound of 1.

For the upper bound, let u_1 and u_3 be the two leaves adjacent to u in G' . Let $\alpha' = \mathcal{S}'(u_1)$ and $\beta' = \mathcal{S}'(u_3)$. We make $S_{G'} = S'_{G'} \sqcup_{\beta'} \{\beta, \gamma\}$. Now we expand the graph G' by first applying Lemma 2.4.34 on u_3 , adding 2 new leaves to it using one additional guard. Next, we add a vertex u_2 while connecting it to u_1 and u_3 and we move γ from R_3 to u_2 which is easy as u_2 is a neighbor of u_3 . The only major change in transitions is that $\mathcal{T}(\alpha, \gamma) =$

$\{(u_1, u_2), (u, u), (u_3, u_3)\}$ instead of $\{(u_1, u), (u, u_3), (u_3, u_2)\}$. No other transitions change, and u behaves the same, so the exchanged graphs are interface equivalent. See Figure 2.18 for strategy S_G .

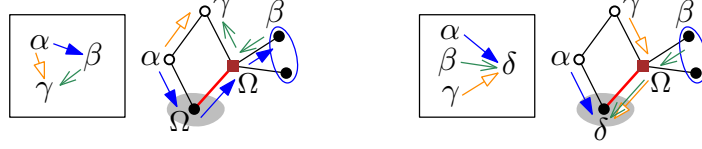


Figure 2.18: Strategy for $(\widehat{X}, \widehat{O}, \widehat{O}, \widehat{2}, \widehat{X})$ leaf cycle

We note that each is traversed at some point and that $\alpha \in N_{u_2, u_3}$, $\beta \in N_{u_1, u_2}$, and $\gamma \in N_{u, u_1}$ so these edges hold Property 2.5.9 and the strategy for G holds Property 2.5.11. The edge $\{u, u_3\}$ does not need to hold the property as it is a special case tackled in Lemma 2.5.29.

We got that $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 1$ and by Lemma 2.4.2 we get that G is defended with one more guard than G' . \square

Lemma 2.5.27. Let G' be G after application of Reduction r_4 . G is defended with 2 more guards than G' .

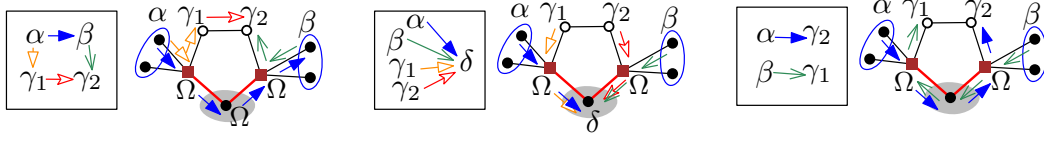
Proof. Using Observation 2.4.8 first on $\{u_1, v_1\}$ where $v_1 \in R_1$, then again on $\{u_3, v_3\}$ where $v_3 \in R_3$, identifying u_2 with u using Observation 2.4.3, and using Reductions m_1 and m_2 to remove loops and multiedges results in lower bound of 2.

For upper bound, repeat exactly the expansion from Lemma 2.5.26 on G' which uses one extra guard. Continue by applying Lemma 2.4.34 on u_1 which adds the leaves R_1 using one extra guard while returning the defending labelled strategy on G . The properties for edges $\{u_1, u_2\}$, $\{u_2, u_3\}$, and interface equivalency still hold from Lemma 2.5.26. However, we can split γ into two states γ_1 and γ_2 which dictates whether $\mathcal{T}(\gamma_i, \delta)$ traverses through $\{(u_2, u_1), (u_1, u)\}$ or $\{(u_2, u_3), (u_3, u)\}$. This ensures Property 2.5.11 for $\{u, u_1\}$ and $\{u, u_3\}$ as former cannot be traversed from γ_2 and latter from γ_1 . Hence, we have $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 2$ and by Lemma 2.4.2 we get that G is defended with two more guards than G' . \square

Lemma 2.5.28. Let G' be G after application of Reduction r_5 . G is defended with 2 more guards than G' .

Proof. Using Observation 2.4.8 first on $\{u_1, v_1\}$ where $v_1 \in R_1$, then again on $\{u_4, v_4\}$ where $v_4 \in R_4$, and last identifying u_2 and u_3 with u using Observation 2.4.3 results in lower bound of 2.

For upper bound, repeat exactly the expansion from Lemma 2.5.27 on G' which uses two extra guards (we do not use part of the proof which proved the property). Then we make $S_{G'} = S'_{G'} \sqcup_{S'(u_2)} \{\gamma_1, \gamma_2\}$, i.e., splitting γ into γ_1 and γ_2 . We expand G' to G by splitting u_2 into u_2 and u_3 (while renaming u_3 to u_4). We preserve a guard of γ_1 on u_2 and


 Figure 2.19: Strategy for $(\widehat{X}, \widehat{2}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X})$ leaf cycle

γ_2 on u_3 . Transition between them will be $\mathcal{T}(\gamma_1, \gamma_2) = \{(u_2, u_3), (u, u)\}$. This is interface equivalent. See Figure 2.19 for strategy S_G .

We have Property 2.5.11 as each edge is traversed and $\{u_1, u_2\}$ cannot be traversed from γ_2 , $\{u_2, u_3\}$ from α , and $\{u_3, u_4\}$ cannot be traversed from γ_1 . We note that the other two cycle edges $\{u, u_1\}$ and $\{u, u_4\}$ are part of the exception which is tackled in Lemma 2.5.29. Hence, we have $\gamma_m^\infty(G) \leq \gamma_m^\infty(G') + 2$ and by Lemma 2.4.2 we get that G is defended with two more guards than G' . \square

Now we tackle the exception in Property 2.5.11 which influences Reductions r_3 and r_5 .

Lemma 2.5.29. The order of reductions can be changed so that in a $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X})$ or $(\widehat{X}, \widehat{2}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{X})$ leaf cycle Property 2.5.9 is not required for edges that connect a \widehat{X} and a $\widehat{2}$ vertex.

Proof. Let us label by e an edge which connects a \widehat{X} and a $\widehat{2}$ vertex. Reductions which require the Property 2.5.9 on an edge are Reductions c_1 , c_4 , and c_5 . If e is not a result of any of these reductions then there is no need for e to hold Property 2.5.9. Otherwise, let us analyze the cases separately.

- Reduction c_1 resulted in e – before reduction we had $(\widehat{X}, \widehat{1}, \widehat{2}, \dots)$ where we can use Reduction c_2 instead. This results in $(\widehat{X}, \widehat{2}, \dots)$ without needing the property for e .
- Reduction c_4 resulted in e – before reduction we had $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{0}, \widehat{2}, \widehat{0}, \widehat{0}, [\widehat{2},]\widehat{X})$. Hence, we may use Reduction c_5 instead. This results in $(\widehat{X}, \widehat{0}, \widehat{0}, \widehat{0}, [\widehat{2},]\widehat{X})$ where e has the property.
- Reduction c_5 resulted in e – before reduction we had $(\widehat{X}, \widehat{0}, \widehat{2}, \widehat{0}, \widehat{2}, \widehat{0}, \widehat{0}, [\widehat{2},]\widehat{X})$ so we may use Reduction c_5 on the second $\widehat{2}$ vertex instead. This results in $(\widehat{X}, \widehat{0}, \widehat{2}, \widehat{0}, [\widehat{2},]\widehat{X})$ where e has the property.

We used other reductions to avoid reaching these leaf components by reductions that would require Property 2.5.9. The first described case can be used at any point. The last two described cases are used on constant leaf components and as the result is different, it follows that their edges hold the property. \square

This concludes the constant component reductions which together with cycle components and approach described in Section 2.5.1 give us a polynomial algorithm to solve M-ETERNAL DOMINATION.

2.6 Future Work

The presented tools could be useful in a future study of the M-ETERNAL DOMINATION on different graph classes. For instance, grids of size $\{3, 5\} \times n$ were extensively studied [89, 106]. We believe it would be interesting to see to which extent the tools could be applied in study of grids of less restricted dimensions.

Another noteworthy class of graphs are the so called dually chordal graphs, for which many domination related problems are polynomial time solvable. It would be interesting to see whether M-ETERNAL DOMINATION remains polynomial time solvable as well.

Furthermore, the computational complexity of the decision variant of the m-eternal domination problem is still mostly unknown as mentioned in the introduction. It remains open whether the problem is in PSPACE and whether it is PSPACE-hard.

2.7 Additional observations

In this short section to discuss several notions connected to complete strategies.

2.7.1 Complete Strategies

We note that if the strategy S_G was a complete graph, then strategy S'_G created by the application of Lemma 2.4.38 is also a complete graph.

Property 2.7.1. A partial labelled strategy $\mathcal{B} = (G, S_G, P, \mathcal{T}, R)$ is *complete* if S_G is a complete graph, i.e., there is $\{\alpha, \beta\} \in \mathbb{F}$ for every $\alpha, \beta \in \Omega$.

We note that there are graphs where every optimal strategy is not complete, see Section 2.7.2 for such an example. Complete strategies can be effectively pruned to contain at most $|V(G)|$ states in the following way.

Lemma 2.7.2. For any complete defending labelled strategy of cardinality k with the minimum number of vertices of S_G it holds $|V(S_G)| \leq |V(G)| - k + 1$.

Proof. Pick an arbitrary complete defending strategy S_G which uses k guards. For each $v \in V(G)$ we shall pick one state $\alpha_v \in V(S_G)$ such that $v \in P(\alpha)$. First, pick any $\alpha \in V(S_G)$ and assign it as state to each $v \in P(\alpha)$. Then, for every $v \in V(G) \setminus P(\alpha)$ assign $\alpha_v \in \mathcal{S}(v)$ as its state. We just picked $|V(G)| - k + 1$ states such that they form a strategy where every pair of states is traversable and which is defending as it covers all the vertices of G . \square

Similar to completeness of a strategy we may talk about the graph class of S_G to describe its properties.

2.7.2 Non-complete Strategy

Observation 2.7.3. An optimal m-Eternal Domination strategy on 5×5 grid uses at least 7 guards.

Proof. Let us denote vertices of the grid by $u_{i,j}$ where $1 \leq i, j \leq 5$.

First, we show a lower bound of 7. Assume for a contradiction that there is a defending strategy S_6 with at most 6 guards. Any state of S_6 needs to dominate all 25 vertices. There must exist a state C where $u_{2,2}$ is occupied. In C there also must be at least one guard in the closed neighborhood of each corner ($u_{1,1}$, $u_{5,1}$, $u_{1,5}$, and $u_{5,5}$). In the grid a vertex may dominate at most 5 vertices and a vertex on the side of the grid may dominate at most 4 vertices. All vertices in the closed neighborhood of corners are on the side of the grid. Additionally, vertex which dominates $u_{1,1}$ may dominate at most 2 new vertices, as $u_{2,2}$ already dominates many of vertices in its neighborhood. In total, the 6 guards of C may dominate at most $2 \cdot 5 + 3 \cdot 4 + 2 = 24$, a contradiction. \square

The upper bound can be shown by construction of a strategy, however, we have no good tools to show that all the strategies are not complete graphs – we found this using a full strategy-space search. A construction which uses 7 guards contains three states and majority of their reflections and rotations, see them on Figure 2.20. In this case, we do not show the strategy, as it contains roughly 20 states (depending on a slight optimization, it may be less) that would contain 190 transitions.

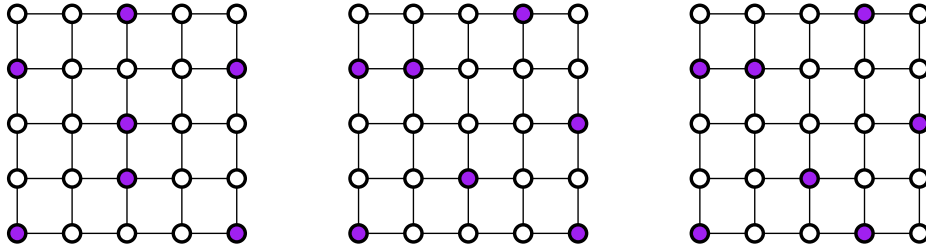


Figure 2.20: The 5×5 grid has 19 m-eternal dominating sets. Each of the configurations can be expressed as a combination of rotations and reflections of exactly one of these 3 basic configurations. Each of the 19 configurations is necessary for the strategy and can move into at most 12 other states.

Bears with Hats

Consider the following hat guessing game. A bear sits on each vertex of a graph G , and a demon puts on each bear a hat colored by one of h colors. Each bear sees only the hat colors of his neighbors. Based on this information only, each bear has to guess g colors and he guesses correctly if his hat color is included in his guesses. The bears win if at least one bear guesses correctly for any hat arrangement.

We introduce a new parameter – fractional hat chromatic number $\hat{\mu}$, arising from the hat guessing game. The parameter $\hat{\mu}$ is related to the hat chromatic number which has been studied before. We present a surprising connection between the hat guessing game and the independence polynomial of graphs. This connection allows us to compute the fractional hat chromatic number of chordal graphs in polynomial time, to bound fractional hat chromatic number by a function of maximum degree of G , and to compute the exact value of $\hat{\mu}$ of cliques, paths, and cycles.

3.1 Introduction

In this chapter, we study a variant of a hat guessing game. In these types of games, there are some entities – players, pirates, sages, or, as in our case, bears. A bear sits on each vertex of graph G . There is some adversary (a demon in our case) that puts a colored hat on the head of each bear. A bear on a vertex v sees only the hats of bears on the neighboring vertices of v but he does not know the color of his own hat. Now to defeat the demon, the bears should guess correctly the color of their hats. However, the bears can only discuss their strategy before they are given the hats. After they get them, no communication is allowed, each bear can only guess his hat color. The variants of the game differ in the bears' winning condition.

The first variant was introduced by Ebert [45]. In this version, each bear gets a red or blue hat (chosen uniformly and independently) and they can either guess a color or pass. The bears see each other, i.e. they stay on vertices of a clique. They win if at least one bear guesses his color correctly and no bear guesses a wrong color. The question is what

is the highest probability that the bears win achievable by some strategy. Soon, the game became quite popular and it was even mentioned in NY Times [97].

Winkler [107] studied a variant where the bears cannot pass and the objective is how many of them guess correctly their hat color. A generalization of this variant for more than two colors was studied by Feige [51] and Aggarwal [2]. Butler et al. [21] studied a variant where the bears are sitting on vertices of a general graph, not only a clique. For a survey of various hat guessing games, we refer to theses of Farnik [50].

In this chapter, we study a variant of the game introduced by Farnik [50], where each bear has to guess and they win if at least one bear guesses correctly. He introduced a hat guessing number HG of a graph G (also named as hat chromatic number and denoted μ in later works) which is defined as the maximum h such that bears win the game with h hat colors. We study a variant where each bear can guess multiple times and we consider that a bear guesses correctly if the color of his hat is included in his guesses. We introduce a parameter *fractional hat chromatic number* $\hat{\mu}$ of a graph G , which we define as the supremum of $\frac{h}{g}$ such that each bear has g guesses and they win the game with h hat colors.

Albeit the hat guessing game looks like a recreational puzzle, connections to more “serious” areas of mathematics and computer science were shown – like coding theory [46, 71], network coding [55, 96], auctions [2], finite dynamical systems [53], and circuits [108]. We exhibit a connection between the hat guessing game and the independence polynomial of graphs, which is our main result. This connection allows us to compute the optimal strategy of bears (and thus the value of $\hat{\mu}$) of an arbitrary chordal graph in polynomial time. We also prove that the fractional hat chromatic number $\hat{\mu}$ is asymptotically equal, up to a logarithmic factor, to the maximum degree of a graph. Finally, we compute the exact value of $\hat{\mu}$ of graphs from some classes, like paths, cycles, and cliques.

We would like to point out that the existence of the algorithm computing $\hat{\mu}$ of a chordal graph is far from obvious. Butler et al. [21] asked how hard is to compute $\mu(G)$ and the optimal strategy for the bears. Note that a trivial non-deterministic algorithm for computing the optimal strategy (or just the value of $\mu(G)$ or $\hat{\mu}(G)$) needs exponential time because a strategy of a bear on v is a function of hat colors of bears on neighbors of v (we formally define the strategy in Section 3.2). It is not clear if the existence of a strategy for bears would imply a strategy for bears where each bear computes his guesses by some efficiently computable function (like linear, computable by a polynomial circuit, etc.). This would allow us to put the problem of computing μ into some level of the polynomial hierarchy, as noted by Butler et al. [21]. On the other hand, we are not aware of any hardness results for the hat guessing games. The maximum degree bound for $\hat{\mu}$ does not imply an exact efficient algorithm computing $\hat{\mu}(G)$ as well. This phenomenon can be illustrated by the edge chromatic number χ' of graphs. By Vizing’s theorem [36, Chapter 5], it holds for any graph G that $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$. However, it is NP-hard to distinguish between these two cases [67].

Organization of this chapter. We finish this section with a summary of results about the variant of the hat guessing game we are studying. In the next section, we present notions

used in this chapter and we define formally the hat guessing game. In Section 3.3, we formally define the fractional hat chromatic number $\hat{\mu}$ and compare it to μ . In Section 3.4, we generalize some previous results to the multi-guess setting. We use these tools to prove our main result in Section 3.5 including the poly-time algorithm that computes $\hat{\mu}$ for chordal graphs. The maximum degree bound for $\hat{\mu}$ and computation of exact values of paths and cycles are provided in Section 3.6.

3.1.1 Related and Follow-up Works

As mentioned above, Farnik [50] introduced a hat chromatic number $\mu(G)$ of a graph G as the maximum number of colors h such that the bears win the hat guessing game with h colors and played on G . He proved that $\mu(G) \leq O(\Delta(G))$ where $\Delta(G)$ is the maximum degree of G .

Since then, the parameter $\mu(G)$ was extensively studied. The parameter μ for multipartite graphs was studied by Gadouleau and Georgiu [54] and by Alon et al. [4]. Szczechla [104] proved that μ of cycles is equal to 3 if and only if the length of the cycle is 4 or it is divisible by 3 (otherwise it is 2). Bosek et al. [17] gave bounds of μ for some graphs, like trees and cliques. They also provided some connections between $\mu(G)$ and other parameters like chromatic number and degeneracy. They conjectured that $\mu(G)$ is bounded by some function of the degeneracy $d(G)$ of the graph G . They showed that such function has to be at least exponential as for every $d \geq 1$ they presented a graph G of $d(G) = d$ such that $\mu(G) \geq 2^d$. This result was improved by He and Li [64] who showed that for every $d \geq 1$ there is a graph G of $d(G) = d$ and $\mu(G) \geq 2^{2^{d(G)-1}}$. Since $\hat{\mu}(G)$ is lower-bounded by $\Omega(\Delta(G)/\log \Delta(G))$ (as we show in Section 3.6) it holds that $\hat{\mu}$ can not be bounded by any function of degeneracy as there are graph classes of unbounded maximum degree and bounded degeneracy (e.g. trees or planar graphs). Recently, Kokhas et al. [82, 83] studied a non-uniform version of the game, i.e., for each bear, there could be a different number of colors of the hat. They considered cliques and almost cliques. They also provided a technique to build a strategy for a graph G whenever G is made up by combining G_1 and G_2 with known strategies. We generalize some of their results and use them as “basic blocks” for our main result.

After the presentation of the preliminary version of our results in [A.2], Latyshev and Kokhas [87] extended ideas presented in this chapter to reason about the standard hat chromatic number. In particular, they found a family of graphs of unbounded maximum degree such that for each graph G in the family holds that $\mu(G) = \frac{4}{3}\Delta(G)$, thus they disproved a conjecture by Alon et al. [4] that $\mu(G) \leq \Delta(G) + 1$.

3.2 Preliminaries

We use standard notions of the graph theory. For an introduction to this topic, we refer to the book by Diestel [36]. We denote a clique as K_n , a cycle as C_n , and a path as P_n , each on n vertices. The maximum degree of a graph G is denoted by $\Delta(G)$, where we shorten it

to Δ if the graph G is clear from the context. The neighbors of a vertex v are denoted by $N(v)$. We use $N[v]$ to denote the closed neighborhood of v , i.e. $N[v] = N(v) \cup \{v\}$. For a set U of vertices of a graph G , we denote $G \setminus U$ a graph induced by vertices $V(G) \setminus U$, i.e., a graph arising from G by removing the vertices in U .

A *hat guessing game* is a triple $\mathcal{H} = (G, h, g)$ where

- $G = (V, E)$ is an undirected graph, called the *visibility graph*,
- $h \in \mathbb{N}$ is a *hatness* that determines the number of different possible hat colors for each bear, and
- $g \in \mathbb{N}$ is a *guessing number* that determines the number of guesses each bear is allowed to make.

The rules of the game are defined as follows. On each vertex of G sits a bear. The demon puts a hat on the head of each bear. Each hat has one of h colors. We would like to point out, that it is allowed that bears on adjacent vertices get a hat of the same color. The only information the bear on a vertex v knows are the colors of hats put on bears sitting on neighbors of v . Based on this information only, the bear has to guess a set of g colors according to a deterministic strategy agreed to in advance. We say bear *guesses correctly* if he included the color of his hat in his guesses. The bears win if at least one bear guesses correctly.

Formally, we associate the colors with natural numbers and say that each bear can receive a hat colored by a color from the set $S = [h] = \{0, \dots, h-1\}$. A *hats arrangement* is a function $\varphi: V \rightarrow S$. A strategy of a bear on v is a function $\Gamma_v: S^{|N(v)|} \rightarrow \binom{S}{g}$, and a *strategy for \mathcal{H}* is a collection of strategies for all vertices, i.e. $(\Gamma_v)_{v \in V}$. We say that a strategy is *winning* if for any possible hats arrangement $\varphi: V \rightarrow S$ there exists at least one vertex v such that $\varphi(v)$ is contained in the image of Γ_v on φ , i.e., $\varphi(v) \in \Gamma_v((\varphi(u))_{u \in N(v)})$. Finally, the game \mathcal{H} is *winning* if there exists a winning strategy of the bears.

As a classical example, we describe a winning strategy for the hat guessing game $(K_3, 3, 1)$. Let us denote the vertices of K_3 by v_0, v_1 and v_2 and fix a hats arrangement φ . For every $i \in [3]$, the bear on the vertex v_i assumes that the sum $\sum_{j \in [3]} \varphi(v_j)$ is equal to i modulo 3 and computes its guess accordingly. It follows that for any hat arrangement φ there is always exactly one bear that guesses correctly, namely the bear on the vertex v_i for $i = \sum_j \varphi(v_j) \pmod{3}$.

Some of our results are stated for a non-uniform variant of the hat guessing game. A non-uniform game is a triple $(G = (V, E), \mathbf{h}, \mathbf{g})$ where $\mathbf{h} = (h_v)_{v \in V}$ and $\mathbf{g} = (g_v)_{v \in V}$ are vectors of natural numbers indexed by the vertices of G and a bear on v gets a hat of one of h_v colors and is allowed to guess exactly g_v colors. Other rules are the same as in the standard hat guessing game. To distinguish between the uniform and non-uniform games, we always use plain letters h and g for the hatness and the guessing number, respectively, and bold letters (e.g. \mathbf{h}, \mathbf{g}) for vectors indexed by the vertices of G .

For our proofs we use two classical results. First one is the inclusion-exclusion principle for computing a size of a union of sets.

Proposition 3.2.1 (folklore). For a union A of sets A_1, \dots, A_n holds that

$$|A| = \sum_{\emptyset \neq I \subseteq \{1, \dots, n\}} (-1)^{|I|+1} \left| \bigcap_{i \in I} A_i \right|.$$

The other one is the rational root theorem, which we use to derive an algorithm for computing an exact value of $\hat{\mu}$, if the value is rational.

Theorem 3.2.2 (Rational root theorem [86]). If a polynomial $a_n x^n + \dots + a_1 x + a_0$ has integer coefficients, then every rational root is of the form p/q where p and q are coprimes, p is a divisor of a_0 , and q is a divisor of a_n .

3.3 Fractional Hat Chromatic Number

From the hat guessing games, we can derive parameters of the underlying visibility graph G . Namely, the *hat chromatic number* $\mu(G)$ is the maximum integer h for which the hat guessing game $(G, h, 1)$ is winning, i.e., each bear gets a hat colored by one of h colors and each bear has only one guess – we call such game a single-guessing game. In this chapter, we study a parameter *fractional hat chromatic number* $\hat{\mu}(G)$ which arises from the hat multi-guessing game and is defined as

$$\hat{\mu}(G) = \sup \left\{ \frac{h}{g} \mid (G, h, g) \text{ is a winning game} \right\}.$$

Observe that $\mu(G) \leq \hat{\mu}(G)$. Farnik [50] and Bosek et al. [17] also study multi-guessing games. They considered a parameter $\mu_g(G)$ that is the maximum number of colors h such that the bears win the game (G, h, g) . The difference between μ_g and $\hat{\mu}$ is the following. If $\mu_g(G) \geq k$, then the bears win the game (G, k, g) and $\hat{\mu} \geq \frac{k}{g}$. If $\hat{\mu}(G) \geq \frac{p}{q}$, then there are $h, g \in \mathbb{N}$ such that $\frac{p}{q} = \frac{h}{g}$ and the bears win the game (G, h, g) . However, it does not imply that the bears would win the game (G, p, q) . In this section, we prove that if the bears win the game (G, h, g) then they win the game (G, kh, kg) for any constant $k \in \mathbb{N}$. The opposite implication does not hold – we discuss a counterexample at the end of this section. Unfortunately, this property prevents us from using our algorithm, which computes $\hat{\mu}$, to compute also μ of chordal graphs.

Moreover, by definition, the parameter $\hat{\mu}$ does not even have to be a rational number. In such a case, for each $p, q \in \mathbb{N}$, it holds that

- If $\frac{p}{q} < \hat{\mu}(G)$ then there are $h, g \in \mathbb{N}$ such that $\frac{p}{q} = \frac{h}{g}$ and the bears win the game (G, h, g) .
- If $\frac{p}{q} > \hat{\mu}(G)$ then the demon wins the game (G, p, q) .

For example, the fractional hat chromatic number $\hat{\mu}(P_3)$ of the path P_3 is irrational. In the case of an irrational $\hat{\mu}(G)$, our algorithm computing the value of $\hat{\mu}$ of chordal graphs

outputs an estimate of $\hat{\mu}(G)$ with arbitrary precision. We finish this section with a proof that the multi-guessing game is in some sense monotone.

Observation 3.3.1. Let $k \in \mathbb{N}$. If a game $\mathcal{H} = (G, h, g)$ is winning, then the game $\mathcal{H}_k = (G, k \cdot h, k \cdot g)$ is winning as well.

Proof. We derive a winning strategy for the game \mathcal{H}_k from a winning strategy for \mathcal{H} . Each bear interprets a color in $[k \cdot h]$ as a pair (i, c) where $i \in [k]$ and $c \in [h]$. Let A_v be guesses of the bear on v in the game \mathcal{H} . For the game \mathcal{H}_k , a strategy of the bear on v is to make guesses $\{(i, c) \mid i \in [k], c \in A_v\}$. It is straight-forward to verify that this is a winning strategy for \mathcal{H}_k . \square

Lemma 3.3.2. Let $(G = (V, E), h, g)$ be a winning hat guessing game. Let r' be a rational number such that $r' \leq h/g$. Then, there exist numbers $h', g' \in \mathbb{N}$ such that $h'/g' = r'$ and the hat guessing game (G, h', g') is winning.

Proof. Let $p, q \in \mathbb{N}$ such that $r' = p/q$ and $\text{GCD}(p, q) = 1$. Let¹ $\ell = \text{LCM}(h, p)$.

Let $\bar{h} = \ell, \bar{g} = \ell \cdot g/h$. By Observation 3.3.1 for $k = \ell/h$, the game (G, \bar{h}, \bar{g}) is winning. Let $h' = \ell$ and $g' = \ell \cdot q/p$. Since $p/q \leq h/g$ by the assumption, it holds that $g' \geq \bar{g}$. Thus, the bears have a strategy for (G, h', g') , as we increased the number of guesses and the hatness does not change ($h' = \bar{h} = \ell$). Moreover, $h'/g' = p/q = r'$. \square

It is straight-forward to prove a generalization of Lemma 3.3.2 for non-uniform games. However, for simplicity, we state it only for the uniform games. By the proof of the previous lemma, we know that we can use a strategy for (G, h, g) to create a strategy for a game $(G, k \cdot h, k \cdot g + \ell)$ for arbitrary $k, \ell \in \mathbb{N}$. A question is if we can do it in general: Can we derive a winning strategy if we decrease the fraction h/g , but the hatness h and the guessing number g are changed arbitrarily? It is true for cliques. We show in Section 3.4 that the bears win the game (K_n, h, g) if and only if $h/g \leq n$. However, it is not true in general. For example, for n large enough it holds that $\hat{\mu}(P_n) \geq 3$, as we show in Section 3.6 that $\hat{\mu}(P_n)$ converges to 4 when n goes to infinity. However, Butler et al. [21] proved that $\mu(T) = 2$ for any tree T . Thus, the bears lose the game $(P_n, 3, 1)$.

3.4 Basic Blocks

In this section, we generalize some results of Kokhas et al. [82, 83] about cliques and strategies for graph products, which we use for proving our main result. The single-guessing version of the next theorem (without the algorithmic consequences) was proved by Kokhas et al. [82, 83].

Theorem 3.4.1. Bears win a game $(K_n = (V, E), \mathbf{h}, \mathbf{g})$ if and only if

$$\sum_{v \in V} \frac{g_v}{h_v} \geq 1.$$

¹GCD stands for the greatest common divisor and LCM stands for the least common multiple.

Moreover, if there is a winning strategy, then there is a winning strategy $(\Gamma_v)_{v \in V}$ such that each Γ_v can be described by two linear inequalities whose coefficients can be computed in linear time.

Proof. The proof follows the proof of Kokhas et al. [83] for the single-guessing game. First, suppose that $\sum_{v \in V} g_v/h_v < 1$ and fix some strategy of bears. A bear on v guesses correctly the color of his hat in exactly (g_v/h_v) -fraction of all possible hat arrangements. Thus, if the sum is smaller than one, there is a hat arrangement where no bear guesses the color of his hat correctly.

Now suppose the opposite inequality holds, i.e., $\sum_{v \in V} g_v/h_v \geq 1$. Let $V(K_n) = \{v_1, \dots, v_n\}$. For simplicity, we denote $h_i = h_{v_i}$ and $g_i = g_{v_i}$. Let $\ell = \text{LCM}(h_1, \dots, h_n)$ and $d_i = \ell/h_i$ (note that $d_i \in \mathbb{N}$). Let the bear on v_i get a hat of color $c_i \in [h_i]$ and

$$s = \sum_{1 \leq i \leq n} c_i \cdot d_i \pmod{\ell}.$$

The bears cover the set $[\ell]$ by disjoint intervals Q_i of length $d_i \cdot g_i$. A bear on v_i makes his guesses according to a hypothesis that s is in an interval Q_i and we will show that he guesses correctly if $s \in Q_i$. More formally, for $b_i = \sum_{j < i} d_j \cdot g_j$ we define the interval Q_i as $\{b_i, \dots, b_i + d_i \cdot g_i - 1\}$. Note that the union of intervals Q_1, \dots, Q_{i-1} is exactly the set $[b_i]$. A bear on v_i computes $s_i = \sum_{v \neq v_i} c_v \cdot d_v$. Then, he guesses all such colors a_i such that $s_i + a_i \cdot d_i \pmod{\ell}$ is in Q_i . Since Q_i contains $d_i \cdot g_i$ consecutive natural numbers and ℓ is divisible by d_i , he makes at most g_i guesses. If s is in Q_i then the bear on v_i guesses the color of his hat correctly, because $s = s_i + c_i \cdot d_i \pmod{\ell}$ and thus the bear on v_i includes the color c_i in his guesses.

Note that the union Q of all intervals Q_i is exactly the set

$$\left\{ 0, \dots, \sum_{1 \leq i \leq n} \frac{\ell \cdot g_i}{h_i} - 1 \right\}.$$

By assumption, we have that $\{0, \dots, \ell - 1\} \subseteq Q$. Since $0 \leq s < \ell$ by definition, it follows that s has to be in some interval Q_i .

For the ‘‘moreover’’ part, the bear on a vertex v_i guesses all colors $a_i \in [h_i]$ such that

$$b_i \leq (s_i + a_i \cdot d_i) \pmod{\ell} < b_i + d_i \cdot g_i.$$

Observe that s_i is a linear function of hat colors of bears sitting on the vertices different from v and the coefficients b_i and d_j can be computed in linear time. \square

By Theorem 3.4.1, we can conclude the following corollary.

Corollary 3.4.2. For each $n \in \mathbb{N}$, it holds that $\hat{\mu}(K_n) = n$.

Kokhas et al. [82] provided another proof of analogue of Theorem 3.4.1 for the single-guessing game, which can be generalized with similar ideas. However, the second proof does not imply a polynomial time algorithm for computing the strategy on cliques.

3.4.1 Graph Products

Further, we generalize a result of Kokhas and Latyshev [82]. In particular, we provide a new way to combine two hat guessing games on graphs G_1 and G_2 into a hat guessing game on graph obtained by gluing G_1 and G_2 together in a specific way.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs, let $S \subseteq V_1$ be a set of vertices inducing a clique in G_1 , and let $v \in V_2$ be an arbitrary vertex of G_2 . The *clique join of graphs G_1 and G_2 with respect to S and v* is the graph $G = (V, E)$ such that $V = V_1 \cup V_2 \setminus \{v\}$; and E contains all the edges of E_1 , all the edges of E_2 that do not contain v , and an edge between every $w \in S$ and every neighbor of v in G_2 . See Figure 3.1 for a sketch of a clique join.

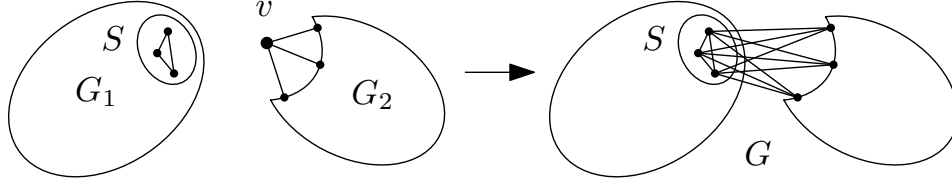


Figure 3.1: The clique join of graphs G_1 and G_2 with respect to S and v .

Lemma 3.4.3. Let $\mathcal{H}_1 = (G_1 = (V_1, E_1), \mathbf{h}^1, \mathbf{g}^1)$ and $\mathcal{H}_2 = (G_2 = (V_2, E_2), \mathbf{h}^2, \mathbf{g}^2)$ be two hat guessing games and let $S \subseteq V_1$ be a set inducing a clique in G_1 and $v \in V_2$. Set G to be the clique join of graphs G_1 and G_2 with respect to S and v . If the bears win the games \mathcal{H}_1 and \mathcal{H}_2 , then they also win the game $\mathcal{H} = (G, \mathbf{h}, \mathbf{g})$ where

$$h_u = \begin{cases} h_u^1 & u \in V_1 \setminus S \\ h_u^2 & u \in V_2 \setminus \{v\} \\ h_u^1 \cdot h_v^2 & u \in S, \text{ and} \end{cases} \quad g_u = \begin{cases} g_u^1 & u \in V_1 \setminus S \\ g_u^2 & u \in V_2 \setminus \{v\} \\ g_u^1 \cdot g_v^2 & u \in S. \end{cases}$$

Proof. Using winning strategies $(\Gamma_v^1)_{v \in V_1}$ and $(\Gamma_v^2)_{v \in V_2}$ for \mathcal{H}_1 and \mathcal{H}_2 respectively, let us construct a winning strategy for \mathcal{H} . For every bear $u \in S$, we interpret his color as a tuple (c_u^1, c_u^2) where $c_u^1 \in [h_u^1]$ and $c_u^2 \in [h_v^2]$. Moreover, we define an imaginary hat color of the bear on vertex v as $s = (\sum_{u \in S} c_u^2) \bmod h_v^2$.

Every bear on $w \in V_1 \setminus S$ plays according to the strategy Γ_w^1 using only the color c_u^1 for his every neighbor $u \in S$. Every bear on $w \in V_2 \setminus \{v\}$ plays according to the strategy Γ_w^2 using the imaginary hat color s of v . And finally, every bear on vertex $w \in S$ computes a set of guesses A_w by playing the strategy Γ_w^1 and a set of guesses B by playing the strategy Γ_v^2 . Since the bear on w can see every other vertex of S , he computes the set

$$B_w = \left\{ \left(c - \sum_{u \in S \setminus \{w\}} c_u^2 \right) \bmod h_v^2 \mid c \in B \right\}.$$

Finally, the bear on w guesses the set $A_w \times B_w$.

Fix an arbitrary hat arrangement. In the simulated hat guessing game \mathcal{H}_1 , there is a vertex u_1 such that the bear on u_1 guessed correctly. If $u_1 \notin S$ then it also guessed correctly

in \mathcal{H} . Likewise, there is a bear on a vertex u_2 in the simulated hat guessing game \mathcal{H}_2 that guessed correctly and we are done if $u_2 \neq v$. The remaining case is when $u_1 \in S$ and $u_2 = v$. Thus, the bear on v includes the color s in his guesses in the game \mathcal{H}_2 . It follows that for each $w \in S$ holds that if (c_w^1, c_w^2) is a hat color of the bear on w , then $c_w^2 \in B_w$. Since $u_1 \in S$, the bear on u_1 includes his hat color $(c_{u_1}^1, c_{u_2}^2)$ in his guesses $A_{u_1} \times B_{u_1}$. \square

We remark that Lemma 3.4.3 generalizes Theorem 3.1 and Theorem 3.5 of [82] not only by introducing multiple guesses but also by allowing for more general ways to glue two graphs together. Thus, it provides new constructions of winning games even for single-guessing games.

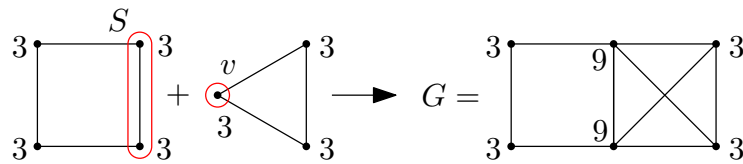


Figure 3.2: Applying Lemma 3.4.3 on winning hat guessing games $(C_4, 3, 1)$ (see [104]) and $(K_3, 3, 1)$, we obtain a winning hat guessing game $(G, \mathbf{h}, 1)$ where G is the result of identifying an edge in C_4 and K_4 , and \mathbf{h} is given in the picture.

3.5 Independence Polynomial

The multivariate *independence polynomial* of a graph $G = (V, E)$ on variables $\mathbf{x} = (x_v)_{v \in V}$ is

$$P_G(\mathbf{x}) = \sum_{\substack{I \subseteq V \\ I \text{ independent set}}} \prod_{v \in I} x_v.$$

First, we describe informally the connection between the multi-guessing game and the independence polynomial. Consider the game (G, h, g) and fix a strategy of bears. Suppose that the demon put on the head of each bear a hat of random color (chosen uniformly and independently). Let A_v be an event that the bear on the vertex v guesses correctly. Then, the probability of A_v is exactly g/h . Moreover, for any independent set I holds that A_v is independent on all events A_w for $w \in I, w \neq v$. Thus, we can use the inclusion-exclusion principle (Proposition 3.2.1) to compute the probability that A_v occurs for at least one $v \in I$, i.e., at least one bear sitting on some vertex of I guesses correctly.

Assume that no two bears on adjacent vertices guess correctly their hat colors at once; it turns out that if we plug $-g/h$ into all variables of the non-constant terms of $-P_G$, then we get exactly the fraction of all hat arrangements on which the bears win. The non-constant terms of P_G correspond (up to sign) to the terms of the formula from the inclusion-exclusion principle. Because of that, we have to plug $-g/h$ into the polynomial P_G .

To avoid confusion with the negative fraction $-g/h$, we define *signed independence polynomial* as $Z_G(\mathbf{x}) = P_G(-\mathbf{x})$, i.e.,

$$Z_G(\mathbf{x}) = \sum_{\substack{I \subseteq V \\ I \text{ independent set}}} (-1)^{|I|} \prod_{v \in I} x_v.$$

We also introduce the monivariate signed independence polynomial $U_G(x)$ obtained by plugging x for each variable x_v of Z_G .

Note that the constant term of any independence polynomial $P_G(\mathbf{x})$ equals to 1, arising from taking $I = \emptyset$ in the sum from the definition of P_G . When $U_G(g/h) = 0$ and no two adjacent bears guess correctly at the same time, then the bears win the game (G, h, g) because the fraction of all hat arrangements, on which at least one bear guesses correctly, is exactly 1, however, the proof is far from trivial.

Slightly abusing the notation, we use $Z_{G'}(\mathbf{x})$ to denote the independence polynomial of an induced subgraph G' with variables \mathbf{x} restricted to the vertices of G' . The independence polynomial P_G can be expanded according to a vertex $v \in V$ in the following way.

$$P_G(\mathbf{x}) = P_{G \setminus \{v\}}(\mathbf{x}) + x_v P_{G \setminus N[v]}(\mathbf{x})$$

The analogous expansions hold for the polynomials Z_G and U_G as well. This expansion follows from the fact that for any independent set I of G , it holds that either v is not in I (the first term of the expansion), or v is in I but in that case, no neighbor of v is in I (the second term). The formal proof of this expansion of P_G was provided by Hoede and Li [66].

For a graph G , we let $\mathcal{R}(G)$ denote the set of all vectors $\mathbf{r} \in [0, \infty)^V$ such that $Z_G(\mathbf{w}) > 0$ for all $0 \leq \mathbf{w} \leq \mathbf{r}$, where the comparison is done entry-wise. For the monivariate independence polynomial U_G , an analogous set to $\mathcal{R}(G)$ would be exactly the real interval $[0, r)$ where r is the smallest positive root of U_G . (Note that $Z_G(\mathbf{0}) = 1$ and $U_G(0) = 1$.)

Our first connection of the independence polynomial to the hat guessing game comes in the shape of a sufficient condition for bears to lose. Consider the following beautiful connection between Lovász Local Lemma and independence polynomial due to Scott and Sokal [99].

Theorem 3.5.1 ([99] Theorem 4.1). Let $G = (V, E)$ be a graph and let $(A_v)_{v \in V}$ be a family of events on some probability space such that for every v , the event A_v is independent of $\{A_w \mid w \notin N[v]\}$. Suppose that $\mathbf{p} \in [0, 1]^V$ is a vector of real numbers such that for each v we have $P(A_v) \leq p_v$ and $\mathbf{p} \in \mathcal{R}(G)$. Then

$$P\left(\bigcap_{v \in V} \bar{A}_v\right) \geq Z_G(\mathbf{p}) > 0.$$

Proposition 3.5.2. A hat guessing game $\mathcal{H} = (G = (V, E), \mathbf{h}, \mathbf{g})$ is losing whenever $\mathbf{r} \in \mathcal{R}(G)$ where $\mathbf{r} = (g_v/h_v)_{v \in V}$.

Proof. Suppose for a contradiction that \mathcal{H} is winning and fix a strategy of the bears. We let the demon assign hat to each bear uniformly at random and independently from the other bears. Let A_v be the event that the bear on the vertex v guesses correctly. Observe, that $P(A_v) = \frac{g_v}{h_v}$ and the probability that the bears lose is precisely $P(\bigcap_{v \in V} \bar{A}_v)$.

Let us show that the event A_v is independent of all events A_w such that $w \notin N[v]$. Observe, that fixing arbitrary hat arrangement φ on $V \setminus \{v\}$ uniquely determines the guesses of bears on all vertices except for $N(v)$. In particular, for every vertex $w \notin N[v]$, we know whether the bear on w guessed correctly and thus the probability of A_w conditioned by φ is either 0 or 1. On the other hand, the probability of A_v conditioned by φ is still $\frac{g_v}{h_v}$. Therefore, A_v is independent of any subset of $\{A_w \mid w \notin N[v]\}$.

The claim follows since the graph G and vector \mathbf{r} satisfies the conditions of Theorem 3.5.1 and we obtain that $P(\bigcap_{v \in V} \bar{A}_v) \geq Z_G(\mathbf{r}) > 0$. Therefore, there exists some hat arrangement in which all bears guess incorrectly. \square

A strategy for a hat guessing game \mathcal{H} is *perfect* if it is winning and in every hat arrangement, no two bears that guess correctly are on adjacent vertices. We remark that perfect strategies exist, for example the strategy for a single-guessing game on a clique K_n and exactly n colors [82], or for a multi-guessing game on a clique K_n and $h/g = n$ (Corollary 3.4.2). The following proposition shows that a perfect strategy can occur only when $\mathbf{r} = (g_v/h_v)_{v \in V}$ lies in some sense just outside of $\mathcal{R}(G)$.

Proposition 3.5.3. If there is a perfect strategy for the hat guessing game $(G, \mathbf{h}, \mathbf{g})$ then for $\mathbf{r} = (g_v/h_v)_{v \in V}$ we have that $Z_G(\mathbf{r}) = 0$ and $Z_G(\mathbf{w}) \geq 0$ for every $0 \leq \mathbf{w} \leq \mathbf{r}$.

Proof. Fix a perfect strategy and set $m = \prod_{v \in V} h_v$ to be the total number of possible hat arrangements. For any subset $S \subseteq V$, let n_S be the number of hat arrangements such that every bear on vertex $v \in S$ guesses correctly (other bears are not forbidden from guessing correctly). We claim that for any independent set $I \subseteq V$, we have $n_I = m \cdot \prod_{v \in I} \frac{g_v}{h_v}$.

Observe that by assigning the hats to the bears on $V \setminus I$, we fix the guesses of all bears on I . Every bear on a vertex $v \in I$ guesses correctly exactly g_v out of h_v of his hat assignments. Thus the total number of hat arrangements where every bear on the independent set I guesses correctly is exactly

$$n_I = \prod_{v \in V \setminus I} h_v \cdot \prod_{v \in I} g_v = m \cdot \prod_{v \in I} \frac{g_v}{h_v}.$$

On the other hand, the perfect strategy guarantees that for any non-empty S that is not an independent set, $n_S = 0$. This allows us to use the inclusion-exclusion principle and count the exact total amount of hat arrangements such that at least one bear guesses correctly

$$\sum_{\emptyset \neq S \subseteq V} (-1)^{|S|+1} n_S = \sum_{\substack{\emptyset \neq I \subseteq V \\ I \text{ independent}}} (-1)^{|I|+1} n_I = m \cdot \sum_{\substack{\emptyset \neq I \subseteq V \\ I \text{ independent}}} (-1)^{|I|+1} \prod_{v \in I} \frac{g_v}{h_v} = m \cdot (1 - Z_G(\mathbf{r})).$$

Finally, the total amount of hat arrangements when at least one bear guesses correctly must be exactly m since the bears win. Therefore, we get $Z_G(\mathbf{r}) = 0$.

We prove the remaining claim in two steps. First, we show that for every induced subgraph G' of G it holds that $Z_{G'}(\mathbf{r}) \geq 0$. To that end, consider a modified hat guessing game where only bears on the vertices of G' are allowed to guess and they play according to the original perfect strategy. By the same argument as before, we can count the total amount of hat arrangements that are guessed correctly by this modified strategy as $m \cdot (1 - Z_{G'}(\mathbf{r}))$. It implies $Z_{G'}(\mathbf{r}) \geq 0$ as the total number of hat arrangements is m .

Now consider any $0 \leq \mathbf{w} \leq \mathbf{r}$. Let v_1, \dots, v_n be an arbitrary ordering of the vertices of G and let us define vectors \mathbf{w}^i for $0 \leq i \leq n$ as

$$w_u^i = \begin{cases} w_u & \text{if } u = v_j \text{ for } j \leq i, \\ r_u & \text{if } u = v_j \text{ for } j > i. \end{cases}$$

Notice that $\mathbf{w}^0 = \mathbf{r}$, $\mathbf{w}^n = \mathbf{w}$, and the vectors \mathbf{w}^i correspond to switching the coordinates of \mathbf{r} into the coordinates of \mathbf{w} one by one. We prove by induction on i that for every induced subgraph G' of G it holds that $Z_{G'}(\mathbf{w}^i) \geq 0$.

We already proved the fact for $i = 0$. Let $i \geq 1$ and let G' be an arbitrary induced subgraph of G . If G' does not contain v_i then $Z_{G'}(\mathbf{w}^i) = Z_{G'}(\mathbf{w}^{i-1}) \geq 0$ and we are done. Otherwise, we have

$$\begin{aligned} Z_{G'}(\mathbf{w}^i) &= Z_{G' \setminus \{v_i\}}(\mathbf{w}^i) - w_{v_i} Z_{G' \setminus N[v_i]}(\mathbf{w}^i) \\ &\geq Z_{G' \setminus \{v_i\}}(\mathbf{w}^{i-1}) - r_{v_i} Z_{G' \setminus N[v_i]}(\mathbf{w}^{i-1}) = Z_{G'}(\mathbf{w}^{i-1}) \geq 0 \end{aligned}$$

where we first partition the independent sets of G' according to their incidence with v_i and then replace \mathbf{w}^i with \mathbf{w}^{i-1} where the inequality holds since $w_{v_i} \leq r_{v_i}$ and $Z_{G' \setminus N[v_i]}(\mathbf{w}^{i-1}) \geq 0$ from induction. Finally, we notice that we obtained the independent polynomial $Z_{G'}$ evaluated in \mathbf{w}^{i-1} and apply induction. Thus, $Z_G(\mathbf{w}) \geq 0$ as $\mathbf{w} = \mathbf{w}^n$ and G is an induced subgraph of itself. \square

Scott and Sokal [99, Corollary 2.20] proved that $Z_G(\mathbf{w}) \geq 0$ for every $0 \leq \mathbf{w} \leq \mathbf{r}$ if and only if \mathbf{r} lies in the closure of $\mathcal{R}(G)$. Therefore, Proposition 3.5.3 further implies that if a perfect strategy for game $(G, \mathbf{h}, \mathbf{g})$ exists, then $\mathbf{r} = (g_v/h_v)_{v \in V}$ lies in the closure of $\mathcal{R}(G)$. And since \mathbf{r} cannot lie inside $\mathcal{R}(G)$ due to Proposition 3.5.2, it must belong to the boundary of the set $\mathcal{R}(G)$.

The natural question is what happens outside of the closure of $\mathcal{R}(G)$. We proceed to answer this question for chordal graphs.

A graph G is *chordal* if every cycle of length at least 4 has a chord. For our purposes, it is more convenient to work with a different equivalent definition of chordal graphs. For a graph $G = (V, E)$, a *clique tree of G* is a tree T whose vertex set is precisely the subsets of V that induce maximal cliques in G and for each $v \in V$ the vertices of T containing v induces a connected subtree. Gavril [57] showed that G is chordal if and only if there exists a clique tree of G .

Theorem 3.5.4. Let $G = (V, E)$ be a chordal graph and let $\mathbf{r} = (r_v)_{v \in V}$ be a vector of rational numbers from the interval $[0, 1]$. If $\mathbf{r} \notin \mathcal{R}(G)$ then there are vectors $\mathbf{g}, \mathbf{h} \in \mathbb{N}^V$ such that $g_v/h_v \leq r_v$ for every $v \in V$ and the hat guessing game $(G, \mathbf{h}, \mathbf{g})$ is winning.

Proof. We prove the theorem by induction on the size of the clique tree of G . Let $0 \leq \mathbf{w} \leq \mathbf{r}$ be a witness that $\mathbf{r} \notin \mathcal{R}(G)$, i.e., $Z_G(\mathbf{w}) \leq 0$.

If G is itself a complete graph, then $Z_G(\mathbf{w}) \leq 0$ implies that $\sum_{v \in V} w_v \geq 1$ and $\sum_{v \in V} r_v \geq \sum_{v \in V} w_v \geq 1$. Thus, if we take the minimal vectors $\mathbf{g}, \mathbf{h} \in \mathbb{N}^V$ such that $g_v/h_v = r_v$ for each v , the assumptions of Theorem 3.4.1 are satisfied and the hat guessing game $(G, \mathbf{h}, \mathbf{g})$ is winning.

Otherwise, the clique tree of G contains at least 2 vertices and we pick its arbitrary leaf C . Let R be the set of vertices such that they belong only to the clique C and S the set of vertices $C \setminus R$. We aim to split the graph into $G' = G[V \setminus R]$ and $G[C]$, apply induction to obtain winning strategies on these graphs, and then combine them into a winning strategy on G .

If $\sum_{v \in C} r_v \geq 1$, then the game is winning already on the clique $G[C]$ due to Theorem 3.4.1. Therefore, we can assume $\sum_{v \in C} r_v < 1$ which implies $\sum_{v \in C} w_v < 1$. We define vectors $\mathbf{w}' = (w'_v)_{v \in V \setminus R}$ and $\mathbf{r}' = (r'_v)_{v \in V \setminus R}$ as

$$w'_v = \begin{cases} w_v/\alpha_w & \text{if } v \in S, \\ w_v & \text{otherwise, and} \end{cases} \quad r'_v = \begin{cases} r_v/\alpha_r & \text{if } v \in S, \\ r_v & \text{otherwise,} \end{cases}$$

where $\alpha_r = 1 - \sum_{v \in R} r_v$ and $\alpha_w = 1 - \sum_{v \in R} w_v$. Observe that $0 < \alpha_r \leq \alpha_w$ and that for every $v \in V \setminus R$ we have $0 \leq w'_v \leq r'_v \leq 1$. In other words, \mathbf{r}' and \mathbf{w}' are both vectors of numbers from $[0, 1]$ such that $\mathbf{w}' \leq \mathbf{r}'$.

To simplify the rest of the proof, we introduce the following notation. For any $u \in V$, let $Z_G(\mathbf{x}; u)$ denote the independence polynomial restricted only to the independent sets containing u , i.e.,

$$Z_G(\mathbf{x}; u) = \sum_{\substack{u \in I \subseteq V \\ I \text{ independent}}} (-1)^{|I|} \prod_{v \in I} x_v.$$

With this in hand, we proceed to show that $Z_{G'}(\mathbf{w}') = Z_G(\mathbf{w})/\alpha_w$.

$$Z_G(\mathbf{w}) = \sum_{v \in R} Z_G(\mathbf{w}; v) + \sum_{v \in S} Z_G(\mathbf{w}; v) + Z_{G \setminus C}(\mathbf{w}) \quad (3.1)$$

$$= \left(1 - \sum_{v \in R} w_v\right) \cdot Z_{G \setminus C}(\mathbf{w}) + \sum_{v \in S} Z_{G \setminus R}(\mathbf{w}; v) \quad (3.2)$$

$$= \alpha_w \cdot Z_{G \setminus C}(\mathbf{w}') + \alpha_w \cdot \sum_{v \in S} Z_{G \setminus R}(\mathbf{w}'; v) \quad (3.3)$$

$$= \alpha_w \cdot Z_{G \setminus R}(\mathbf{w}') = \alpha_w \cdot Z_{G'}(\mathbf{w}') \quad (3.4)$$

In (3.1), we partition the independent sets in G depending on their incidence with C . The line (3.2) follows since every independent set intersecting R in G can be written as a union of $v \in R$ and an independent set in $G \setminus C$ which allows us to collect the first and third terms. At the same time, all independent sets intersecting S in G can be regarded as independent sets intersecting S in $G \setminus R$. In (3.3), we replace \mathbf{w} with \mathbf{w}' which scales each term in the second sum by the factor $w_v/w'_v = \alpha_w$. Finally, notice that the terms in (3.3) describe (up to scaling by α_w) the independent sets in $G \setminus R$ partitioned according to their incidence with S . We collect them in (3.4).

Since $\alpha_w > 0$ and $Z_G(\mathbf{w}) \leq 0$, we have $Z_{G'}(\mathbf{w}') \leq 0$ which witnesses that $\mathbf{r}' \notin \mathcal{R}(G')$. Therefore, we can apply induction on G' and \mathbf{r}' to obtain functions \mathbf{h}', \mathbf{g}' such that the hat guessing game $(G', \mathbf{h}', \mathbf{g}')$ is winning and $g'_v/h'_v \leq r'_v$ for each vertex v .

Let G'' be the graph obtained from the clique $G[C]$ by contracting S to a single vertex u and define the vector $\mathbf{r}'' = (r''_v)_{v \in R \cup \{u\}}$ as

$$r''_v = \begin{cases} r_v & \text{if } v \in R, \\ \alpha_r & \text{if } v = u. \end{cases}$$

Observe that G is precisely the clique join of G' and G'' with respect to S and w . Since $r''_u + \sum_{v \in R} r''_v = 1$, we can take the minimal vectors $\mathbf{h}'', \mathbf{g}'' \in \mathbb{N}^V$ such that $g''_v/h''_v = r_v$ for every v and apply Theorem 3.4.1 on G'' to show that the hat guessing game $(G'', \mathbf{h}'', \mathbf{g}'')$ is winning. Finally, we construct the desired winning strategy by combining the two graphs and their respective strategies using Lemma 3.4.3 since $r'_v \cdot r''_v = r_v$ for every $v \in S$. \square

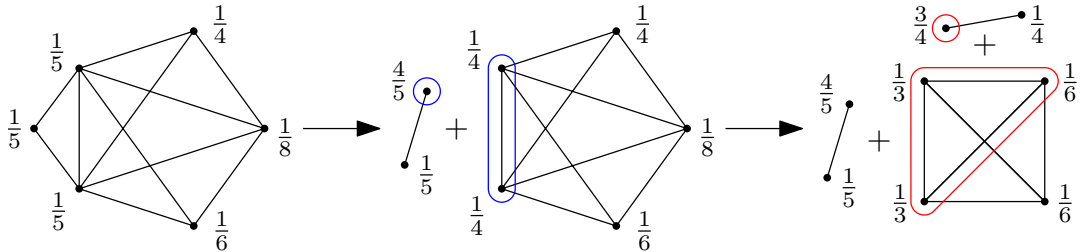


Figure 3.3: Application of Theorem 3.4.1 on a chordal graph G with vector $\mathbf{r} \in \mathcal{R}(G)$. In each step, we highlight the clique S and vertex w that are used for Lemma 3.4.3 to inductively build a strategy for G from strategies on cliques given by Theorem 3.4.1.

Theorem 3.5.4 applied for the uniform polynomial U_G immediately gives us the following corollary.

Corollary 3.5.5. For any chordal graph G , the fractional hat chromatic number $\hat{\mu}(G)$ is equal to $1/r$ where r is the smallest positive root of $U_G(x)$.

Proof. Theorem 3.5.4 implies that $\hat{\mu}(G) \geq 1/r$. For the other direction, let $(w_i)_{i \in \mathbb{N}}$ be a sequence of rational numbers such that $w_i < r$ for every i and $\lim_{i \rightarrow \infty} w_i = r$. Set $\mathbf{w}_i = (w_i)_{v \in V}$. Scott and Sokal [99, Theorem 2.10] prove that $\mathbf{r} \in \mathcal{R}(G)$ if and only if there

is a path in $[0, \infty)^V$ connecting $\mathbf{0}$ and \mathbf{r} such that $Z_G(\mathbf{p}) > 0$ for any \mathbf{p} on the path. Taking the path $\{\lambda \mathbf{w}_i \mid \lambda \in [0, 1]\}$, we see that $Z_G(\lambda \mathbf{w}_i) = U_G(\lambda \cdot w_i) > 0$ and thus $\mathbf{w}_i \in \mathcal{R}(G)$ for every i . Therefore by Proposition 3.5.2, the hat guessing game (G, h, g) is losing for any h, g such that $g/h = w_i$ and $\hat{\mu}(G) \leq 1/w_i$ for every i . It follows that $\hat{\mu}(G) \leq 1/r$. \square

We would like to remark that the proof of Theorem 3.5.4 (and also Theorem 3.4.1) is constructive in the sense that given a graph G and a vector \mathbf{r} it either greedily finds vectors $\mathbf{g}, \mathbf{h} \in \mathbb{N}^V$ such that $g_v/h_v \leq r_v$ together with a succinct representation of a winning strategy on $(G, \mathbf{h}, \mathbf{g})$ or it reaches a contradiction if $\mathbf{r} \in \mathcal{R}(G)$. Moreover, it is easy to see that it can be implemented to run in polynomial time if the clique tree of G is provided. Combining it with the well-known fact that a clique tree of a chordal graph can be obtained in polynomial time (see Blair and Peyton [12]) we get the following corollary.

Corollary 3.5.6. There is a polynomial-time algorithm that for a chordal graph $G = (V, E)$ and vector \mathbf{r} decides whether $\mathbf{r} \in \mathcal{R}(G)$. Moreover, if $\mathbf{r} \notin \mathcal{R}(G)$ it outputs vectors $\mathbf{h}, \mathbf{g} \in \mathbb{N}^V$ such that $g_v/h_v \leq r_v$ for every $v \in V$, together with a polynomial-size representation of a winning strategy for the hat guessing game $(G, \mathbf{h}, \mathbf{g})$.

This result is consistent with the fact that chordal graphs are in general well-behaved with respect to Lovász Local Lemma – Pegden [92] showed that for a chordal graph G , we can decide in polynomial time whether a given vector \mathbf{r} belongs to $\mathcal{R}(G)$. We finish this section by presenting an algorithm that computes hat chromatic number of chordal graphs.

Theorem 3.5.7. There is an algorithm \mathcal{A} such that given a chordal graph G as an input, it approximates $\hat{\mu}(G)$ up to an additive error $1/2^k$. The running time of \mathcal{A} is $2k \cdot \text{poly}(n)$, where n is the number of vertices of G . Moreover, if $\hat{\mu}(G)$ is rational, then the algorithm \mathcal{A} outputs the exact value of $\hat{\mu}(G)$.

Proof. First, suppose that $\hat{\mu}(G)$ is rational. Let $\hat{\mu}(G) = q/p$ for coprimes $p, q \in \mathbb{N}$. By Corollary 3.5.5, $1/\hat{\mu}(G) = p/q$ is the smallest positive root of the polynomial U_G . Let $U_G(x) = a_d x^d + \dots + a_1 x + a_0$. Note that $a_0 = 1$ and for each $i \leq d$ holds that $|a_i| \leq 2^n$ because $|a_i|$ is exactly the number of independent sets of size i in the graph G . By the rational root theorem (Theorem 3.2.2), it holds that $p = 1$ and $q \leq 2^n$.

The algorithm \mathcal{A} repeats a halving procedure which works as follows. We set the initial bounds $\ell_0 = 0$ and $u_0 = 1$. In a step i , let $r_i = (\ell_i + u_i)/2$. We run the algorithm by Corollary 3.5.6 to test if there are $h_i, g_i \in \mathbb{N}$ such that $g_i/h_i \leq r_i$ and the game $\mathcal{H}_i = (G, h_i, g_i)$ is winning. If so, we set new bounds $\ell_{i+1} = \ell_i$ and $u_{i+1} = r_i$. On the other hand, if \mathcal{H}_i is not winning then we set $\ell_{i+1} = r_i$ and $u_{i+1} = u_i$. Thus, for each i it holds that $\ell_i \leq 1/\hat{\mu}(G) \leq u_i$.

We make $s = \max\{2k, 3n\}$ steps. The length of the real interval $I_s = [\ell_s, u_s]$ is at most $1/2^{3n}$. It is easy to verify that the interval I_s contains at most one rational number $1/q$ for $q \leq 2^n$. If so, we output the number q . Otherwise, we output a number t such that $1/t$ is an arbitrary number in I_s .

If $\hat{\mu}(G)$ is rational, then by Corollary 3.5.5 and by the discussion above we found its value. Otherwise, we know that $|1/\hat{\mu}(G) - 1/t| \leq 1/2^s$ as the length of I_s is exactly $1/2^s$. Since $s \geq 3n$ and $1/t \geq 1/n$ by Corollary 3.4.2, it follows by easy calculation that $|\hat{\mu}(G) - t| \leq 1/2^{s/2} \leq 1/2^k$. Thus, even if $\hat{\mu}(G)$ is irrational, then we estimate it with precision $1/2^k$.

We ran the halving procedure at most $2k$ -times and during each step we run the poly-time algorithm given by Corollary 3.5.6. Thus, the running time of \mathcal{A} is at most $2k \cdot \text{poly}(n)$. \square

3.6 Applications

In this section, we present applications of the relation between the hat guessing game and independence polynomials which was presented in the previous section.

3.6.1 Fractional Hat Chromatic Number is Almost Linear in the Maximum Degree

First, we prove that $\hat{\mu}(G)$ is asymptotically equal to $\Delta(G)$ up to a logarithmic factor.

Proposition 3.6.1. The fractional hat chromatic number of any graph $G = (V, E)$ is at least $\Omega(\Delta/\log \Delta)$.

Proof. Let H be a subgraph of G . Note that $\hat{\mu}(H) \leq \hat{\mu}(G)$ as the bears can use a winning strategy for H in G . Let S be a star of $\Delta(G) = \Delta$ leaves. The graph G contains S as a subgraph. We prove the proposition by giving a lower bound for $\hat{\mu}(S)$.

By Corollary 3.5.5, we have that $r = 1/\hat{\mu}(S)$ is the smallest positive root of $U_S(x)$. The independence polynomial of S is

$$U_S(x) = -x + \sum_{i=0}^{\Delta} \binom{\Delta}{i} (-x)^i = (1-x)^\Delta - x.$$

The term $-x$ is given by the independent set containing only the vertex of degree Δ . The sum is given by all independent sets consisting of leaves of S . Thus, it must hold that $(1-r)^\Delta = r$. By simple calculation, we conclude that $r = \Theta(\log \Delta/\Delta)$, which implies the assertion of the proposition. \square

Farnik [50] proved that $\mu_g(G) \in O(g \cdot \Delta(G))$, from which we can deduce that $\hat{\mu}(G) \in O(\Delta(G))$. It gives with Proposition 3.6.1 the following corollary that $\hat{\mu}(G)$ is almost linear in $\Delta(G)$.

Corollary 3.6.2. For any graph G , it holds that $\hat{\mu}(G) \in \Omega(\Delta/\log \Delta)$ and $\hat{\mu}(G) \in O(\Delta)$.

3.6.2 Paths and Cycles

In this section, we discuss the precise value of $\hat{\mu}$ of paths and cycles. It follows from Corollary 3.6.2, that $\hat{\mu}(P_n)$ and $\hat{\mu}(C_n)$ are some constants. We prove that the fractional hat chromatic number of paths and cycles goes to 4 with their increasing length.

For a proof, we need a version of Lovász local lemma proved by Shearer.

Lemma 3.6.3 (Shearer [102]). Let A_1, \dots, A_n be events such that each event is independent on all but at most d other events. Let the probability of any events A_i is at most p . If $d > 1$ and $p < \frac{(d-1)^{d-1}}{d^d}$, then there is non-zero probability that none of the events A_1, \dots, A_n occurs.

Proposition 3.6.4. $\lim_{n \rightarrow \infty} \hat{\mu}(P_n) = \lim_{n \rightarrow \infty} \hat{\mu}(C_n) = 4$

Proof. First, we prove the lower bound for paths. Let $\varepsilon > 0$. We construct a sufficiently long path $P = (V, E)$ and vectors $\mathbf{h}, \mathbf{g} \in \mathbb{N}^V$ such that a hat guessing game $(P, \mathbf{h}, \mathbf{g})$ is winning and $g_v/h_v \leq 1/4 + \varepsilon$. Thus, we can conclude that for every $\delta > 0$ there is n such that $\hat{\mu}(P_n) \geq 4 - \delta$, i.e., $\lim_{n \rightarrow \infty} \hat{\mu}(P_n) \geq 4$. The same lower bound holds for cycles as they contain paths as subgraphs.

We construct the path P iteratively. Let P^0 be a path consisting of one edge $e_0 = \{v_0, u_0\}$. We set $\mathbf{g}_{v_0}^0 = \mathbf{g}_{u_0}^0 = 1$ and $\mathbf{h}_{v_0}^0 = \mathbf{h}_{u_0}^0 = 2$. By Theorem 3.4.1, the game $(P^0, \mathbf{h}^0, \mathbf{g}^0)$ is winning.

Now, we want to construct a game $\mathcal{H}_{i+1} = (P^{i+1}, \mathbf{h}^{i+1}, \mathbf{g}^{i+1})$ from $(P^i, \mathbf{h}^i, \mathbf{g}^i)$. Let v_i and u_i be the endpoints of P^i . We will maintain the invariant that $g_{v_i}^i = g_{u_i}^i$ and $h_{v_i}^i = h_{u_i}^i$ and let us denote the ratio $g_{v_i}^i/h_{v_i}^i$ by r_i . We construct the paths P^i in a way such that $r_i = \frac{1}{2} - i \cdot \varepsilon$. Note that this equality holds for the game $(P^0, \mathbf{h}^0, \mathbf{g}^0)$.

Let P' be a path consisting of one edge $e' = \{w, w'\}$ and we set \mathbf{g}' and \mathbf{h}' in such a way that $g'_w/h'_w = 1/2 + (i+1) \cdot \varepsilon$ and $g'_{w'}/h'_{w'} = 1/2 - (i+1) \cdot \varepsilon$. Again by Theorem 3.4.1, the game $(P', \mathbf{h}', \mathbf{g}')$ is winning. To create the path P^{i+1} , we join two copies of P' to P^i using Lemma 3.4.3. More formally, we join one copy of P' by identifying w and u_i and the second copy by identifying w and v_i . Thus, the endpoints u_{i+1} and v_{i+1} of P^{i+1} are copies of w' . By Lemma 3.4.3, we get a winning game $\mathcal{H}_{i+1} = (P^{i+1}, \mathbf{h}^{i+1}, \mathbf{g}^{i+1})$. For a sketch of construction of the game \mathcal{H}_{i+1} , see Figure 3.4. Note that indeed $r_{i+1} = \frac{1}{2} - (i+1) \cdot \varepsilon$.

We end this process after $k = \lceil \frac{1}{4\varepsilon} \rceil$ steps. Thus, it holds that $r_k = \frac{1}{2} - k \cdot \varepsilon \leq \frac{1}{4}$. On the other hand, it holds for each $0 \leq i < k$ by Lemma 3.4.3 that

$$\frac{g_{v_i}^k}{h_{v_i}^k} = \frac{g_{u_i}^k}{h_{u_i}^k} = \left(\frac{1}{2} - i \cdot \varepsilon \right) \cdot \left(\frac{1}{2} + (i+1) \cdot \varepsilon \right) = \frac{1}{4} + \frac{\varepsilon}{2} - i(i+1)\varepsilon^2.$$

Thus, for each vertex v of P^k holds that $\frac{g_v^k}{h_v^k} \leq \frac{1}{4} + \varepsilon$ as claimed.

Now, we prove the upper bound. Let $\mathcal{H} = (G, h, g)$ be a game such that G is a path or a cycle and $\frac{h}{g} > 4$. We will prove that bears lose \mathcal{H} , which implies that $\lim_{n \rightarrow \infty} \hat{\mu}(P_n), \lim_{n \rightarrow \infty} \hat{\mu}(C_n) \leq 4$. Let us fix some strategy of bears and the demon gives each bear a hat of random color (chosen uniformly and independently). We denote A_v an

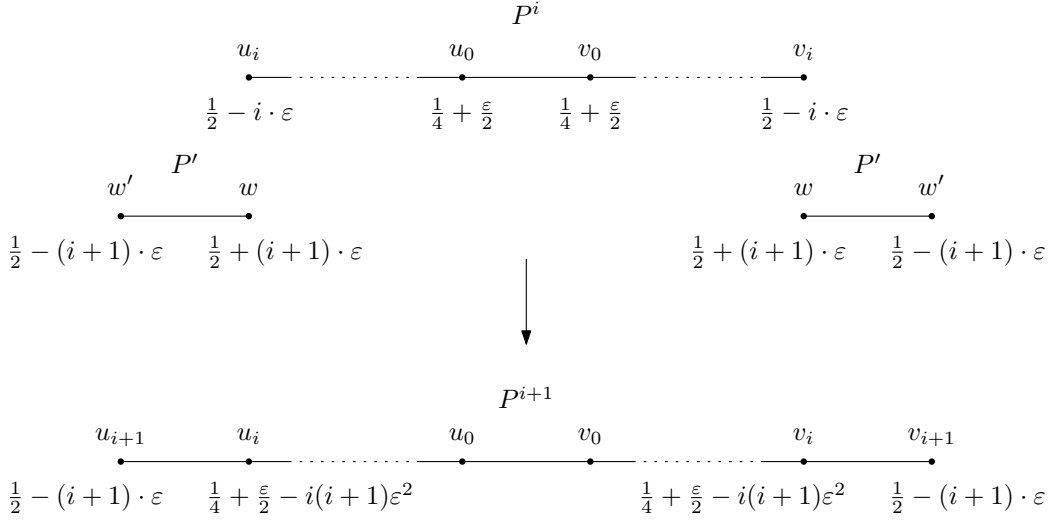


Figure 3.4: A sketch of construction of the game \mathcal{H}_{i+1} . The formulas below vertices are the fractions g_v/h_v .

event that the bear on v guesses correctly. Then, $\Pr[A_v] = \frac{g}{h} < \frac{1}{4}$. Since the maximum degree in G is 2, each event A_v might depend only on at most 2 other events. By Lemma 3.6.3, for events $(A_v)_{v \in V(G)}$ and $d = 2$, we have that no event A_v occurs with non-zero probability. Thus, there is a hat arrangement such that no bear guesses correctly. \square

We remark that Proposition 3.6.4 follows also from the results of Scott and Sokal [99] as they proved that the small positive roots of U_{P_n} and U_{C_n} go to $1/4$ when n goes to infinity. However, their proof is purely algebraic whereas we provide a combinatorial proof.

Further, we discuss the value of $\hat{\mu} = \hat{\mu}(P_3)$. By Corollary 3.5.5, we have that $1/\hat{\mu}$ is the smallest positive root of $U_{P_3}(x) = x^2 - 3x + 1$. Thus, $1/\hat{\mu} = (3 - \sqrt{5})/2$. By Theorem 3.5.4, it holds that for any $p, q \in \mathbb{N}$ such that $\hat{\mu} \leq p/q$ there are $g, h \in \mathbb{N}$ such that $p/q = h/g$ and the game (P_3, h, g) is winning. However, the strategy from the proof gives us $h = p \cdot (p - q)$ and $g = q \cdot (p - q)$. We present a sequence $(h_i/g_i)_{i \in \mathbb{N}}$ such that the sequence goes to $\hat{\mu}$, for each i the numbers h_i and g_i are coprime, and the game (P_3, h_i, g_i) is winning for each i . Thus, we present a strategy that is in some sense more efficient than the strategy given by the proof of Theorem 3.5.4 as the general strategy for P_3 does not produce numbers g and h which are coprimes.

First, we present the strategy for P_3 . Note that if $1 \geq g/h \geq 1/\hat{\mu}$ (for $g, h \in \mathbb{N}$) then $U_{P_3}(g/h) = (g/h)^2 - 3g/h + 1 < 0$. We change the inequality to $g^2 - 3gh + h^2 < 0$ and prove that for each g and h , which satisfy the previous inequality, there is a winning strategy for (P_3, h, g) .

Lemma 3.6.5. Let $g, h \in \mathbb{N}$ such that $g^2 - 3gh + h^2 < 0$. Then, the bears win the game (P_3, h, g) .

Proof. Let $V(P_3) = \{u, v, w\}$ where v and w are the endpoints of the path P_3 . We identify the colors with a set $C = \{0, \dots, h - 1\}$. Let the bear on v get a hat of color c_v . The

bear on u makes guesses $A_u = \{c_v, c_v - 1, \dots, c_v - (g - 1)\}$. The bear on w makes guesses $A_w = \{c_v, c_v - \lfloor h/g \rfloor, \dots, c_v - \lfloor (g - 1) \cdot h/g \rfloor\}$, where $\lfloor x \rfloor$ is the nearest integer to x (i.e., standard rounding). We compute the guessed colors modulo h .

The bear on v computes two sets of colors I_u and I_w based on the hat colors of bears on u and w such that he will not guess the colors from $I_u \cup I_w$ because if $c_v \in I_u \cup I_w$ then the bear on u or the bear on w would guess correctly (or maybe both of them). The guesses of the bear on u is an interval in the set C . However, the guesses of the bear on w are spread through C as evenly as possible. Thus, the intersection $I_u \cap I_w$ is small and $I_u \cup I_w$ is large.

More formally, let c_u and c_w be hat colors of the bears on u and w , respectively. Then, $I_u = \{c_u, c_u + 1, \dots, c_u + (g - 1)\}$, and $I_w = \{c_w, c_w + \lfloor h/g \rfloor, \dots, c_w + (g - 1) \cdot \lfloor h/g \rfloor\}$. Again, we compute the elements in the sets modulo h . Note that if $c_v \in I_u$ then the bear on u guesses correctly because in that case $c_v = c_u + t \pmod{h}$ for some $t < g$ and thus $c_u \in A_u$. An analogous property holds for c_w . Thus, the bear on v does not have to guess the colors from $I_u \cup I_w$.

We will prove that $|C \setminus (I_u \cup I_w)| \leq g$. Thus, the bear on v can guess all colors outside I_u and I_w and makes at most g guesses. First, we prove that $|I_u \cap I_w| \leq 3g - h$. Suppose opposite, that $|I_u \cap I_w| > 3g - h$. In such a case, there must be k such that both colors $c_u + \lfloor k \cdot h/g \rfloor$ and $c_w + \lfloor (k + 3g - h) \cdot h/g \rfloor$ belong to I_u . This implies that $\lfloor (k + 3g - h) \cdot h/g \rfloor - \lfloor k \cdot h/g \rfloor \leq g - 1$. Applying bounds on the rounded terms, we obtain

$$\begin{aligned} g - 1 &\geq \left\lfloor (k + 3g - h) \cdot \frac{h}{g} \right\rfloor - \left\lfloor k \cdot \frac{h}{g} \right\rfloor \\ &\geq (k + 3g - h) \cdot \frac{h}{g} - 0.5 - k \cdot \frac{h}{g} - 0.5 \\ &= (3g - h) \cdot \frac{h}{g} - 1. \end{aligned}$$

The final inequality implies $g^2 - 3gh + h^2 \geq 0$ which contradicts the assumption of the lemma. Therefore, the size of the intersection $I_u \cap I_w$ is at most $3g - h$. It follows that the size of the union $I_u \cup I_w$ is at least $2g - (3g - h) = h - g$ and $|C \setminus (I_u \cup I_w)| \leq g$. \square

Let F_i be the i -th Fibonacci number². We define $h_i = F_{2i}$ and $g_i = F_{2i-2}$. Now, we prove the sequence $(g_i/h_i)_{i \in \mathbb{N}}$ has the sought properties.

Lemma 3.6.6. For each $i \in \mathbb{N}$ it holds that $\frac{h_i}{g_i} \leq \hat{\mu}$. Moreover,

$$\lim_{i \rightarrow \infty} \frac{h_i}{g_i} = \hat{\mu}.$$

Proof. Note that $1/\hat{\mu} = 1 - \frac{\sqrt{5}-1}{2} = 1 - \frac{1}{\varphi}$, where φ is the golden ratio, i.e., $\varphi = \frac{1+\sqrt{5}}{2}$. It is well-known that fractions $\frac{F_i}{F_{i-1}}$ go to φ . Moreover, $\frac{F_{2i}}{F_{2i-1}} \geq \varphi$. Thus, for each $i \in \mathbb{N}$ it holds

² $F_0 = F_1 = 1$ and $F_{i+1} = F_{i-1} + F_i$.

that

$$\frac{1}{\hat{\mu}} = 1 - \frac{1}{\varphi} \leq 1 - \frac{F_{2i-1}}{F_{2i}} = \frac{F_{2i-2}}{F_{2i}} = \frac{g_i}{h_i}.$$

and the fractions $\frac{h_i}{g_i}$ indeed go to $\hat{\mu}$. □

Lemma 3.6.7. For each $i \in \mathbb{N}$ holds that

$$g_i^2 - 3g_i h_i + h_i^2 = -1. \quad (3.5)$$

Proof. By definition $h_i = F_{2i}$ and $g_i = F_{2i-2}$. We use the closed form expression for Fibonacci numbers

$$F_n = \frac{\varphi^{n+1}}{\sqrt{5}} - \frac{(1-\varphi)^{n+1}}{\sqrt{5}}.$$

We plug it into the left-hand side of Equation 3.5.

$$\begin{aligned} F_{2i-2}^2 - 3F_{2i-2}F_{2i} + F_{2i}^2 &= \\ \frac{1}{5} \cdot \left(\varphi^{2(2i-1)} - 2\varphi^{2i-1}(1-\varphi)^{2i-1} + (1-\varphi)^{2(2i-1)} \right. \\ &\quad - 3(\varphi^{2i-1} - (1-\varphi)^{2i-1})(\varphi^{2i+1} - (1-\varphi)^{2i+1}) \\ &\quad \left. + \varphi^{2(2i+1)} - 2\varphi^{2i+1}(1-\varphi)^{2i+1} + (1-\varphi)^{2(2i+1)} \right) \end{aligned}$$

We rearrange this expression by powers of φ and $(1-\varphi)$.

$$\begin{aligned} \frac{1}{5} \cdot \left(\varphi^{2(2i-1)}(1-3\varphi^2+\varphi^4) \right. \\ \quad + \varphi^{2i-1}(1-\varphi)^{2i-1}(-2+3(1-\varphi)^2+3\varphi^2-2\varphi^2(1-\varphi)^2) \\ \quad \left. + (1-\varphi)^{2(2i-1)}(1-3(1-\varphi)^2+(1-\varphi)^4) \right) \end{aligned}$$

By plugging $\varphi = \frac{1+\sqrt{5}}{2}$ we get the following equations.

$$\begin{aligned} 1 - 3\varphi^2 + \varphi^4 &= 0 \\ 1 - 3(1-\varphi)^2 + (1-\varphi)^4 &= 0 \\ -2 + 3(1-\varphi)^2 + 3\varphi^2 - 2\varphi^2(1-\varphi)^2 &= 5 \\ \varphi \cdot (1-\varphi) &= -1 \end{aligned}$$

From these equations and the expression above follows that the left-hand side of Equation 3.5 is equal to -1 . □

Observation 3.6.8. For each $i \in \mathbb{N}$ the numbers h_i and g_i are coprime.

Proof. By definition, $g_i = F_{2i-2}$ and $h_i = F_{2i}$.

$$\text{GCD}(F_{2i-2}, F_{2i}) = \text{GCD}(F_{2i-2}, F_{2i-1} + F_{2i-2}) = \text{GCD}(F_{2i-2}, F_{2i-1})$$

It is easy to prove by induction that for each $i \in \mathbb{N}$ it holds that $\text{GCD}(F_{i-1}, F_i) = 1$. □

Online Ramsey Numbers

An online Ramsey game is a game between Builder and Painter, alternating in turns. They are given a fixed graph H and an infinite set of independent vertices G . In each round Builder draws a new edge in G and the Painter colors it either red or blue. Builder wins if after some finite round there is a monochromatic copy of the graph H , otherwise, Painter wins. The online Ramsey number $\tilde{r}(H)$ is the minimum number of rounds such that Builder can force a monochromatic copy of H in G . This is an analogy to the size-Ramsey number $\bar{r}(H)$ defined as the minimum number such that there exists graph G with $\bar{r}(H)$ edges where for any edge two-coloring G contains a monochromatic copy of H .

In this chapter, we introduce the concept of induced online Ramsey numbers: the induced online Ramsey number $\tilde{r}_{\text{ind}}(H)$ is the minimum number of rounds Builder can force an induced monochromatic copy of H in G . We prove asymptotically tight bounds on the induced online Ramsey numbers of paths, cycles and two families of trees. Moreover, we provide a result analogous to Conlon [On-line Ramsey Numbers, *SIAM J. Discr. Math.* 2009], showing that there is an infinite family of trees $T_1, T_2, \dots, |T_i| < |T_{i+1}|$ for $i \geq 1$, such that

$$\lim_{i \rightarrow \infty} \frac{\tilde{r}(T_i)}{\bar{r}(T_i)} = 0.$$

4.1 Introduction

For a graph H , the Ramsey number $r(H)$ is the smallest integer n such that in any two-coloring of edges of the complete graph K_n , there is a monochromatic copy of H . The size-Ramsey number $\bar{r}(H)$, introduced by Erdős, Faudree, Rousseau, and Schelp [48], is the smallest integer m such that there exists a graph G with m edges such that for any two-coloring of the edges of G one will always find a monochromatic copy of H .

There are many interesting variants of the usual Ramsey function. One important concept is the *induced Ramsey number* $r_{\text{ind}}(H)$, which is the smallest integer n for which there is a graph G on n vertices such that every edge two-coloring of G contains an induced monochromatic copy of H . Erdős [49] conjectured the existence of a constant c such that

every graph H with n vertices satisfies $r_{\text{ind}}(H) \leq 2^{cn}$, which would be best possible. In 2012, Conlon, Fox and Sudakov [26] proved that there is a constant c such that every graph H with n vertices satisfies $r_{\text{ind}}(H) \leq 2^{cn \log n}$. The proof uses a construction of explicit pseudorandom graphs, as opposed to random graph construction techniques used by previous attempts. For more on the topic see the excellent review by Conlon, Fox, and Sudakov [27].

The induced size-Ramsey number $\bar{r}_{\text{ind}}(H)$ is an analog of the size-Ramsey number: we define $\bar{r}_{\text{ind}}(H)$ as the smallest integer m such that there exists a graph G with m edges such that for any two-coloring of the edges of G there is always a monochromatic copy of H . In 1983, Beck [6], using probabilistic methods, proved the surprising fact that $\tilde{r}(P_n) \leq cn$, where P_n is a path of length n and c is an absolute constant. An even more surprising result came by Haxell, Kohayakawa, and Łuczak [62], who studied the induced size-Ramsey number of cycles showing that $\bar{r}_{\text{ind}}(C_n) = \mathcal{O}(n)$. However, the proof uses random graph techniques and regularity lemma and does not provide any reasonably small multiplicative constant.

In this chapter, we study the online variant of size Ramsey number which was introduced independently by Beck [8] and Kurek and Ruciński [85]. The best way to define it is in term of a game between two players, Builder and Painter. An infinite set of vertices is given, in each round Builder draws a new edge and immediately it is colored by Painter in either red or blue. The goal of Builder is to force Painter to obtain a monochromatic copy of a fixed graph H (called *target graph*). The minimum number of edges which Builder must draw in order to obtain such monochromatic copy of H , assuming optimal strategy of Painter, is known as the online Ramsey number $\tilde{r}(H)$. The graph G , which is being built by Builder, is called *background graph*. The online Ramsey number is guaranteed to exist because Builder can simply create a big complete graph $K_{r(H)}$, which by Ramsey theorem trivially contains a monochromatic copy of H .

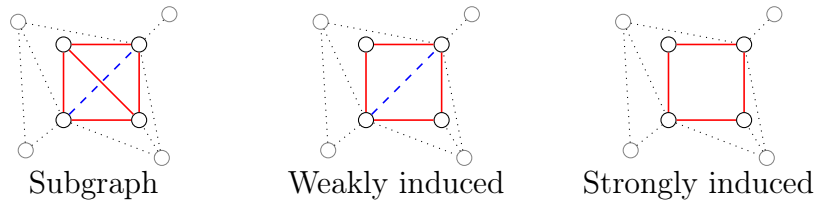
The winning condition for Builder is to obtain a copy of the target graph H .

ONLINE RAMSEY NUMBER

Input: A graph H

Question: $\tilde{r}(H)$ – the minimum number of rounds of the Builder-Painter game
Builder has a strategy to obtain a monochromatic subgraph H in G .

However, there are more different notions of “being a copy”. This leads us to the following definitions. See Figure 4.1 for an illustration.

Figure 4.1: Various notions of C_4 being a copy in G **(STRONGLY) INDUCED ONLINE RAMSEY NUMBER***Input:* A graph H *Question:* $\tilde{r}_{\text{ind}}(H)$ – the minimum number of rounds of the Builder-Painter game such that Builder has a strategy to obtain a monochromatic induced subgraph H in G .**WEAKLY INDUCED ONLINE RAMSEY NUMBER***Input:* A graph H *Question:* $\tilde{r}_{\text{wind}}(H)$ – the minimum number of rounds of the Builder-Painter game such that Builder has a strategy providing a color c such that G restricted to edges of color c contains an induced subgraph H .

If there is no strategy of Builder to obtain the copy of H , we define the respective number as ∞ .

Note that for any graph H we have $\tilde{r}(H) \leq \tilde{r}_{\text{wind}}(H) \leq \tilde{r}_{\text{ind}}(H)$. Also, these online Ramsey numbers lower bound respective size-Ramsey numbers.

In 2008 Grytczuk, Kierstead, and Prałat [60] studied the online Ramsey number of paths, obtaining $\tilde{r}(P_n) \leq 4n - 3$, where P_n is a path with n edges, providing an interesting counterpart to the result of Beck [6]. Also, the result by Haxell, Kohayakawa, and Łuczak. [62] on induced size-Ramsey number of cycles naturally bounds the online version as well, but with no reasonable multiplicative constant.

In this chapter, we study the induced online Ramsey number of paths, cycles, and trees. The summary of the results for paths and cycles is as follows.

Theorem 4.1.1. Let P_n denote the path of length n and let C_n denote a cycle with n vertices. Then

- $\tilde{r}_{\text{ind}}(P_n) \leq 28n - 27$,
- $\tilde{r}_{\text{ind}}(C_n) \leq 367n - 27$ for even n ,
- $\tilde{r}_{\text{ind}}(C_n) \leq 735n - 27$ for odd n .

A *spider* $\sigma_{k,\ell}$ is a union of k paths of length ℓ sharing exactly one common endpoint. We further show that $\tilde{r}_{\text{ind}}(\sigma_{k,\ell}) = \Theta(k^2\ell)$ and $\tilde{r}(\sigma_{k,\ell}) = \Theta(k^2\ell)$.

Although we know that $\tilde{r}(H) \leq \bar{r}(H)$, it is a challenging task to identify classes of graphs for which there is an asymptotic gap between both numbers. For complete graphs, Chvátal observed (see [48]) that $\bar{r}(K_t) = \binom{r(K_t)}{2}$. The basic question, attributed to Rödl (see [85]), is to show $\lim_{t \rightarrow \infty} \tilde{r}(K_t)/\bar{r}(K_t)$, or put differently, to show that $\tilde{r}(K_t) = o(\binom{r(K_t)}{2})$. This conjecture remains open, but in 2009 Conlon [25] showed there exists $c > 1$ such that for infinitely many t ,

$$\tilde{r}(K_t) \leq c^{-t} \binom{r(K_t)}{2}.$$

In this chapter, we contribute to this topic by showing that there is an infinite family of trees T_1, T_2, \dots , with $|T_i| < |T_{i+1}|$ for $i \geq 1$, such that

$$\lim_{i \rightarrow \infty} \frac{\tilde{r}(T_i)}{\bar{r}(T_i)} = 0,$$

thus exhibiting the desired asymptotic gap. In fact, we prove a stronger statement, exhibiting the asymptotic gap even for the induced online Ramsey number.

4.2 Induced Paths

In this section, we present an upper bound on the induced online Ramsey number of paths. This is the main building block for induced strategies we obtain later.

Theorem 4.2.1. Let P_n be a path of length n . Then $\tilde{r}_{\text{ind}}(P_n) \leq 28n - 27$.

Proof. First we build the set I of $2(7n - 7) - 1$ isolated edges, then at least $7n - 7$ have the same color, we say this color is *abundant* in I .

Let R^0 and B^0 be the initial paths of lengths 0. In s -th step we have a red induced path $R^s = (r_0, \{r_0, r_1\}, r_1, \dots, r_i)$ of length i and a blue induced path $B^s = (b_0, \{b_0, b_1\}, b_1, \dots, b_j)$ of length j . We denote the concatenation of paths P and Q by $P \cup Q$. The removal of vertices and incident edges is denoted by $P \setminus \{v\}$. We define the potential of s -th step $p^s = 3a + 4o$ where a is the length of the path in color which is abundant in I and o is the length of path in the other color. Further, we show that we are able to maintain the invariant that there are no edges between R^s and B^s and that $p^{s+1} > p^s$.

Assume, without loss of generality, that the blue edges are abundant in I . Let $g = \{x, y\}$ be an unused blue edge from I . One step of Builder is as follows. Builder creates an edge $e = \{r_i, b_j\}$. If Painter colored e red then Builder creates an edge $f = \{b_j, x\}$, however if e is blue then Builder creates $f = \{r_i, x\}$.

Depending on how edges e and f were colored we end up in one of the following four scenarios.

$$(B^{s+1}, R^{s+1}) = \begin{cases} (B^s \cup (e, r_i, f, x, g, y), R^s \setminus \{r_i, r_{i-1}\}) & \text{if } e \text{ and } f \text{ are blue} \\ (B^s \setminus \{b_j\}, R^s \cup (f, x)) & \text{if } e \text{ is blue and } f \text{ is red} \\ (B^s \cup (f, x, g, y), R^s \setminus \{r_i\}) & \text{if } e \text{ is red and } f \text{ is blue} \\ (B^s \setminus \{b_j, b_{j-1}\}, R^s \cup (e, b_j, f, x)) & \text{if } e \text{ and } f \text{ are red} \end{cases}$$

These different cases are also depicted in Figure 4.2. Each of these scenarios lead to a following change in potential.

$$p^{s+1} = \begin{cases} 3(|B^s| + 3) + 4(|R^s| - 2) = p^s + 1 & \text{if } e \text{ and } f \text{ are blue} \\ 3(|B^s| - 1) + 4(|R^s| + 1) = p^s + 1 & \text{if } e \text{ is blue and } f \text{ is red} \\ 3(|B^s| + 2) + 4(|R^s| - 1) = p^s + 2 & \text{if } e \text{ is red and } f \text{ is blue} \\ 3(|B^s| - 2) + 4(|R^s| + 2) = p^s + 2 & \text{if } e \text{ and } f \text{ are red} \end{cases}$$

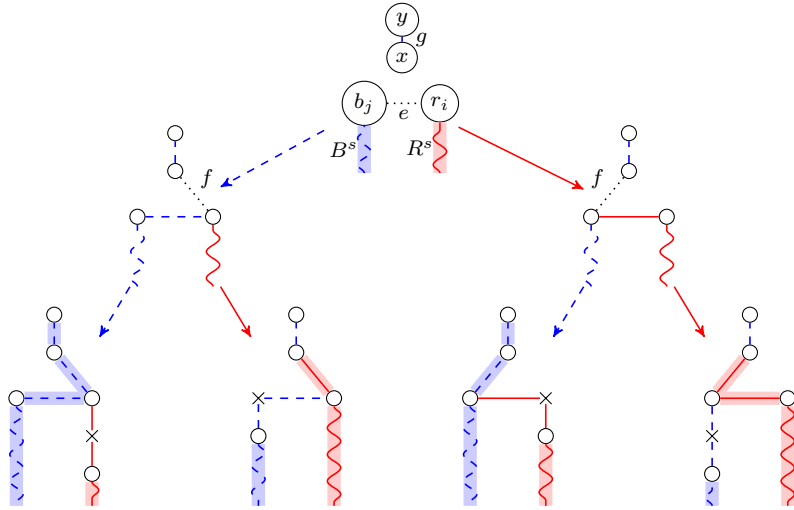


Figure 4.2: One step in creating an induced monochromatic P_n

We obtained a pair of paths B^{s+1}, R^{s+1} such that $p^{s+1} > p^s$ and the invariant holds. The maximum potential for which Builder did not win yet is $p^s = 7n - 7$. Therefore, there are no more than $7n - 6$ steps to finish one monochromatic induced path of length n . To create the initial set I Builder creates $2(7n - 7) - 1$ isolated edges. In each step, Builder creates two edges. The total number of edges created by Builder is no more than $2(7n - 6) + 2(7n - 7) - 1 = 28n - 27$. \square

4.3 Cycles and Induced Cycles

In this section, we present a linear constructive upper bound on the online Ramsey number of cycles $\tilde{r}(C_n)$ and induced cycles $\tilde{r}_{\text{ind}}(C_n)$.

Theorem 4.3.1. Let C_n be a cycle on n vertices, where n is even. Then, $\tilde{r}_{\text{ind}}(C_n) \leq 367n - 27$.

Proof. First, Builder obtains disjoint paths $\rho_1, \rho_2, \dots, \rho_9$ of length $4n/3 - 1$ and one path ρ_{10} of length $n - 2$. Instead of using Theorem 4.2.1 to create these paths separately it is more efficient to create a P_{13n} using at most $28(13n) - 27$ edges and define paths $\rho_1, \rho_2, \dots, \rho_{10}$ as an induced subgraph of P_{13n} . Let the P_{13n} be without loss of generality red. Let $\rho_{i,j}$ denote the j -th vertex of path ρ_i .

In the procedure, Builder will either create a red C_n using $\rho_1, \rho_2, \dots, \rho_{10}$ or three blue paths of length $n/2$ which all start in u and end in either $\rho_{10,1}$ or $\rho_{10,n-1}$. Two of the three paths necessarily share both endpoint and will form a blue C_n . Each blue path will be built within a separate triplet of paths from $\rho_1, \rho_2, \dots, \rho_9$ and alternate between them with each added vertex.

Let us run the following procedure three times – once for each $k \in \{1, 2, 3\}$. Let us set $p = \rho_{3k-2}$, $q = \rho_{3k-1}$ and $r = \rho_{3k}$ (for each run separately). Let us define cyclic order of these paths to be (p, q, r, p) which defines a natural successor for each path. Builder does the following three steps, which are also depicted in Figure 4.3.

1. Create edges $\{u, p_1\}$ and $\{u, p_{n-1}\}$. If both of these edges are red, Builder wins immediately. If that is not the case, then at least one edge $\{u, v_1\}$ where $v_1 \in \{p_1, p_{n-1}\}$ is blue.
2. Now for i from 1 to $n/2 - 1$ we do as follows.
 - Let $j := 2\lfloor i/3 \rfloor$, i.e., every three steps we move by 2 vertices. Let $t \in \{p, q, r\}$ such that $v_i \in t$ and set s to be the successor of t .
 - We create edges $\{v_i, s_{j+1}\}$ and $\{v_i, s_{j+n-1}\}$. If both are red, Builder wins, otherwise take an edge $\{v_i, v_{i+1}\}$ where $v_{i+1} \in \{s_{j+1}, s_{j+n-1}\}$ is blue.
3. Finish the path $(u, v_1, v_2, \dots, v_{n/2-1})$ by creating $\{v_{n/2-1}, \rho_{10,1}\}$ and $\{v_{n/2-1}, \rho_{10,n-1}\}$. Again, if both edges are red, Builder wins immediately. Otherwise, Builder creates a blue path from u to $\rho_{10,1}$ or to $\rho_{10,n-1}$.

If the final circle is red, then it is induced because the initial path is induced and we neither create edges connecting two vertices of ρ_k to itself, nor edges connecting v_i to any vertices between endpoints of the cycle. If the blue cycle is created, then it is induced because we use only odd vertices on $\rho_1, \rho_2, \dots, \rho_9$ for creating the three blue paths and no edges are created between vertices which are further than 1 apart on these blue paths.

Note that the length of paths $\rho_1, \rho_2, \dots, \rho_9$ is sufficient because they need to be at least $2\lfloor \frac{n/2-1}{3} \rfloor + (n-2) \leq \frac{4n-8}{3} \leq 4n/3 - 1$.

By Theorem 4.2.1 we can create the initial induced P_{13n} in $28(13n) - 27$ rounds. For each blue path we used at most n edges so there are at most $3n$ additional edges, hence, $\tilde{r}(C_n) \leq 367n - 27$. \square

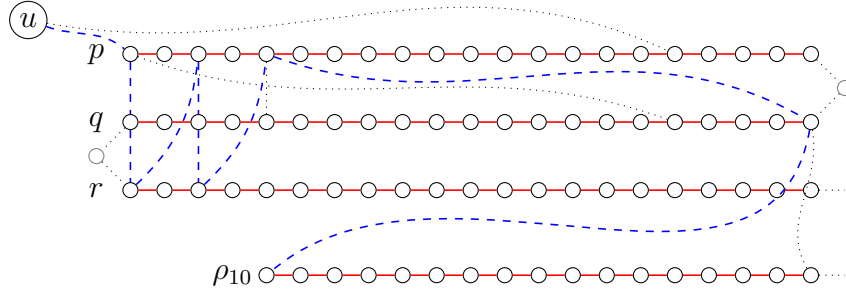


Figure 4.3: Creation of $\rho_{n/2}$ for $n = 18$.

Theorem 4.3.2. Let C_n be a cycle on n vertices, where n is odd. Then $\tilde{r}_{\text{ind}}(C_n) \leq \tilde{r}_{\text{ind}}(C_{2n}) + n$.

Proof. First, we create a monochromatic cycle C_{2n} . Assume without loss of generality that this cycle is blue. Let $c_0, c_1, \dots, c_{2n-1}$ denote vertices on the C_{2n} in the natural cyclic order and let c_i for any $i \geq 2n$ denote vertex c_j , $j = i \bmod 2n$. We join two vertices which lie $n - 1$ apart on the even cycle by creating an edge $\{c_0, c_{n-1}\}$. If the edge is blue it forms a blue C_n with part of the blue even cycle, see Figure 4.4. If the edge is red we can continue and create an edge $\{c_{n-1}, c_{2(n-1)}\}$ and use the same argument. This procedure can be repeated n times finishing with the edge $\{c_{(n-1)(n-1)}, c_{n(n-1)}\}$ where $c_{n(n-1)} = c_0$ because $\text{GCD}(n - 1, 2n) = 2$ and we visit only even vertex.

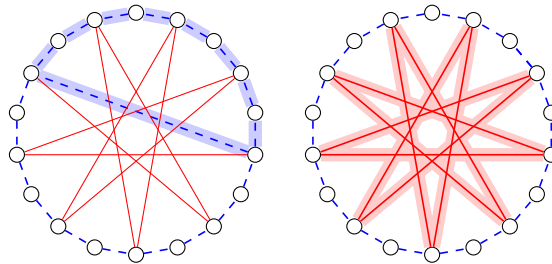


Figure 4.4: The two cases which arise after the final step of building C_9 .

Let E be all the new red edges we just created, i.e., $E = \{\{c_i, c_{i+n-1}\} \mid i \in J\}$ where $J = \{j(n - 1) \mid j \in \{0, 1, \dots, n - 1\}\}$. Since $\text{GCD}(n - 1, 2n) = 2$ it follows that the edges of E complete a cycle $C'_n = (\{c_0, c_2, \dots, c_{2n-2}\}, E)$, see Figure 4.4.

Since the C_{2n} is induced then it follows that the target C_n will be induced as well.

We first used Theorem 4.3.1 to create an even cycle C_{2n} then we added n edges to form the C' . This gives us an upper bound for induced odd cycles $\tilde{r}_{\text{ind}}(C_n) \leq \tilde{r}_{\text{ind}}(C_{2n}) + n \leq 735n - 27$. \square

Non-induced Cycles

Although the induced cycle strategies are asymptotically tight we can get better constants for the non-induced cycles. For even cycles, we can use the non-induced path strategy to create the initial $P_{17n/2}$ in $4(17n/2) - 3$ rounds. Then we add $3n/2$ edges in the similar fashion as for the induced cycles however we can squeeze them more tightly as depicted in the Figure 4.5.

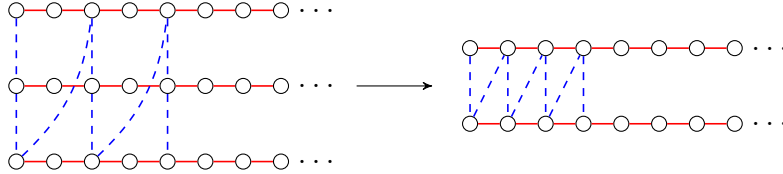


Figure 4.5: More efficient construction for even non-induced cycles.

Using this method the paths $\rho_1, \rho_2, \dots, \rho_6$ need only $5n/4$ vertices each, therefore the initial path $P_{17n/2}$ is sufficient. This gives us $\tilde{r}(C_n) \leq 71n/2 - 3$ for even n which directly translates to odd cycles and gives us $\tilde{r}(C_n) \leq \tilde{r}(C_{2n}) + n \leq 72n - 3$ for odd n .

4.4 Tight Bounds for a Family of Trees

We first prove a general lower bound for the online Ramsey number of graphs. It will be used to show the tightness of bounds in this section.

Lemma 4.4.1. The $\tilde{r}(H)$ is at least $VC(H) \cdot (\Delta(H) - 1)/2 + |E(H)|$ where $VC(H)$ is the vertex cover, $\Delta(H)$ is the maximum degree in H , and $|E(H)|$ is the number of edges.

Proof. Let $\deg_b(v)$ be the number of blue edges incident to the vertex v . Let us define the Painter's strategy against the target graph H as follows.

1. If both incident vertices have $\deg_b < \Delta(H) - 1$ then color the edge blue,
2. otherwise color the edge red.

It is clear that Builder cannot create H in blue color because the blue graph can contain only vertices with degree at most $\Delta(H) - 1$. To obtain a red edge it has to have at least one incident vertex with high blue degree. The minimum number of vertices with high blue degree which are required to complete H is $VC(H)$, therefore, Builder has to create at least $VC(H) \cdot (\Delta(H) - 1)/2$ blue edges. Then Builder has to create at least $|E(H)|$ edges to complete the target graph in red color. \square

Let us define a *spider* $\sigma_{k,\ell}$ for $k \geq 3$ and $\ell \geq 2$ as a union of k paths of length ℓ that share exactly one common endpoint. Let a *center* of $\sigma_{k,\ell}$ denote the only vertex with degree equal to k .

In the following theorem, we obtain an upper bound on $\tilde{r}(\sigma_{k,\ell})$ that asymptotically matches the lower bound from Lemma 4.4.1.

Theorem 4.4.2. $\tilde{r}_{\text{ind}}(\sigma_{k,\ell}) = \Theta(k^2\ell)$.

Proof. We describe Builder's strategy for obtaining an induced monochromatic $\sigma_{k,\ell}$. We start by creating an induced monochromatic path of length $k^2(2\ell + 1)$ which is without loss of generality blue. This path contains k^2 copies of $P_{2\ell}$ as an induced subgraph. Let $P_{i,j}$ denote the j -th vertex on path P_i . Let $\mathbb{P}^1, \mathbb{P}^2, \dots, \mathbb{P}^k$ be k sets where each contains k disjoint induced paths. Let u be a previously unused vertex. Now for each \mathbb{P}^j we do the following procedure, see Figure 4.6 for an illustration.

1. Let $\{P^1, P^2, \dots, P^k\} = \mathbb{P}^j$.
2. Create edges $\{\{u, w\} \mid w \in \{P_1^1, P_1^2, \dots, P_1^k\}\}$. If there are k blue edges there is a $\sigma_{k,\ell}$ with the center in u . If that is not the case, then there is at least one red edge $e^1 = \{u, v^1\}$ where $v^1 \in \{P_1^1, P_1^2, \dots, P_1^k\}$.
3. For i from 2 to ℓ we do as follows.
 - For $v^{i-1} \in P^z$ create edges $\{\{v^{i-1}, w\} \mid w \in \{P_i^1, P_i^2, \dots, P_i^k\} - P_i^z\}$. If all of these edges are blue we have a $\sigma_{k,\ell}$ with the center in v^{i-1} , otherwise there is a red edge $\{v^{i-1}, v^i\}$ where $v^i \in \{P_i^1, P_i^2, \dots, P_i^k\}$.
4. We obtained a red induced path $L^j = (u, \{u, v^1\}, \dots, v^\ell)$.

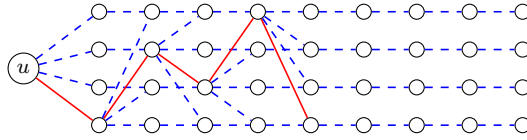


Figure 4.6: Building one red leg of a spider $\sigma_{4,5}$.

If all iterations end up in obtaining a path L^j we have k induced paths of length ℓ which all start in u and together they form a $\sigma_{k,\ell}$ with the center in u .

We built a path $P_{k^2(2\ell+1)}$ using Theorem 4.2.1 using at most $28(k^2(2\ell + 1)) - 27$ edges. During iterations, we created at most $k\ell(k - 1)$ edges. Therefore, we either got a blue $\sigma_{k,\ell}$ during the process or a red $\sigma_{k,\ell}$ after using no more than $\tilde{r}_{\text{ind}}(\sigma_{k,\ell}) \leq 57k^2\ell + 28k^2 - k\ell - 27 = \mathcal{O}(k^2\ell)$ rounds.

The lower bound of Lemma 4.4.1 gives us $\Omega(k^2\ell)$ so $\tilde{r}_{\text{ind}}(\sigma_{k,\ell}) = \Theta(k^2\ell)$. \square

We can get the bound on non-induced spiders in a similar way, however, we can use several tricks to get a bound which is not far from the lower bound.

Theorem 4.4.3. $\tilde{r}(\sigma_{k,\ell}) \leq k^2\ell + 15k\ell + 2k - 12 = \mathcal{O}(k^2\ell)$.

Proof. We create a path $P_{4k\ell}$ using strategy by Grytczuk et al. [60] in $4(4k\ell) - 3$ rounds and split it into $2k$ paths of length 2ℓ . We follow the same strategy as in the induced case, however, we work over the same set of paths in all iterations and we exclude those vertices

which are already used by some path. Choosing $2k$ paths guarantees that we have big enough set even for the last iteration. We create $2k$ edges from u and then we use $k\ell(k-1)$ to create the red paths. We either get a blue $\sigma_{k,\ell}$ in the process or a red $\sigma_{k,\ell}$ after using no more than $k^2\ell + 15k\ell + 2k - 12$ rounds. \square

4.5 Family of Induced Trees with an Asymptotic Gap

In 2009 Conlon [25] showed that the online Ramsey number and the size-Ramsey number differ asymptotically for an infinite number of cliques. In this section, we present a family of trees which exhibit the same property, i.e., their induced online Ramsey number and size-Ramsey number differ asymptotically.

Definition 4.5.1. Let *centipede* $S_{k,\ell}$ be a tree consisting of a path P_ℓ of length ℓ where each of its vertices has k adjacent leaves, i.e., a thorn-regular caterpillar.

Note that $S_{k,\ell}$ has $(k+1)(\ell+1)$ vertices and its maximum degree is $k+2$. We will show that $S_{k,\ell}$ exhibits a small induced online Ramsey number.

The proof uses the potential method to bound the number of created edges. We proceed in steps where each step either makes the centipede longer or we get a vertex which has k incident edges in both colors (so-called colorful star). When the potential reaches a certain threshold we either get sufficiently long centipede or enough of the colorful stars to run the induced path strategy, which enforces an induced monochromatic centipede.

Theorem 4.5.2. $\tilde{r}_{\text{ind}}(S_{k,\ell}) \leq 678k\ell - 652k + 476\ell - 434 = \mathcal{O}(k\ell)$.

Proof. We will need a “degree-type” notion. Let $G = (V, E)$ be a graph whose edges are colored red and blue. Let $U \subseteq V$. For a vertex $v \in V$ let $\overline{\text{deg}}(v, U)$ be a degree outside U , i.e., $\overline{\text{deg}}(v, U) = |N(v) \setminus U|$. Let $\overline{\text{deg}}_{\text{b}}(v, U)$ and $\overline{\text{deg}}_{\text{r}}(v, U)$ be a vertex degree outside U in blue or red color, respectively. Formally,

$$\overline{\text{deg}}_{\text{b}}(v, U) = \left| \{u \in N(v) \setminus U : \{u, v\} \text{ is blue}\} \right|,$$

and similarly for $\overline{\text{deg}}_{\text{r}}(v, U)$.

A *center* of a star S_k is the vertex of degree k . A *center* of a union of stars are centers of all stars in the union. A *colorful star* is a star such that for its center v holds that $\overline{\text{deg}}_{\text{b}}(v) \geq k$ and $\overline{\text{deg}}_{\text{r}}(v) \geq k$. Let H be a centipede or a union of stars. We denote a center of H by $c(H)$.

We will proceed in steps. Let a superscript X^i of any set X denote the state of the set in i -th step. Also, let $X^{i+1} = X^i$ if not mentioned otherwise.

We will gradually build two centipedes (one red, one blue) and a set of colorful stars. Let R^i (B^i) be a red (blue) centipede in i -th step. First, we assume that both R^i and B^i are nonempty. We show later a strategy for the case where R^i or B^i is empty (i.e., centipede of length 0).

Let Q_r^i be a union of colorful stars such that for each star $S \in Q_r^i$ we have $c(S) \in c(R^j)$ for some $j < i$, i.e., the center of S were in the center of the red centipede in some previous step. The Q_b^i is defined similarly, it is a union of colorful stars such that for each star $S \in Q_b^i$ we have $c(S) \in c(B^j)$ for some $j < i$. Let $U^i = R^i \cup B^i \cup Q_r^i \cup Q_b^i$. For $v \in c(R^i)$ let $\overline{\deg}_o(v)$ be $\overline{\deg}_b(v, U^i)$, i.e., blue degree of v outside centipedes and colorful stars. Similarly, let $\overline{\deg}_o(v)$ be $\overline{\deg}_r(v, U^i)$ for $v \in c(B^i)$ (o stands for the ‘‘other’’ color). In each step, we either make one centipede longer by 1, add one colorful star to Q_r or Q_b , or increase $\overline{\deg}_o(v)$ of $v \in c(R^i) \cup c(B^i)$. One step will proceed as follows.

1. Let u and v be endpoints of $c(R^i)$ and $c(B^i)$ respectively.
2. Create an edge $e = \{u, x\}$ where x is a previously unused vertex.
3. If e is blue set $w := u$, if e is red create an edge $f = \{v, x\}$ and set $w := v$.
4. Perform one of the following steps.
 - a) If e is red and f is blue, create edges from x until k of them are in the same color and then add x to respective centipede center set.
 - b) Either e is blue, or both e and f are red,
 - i. if $\overline{\deg}_o(w) < k$, the $\overline{\deg}_o(w)$ was increased by 1,
 - ii. or $\overline{\deg}_o(w) \geq k$, we have a colorful star with center in w , therefore we move w from its centipede center set to respective colorful star set, i.e., $c(Q_r^{i+1}) = c(Q_r^i) \cup \{u\}$ and $c(R^{i+1}) = c(R^i) \setminus \{u\}$ if $w = u$, or $c(Q_b^{i+1}) = c(Q_b^i) \cup \{v\}$ and $c(B^{i+1}) = c(B^i) \setminus \{v\}$ if $w = v$.

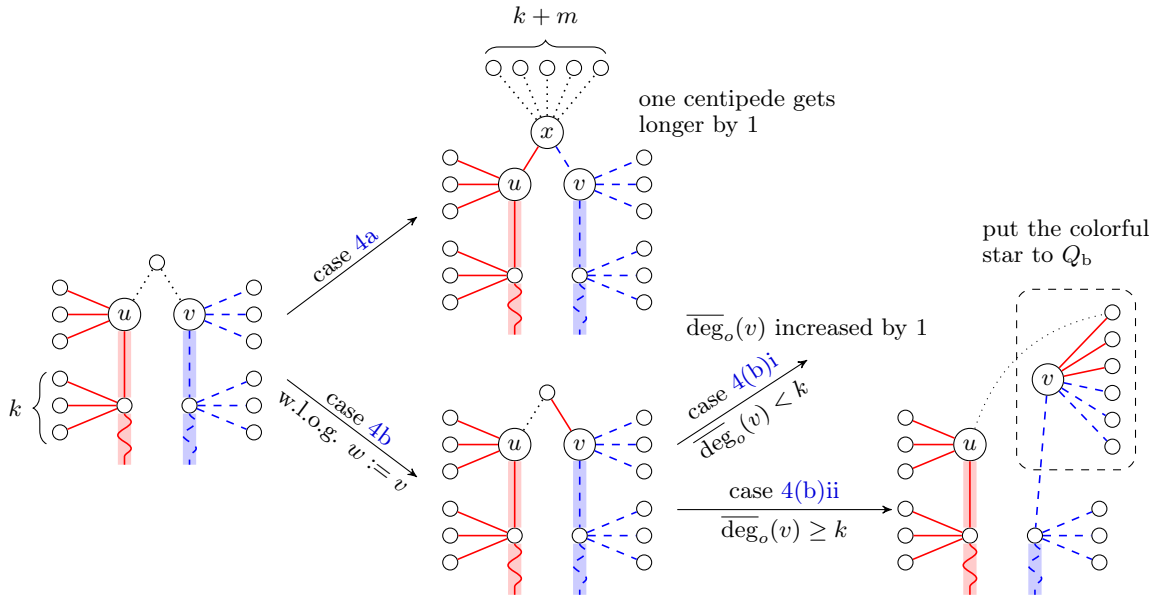
See Figure 4.7 for an illustration of various cases during one step.

Let p^i be a potential in i -th step defined as

$$p^i = (|c(R^i)| + |c(B^i)|)(k + 2) + (|c(Q_r^i)| + |c(Q_b^i)|)(3k + 2) + 2 \sum_{v \in c(R^i) \cup c(B^i)} \overline{\deg}_o(v).$$

For all the outcomes of one step the potential will increase by at least the number of created edges.

- In case 4a we create $2 + k + m$ edges. $k + 1$ edges extend one centipede by one star, one edge is not used, and $m \leq k - 1$ edges are additional edges of the other color on the star. Extending one centipede by a star with m edges in other color increases p by $(k + 2) + 2m$.
- In case 4(b)i we create at most 2 edges, increasing $\overline{\deg}_o$ of one vertex by one, which increases p by 2.
- In case 4(b)ii we create at most 2 edges, making one centipede shorter by one, however adding one colorful star to either Q_r or Q_b so p increases by $(3k + 2) - (k + 2) - 2(k - 1) = 2$.


 Figure 4.7: One step of building a $S_{k,\ell}$ where $k = 3$

Note that the graphs induced by $c(R^i)$ and $c(B^i)$ are paths. These graphs are altered by adding one vertex at the end or moving end-vertex to respective $c(Q^i)$ set. It follows that the graphs induced by $c(Q_r^i)$ and $c(Q_b^i)$ are forests.

Assume that after many steps we end up with $|c(R^i)| = |c(B^i)| = \ell$, $|c(Q_r^i)| = |c(Q_b^i)| = 4(28\ell - 27) - 1$, $\overline{\deg}_o(v) = k - 1$ for all $v \in c(R^i) \cup c(B^i)$, and we did not win yet. In such situation the potential is

$$\begin{aligned} p^i &= 2\ell(k + 2) + 2(4(28\ell - 27) - 1)(3k + 2) + 2(k - 1)2\ell \\ &= 678k\ell - 654k + 448\ell - 436. \end{aligned}$$

We now perform one last step. If the last step did not result in a win then either Q_r^{i+1} or Q_b^{i+1} contains $4(28\ell - 27)$ colorful stars. As star centers form forests, there is an independent set which contains at least half of them. We take the $2(28\ell - 27)$ independent colorful star centers and perform the induced path strategy on them (each edge of the strategy uses at most 2 vertices). This takes $28\ell - 27$ additional edges. An induced path over colorful star centers creates the desired centipede.

The final step might add various number of rounds to our strategy depending on the case which we end up in. Case 4a demands at most $1 + 2k$ edges to win. Case 4(b)i cannot happen because $\overline{\deg}_o(v) = \overline{\deg}_o(u) = k - 1$. And case 4(b)ii demands that we add 2 edges and then perform the path strategy using at most $\tilde{r}_{\text{ind}}(P_\ell) \leq 28\ell - 27$ edges. Taking maximum over the number of moves we get the final upper bound on the number of edges $\tilde{r}_{\text{ind}}(S_{k,\ell}) \leq 678k\ell - 652k + 476\ell - 434$ which is $\mathcal{O}(k\ell)$.

We now discuss why the final centipede is induced. First, let us partition all vertices used in the strategy into three groups: $\mathcal{R} = c(R) \cup c(Q_r)$, $\mathcal{B} = c(B) \cup c(Q_b)$, and \mathcal{O} (which contains all the remaining vertices). Note that in each step some vertices are added to the

groups but once assigned they never change their group. Vertices in \mathcal{R} and \mathcal{B} are always added to $c(R)$ or $c(B)$ and then they might be moved into $c(Q_r)$ and $c(Q_b)$, respectively. Vertices in \mathcal{O} are used during one step and are never used again. Specifically, in case 4a there are $k + m$ vertices created and all of them are connected to one center vertex (in $c(R)$ or $c(B)$); in case 4b one new vertex is connected to at most one vertex from \mathcal{R} and another one from \mathcal{B} . Assume that the centipede is in red color (the argument is the same for blue). The centipede either appears with centers in $c(R)$ or $c(Q_r)$. If the former occurred, then the centers of $c(R)$ induce a path. If the latter occurred, then the vertices of $c(Q_r)$ we used in the induced path strategy were independent. In both cases, the leaves of the centipede appear in \mathcal{O} . These vertices have at most one edge to \mathcal{R} and have no edges between each other.

If R or B is empty then the strategy changes slightly. In all the cases we omit creation of edge e if $R = \emptyset$ and f if $B = \emptyset$. If e is omitted the algorithm assumes it exists and it was colored red when deciding what edges to draw next. The same is done for omitted f by assuming it was blue. We observe that all the steps stay the same and the potential increases in the same manner but we created fewer edges which does not break the obtained upper bound. \square

Due to Beck [7] we have a lower bound for trees T which is $\bar{r}(T) \geq \beta(T)/4$ where $\beta(T)$ is defined as

$$\beta(T) = |T_0|\Delta(T_0) + |T_1|\Delta(T_1),$$

where T_0 and T_1 are partitions of the unique bipartitioning of T . Our family of trees have $\beta(S_{k,\ell}) \approx (\ell/2 + k\ell/2)(k + 2) = \Theta(k^2\ell)$, which gives us the lower bound on their size-Ramsey number $\bar{r}(S_{k,\ell}) = \Omega(k^2\ell)$.

Since by Theorem 4.5.2 we have $\tilde{r}_{\text{ind}}(S_{k,\ell}) = \mathcal{O}(k\ell)$ the online Ramsey number for $S_{k,\ell}$ is asymptotically smaller than its size-Ramsey number. The induced variants bound the non-induced ones as $\tilde{r}(S_{k,\ell}) \leq \tilde{r}_{\text{ind}}(S_{k,\ell})$ and $\bar{r}(G) \leq \bar{r}_{\text{ind}}(G)$. This implies an asymptotic gap between $\bar{r}(S_{k,\ell})$ and $\tilde{r}(S_{k,\ell})$ as well as a gap between $\tilde{r}_{\text{ind}}(S_{k,\ell})$ and $\bar{r}_{\text{ind}}(S_{k,\ell})$.

Corollary 4.5.3. There is an infinite sequence of trees T_1, T_2, \dots such that $|T_i| < |T_{i+1}|$ for each $i \geq 1$ and

$$\lim_{i \rightarrow \infty} \frac{\tilde{r}(T_i)}{\bar{r}(T_i)} = 0 \quad \text{and} \quad \lim_{i \rightarrow \infty} \frac{\tilde{r}_{\text{ind}}(T_i)}{\bar{r}_{\text{ind}}(T_i)} = 0.$$

Group Identification

Information diffusion in social networks is a well-studied concept in social choice theory. We propose the study of the diffusion of two secrets in a heterogeneous environment, that is, there are two different networks with the same set of agents (e.g., the structure of the set of followers might be different in two distinct social networks).

Formally, our model combines two group identification processes for which we do have independent desiderata – either constructive, where we would like a given group of agents to be exposed to a secret, or destructive, where a given group of agents should not be exposed to a secret. To be able to reach these targets, we can either delete an agent or introduce a previously latent agent.

As the problem is NP-hard for multiple settings, we propose a parameterized study with respect to most of the natural parameters, the number of influenced agents, the size of the required/protected agent sets, and the duration of the diffusion process. We complement our hardness results with XP algorithms and respective running-time lower bounds based on famous Exponential-Time Hypothesis.

5.1 Introduction

We have a group of agents \mathcal{A} and two social networks (this is formally captured by two directed graphs with identical vertex set \mathcal{A}). We use the liberal starting rule [73], known from group identification, to model the spread of the two secrets at the beginning to certain agents (these agents have a self-loop). Each agent knowing (upon learning) a secret will share it in the appropriate network. Finally, we are given two groups of agents D_1 and D_2 and a positive integer k , we want to decide if there is a group of at most k agents X such that if we remove these agents, the agents in D_1 will not learn the first secret and the agents in D_2 will not learn the second secret.

It is apparent that there are many natural points where one can modify the DLDD problem. One might consider the *constructive-destructive* variant of the problem where we want one group of agents to learn the first secret while preventing the other group from learning the other secret – the DLCD problem.

Related Work Our work is at the intersection of network dynamics (in particular, secret spreading) and group identification (as it uses the LSR rule as the spread model). There is vast work on secret spread in a social network; one of the most discussed models leads (in the discrete case) to the so-called target set selection introduced by Kempe et al. [74]. Here, we run a certain activation process on a weighted undirected graph; it is worth noting that our use of LSR leads to a similar process (see Equation (5.1)) in a directed graph. The TARGET SET SELECTION problem (finding a small activation set) is a notoriously hard problem from the perspective of both classical and parameterized complexity [5, 10, 15, 22, 24, 31, 33, 38, 41, 43, 91, 95]. However, in our work the activation sets are given and our task is to secure certain targets – a task recently studied by Cordasco et al. [32]. In the group identification [37, 73] line of work the most important for us are the recent works [13, 47, 109] on manipulation of the outcome. Works of Yang et al. [47, 109] studies the complexity of destructive or constructive control; it is important to note that both these goals are solvable in polynomial time for LSR (as well as for the consent starting rule – CSR [98]). Boehmer et al. [13] then introduced the combination of the two goals (constructive and destructive) in the same social network (i.e., for the same identification process) and studied the problems from both classical and parameterized perspective. The spread of a computer viruses via e-mail networks can be modeled using LSR as a group-identification process as analyzed by Newman et al. [90].

More secrets In our work, we assume two different social networks on the same set of agents. It is worth noting that this setting was already proposed in the group identification [23], however, the problem studied therein is to find a partition of the agents into disjoint groups of socially qualified agents. Recently, the target set selection problem has been studied with more secrets [56, 79]. There the task is again to find an initial set of agents (in a weighted digraph) so that the spread of the two (or more [9]) secrets is equalized, that is, the agents either end up leaning none of the two or both (for more secrets there are indeed even more complicated settings). One should point out that, similar to the problems studied in this chapter, the model [9, 56] assumes secret-specific weights. In our work, however, we do have different goals for different secrets (and include preventing of learning the secret).

5.1.1 Contribution and Organization

Our Contribution We study a new variant of the group identification problem with more secrets. In the first part of our chapter, we offer the reader a formal definition of the problem and its variants. Later we discuss some of the easy settings of the problem. For such variants, we show that these belong to the complexity class \mathcal{P} , i.e., they are polynomial time solvable with respect to the number of agents. Aside some settings solvable in polynomial time, for most of the variants, we show the hardness of such settings in terms of both classical computational complexity and parameterized complexity. Among other things, we prove that all assumed both DLDD and DLCD variants of the studied problems are NP-complete. We parameterize these variants using (different combinations of) the

most natural parameters and we show that for some settings, such as DLCD where the only parameter is number k of agents we are allowed to affect, the problem is $W[1]$ -hard, and we provide XP algorithms for these settings. For all of them we prove that under some reasonable theoretical assumptions, our algorithms are optimal.

Organization This chapter is organized as follows. In Section 5.2, we define the GROUP IDENTIFICATION PROBLEM and its variants. Next, in Section 5.3 we introduce a notation which will help us define various hardness constructions. Finally, in Section 5.4 we go over several variants of the GROUP IDENTIFICATION PROBLEM and show upper and lower bounds in regards to several parameters.

5.2 Preliminaries

In a GROUP IDENTIFICATION PROBLEM (GIP for short) we are given a *social network* $\mathcal{N} = (\mathcal{A}, \varphi)$, where \mathcal{A} is a set of agents and $\varphi: \mathcal{A} \times \mathcal{A} \rightarrow \{0, 1\}$ is a *qualification profile*. A *qualification graph* $G_{\mathcal{A}, \varphi}$ is an oriented graph with loops and without multiedges where $V(G_{\mathcal{A}, \varphi}) = \mathcal{A}$ and $E(G_{\mathcal{A}, \varphi}) = \{(a, b) \mid \varphi(a, b) = 1\}$. If \mathcal{A} and/or φ are clear from the context, we shorten $G_{\mathcal{A}, \varphi}$ to G_φ , or simply G .

We study the problem according to a *qualification rule* $\varrho: \mathcal{N} \rightarrow \mathcal{P}(\mathcal{A})$ which formalize the way how the set $S_0 \subseteq \mathcal{A}$ of *starting agents* is found. We say that agent $a \in \mathcal{A}$ *qualifies* agent $b \in \mathcal{A}$ if and only if $\varphi(a, b) = 1$. Let $Q_\varphi^+(a) = \{b \in \mathcal{A} \mid \varphi(a, b) = 1\}$ and $Q_\varphi^-(a) = \{b \in \mathcal{A} \mid \varphi(a, b) = 0\}$. Then the dynamic *qualification process* continues as follows

$$\begin{aligned} S_0 &= \varrho(\mathcal{A}, \varphi) \\ S_i &= S_{i-1} \cup \{a \in \mathcal{A} \mid \exists b \in S_{i-1}: \varphi(b, a) = 1\}. \end{aligned} \quad (5.1)$$

When the above process stabilizes, i.e., when $S_i = S_{i+1}$, we terminate and set $Q^{\text{fin}} = S_i$ to be the set of *socially qualified agents*. All agents which are not socially qualified are *socially disqualified*. We denote the number of steps (or rounds) before the process stabilizes as \mathcal{T} .

If there are more qualification profiles φ_1 and φ_2 , then we denote $Q_{\varphi_1}^{\text{fin}}$ the set of socially qualified agents for profile φ_1 (and $Q_{\varphi_2}^{\text{fin}}$ for φ_2). We stress here that the two processes do not interact (besides sharing the agents), and *consensus-start-respecting rule* ϱ^{CSR} chooses $S = \{a \in \mathcal{A} \mid (\forall b \in \mathcal{A}) \varphi(b, a) = 1\}$.

It is easy to evaluate whether the group identification target is satisfied. We are interested in *controlling the outcome*, that is, the task is to decide whether one can use a limited number of changes to the instance so that a desired target is met. Some of the possible targets are: *Constructive* where we are given a subset of agents $C \subseteq \mathcal{A}$ and we want them all to be socially qualified ($C \subseteq Q^{\text{fin}}$), and similarly, *Destructive* where the set $D \subseteq \mathcal{A}$ should be socially disqualified ($D \cap Q^{\text{fin}} = \emptyset$). It is worth noting that constructive/destructive in the context of control (e.g., in control of electorates) usually refers to

the action related to agents (voters); here we reserve these for the target as was done by Boehmer et al. [13]. We try to reach the target by changing the instance by controlling at most k agents. Main way to change the instance is via controlling the *latent agents* – these agents are in the set but do not participate in spreading the secret. There are several different *control operations* we can use to accomplish our target:

- Agent addition: some agents of the instance are labelled as latent and we can persuade them to participate in the process by controlling them.
- Agent deletion: controlling an agent makes him latent.

We are interested in the case where the control is searched for in two or more group identification instances over the same set of agents. In this model, every agent has some preferences for both GIPs but these instances cannot be solved separately because the control affects both instances at the same time (their common agent set).

Let \odot be a control operation, f and g two qualification target deciding functions, and ϱ a qualification rule. We formally define the studied problem as follows.

(\odot, f, g, ϱ) -GROUP IDENTIFICATION PROBLEM

Input: A set of agents \mathcal{A} , two qualification profiles φ_1 and φ_2 , a positive integer k , and two subsets of agents $A_1, A_2 \subseteq \mathcal{A}$.

Question: Is there a set $X \subseteq \mathcal{A}$ with $|X| \leq k$ such that qualification process started with $S_0^1 = \varrho(\mathcal{A} \odot X, \varphi_1)$, $S_0^2 = \varrho(\mathcal{A} \odot X, \varphi_2)$ terminates with $f(Q_{\mathcal{A} \odot X, \varphi_1}^{\text{fin}}, A_1) = 1$ and $g(Q_{\mathcal{A} \odot X, \varphi_2}^{\text{fin}}, A_2) = 1$?

To exemplify our definition we note that the DLDD problem is in fact (\odot, f, g, ϱ) -GROUP IDENTIFICATION PROBLEM, where \odot is agent deletion operation, both f and g check whether agents from A_1 and A_2 are socially disqualified, and ϱ is liberal-start-respecting rule.

Throughout this chapter we denote by \oplus the agent addition operation, by \ominus we denote the agent deletion operation. We abbreviate constructive and destructive target checking functions as κ and δ , respectively.

5.2.1 Computational Complexity

For our NP-hardness results we use polynomial reductions from the famous 3-SAT problem which is known to be NP-complete [30].

Definition 5.2.1. In the 3-SAT problem we are given a formula \mathcal{F} in CNF such that there are at most three literals in each clause. The goal is to find a satisfying assignment $\pi: \text{var}(\mathcal{F}) \rightarrow \{0, 1\}$, where $\text{var}(\mathcal{F})$ denotes the set of variables of \mathcal{F} .

We denote the i -th clause of \mathcal{F} as C_i and the j -th literal of this clause as $C_{i,j}$. Let $x_j \in \text{var}(\mathcal{F})$ be a variable. Then ℓ_j and $\neg\ell_j$ denotes a set of positive and negative literals of the variable x_j , respectively.

5.2.2 Parameterized Complexity

In parameterized algorithmics the input of the problem is given along with a numeric value (the *parameter*) which expresses certain properties of the instance, e.g., the size of the sought solution. Positive results focus on finding either fixed-parameter-tractable (FPT for short) algorithms (those admitting running times of the form $f(k) \cdot \text{poly}(n)$) or XP algorithms ($n^{f(k)}$), where n is the size of the input, k is the parameter value, and f is a computable function.

We investigate many *settings* where certain parts of the input may appear either as parameters, constants, or unrestricted. Let us denote those cases as follows.

- **const** – X is a fixed constant which is independent of the input instance,
- **param** – X is given as a parameter of the problem,
- **input** – X is not constrained, i.e., it may depend on the input.

It is trivial to see that a problem where a parameter is **const** is not harder than when the parameter is **param**. The same relation is also true for **param** and **input**. This connection allows us to infer complexity for many settings which are not explicitly stated: hardness results apply to all settings which are not easier and positive results apply to all settings which are not harder. Also, whenever a setting is XP, then we infer that by changing all parameters from **param** to **const** we get a setting which is in P. This comes directly from the definition of XP as complexity class with $n^{\mathcal{O}(f(k))}$ where k is a parameter. Similarly, for setting which are not easier than some W[1]-hardness setting with changing of **param** to **input** becomes NP-hard.

Among the negative results, the goal is to characterize computational problems in terms of W-hierarchy. It holds that $W[0] = \text{FPT}$. Similarly to classical complexity theory, it is widely believed that there are no FPT algorithms for problems that are W[1]-hard [35, Chapter 13].

Analogously to polynomial reduction one can show that a problem belongs to a class of W-hierarchy by reduction from another problem out of this complexity class.

Definition 5.2.2 (Parameterized reduction [35]). Let A, B be two parameterized problems. A *parameterized reduction* from problem A to problem B is an algorithm \mathcal{A} that, given an instance (x, k) of A , outputs modified instance (x', k') of the problem B such that

1. $(x, k) \in A \iff (x', k') \in B$,
2. $k' \leq g(k)$, and
3. \mathcal{A} works in time $f(k) \cdot |x|^{\mathcal{O}(1)}$,

where $f, g: \mathbb{N} \rightarrow \mathbb{N}$ are some computable functions.

To show $W[1]$ -hardness of GIP variants, we use a parameterized reduction from the k -MULTICOLORED INDEPENDENT SET, PARTITIONED SUBGRAPH ISOMORPHISM, GRID TILING, or the GRID TILING WITH \leq problem defined below, which are well-known $W[1]$ -hard problems [35, 94].

Definition 5.2.3. In the k -MULTICOLORED INDEPENDENT SET problem we are given a graph G' and a mapping $c: V(G') \rightarrow \{1, \dots, k\}$. The goal is to decide whether there exists an independent set I of size k such that no two distinct vertices $u, v \in I$ have the same color, i.e., $c(u) \neq c(v)$.

Definition 5.2.4. In the PARTITIONED SUBGRAPH ISOMORPHISM problem (PSI) we are given two graphs G' and H' and a mapping $c: V(G') \rightarrow V(H')$. The task is to decide whether there exists a mapping $d: V(H') \rightarrow V(G')$ such that $c \circ d$ is the identity mapping (on vertices of H').

Definition 5.2.5. In the GRID TILING problem we are given an integer k , an integer n , and a collection \mathcal{S} of k^2 nonempty sets $S_{i,j} \subseteq [n] \times [n]$ ($1 \leq i, j \leq k$). The task is to find, for each $1 \leq i, j \leq k$, a pair $s_{i,j} \in S_{i,j}$ such that

- if $s_{i,j} = (a, b)$ and $s_{i+1,j} = (a', b')$, then $a = a'$,
- if $s_{i,j} = (a, b)$ and $s_{i,j+1} = (a', b')$, then $b = b'$.

Definition 5.2.6. In the GRID TILING WITH \leq problem have the same input as to GRID TILING, but the task is to find, for each $1 \leq i, j \leq k$, a pair $s_{i,j} \in S_{i,j}$ such that

- if $s_{i,j} = (a, b)$ and $s_{i+1,j} = (a', b')$, then $a \leq a'$,
- if $s_{i,j} = (a, b)$ and $s_{i,j+1} = (a', b')$, then $b \leq b'$.

We refer the interested reader to standard textbooks in this area, e.g., [35, 40].

5.3 Node and Composition Notation

We define a node notation which we use to construct graph gadgets from the ground up without excess description of their vertices and edges.

Let a *node* of a graph G be a triplet $N = (B, S, T)$ where B is an induced subgraph of G , i.e., $V(B) \subseteq V(G)$ and $E(B) = E(G) \cap \binom{V(B)}{2}$, and $S, T \subseteq V(B)$ are *source* and *sink* vertices, respectively. Let N_i, N_j be two nodes. A directed edge (N_i, N_j) corresponds to a directed complete bipartite subgraph from vertices T_i to vertices S_j , see Figure 5.1.

Definition 5.3.1. Let the *parallel node composition* of nodes $N_i = (B_i, S_i, T_i)$ and $N_j = (B_j, S_j, T_j)$ be a new node $N_i \uplus N_j$ (disjoint union) defined as $(B_i \uplus B_j, S_i \uplus S_j, T_i \uplus T_j)$.

Let the *serial node composition* of $N_i = (B_i, S_i, T_i)$ and $N_j = (B_j, S_j, T_j)$ be a new node $N_i \dot{\cup} N_j$ defined as

$$N_i \dot{\cup} N_j = \left((V(B_i) \uplus V(B_j), E(B_i) \uplus E(B_j) \uplus \{(t, s) \mid t \in T_i, s \in S_j\}), S_i, T_j \right).$$

By using the parallel composition on several nodes we mean disjoint union of all of them, and by the serial composition we mean building a directed path over all of them in the given order. Any vertex or vertex set S may also be used as an *elementary node* $((\{S\}, \emptyset), S, S)$. An example of serial node composition is shown in Figure 5.1. We use the composition notation to describe edge-structure and to show properties of induced subgraphs of G .

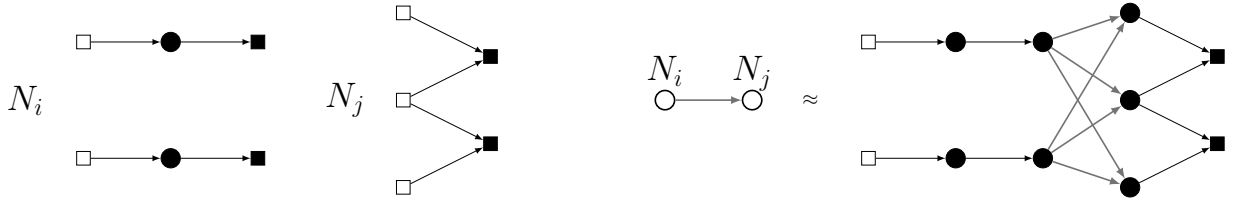


Figure 5.1: Serial composition of nodes N_i and N_j adds an (N_i, N_j) edge which represents a directed complete bipartite graph from sinks of N_i to sources of N_j .

Definition 5.3.2. The *cut-weight* of a node $N = (B, S, T)$, denoted by $w(N)$, is the size of minimum vertex separator between sources set S and sinks set T in the subgraph B .

We proceed with several observations which serve as building blocks for our future argumentation.

Observation 5.3.3. A path can be built as a serial node composition of elementary nodes consisting of its vertices.

Observation 5.3.4. A node consisting of a directed path P on n vertices with source on the first vertex and sink on the last one has cut-weight equal to 1.

Observation 5.3.5. For $N_k = N_i \uplus N_j$ we have $w(N_k) = w(N_i) + w(N_j)$. For $N_k = N_i \dot{\cup} N_j$ it holds $w(N_k) = \min\{w(N_i), w(N_j)\}$.

Proof. We prove the observation by the contradiction. Assuming that we hit less than $\min\{w(N_i), w(N_j)\}$ vertices leaves at least one path in each component, and by concatenating them we get a path from the source to the sink of N_k , a contradiction. \square

5.4 Liberal Starting Rule

In the *liberal-start-respecting rule*, denoted ϱ^{LSR} , we construct the set of starting agents as $S_0 = \{a \in \mathcal{A} \mid \varphi(a, a) = 1\}$ and then we proceed with standard qualification process

Problem	Result type	Obtained by	Shown by
general	XP	try all solutions	Observation 5.4.1
DLCD	$ D = 1$	–	Observation 5.4.3
	NP-complete	red. 3-SAT	Theorem 5.4.4
	XP	cut paths & flow	Lemma 5.4.6
	W[1]-hard	red. IND. SET	Lemma 5.4.8
DLDD	W[1]-hard	red. GRID TILING $_{\leq}$	Lemma 5.4.9
	W[1]-hard	red. GRID TILING	Lemma 5.4.10
	FPT	cut paths & check	Lemma 5.4.7
ALCD	$ D = 1$	–	Observation 5.4.12
	NP-complete	Theorem 5.4.4	Theorem 5.4.13
	XP	add paths & check	Lemma 5.4.14
ALCC	W[1]-hard	Lemma 5.4.10	Lemma 5.4.15
	W[1]-hard	otherwise trivial	Observation 5.4.17
	NP-complete	red. 3-SAT	Theorem 5.4.19
	W[1]-hard	red. PSI	Lemma 5.4.20
ALCC	FPT	cut paths	Lemma 5.4.21
	W[1]-hard	red. PSI	Lemma 5.4.24
	W[1]-hard	red. PSI	Lemma 5.4.25
	XP	add paths & solve	Lemma 5.4.26

Table 5.1: Results summary for GROUP IDENTIFICATION PROBLEM.

as prescribed by Equation (5.1). We give an overview of the results and used methods in Table 5.1.

A detailed list of the results along with their settings are presented in tables in each section separately. The lists of results are accompanied with Hasse diagrams of the settings that give an overall complexity picture for the respective problem.

5.4.1 Easy variants

There are several variants of GIP we do not dedicate the whole section to, as they are very simple to solve. All the single qualification profile variants with Add/Delete agent operation and Constructive/Destructive target were showed by Erdélyi et al. [47] to be solvable in polynomial time. Only some of the results translate to multiple qualification profile variant as it may be easy to decide that a setting is either solved or impossible straight away.

- Delete agent operation & two constructive targets: Either both targets are met at the start or it is impossible.
- Add agent operation & two destructive targets: Either both targets are met at the start or it is impossible.

We show that all the remaining variants are NP-hard. The first algorithm we propose is captured in the following observation which is directly applicable to all the studied cases parameterized by the solution size.

Observation 5.4.1. All the DLCD, DLDD, ALCC, and ALCD problems parameterized by the solution size k are in complexity class XP as we can check all the possible solutions in $n^{\mathcal{O}(k)}$ time.

5.4.2 Agent deletion: Constructive-Destructive

Notation 5.4.2. By DLCD we denote $(\Theta, \kappa, \delta, \varrho^{\text{LSR}})$ -GROUP IDENTIFICATION PROBLEM. For the rest of this section, we set aliases A_1 as C , and A_2 as D .

Note that any agent in $C \cap D$ must not be deleted which allows us to do the following simplification.

Observation 5.4.3. In every instance of DLCD one can reduce the size of D to 1 while increasing $|C|$ by one.

Proof. Create a new auxiliary sink vertex t' and create edges $\{(d, t') \mid d \in D\}$ in φ_2 , set $D' = \{t'\}$, and add t' as φ_1 's starting vertex ($(t', t') \in \varphi_1$) and also as its target ($t' \in C$). Agent t' cannot be deleted as it would make the satisfaction of the first selection rule impossible. \square

The remainder of this section is filled with results that stemmed from investigating influence of $|C|$, k , and \mathcal{I} secondary input measures on the complexity of DELETE AGENT LSR CONSTRUCTIVE-DESTRUCTIVE GIP. These are summarized in Table 5.2 and the complete complexity picture can be seen on Figure 5.2.

$ C $	k	\mathcal{I}	Class	Shown by
const	input	input	NP-complete	Theorem 5.4.4
input	input	const	NP-complete	Corollary 5.4.5
param	input	param	XP	Lemma 5.4.6
input	param	input	XP	Observation 5.4.1
input	param	param	FPT	Lemma 5.4.7
input	param	input	W[1]-hard	Lemma 5.4.8
const	param	input	W[1]-hard	Lemma 5.4.9
const	input	param	W[1]-hard	Lemma 5.4.10
param	input	const	W[1]-hard	Lemma 5.4.10

Table 5.2: Results summary for the DELETE AGENT LSR CONSTRUCTIVE-DESTRUCTIVE GIP.

First, we establish hardness of the problem.

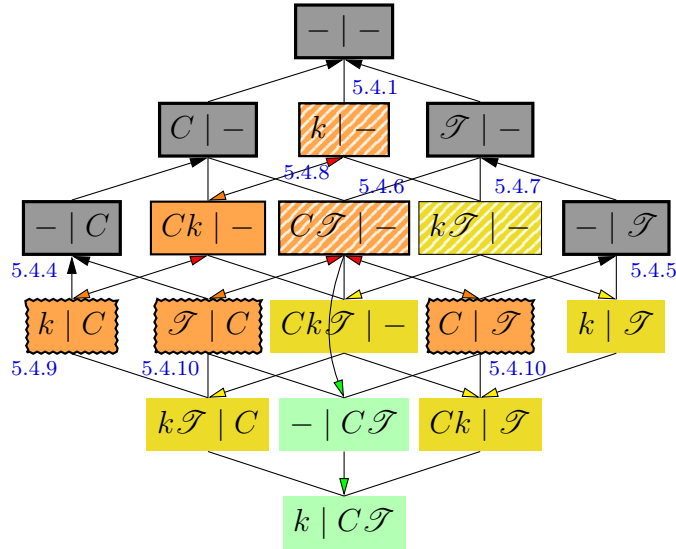


Figure 5.2: Hasse diagram for results in the DELETE AGENT LSR CONSTRUCTIVE-DESTRUCTIVE GIP for cases where k is not a constant. **Legend:** each box represents a setting, and contains two lists of secondary measures; first list contains the (combined) parameters and second list contains constants; boxes are connected if their complexities are comparable, arrows along the connections signify inference of results; very thick border means NP-hardness, thick border is for W[1]-hardness; colors represent tractability class – from light to dark we have P (light green), FPT (yellow), XP (orange), and NP (gray); stripped fill and jagged border signify our results in tractability (XP, FPT) and hardness (W[1]-hard), respectively; We write C instead of $|C|$ for clarity of the diagram. Tractability and hardness results are written above and below the setting box, respectively.

Theorem 5.4.4. DLCD is NP-complete even if the size of the constructive set $|C|$ and the size of the destructive set $|D|$ is a fixed constant. Unless ETH fails, there is no algorithm solving DLCD in $f(|C|+|D|)^{o(n+m)} \cdot (n+m)^{f(|C|+|D|)}$ time, where n and m are cardinalities of the vertex and edge sets of the input graph, respectively, and f is any computable function.

Proof. We show the NP-hardness via reduction from the 3-SAT problem.

Each literal occurrence in \mathcal{F} is represented by a vertex in the qualification graph G . Our construction ensures that the agent $v \in V(G)$ can be controlled only if the respective literal evaluates to true.

Our construction uses two gadgets. The first gadget over the edges of φ_1 ensures that we cannot control both (agents corresponding to) positive literals ℓ_j and negative literals $\neg\ell_j$ of a variable x_j at the same time. The second gadget over the edges of φ_2 ensures that each clause C_i is satisfied. We set the budget k to the number of clauses so that every clause is satisfied by at least one literal occurrence.

The first gadget N_{x_j} consists of a parallel composition of two directed paths. The first path goes over vertices ℓ_j and the second goes over vertices $\neg\ell_j$ (in any order). By Observations 5.3.4 and 5.3.5 the first gadget has cut-weight equal to 2. The second gadget

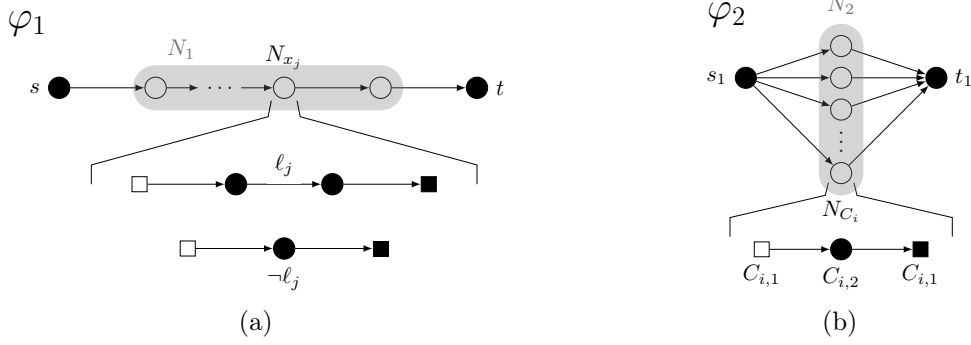


Figure 5.3: Node compositions representing construction of edges of G_{φ_1} in 5.3a and G_{φ_2} in 5.3b illustrating NP-hardness reduction for DLCDC.

N_{C_i} consists of a path over vertices $\{C_{i,1}, C_{i,2}, C_{i,3}\}$, i.e., all vertices representing literals of clause C_i . By Observation 5.3.4 the second gadget has cut-weight equal to 1.

Now, we construct the graph G using the above gadgets. The set of vertices $V(G)$ contains a vertex for every literal occurrence in \mathcal{F} and two auxiliary vertices – one global source s and one global sink t .

G_{φ_1} contains a first gadget node N_{x_j} over vertices of ℓ_j and $\neg \ell_j$ for every variable x_j . Let N_1 be the serial composition of $(N_{x_j} \mid x_j \in \mathcal{F})$ (in any order). We add node-edges (s, N_1) and (N_1, t) , see Figure 5.3a. We add (s, s) to φ_1 and set $C = \{t\}$, hence at least one path from s to t must remain uncut, in particular, we cannot remove s nor t . This means that every node N_{x_j} must have at least one uncut path. The cut path is the chosen valuation, so an uncut path over ℓ_j or $\neg \ell_j$ represents that x_j has either false or true valuation, respectively.

G_{φ_2} contains a second gadget node N_{C_i} for every clause C_i . The N_{C_i} nodes are combined using parallel node composition to create a node N_2 , see Figure 5.3b. We also add node-edges (s, N_2) and (N_2, t) . We add (s, s) to φ_2 and set $D = \{t\}$. This forces a cut of at least one vertex in every clause, representing the literal occurrence which satisfies the clause.

By choosing the budget to be equal to the number of clauses of \mathcal{F} , φ_2 forces the controller to choose exactly one vertex from each clause. φ_1 makes it impossible to choose opposite literals of x_j . Hence, the controlled vertices which satisfy the group selection correspond to a satisfying assignment for the original 3-SAT formula \mathcal{F} , which concludes the NP-hardness of DLCDC. As checking if the solution solves the instance is easy, we have $\text{DLCDC} \in \text{NP}$.

The output of the reduction is DLCDC instance with $|G| \in \mathcal{O}(m)$, $|C| = 1$, and $|D| = 1$. Therefore, algorithm solving DLCDC in $f(|C| + |D|)^{\mathcal{O}(n+m)} \cdot (n + m)^{f(|C| + |D|)}$ time implies algorithm for 3-SAT running in $2^{\mathcal{O}(m)}$ time, which contradicts ETH. \square

The proof can also be slightly changed to obtain NP-completeness for a different setting.

Corollary 5.4.5. The DLCDC problem is NP-complete even if the number of rounds \mathcal{T} is a fixed constant. Moreover, DLCDC does not admit an algorithm running in $f(\mathcal{T})^{o(n+m)} \cdot (n+m)^{f(\mathcal{T})}$ time for any computable function f .

Proof. We follow the same argument as in the proof of Theorem 5.4.4, however, there are two main differences. First, we reduce from (3, 4)-SAT (at most 3 literals in each clause and at most 4 variable occurrences for each variable), which is known to be NP-complete [105]. This ensures that the paths used in the first gadget are constant-length. Second, we split the vertex t into k vertices t_1, \dots, t_k so that every first gadget can have its own sink. Now, we have the depth over φ_1 at most 6, and the depth over φ_2 still remains 5, hence $\mathcal{T} = 6$.

We recall that any 3-SAT formula with m clauses can be transformed into an equivalent (3, 4)-SAT instance with $\mathcal{O}(m)$ clauses [105]. The presented reduction produces instance with $|G| \in \mathcal{O}(m)$ and $\mathcal{T} = 6$. Hence, algorithm for DLCDC running in time $f(\mathcal{T})^{o(n+m)} \cdot (n+m)^{f(\mathcal{T})}$ for some computable function f implies existence of a $2^{\mathcal{O}(m)}$ -time algorithm for 3-SAT, which contradicts ETH. \square

If both parameters are constants, by the following lemma we obtain a poly-time algorithm.

Lemma 5.4.6. The DLCDC problem parameterized by the combined parameter $|C|$ and the number of rounds \mathcal{T} can be solved in $n^{\mathcal{O}(\mathcal{T} \cdot |C|)}$ time.

Proof. The algorithm guesses the paths leading from sources to vertices of C in φ_1 . It fixes that these paths will be reachable by the opinion in φ_1 and that their vertices cannot be removed. Then it checks how many vertices we need to remove so that no vertex in D is reached. This can be done by the standard max-flow/min-cut algorithm with a alteration to remove vertices (done by splitting each vertex in twine, one preserving in-edges and one preserving out-edges; a single edge between them represents the vertex). In the min-cut algorithm we can simulate that a vertex cannot be removed by setting respective edge-weight to $k + 1$. There are at most $n^{\mathcal{T}}$ paths of length \mathcal{T} in G , and we choose $|C|$ of them which makes $(n^{\mathcal{T}})^{|C|}$ possible combinations in total. Max-flow/min-cut algorithm together with the graph preprocessing is polynomial, so we have total $n^{\mathcal{O}(\mathcal{T} \cdot |C|)}$ running time. \square

Lemma 5.4.7. The DLCDC problem is fixed-parameter tractable parameterized by the combined parameter k and \mathcal{T} .

Proof. We will have a branching procedure as follows. Let us run the Breadth First Search (BFS) algorithm on φ_2 to find some path $P = (p_1, \dots, p_m)$ from source to target in $\mathcal{O}(n)$ time, where n is the total size of the graph. If no path P exists, then we check whether the graph on edges φ_1 can reach all vertices in C . If yes, then we return that this branch is a solution, otherwise continue the procedure as follows. If the budget is zero then return that the current branch does not contain a solution. The path P has length at most \mathcal{T} and at least one of its vertices must be chosen to the solution, otherwise the target p_m would learn the secret from source p_1 . Let us choose a vertex $u \in P$ (we branch here for every choice of u) to be included in the solution. Now, run this procedure again for DLCDC on the graph

where u is removed (marked not to be traversed by the BFS) and budget is reduced by 1. If the procedure finds a solution then add u to it and return the new solution.

The above procedure branches on at most \mathcal{I} vertices of P . The biggest recursion depth is k as the budget gets decreased by one with each sub-procedure, and for $k = 0$ the procedure does not call any sub-procedure. The number of procedure calls is at most $\sum_{i=0}^k \mathcal{I}^i = \frac{\mathcal{I}^{k+1}}{\mathcal{I}-1}$ and each procedure calls BFS and several constant-time operations. The total time is $\mathcal{O}(\mathcal{I}^{k+1}n)$ which is FPT in combined parameter k and \mathcal{I} . \square

Since the preceding results illustrate that the DLCDC problem is expected to be intractable, we focus on its parameterized analysis.

Lemma 5.4.8. The DLCDC problem is W[1]-hard when parameterized by the solution size k . Assuming ETH, there is no $f(k) \cdot n^{o(k)}$ -time algorithm for the DLCDC problem for any computable function f .

Proof. We reduce from k' -MULTICOLORED INDEPENDENT SET on a graph G' . The controlled agents in G are in 1-to-1 correspondence with the independent set in G' . The φ_1 ensures that no two incident vertices can be chosen, and φ_2 ensures that at most one vertex is chosen in each color class.

Let the vertices of G contain a vertex-vertices u for every $u \in V(G')$ and an edge-vertices e for every $e \in E(G')$, and two auxiliary vertices s and t . Let C be all the edge-vertices and t , and let D be the destructive target D , i.e.,

$$V(G) = V(G') \cup E(G') \cup \{s, t\}, \quad C = E(G') \cup \{t\}, \quad D = \{t\}.$$

Note that the vertices $E(G')$, s , and t cannot be controlled, as that would make satisfying φ_1 impossible. φ_1 shall have only one starting vertex s , and lead edges from s to vertices $V(G')$, and then to the respective incident edges in G' , i.e.,

$$E(G_{\varphi_1}) = \{(s, s), (s, t)\} \cup \{(s, v) \mid v \in V(G')\} \cup \{(v, e) \mid v \in e \wedge e \in E(G')\},$$

see Figure 5.4. Two vertices which are incident in G' cannot be chosen to be controlled as that would make the respective edge in $E(G')$ an unsatisfiable vertex in C .

$E(G_{\varphi_2})$ contains (s, s) and a directed path from s to t going through all same-colored vertices of G' (in any order) for every color in G' . As s nor t cannot be controlled, the solution is forced to control at least one vertex from each of these paths.

Setting the budget $k = k'$ together with φ_2 ensures that exactly one vertex in each color class is controlled, and φ_1 ensures it is an independent set.

Let f be any computable function. It is known, assuming ETH, that there is no $f(k) \cdot n^{o(k)}$ -time algorithm for the k' -MULTICOLORED INDEPENDENT SET problem [35]. As we set $k = k'$ in the construction, it follows that algorithm solving DLCDC in time $f(k) \cdot n^{o(k)}$ leads to the algorithm solving k' -MULTICOLORED INDEPENDENT SET in the same time complexity, which is contradiction. \square

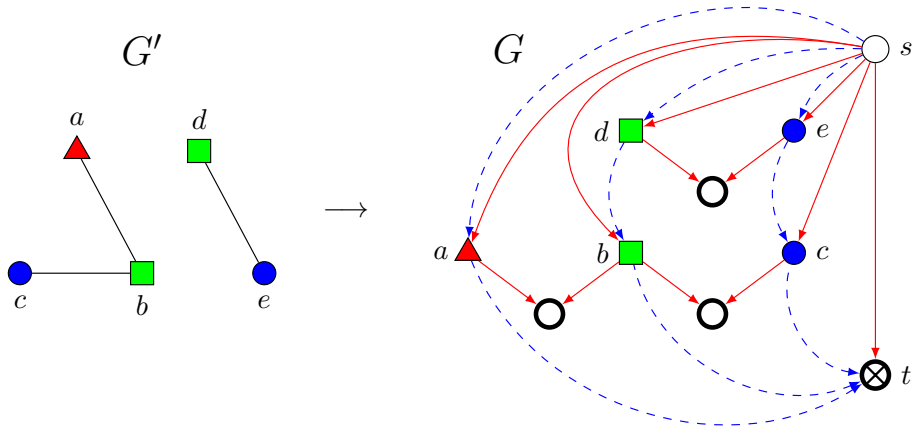


Figure 5.4: An illustration of $W[1]$ -hardness reduction from k -MULTICOLORED INDEPENDENT SET to DLCDC group selection. Colors over vertices $\{a, b, c, d, e\}$ are represented with colors and shapes; thick-bordered vertices in G are in C ; the vertex t marked with a cross belongs also to D ; full-red edges belong to φ_1 and dashed-blue edges to φ_2 .

The previous result gives reasonably tight lower bound assuming ETH compared to the naive XP algorithm. What follows are $W[1]$ -hardness bounds that complete the complexity picture for DLCDC.

Lemma 5.4.9. DLCDC parametrized by the parameter k is $W[1]$ -hard, even if $|C|$ is a fixed constant.

Proof. We shall reduce from GRID TILING WITH \leq , recall notation from Definition 5.2.6. Let us create a graph G where for each $s_{i,j} \in S_{i,j}$ we have an *entry gadget*. Each entry gadget consists of four vertices which we denote as: input vertical v_{in} , output vertical v_{out} , input horizontal h_{in} , and output horizontal h_{out} . For an entry (a, b) we say that both vertical vertices represent a and both horizontal vertices represent b . There will be one global starting vertex s with self-loop in φ_1 and φ_2 . For each tile $S_{i,j}$ we have a set of 5 vertices $d_{i,j}$ which belong to the destructive target D . Also, for each tile $S_{i,j}$ we have two vertices – a vertices vertical sink $v_{i,j}$ and a horizontal sink $h_{i,j}$.

For each tile $S_{i,j}$ let (e_1, \dots, e_m) be an arbitrary linear order over its entry gadgets. Let φ_2 contain a serial composition $(s, e_1, \dots, e_m, d_{i,j})$. We set the budget to $4k^4$ which forces us to remove all vertices of exactly one entry gadget in each tile, otherwise there would be a path from s to D in φ_2 .

Now, let us describe how to ensure the inequality conditions of the chosen entry gadgets with some intuition. We shall build an *inequality gadget* for each pair of adjacent tiles. We then concatenate them in a way that there is one long path – first it passes every vertically adjacent tiles, then every horizontally adjacent tiles, and it ends up in the only vertex in C . As there is only one source and one target for φ_1 we must have an open path through

every gadget present for every pair of adjacent tiles. The gadget ensures inequality of the chosen (removed) entries as dictated by the conditions in GRID TILING WITH \leq .

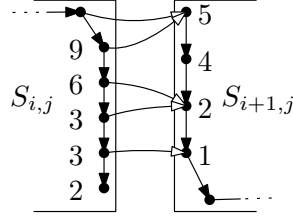


Figure 5.5: Example of the inequality gadget for the GRID TILING WITH \leq reduction.

The inequality gadget consists of two oriented paths one from each tile (e.g. tiles $S_{i,j}$ and $S_{i+1,j}$). The paths go through respective vertices in order which represents non-increasing numbers in the entries. As the global path connects to the first vertex of the first path, continues through the gadget, and leaves through the ending vertex of the second path. To be able to pass, the removed entries of $S_{i,j}$ and $S_{i+1,j}$ must not constitute a cut in the gadget. For each vertex of the first path we connect its predecessor to a vertex which represents a strictly smaller number in the second path. By this, we can pass through if and only if the second chosen entry is at least the value of the first chosen entry. See Figure 5.5 for an example of the inequality gadget.

Formally, the construction goes like this. For each pair of adjacent tiles let us focus on output vertical vertices of entry gadgets of $S_{i,j}$ and on input vertical vertices of entry gadgets of $S_{i+1,j}$. (Similarly, for the *horizontal case* we focus on output and input horizontal vertices of entry gadgets for $S_{i,j}$ and $S_{i,j+1}$, respectively.) Let us label the respective output vertices of $S_{i,j}$ by O and input vertices of $S_{i+1,j}$ by Y . Let O^\uparrow and Y^\uparrow label an ordering of O and Y such that the values represented by respective vertices are non-decreasing. For each pair adjacent tiles let φ_1 contain the following edges.

- $(o_{\ell+1}, o_\ell)$ for $o_\ell \in O^\uparrow$,
- $(y_{\ell+1}, y_\ell)$ for $y_\ell \in Y^\uparrow$,
- $(o_{\ell-1}, y_{<o_\ell})$ for each $o_\ell \in O^\uparrow$,
- $(v_{i,j}, o_m)$ and $(y_0, v_{i+1,j})$ (for horizontal case add $(h_{i,j}, o_m)$ and $(y_0, h_{i,j+1})$ instead),

where o_m is the last element of O^\uparrow , and $y_{<o_i}$ is the maximal element of Y^\uparrow which is smaller than o_i . Moreover, we add edges $(s, v_{1,1})$, $(v_{k,k}, h_{1,1})$ and edges $(v_{i,k}, v_{i+1,1})$ for $i \in \{1, \dots, k\}$ and $(v_{k,j}, v_{1,j+1})$ for $j \in \{1, \dots, k\}$. We set $C = \{h_{k,k}\}$.

We showed how to ensure that exactly one entry is chosen for each tile and that they comply to the inequalities of GRID TILING WITH \leq . The chosen entries constitute a solution to GRID TILING WITH \leq . \square

Lemma 5.4.10. DLCD parametrized by the parameter $|C|$ is W[1]-hard, even if \mathcal{T} is a fixed constant. Also, DLCD parametrized by the parameter \mathcal{T} is W[1]-hard, even if $|C|$ is a fixed constant.

Proof. We shall reduce from GRID TILING. Construction is similar to Lemma 5.4.9. Let us create a graph G where for each $s_{i,j} \in S_{i,j}$ we have an *entry gadget*. There will be one global starting vertex s with a self-loop in φ_1 and φ_2 . For each $S_{i,j}$ we have a special vertex $c_{i,j}$ which belongs to both the constructive target C and the destructive target D . Each entry gadget consists of four vertices which we denote as: input vertical v_{in} , output vertical v_{out} , input horizontal h_{in} , and output horizontal h_{out} . Each entry gadget is connected in φ_1 with a path $(s, v_{\text{in}}, v_{\text{out}}, h_{\text{in}}, h_{\text{out}}, c_{i,j})$ and is connected in φ_2 by (s, v_{out}) , (s, h_{out}) , $(v_{\text{in}}, c_{i,j})$, and $(h_{\text{in}}, c_{i,j})$.

Let us have $s_{i,j} \in S_{i,j}$ and $s_{i+1,j} \in S_{i+1,j}$ with respective entry gadgets consisting of $v_{\text{in}}, v_{\text{out}}, h_{\text{in}}, h_{\text{out}}$ and $v'_{\text{in}}, v'_{\text{out}}, h'_{\text{in}}, h'_{\text{out}}$, respectively. Let $(a, b) = s_{i,j}$ and $(a', b') = s_{i+1,j}$ if $a \neq a'$ then we connect vertices $(v_{\text{out}}, v'_{\text{in}})$ in φ_2 . Similarly, for pair of tiles $S_{i,j}$ and $S_{i,j+1}$, however, in this case if $b \neq b'$ we connect $(h_{\text{out}}, h'_{\text{in}})$ in φ_2 .

We set the budget to be $|V(G)|$ (it does not restrict the solution in any way). Note that in each tile $S_{i,j}$ in at least one entry gadget no vertex is deleted because $c_{i,j}$ must be reachable in φ_1 – the untouched entry gadgets represent the chosen entries in GRID TILING. Moreover, if we choose entry gadgets of neighboring tiles $S_{i,j}$ and $S_{i+1,j}$ that represent entries which cannot form a solution of GRID TILING (as $a \neq a'$) then (and only then) we did not remove vertices on a path $(s, v_{\text{out}}, v'_{\text{in}}, c_{i+1,j})$ in φ_2 , so a vertex in D learned the secret; hence this may not be a solution of the instance (and similarly for $S_{i,j}$ and $S_{i,j+1}$). All paths in φ_1 contain at most 5 edges and paths in φ_2 contain at most 3 edges, so \mathcal{T} is bounded by a constant. Number of vertices in D is reduced via Observation 5.4.12.

This gave us $W[1]$ -hardness for parameter $|C|$ even if \mathcal{T} is a constant. To reach parameter \mathcal{T} with a constant $|C|$ we simply remove all vertices $c_{i,j}$ from C and add one global sink t to C . We concatenate vertices $c_{i,j}$ so that as each tile is passed by the signal in φ_1 it then goes from $c_{i,j}$ to the gadgets of $S_{i+1,j}$, and so on. More precisely, the four vertices of entries are connected in φ_2 by $(c_{i',j'}, v_{\text{out}})$, $(c_{i',j'}, h_{\text{out}})$, $(v_{\text{in}}, c_{i,j})$, and $(h_{\text{in}}, c_{i,j})$, where $c_{i',j'}$ signifies the special vertex of the previously processed tile. Once all the tiles in both directions are passed by the signal, the last $c_{i,j}$ connects to t in φ_1 . As t must be reached there must be at least one non-cut path through all the gadgets, which gives us exactly the same solution as the construction above. This traded off size of the constructive set $|C| = 1$ for maximum length of the process $\mathcal{T} = 5k^2$. \square

5.4.3 Agent addition: Constructive-Destructive

Notation 5.4.11. By ALCD we denote $(\oplus, \kappa, \delta, \varrho^{\text{LSR}})$ -GROUP IDENTIFICATION PROBLEM. For the rest of this section, we set aliases A_1 as C , and A_2 as D .

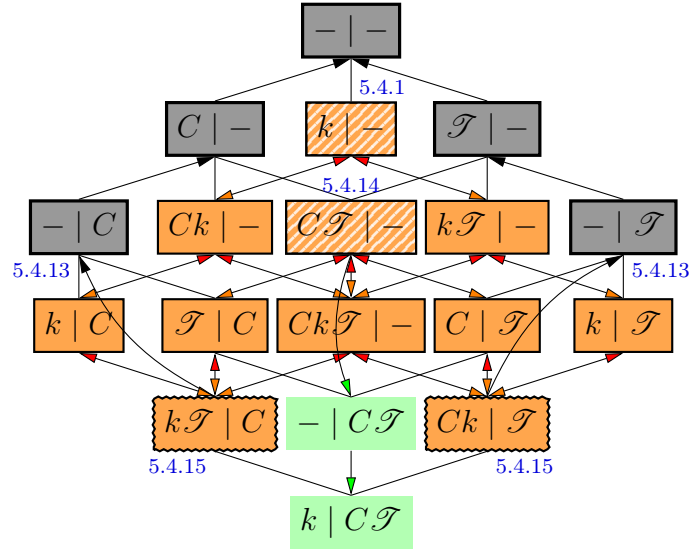
The ALCD problem is in some sense “dual” to the DLCD problem. The main difference is that majority of the agents are latent at the beginning and that in the construction we swap parallel and serial composition. See the list of results for ALCD in Table 5.3 and its complexity picture in Figure 5.6.

First, we mirror the Observation 5.4.3 about the bounded size of the set D .

Observation 5.4.12. In every instance of ALCD one can reduce the size of D to 1.

C	k	\mathcal{T}	Result	Shown by
const	input	input	NP-complete	Theorem 5.4.13
input	input	const	NP-complete	Theorem 5.4.13
param	input	param	XP	Lemma 5.4.14
input	param	input	XP	Observation 5.4.1
param	param	const	W[1]-hard	Lemma 5.4.15
const	param	param	W[1]-hard	Lemma 5.4.15

Table 5.3: Results for ADD AGENT LSR CONSTRUCTIVE-DESTRUCTIVE GIP


 Figure 5.6: Hasse diagram for results in the ADD AGENT LSR CONSTRUCTIVE-DESTRUCTIVE GIP for cases where k is not a constant. Notation is the same as in Figure 5.2.

Proof. Create a new auxiliary non-latent sink vertex t' and create edges $\{(d, t') \mid d \in D\}$ in φ_2 , set $D' = \{t'\}$. There is no way to get rid of the t' vertex. The solution reaches any vertex in D in the old instance if and only if it reaches D' in the new instance. \square

Again, we start by establishing that the problem is NP-complete.

Theorem 5.4.13. The ALCD problem is NP-complete even if the size of the constructive set $|C|$, or the number of rounds \mathcal{T} is a fixed constant.

Proof. To prove NP-completeness alone, the construction used in the proof of Theorem 5.4.4 can be used as well. It consists of serial and parallel node compositions, and it can be easily changed to a construction which works for ALCD in the following way.

- Put all vertices except source and sink to the latent set,
- swap each use of serial node composition for parallel one, and vice-versa,

- swap C with D .

On the one hand, this way each clause becomes a triplet of parallel vertices. All clauses are put in serial composition in φ_1 , and a solution must control at least one agent in each clause to form a path from s to t . On the other hand, the literals form a $(s, \ell_i, -\ell_i, t)$ serial composition, meaning that choosing a positive and negative valuation of the same variable would create a path from s to t in φ_2 , which is forbidden.

We see that the construction uses $C = D = \{t\}$ so when these sets are of constant size the problem is still NP-hard. Similarly to the proof of Theorem 5.4.4, we can change the construction by creating a separate sink in C for every clause which implies that for a constant \mathcal{T} the problem is NP-hard. \square

And finally, minor change of the proof of Lemma 5.4.6 gives us the following XP algorithm.

Lemma 5.4.14. The ALCD problem parameterized by the combined parameter $|C|$ and number of rounds \mathcal{T} can be solved in $n^{\mathcal{O}(\mathcal{T} \cdot |C|)}$ time.

Proof. This approach is somewhat similar to the proof of Lemma 5.4.6. The algorithm guesses the paths leading from sources to vertices of C in φ_1 . Now it chooses to control all latent agents that lie on the chosen paths and checks that there are at most k of them. Then it checks that no vertex in D is reached by the standard BFS algorithm. There are at most $n^{\mathcal{T}}$ paths of length \mathcal{T} in G , and we choose $|C|$ of them which makes $(n^{\mathcal{T}})^{|C|}$ possible combinations in total. BFS algorithm is polynomial, so we have total $n^{\mathcal{O}(\mathcal{T} \cdot |C|)}$ running time. \square

Lemma 5.4.15. ALCD parameterized by the combined parameter k and $|C|$ is W[1]-hard, even if \mathcal{T} is a fixed constant. And ALCD parameterized by the combined parameter k and \mathcal{T} is W[1]-hard, even if $|C|$ is a fixed constant.

Proof. This construction is very similar to Lemma 5.4.10, however, we need to change the budget to accommodate the change of settings. Let us give the argument in full so we do not omit any necessary detail. We reduce from GRID TILING. Let us create a graph G where for each $s_{i,j} \in S_{i,j}$ we have an *entry gadget*. There will be one global starting vertex s with a self-loop in φ_1 and φ_2 . For each $S_{i,j}$ we have a special vertex $c_{i,j}$ which belongs to both the constructive target C and the destructive target D . Each entry gadget consists of four latent vertices which we denote as: input vertical v_{in} , output vertical v_{out} , input horizontal h_{in} , and output horizontal h_{out} . Each entry gadget is connected in φ_1 with a path $(s, v_{\text{in}}, v_{\text{out}}, h_{\text{in}}, h_{\text{out}}, c_{i,j})$ and is connected in φ_2 by (s, v_{out}) , (s, h_{out}) , $(v_{\text{in}}, c_{i,j})$, and $(h_{\text{in}}, c_{i,j})$.

Let us have $s_{i,j} \in S_{i,j}$ and $s_{i+1,j} \in S_{i+1,j}$ with respective entry gadgets consisting of $v_{\text{in}}, v_{\text{out}}, h_{\text{in}}, h_{\text{out}}$ and $v'_{\text{in}}, v'_{\text{out}}, h'_{\text{in}}, h'_{\text{out}}$, respectively. Let $(a, b) = s_{i,j}$ and $(a', b') = s_{i+1,j}$ if $a \neq a'$ then we connect vertices $(v_{\text{out}}, v'_{\text{in}})$ in φ_2 . Similarly, for pair of tiles $S_{i,j}$ and $S_{i,j+1}$, however, in this case if $b \neq b'$ we connect $(h_{\text{out}}, h'_{\text{in}})$ in φ_2 .

We set the budget to be $4k^2$. Note that in each tile $S_{i,j}$ at least one entry gadget needs to be activated because $c_{i,j}$ must be reachable in φ_1 – the activated entry gadgets represent the chosen entries in GRID TILING. The budget is tight so we may activate only one entry gadget in each tile. Moreover, if we choose entry gadgets of neighboring tiles $S_{i,j}$ and $S_{i+1,j}$ that represent entries which cannot form a solution of GRID TILING (as $a \neq a'$) then (and only then) we activated vertices on a path $(s, v_{\text{out}}, v'_{\text{in}}, c_{i+1,j})$ in φ_2 , so a vertex in D learned the secret; hence this may not be a solution of the instance (and similarly for $S_{i,j}$ and $S_{i,j+1}$). All paths in φ_1 contain at most 5 edges and paths in φ_2 contain at most 3 edges, so \mathcal{T} is bounded by a constant. Number of vertices in D is reduced via Observation 5.4.12.

This gave us $W[1]$ -hardness for combined parameter k and $|C|$ even if \mathcal{T} is a constant.

To reach parameter \mathcal{T} with a constant $|C|$ we simply remove all vertices $c_{i,j}$ from C and add one global sink t to C . We concatenate vertices $c_{i,j}$ so that as each tile is passed by the signal in φ_1 it then goes from $c_{i,j}$ to the gadgets of $S_{i+1,j}$, and so on. More precisely, the four vertices of entries are connected in φ_2 by $(c_{i',j'}, v_{\text{out}})$, $(c_{i',j'}, h_{\text{out}})$, $(v_{\text{in}}, c_{i,j})$, and $(h_{\text{in}}, c_{i,j})$, where $c_{i',j'}$ signifies the special vertex of the previously processed tile. Once all the tiles in both directions are passed by the signal, the last $c_{i,j}$ connects to t in φ_1 . As t must be reached there must be at least one non-cut path through all the gadgets, which gives us exactly the same solution as the construction above. This traded off size of the constructive set $|C| = 1$ for maximum length of the process $\mathcal{T} = 5k^2$. \square

5.4.4 Agent deletion: Destructive-Destructive

Notation 5.4.16. By DLDD we denote $(\ominus, \delta, \delta, \rho^{\text{LSR}})$ -GROUP IDENTIFICATION PROBLEM. For the rest of this section, we set aliases A_1 as D_1 , and A_2 as D_2 .

Observation 5.4.17. Let \mathcal{I} be an instance of the DLDD problem. If $k \geq |D_1| + |D_2|$, then \mathcal{I} is a *yes*-instance.

Proof. If $k \geq |D_1| + |D_2|$, then we can directly delete all agents from $D_1 \cup D_2$ which solves the instance. \square

This directly gives an XP algorithm for the DLDD problem parameterized by $|D_1|$ and $|D_2|$.

Corollary 5.4.18. The DLDD problem is in XP parameterized by the combined parameter $|D_1|$ and $|D_2|$.

Proof. By Observation 5.4.17 the problem is trivial if $k \geq |D_1| + |D_2|$. Assuming $k \leq |D_1| + |D_2|$, the solution size is bounded the combined parameter and we can run the algorithm from Observation 5.4.1. \square

Observe that the complexity results do not change when we swap the roles of D_1 and D_2 , however, as they might be mutually incomparable we prove only one variant but we present both variants in the results overview of Table 5.4. Also, see Figure 5.7 for the complexity picture.

D_1	D_2	k	\mathcal{T}	Result	Shown by
const	input	input	const	NP-complete	Theorem 5.4.19
input	const	input	const	NP-complete	(Theorem 5.4.19)
param	param	input	input	XP	Corollary 5.4.18
input	input	param	input	XP	Observation 5.4.1
const	param	param	input	W[1]-hard	Lemma 5.4.20
param	const	param	input	W[1]-hard	(Lemma 5.4.20)
input	input	param	param	FPT	Lemma 5.4.21
param	param	input	param	FPT	Corollary 5.4.22

Table 5.4: Results summary for the DELETE AGENT LSR DESTRUCTIVE-DESTRUCTIVE GIP.

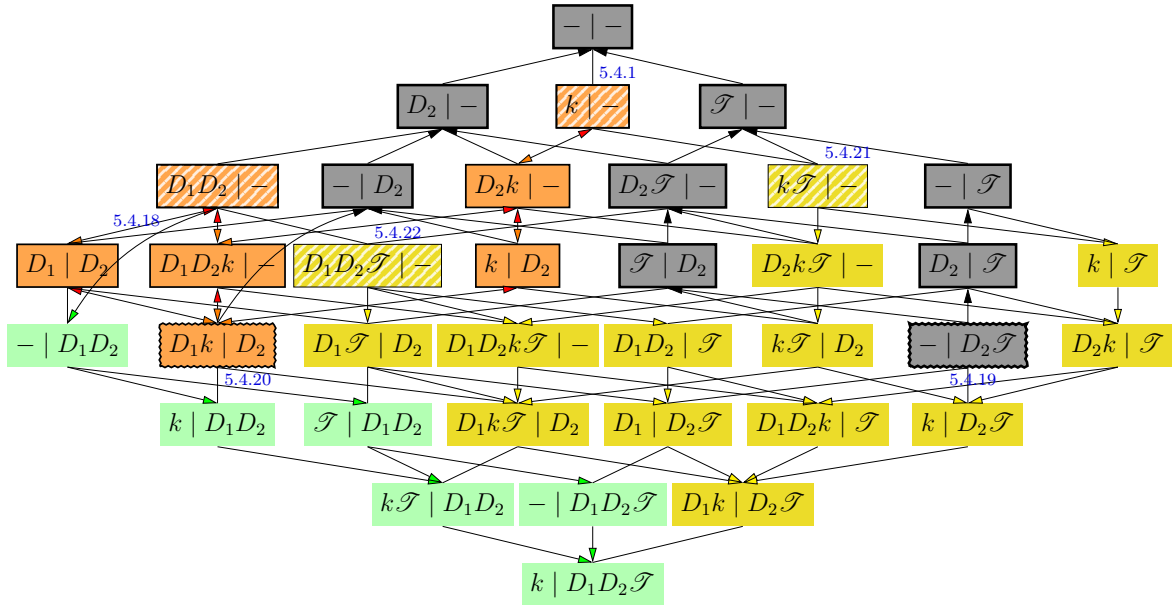


Figure 5.7: Hasse diagram for results in the DELETE AGENT LSR DESTRUCTIVE-DESTRUCTIVE GIP for cases where k is not a constant, and with D_1D_2 symmetries shown only once. Notation is the same as in Figure 5.2.

Theorem 5.4.19. DLDD is NP-complete even if both $|D_2|$ and the number of rounds \mathcal{T} are fixed constants. Unless ETH fails, there is no algorithm solving DLDD in $f(|D_2| + \mathcal{T})^{o(n+m)} \cdot (n + m)^{f(|D_2| + \mathcal{T})}$ time for any computable function f .

Proof. We present reduction from 3-SAT to DLDD which shows its NP-hardness. Each literal occurrence in \mathcal{F} is represented by a vertex in the qualification graph G . Our construction ensures that whenever a literal evaluates to true the respective agent $v \in V(G)$ is controlled, and vice-versa.

We construct two gadgets. First ensures that either all positive literals ℓ_j or all negative literals $\neg\ell_j$ are chosen for every variable x_j . Second ensures that at least one literal is chosen

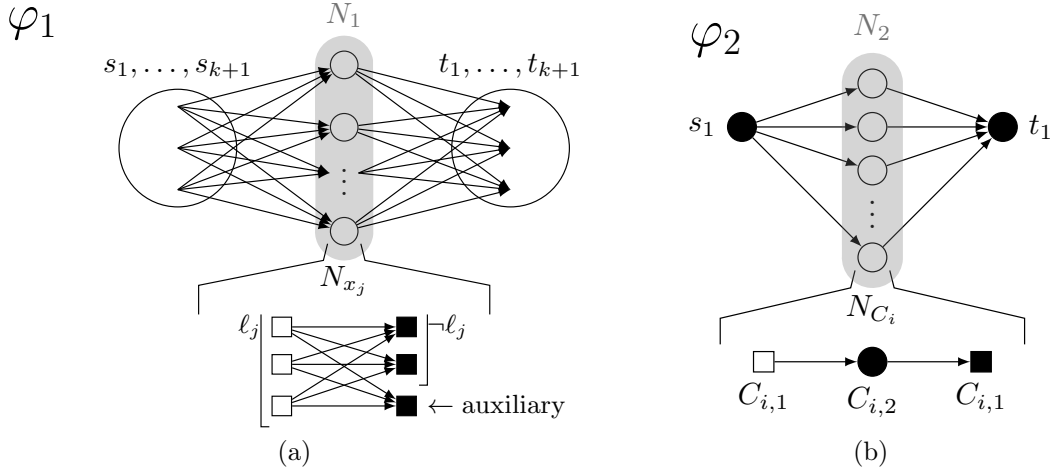


Figure 5.8: Node compositions representing construction of G_{φ_1} edges in 5.8a and G_{φ_2} edges in 5.8b, illustrating NP-hardness reduction for DLDD group selection.

in every clause.

First, let $L_j = \ell_j$ and $\neg L_j = \neg \ell_j$. Add $||\ell_j| - |\neg \ell_j||$ new auxiliary vertices to the smaller set out of L_j and $\neg L_j$ to equalize their size. The first gadget is a serial composition $N_{x_j} = L_j \dot{\cup} \neg L_j$. By Observation 5.3.5 the first gadget has cut-weight equal to the minimum cut-weight of individual components, which is $\max\{\ell_j, \neg \ell_j\}$. The second gadget N_{C_i} consists of a directed path over literal vertices of a clause, formally:

$$N_{C_i} = \left((\{C_{i,1}, C_{i,2}, C_{i,3}\}, \{(C_{i,1}, C_{i,2}), (C_{i,2}, C_{i,3})\}), \{C_{i,1}\}, \{C_{i,3}\} \right).$$

We construct the graph G using the gadgets. $V(G)$ has a vertex for every literal occurrence in \mathcal{F} and several auxiliary source and sink vertices. As the solution may simply remove all sources or sinks we need the sets to be of size at least $k + 1$ (bigger than the budget whose value is determined later), so set the global sources $S = \{s_1, \dots, s_{k+1}\}$ and global sinks $T = \{t_1, \dots, t_{k+1}\}$. See Figure 5.8 for the construction.

G_{φ_1} contains the first gadget node N_{x_j} for every variable x_j . Let N_1 be parallel composition of N_{x_j} for all variables x_j . We add node-edges (S, N_1) and (N_1, T) . Add loops to all vertices of S so that they are sources for φ_1 and set $D_1 = T$, see Figure 5.8a.

G_{φ_2} contains the second gadget node N_{C_i} for every clause C_i . Let N_2 be a parallel composition of all N_{C_i} nodes and add node-edges (S, N_2) and (N_2, T) . Add loop to all sources S in φ_2 and set $D_2 = \{t_1\}$, see Figure 5.8b.

We set the budget $k = \sum_j \max\{|\ell_j|, |\neg \ell_j|\}$, so that in every first gadget the solution have to control either the whole L_j or $\neg L_j$ and has no spare budget. Each second gadget must be cut to ensure satisfaction of φ_2 meaning that in each clause node N_{C_i} there is one cut vertex representing the literal which satisfied it. Note that here we can set $D_2 = \{t_1\}$ as a sink as the first gadget forbids the solution from choosing vertices other than ℓ_j or $\neg \ell_j$.

As checking whether D_1 or D_2 is reachable from respective start vertices for a given solution is trivial, it follows that DLDD is NP-complete.

The output of the presented reduction is an instance with $|G| \in \mathcal{O}(m)$, $|D_2| = 1$, and $\mathcal{T} = 4$. Hence, algorithm for DLDD running in time $f(|D_2| + \mathcal{T})^{o(n+m)} \cdot (n+m)^{f(|D_2|+\mathcal{T})}$ for some computable function f implies existence of a $2^{\mathcal{O}(m)}$ -time algorithm for 3-SAT, which contradicts ETH. \square

Lemma 5.4.20. The DLDD problem is W[1]-hard when parameterized by the combined parameter $|D_2|$ and solution size k , even if $|D_1|$ is a fixed constant. Assuming ETH, there is no $f(k + |D_2|) \cdot n^{o((k+|D_2|)/\log(k+|D_2|))}$ -time algorithm for DLDD.

Proof. We present a reduction from PSI over G' colored by vertices of H' , see Definition 5.2.4 for notation. First, we preprocess G' by removing all vertices u which do not have neighbors of colors that occur as vertices in H' in the neighborhood of $c(u)$ (such vertices cannot be in a solution). Let us have two vertices in the qualification graph G for both orientations of every edge in G' , i.e., put $\{(u, v), (v, u) \mid \{u, v\} \in E(G')\}$ into $V(G)$. The controlled agents correspond to the edges of the PSI solution. We need to ensure the following two conditions.

1. If we control (u, v) in G , then we must control (v, u) as well, and
2. if we control (u, v) , then u is the only vertex of color $c(u)$ among the vertices inside the chosen edges.

For both these conditions we introduce a gadget securing it. Let us have (u, v) and (v, u) as a node $N_{\{u,v\}}$. We create the first gadget N_{c_1, c_2} by a serial composition of all nodes $N_{\{u,v\}}$ such that $\{u, v\} \in E(G')$ and $c(u) = c_1$ and $c(v) = c_2$. Note that the first gadget has cut-weight 2 by Observation 5.3.5. We use this gadget to ensure the first condition later.

Now, we build the second gadget gradually from the smallest components. Let us have a node N_{u, c_1} which contains an oriented path over all vertices representing edges from u in G' ending in vertices with color c_1 (in any order), i.e., a path over $\{(u, v) \mid \{u, v\} \in E(G') \wedge c(v) = c_1\}$. Let the $N_{u, Q}$ be a node which creates a parallel composition of nodes $\{N_{u, c_1} \mid c_1 \in Q\}$ where Q is a set of colors. Note that the vertex cut of $N_{u, Q}$ must pick one vertex from each N_{u, c_1} , and by Observation 5.3.5 has cut-weight $|Q|$. The second gadget $N_{c_1, Q}$ is made by making a serial composition of $N_{u, Q}$ nodes for all u where $c(u) = c_1$. The second gadget still has cut-weight $|Q|$ as the cut-weight of all components is the same.

The edges of G are constructed as follows. Set the global sources $S = \{s_1, \dots, s_{k+1}\}$ and global sinks $K = \{t_1, \dots, t_{k+1}\}$ (the value of the budget k is determined later). G_{φ_1} contains a node N_1 which is a parallel composition of the first gadgets for every pair of colors $\{c_1, c_2\} \in E(H')$ and there are additional node-edges (S, N_1) and (N_1, K) . G_{φ_2} contains a node N_2 which is a parallel composition of the second gadgets $N_{c_1, N(c_1)}$ for every $c_1 \in V(H')$ and $N(c_1)$ being neighborhood of c_1 in H' (not to be confused with nodes), and also contains node-edges (S, N_2) and (N_2, K) . See Figure 5.9 for an illustration of the reduction.

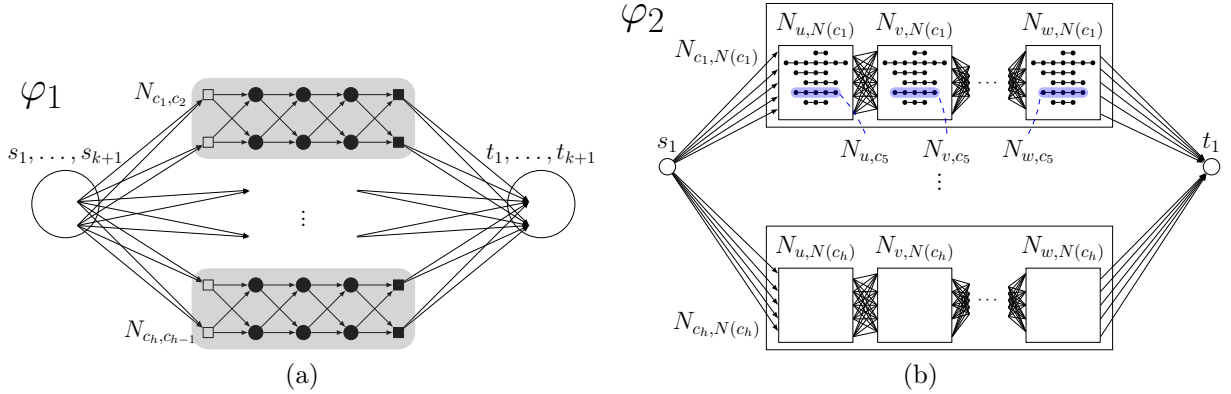


Figure 5.9: Node compositions representing construction of edges of G_{φ_1} in 5.9a and G_{φ_2} in 5.9b used in $W[1]$ -hardness proof for DLDD; $h = |V(H')|$.

The first gadget must be cut by at least 2 vertices using a pair of vertices which represent edges in the opposite direction. The first gadgets are put in parallel so that the solution must cut each of them. The budget $k = 2 \cdot |E(H')|$, i.e., twice the number of edges in H' , which is the minimal number of vertices needed to cut the first gadgets by Observation 5.3.5. The second gadgets represent choice of a vertex of a given color. As the second gadgets are all put in parallel, and each has fixed cut-weight no matter where it is cut, the total cut-weight over the construction is $\sum_{c_1 \in H'} |N(c_1)|$ which equals $2 \cdot |E(H')|$ (by hand-shake lemma) so it is tight with the budget, and other vertices cannot be cut.

The sets D_1 and D_2 need to cover t_1, \dots, t_{k+1} so at least one of these must be parameterized, however, one may be of constant size. The budget k is dependent on the size of the parameter $|H'|$, and the number of stabilization steps \mathcal{T} is bounded only by the input size.

It is known that algorithm for PSI running in time $f(|H'|) \cdot n^{(|H'|/\log |H'|)}$ contradicts ETH [88]. In the proposed parameterized reduction, we have $k = 2|H'|$ and $|D_2| = k$. Hence, algorithm for DLDD running in $f(k + |D_2|) \cdot n^{o((k+|D_2|)/\log(k+|D_2|))}$ time contradicts ETH. \square

Lemma 5.4.21. The DLDD problem is fixed-parameter tractable parameterized by the combined parameter k and \mathcal{T} .

Proof. Let the procedure be as follows. Let us run the Breadth First Search (BFS) algorithm on φ_1 and φ_2 to find some path $P = (p_1, \dots, p_m)$ from source to target in $\mathcal{O}(n)$ time, where n is the total size of the graph. If no path P exists then the procedure returns \emptyset as a solution. If the budget is zero then return that the solution does not exist. The path P has length at most \mathcal{T} and at least one of its vertices must be chosen to the solution, otherwise the target p_m would learn the secret from source p_1 . Let us choose a vertex $u \in P$ (we branch here for every choice of u) to be included in the solution. Now, run this procedure again for DLDD where u is removed (marked not to be traversed by the BFS) and budget is reduced by 1. If the procedure finds a solution then add u to it and return it.

The above procedure branches on at most \mathcal{T} vertices of P . The biggest recursion depth is k as the budget gets decreased by one with each sub-procedure, and for $k = 0$ the procedure does not call any sub-procedure. The number of procedure calls is at most $\sum_{i=0}^k \mathcal{T}^i = \frac{\mathcal{T}^{k+1}}{\mathcal{T}-1}$ and each procedure calls BFS twice and several constant-time operations. The total time is $\mathcal{O}(\mathcal{T}^{k+1}n)$ which is FPT in combined parameter k and \mathcal{T} . \square

By combining Observation 5.4.17 and Lemma 5.4.21, we obtain the following algorithm for slightly different setting.

Corollary 5.4.22. The DLDD problem is fixed-parameter tractable parameterized by the combined parameter $|D_1|$, $|D_2|$, and number of rounds \mathcal{T} .

Proof. By Observation 5.4.17 we have $k \leq |D_1| + |D_2|$. Thus, the solution size is bounded and we can run algorithm from Lemma 5.4.21 which completes the proof. \square

5.4.5 Agent addition: Constructive-Constructive

Notation 5.4.23. By ALCC we denote $(\oplus, \kappa, \kappa, \varrho^{\text{LSR}})$ -GROUP IDENTIFICATION PROBLEM. For the rest of this section, we set aliases A_1 as C_1 , and A_2 as C_2 .

Similarly to DLDD, C_1 and C_2 can be swapped and the result stays the same. For the list of results see Table 5.5, and for a complexity diagram see Figure 5.10

C_1	C_2	k	\mathcal{T}	Result	Shown by
param	input	input	param	XP	Lemma 5.4.26
input	param	input	param	XP	Lemma 5.4.26
input	input	param	input	XP	Observation 5.4.1
param	param	param	const	W[1]-hard	Lemma 5.4.25
const	const	param	param	W[1]-hard	Lemma 5.4.24

Table 5.5: Results for ADD AGENT LSR CONSTRUCTIVE-CONSTRUCTIVE GIP

NP-hardness of ALCC variant follows from similar reduction to the one presented in the proof of Theorem 5.4.19. We do not present it explicitly because, similarly to the ALCD, we show W[1]-hardness that implies NP-hardness of ALCC.

Lemma 5.4.24. ALCC parameterized by the combined parameter the solution size k and the number of rounds \mathcal{T} is W[1]-hard even if $|C_1|$ and $|C_2|$ are fixed constants. Moreover, if ALCC can be solved in time $f(k + \mathcal{T}) \cdot n^{o((k+\mathcal{T})/\log(k+\mathcal{T}))}$, where f is an arbitrary computable function, then ETH fails.

Proof. The proof is very similar to the proof of Lemma 5.4.20. We shall not present the proof in its entirety and we rather mention the main differences from that proof instead. We do the same preprocessing step for the given PSI instance and even create the same set of vertices. However, we need only one vertex s and one t instead of the used sets, which are also the only non-latent agents. The edges are added to the graph in a similar way but

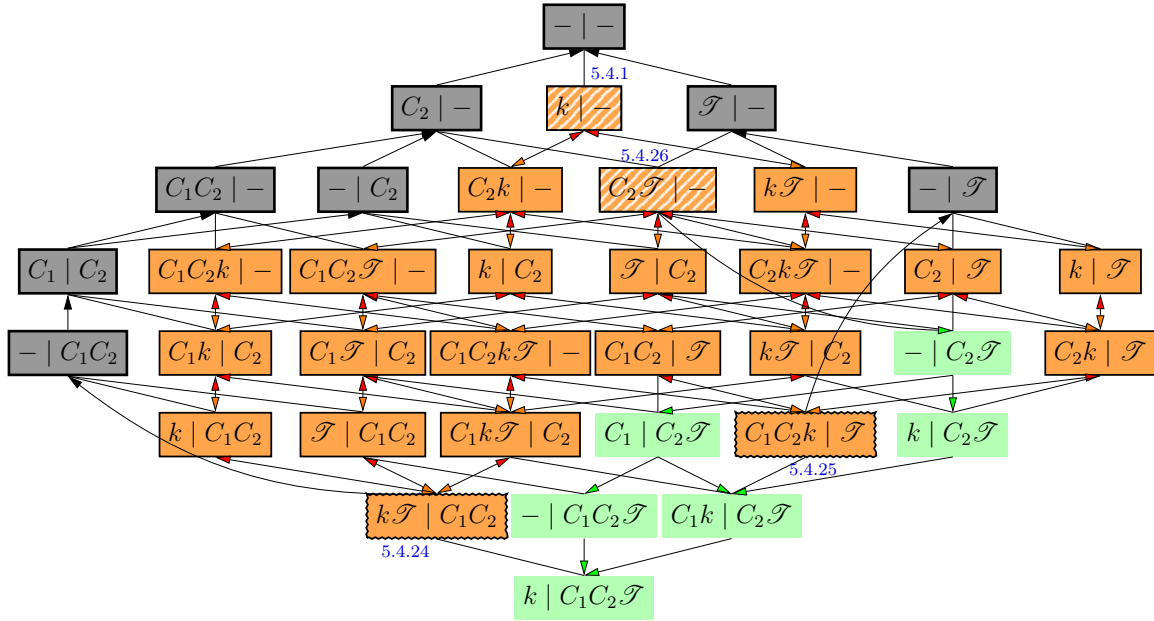


Figure 5.10: Hasse diagram for results in the ADD AGENT LSR CONSTRUCTIVE-CONSTRUCTIVE GIP for cases where k is not a constant, and with C_1C_2 symmetries shown only once. Notation is the same as in Figure 5.2.

the main difference is the following: when parallel composition was used in Lemma 5.4.20, we apply the serial composition and vice-versa. Instead of creating a cut, we are building a path over the controlled (in this case added) agents. The rest of the proof is the same and budget remains also the same. $C_1 = C_2 = \{t\}$ so their sizes are both bounded by a constant. The length of a longest oriented path is bounded by the number of active agents which is at most $2 + k$. \square

We conclude this section with $W[1]$ -hardness result and another XP algorithm, which illustrates well the computational hardness of the ALCC problem.

Lemma 5.4.25. ALCC parameterized by the combined parameter $|C_1|, |C_2|$, and solution size k is $W[1]$ -hard, even if number of rounds \mathcal{T} is a fixed constant. Moreover, if ALCC can be solved in time $f(|C_1| + |C_2| + k) \cdot n^{o((|C_1|+|C_2|+k)/\log(|C_1|+|C_2|+k))}$, where f is an arbitrary computable function, then ETH fails.

Proof. We do the reduction from PSI on a graph G' when searching for H' . The construction is similar to a reduction of PSI to STRONGLY CONNECTED STEINER SUBGRAPH [35, Theorem 13.33]. First, preprocess G' and H' by removing isolated vertices of H' and so removing all vertices of respective colors in G' .

$V(G)$ shall contain four main types of vertices:

- edge-vertices $\{u, v\} \in V(G)$ for every $\{u, v\} \in E(G')$,
- diedge-vertices $(u, v), (v, u) \in V(G)$ for every $\{u, v\} \in E(G')$,

- vertex-vertices $u \in V(G)$ for every $u \in V(G')$, and
- color-pair-vertices $(c_i, c_j) \in V(G)$ for every $c_i, c_j \in V(H')$ such that $c_i \neq c_j$.

Also add s which is the source vertex for φ_1 and φ_2 to $V(G)$. We set $C_1 = C_2 =$ all the color-pair-vertices. The vertices which are not participating in the selection, but can be controlled to participate, are all the vertex-vertices and edge-vertices.

We add several edges to φ_1 :

- (s, u) for each vertex-vertex $u \in V(G')$,
- $(u, (u, v))$ from a vertex-vertex u to each diedge-vertex from u , and
- $((u, v), (c_i, c_j))$ for every diedge-vertex u, v such that $c(u) = c_i$ and $c(v) = c_j$ (only if (c_i, c_j) color-pair-vertex exists).

We add the following edges to φ_2 :

- $(s, \{u, v\})$ for each edge-vertex $\{u, v\} \in V(G')$,
- $(\{u, v\}, (u, v))$ and $(\{u, v\}, (v, u))$ from each edge-vertex to respective diedge-vertices,
- and same as for φ_1 , $((u, v), (c_i, c_j))$ for every diedge-vertex u, v such that $c(u) = c_i$ and $c(v) = c_j$ (only if (c_i, c_j) color-pair-vertex exists).

See the vertices and edges depicted on the Figure 5.11.

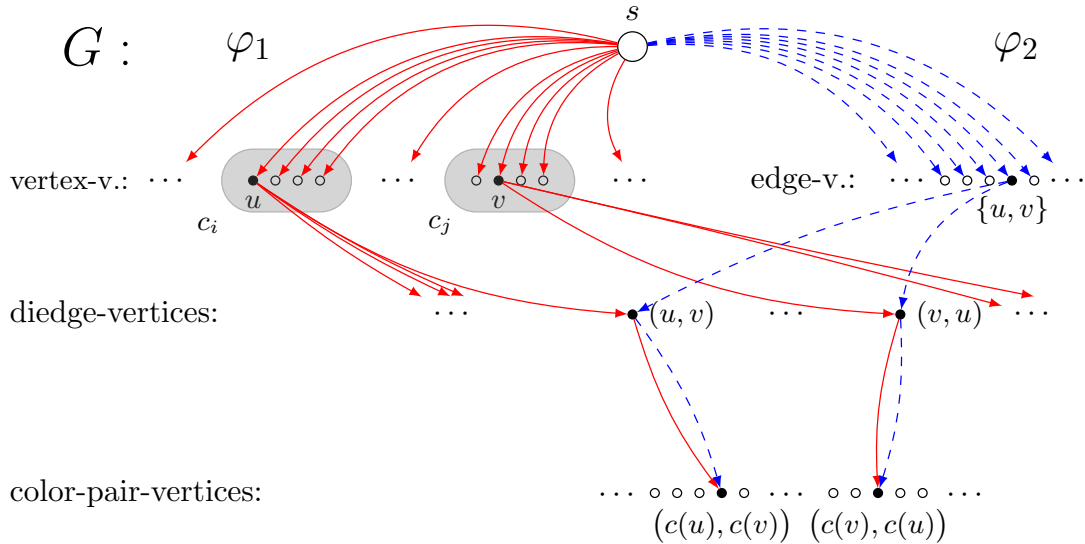


Figure 5.11: Illustration of STRONGLY CONNECTED STEINER SUBGRAPH-like $W[1]$ -hardness reduction for ALCC from PARTITIONED SUBGRAPH ISOMORPHISM problem.

Set the budget to $k = |V(H')| + 3 \cdot |E(H')|$ and we show that the solution must choose such vertices that those represent the subgraph of G' which is isomorphic to H' . First, note that diedge-vertices have out-degree exactly 1 for both φ_1 and φ_2 , meaning that we have to pick at least $|C_1|$ ($= |C_2|$) of them to satisfy the requirements; more precisely, for each tuple of colors (c_1, c_2) one diedge-vertex (u, v) where $c(u) = c_1$ and $c(v) = c_2$ must be chosen. By definition $|C_1| = 2 \cdot |E(H')|$, hence the remaining budget to choose vertex-vertices and edge-vertices is at most $|V(H')| + |E(H')|$.

We see that to satisfy φ_1 we need to select at least one vertex of each color among vertex-vertices so that respective outgoing edges are reached (note their degree in H' is at least 1 because of the preprocessing step). Similarly, for φ_2 we need to select at least $|E(H')|$ vertices of edge-vertices as their out-degree is 2 and we need to reach at least $2 \cdot |E(H')|$ diedge-vertices. We see that the budget is tight so in each group of vertices we must select exactly the prescribed number of vertices. It follows, that the set of diedge-vertices corresponds to both orientations of the selected edge-vertices, and set of vertex-vertices correspond to starting vertices of the diedge-vertices. Such selection of vertices in G describes a subgraph in G' isomorphic to H' as each satisfied color-pair-vertex corresponds to an edge-vertex and vertex-vertices which satisfied it. \square

Lemma 5.4.26. The ALCC problem is in XP parameterized by the combined parameter $|C_1|$ and number of rounds \mathcal{T} .

Proof. There are at most $n^{\mathcal{T}}$ different paths of length \mathcal{T} . We can search through all combinations of $|C_1|$ paths of length at most \mathcal{T} in time $\mathcal{O}(n^{\mathcal{T} \cdot |C_1|})$. For every combination we activate latent agents on these paths and know that all vertices in C_1 are reached in φ_1 . The remaining budget shall be used to resolve vertices in C_2 using the known polynomial-time algorithm for single constructive goal.

The solution must lie inside these combinations because every vertex in C_1 and C_2 must be satisfied by at least one path of length at most \mathcal{T} . Thus, we have algorithm running in $n^{\mathcal{O}(\mathcal{T} \cdot |C_1|)}$ time. \square

5.5 Future work

We have proposed the study of the combination of controlling two simultaneous group identification processes with constructive and destructive targets. It should be noted that even though our XP algorithms are rather simple for most of them our W[1]-hardness results suggest that one should not expected better running times as this would contradict the ETH [69]; see also [88]. It is worth mentioning that the group identification is close to committee elections, however, here the set of voters coincides with the set of alternatives and, most importantly, the size of the committee (i.e., the socially qualified agents) is not fixed in advance. In the presented work we solely use the liberal starting rule; a natural question arises since there are more rules commonly used for group identification. The most promising direction should be study of the consent starting rule, as Erdélyi et al. [47] shown that with one secret constructive/destructive targets are polynomial-time solvable.

It is not hard to see that some of our NP-hardness results should be easily transformed for this case as well. Notably this should remain the case as long as the rules used for spread of the two secrets remain the same. Thus, one should ask what if different rules are used for different secrets?

The key ingredient of our study is that the LSR rule starts a certain activation process from an initial set of agents (note that this is the case for CSR as well). We usually wait until the activation process stabilizes, however, for many processes it might be natural to limit the duration of the activation process; thus, yielding a new parameter to be studied. Last but not least, we believe a more complicated processes should be investigated in future work – namely, what if there is a certain interaction between the two secrets (e.g., each agent can only learn one secret)?

Conclusions

6.1 Contributions of the Dissertation Thesis

Contributions. In this dissertation thesis, we achieved several advances in combinatorial and algorithmic games on graphs. We managed to solve the M-ETERNAL DOMINATION on cactus graphs while providing several tools for achieving lower and upper bounds. We showed that HAT CHROMATIC NUMBER on chordal graphs is tractable by introducing FRACTIONAL HAT CHROMATIC NUMBER and by obtaining a connection to the independence polynomial of a graph. We introduced a stricter variant of the ONLINE RAMSEY NUMBER where the results must appear induced and we resolved it in an asymptotically optimal fashion for a few graph classes while giving solutions also for the original problem. We showed how secondary measures of the input influence tractability of a generalized GROUP IDENTIFICATION PROBLEM with two secrets and constructive/destructive target sets by providing a complete parameterized complexity analysis.

Attributions. The author acknowledges that this work is based on published and unpublished papers where the contribution is split between him and his coauthors. In all the works, the author played a major role in obtaining the concepts and methods. Though it is not easy to attribute contributions in collaborative work, we attempt to give an overall picture.

The investigation of M-ETERNAL DOMINATION was initiated by J. M. Kříšťan and it was later collaboratively developed with the author and under the supervision of T. Valla to gain methods and results that were presented in [A.1]. There we showed how to solve the problem for the class of Christmas cactus graphs. Further continuation of this work led to developing more refined tools which serve to show bounds and were used to solve any cactus graph. We presented these results in Chapter 2.

The work on the HAT CHROMATIC NUMBER was published in [A.2]. These investigations started on KAMAK 2019 together with Miloš Chromý, Michał Debski, Sophie Rehberg, Pavel Valtr, Michal Opler, and Pavel Dvořák. The attempt was to solve things shown Section 3.4 and several other open problems but at a later date we found a preprint

of Kokhas et al. [81] which addressed many of these issues. We continued to work on the problem with M. Opler and P. Dvořák and developed the notion of fractional variant and its connection to the independence polynomial. This led us also to generalizations of known theorems, all of which are shown in Chapter 3.

The ONLINE RAMSEY THEORY work, which we published in [A.3], stems from the original work of the author. There was a significant attempts to solve other open problems together with P. Dvořák and T. Valla, however, it led only to minor advancements which are not presented in this work. Chapter 4 shows the published results in a more polished form.

Last, Chapter 5 presents results in GROUP IDENTIFICATION and is collaborative work with D. Knop and Š. Schierreich. The problem statement, context, and direction comes from D. Knop, the results come mainly from the author and Š. Schierreich. The author analysed which settings were open during our investigation and played a crucial role in obtaining reductions and algorithms to solve them. This led to a complete picture of the parameterized complexity structure of the problem. These results were presented in the form of an extended abstract [A.4].

6.2 Future Work

We mentioned possible future directions in respective chapters, however, let us provide a brief overview of the possible future directions within the investigated problems.

ETERNAL DOMINATION. Future investigation can aim at a natural question: Does M-ETERNAL DOMINATION lie in PSPACE? Another target is to extend the graph class where the problem can be solved efficiently. That is, is there a polynomial algorithm to solve the problem on series-parallel graphs, or at least outer-planar graphs? Intractability of this problem should also be investigated, showing hardness for the most specific graph classes possible. We also attempted to solve the problem parameterized by tree-width, however, there seems to be no clear way how to do it. Parameterization via other graph parameters should be investigated, e.g., tree-depth. Another question we find quite natural is: If attacks are announced k turns in advance, how much does this influence the number of necessary guards?

HAT CHROMATIC NUMBER. We got several results by the connection to the independence polynomial, however, we would be interested whether other results can be achieved by exploiting this connection. We also think that FRACTIONAL HAT CHROMATIC NUMBER can be used to get further results which seemed previously unreachable.

ONLINE RAMSEY NUMBER. The first direction which could be developed in this area is general lower bounds. We are aware of only one [1]. Also, we think further investigations should see if “big” structures may be built asymptotically faster when some “small” helping substructure is already present.

GROUP IDENTIFICATION. There are several research directions. One, we find natural to study, is to investigate consent starting rule which was shown tractable for a single secret setting by Erdélyi et al. [47]. Our NP-hardness reductions shall be very similar for that setting, but, we wonder what will the complete complexity picture look like. One can introduce more involved process and study how it influences the complexity – usually, the more real-life the process is, the harder it becomes; this includes interaction between the secrets.

Bibliography

- [1] Grzegorz Adamski and Małgorzata Bednarska-Bzdęga. Online size Ramsey numbers: Odd cycles vs connected graphs. *arXiv*, abs/2111.14147, November 2021.
- [2] Gagan Aggarwal, Amos Fiat, Andrew V. Goldberg, Jason D. Hartline, Nicole Immorlica, and Madhu Sudan. Derandomization of auctions. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 619–625. Association for Computing Machinery, May 2005. doi:[10.1145/1060590.1060682](https://doi.org/10.1145/1060590.1060682).
- [3] Michael Albert, Richard Nowakowski, and David Wolfe. *Lessons in Play: An Introduction to Combinatorial Game Theory*. A K Peters/CRC Press, Wellesley, Mass, July 2007. doi:[10.1201/b10691](https://doi.org/10.1201/b10691).
- [4] Noga Alon, Omri Ben-Eliezer, Chong Shangguan, and Itzhak Tamo. The hat guessing number of graphs. *J. Comb. Theory, Series B*, 144:119–149, September 2020. doi:[10.1016/j.jctb.2020.01.003](https://doi.org/10.1016/j.jctb.2020.01.003).
- [5] Cristina Bazgan, Morgan Chopin, André Nichterlein, and Florian Sikora. Parametrized inapproximability of target set selection and generalizations. *Computability*, 3(2):135–145, 2014. doi:[10.3233/COM-140030](https://doi.org/10.3233/COM-140030).
- [6] József Beck. On size Ramsey number of paths, trees, and circuits. I. *Journal of Graph Theory*, 7(1):115–129, 1983.
- [7] József Beck. On size Ramsey number of paths, trees and circuits. II. In *Mathematics of Ramsey theory*, pages 34–45. Springer, 1990.
- [8] József Beck. Achievement games and the probabilistic method. *Combinatorics, Paul Erdős is Eighty*, 1:51–78, 1993.
- [9] Ruben Becker, Federico Corò, Gianlorenzo D’Angelo, and Hugo Gilbert. Balancing spreads of influence in a social network. *AAAI*, 34(01):3–10, 2020.

- URL: <https://ojs.aaai.org/index.php/AAAI/article/view/5327>, doi:10.1609/aaai.v34i01.5327.
- [10] Oren Ben-Zwi, Danny Hermelin, Daniel Lokshtanov, and Ilan Newman. An exact almost optimal algorithm for target set selection in social networks. In *EC 2009*, pages 355–362. ACM, 2009. doi:10.1145/1566374.1566424.
- [11] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays*, volume 1. A K PETERS LTD (MA), January 2001. doi:10.1201/9780429487330.
- [12] Jean R. S. Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In Alan George, John R. Gilbert, and Joseph W. H. Liu, editors, *Graph Theory and Sparse Matrix Computation*, pages 1–29, New York, NY, 1993. Springer. doi:10.1007/978-1-4613-8369-7_1.
- [13] Niclas Boehmer, Robert Brederbeck, Dušan Knop, and Junjie Luo. Fine-grained view on bribery for group identification. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 67–73. ijcai.org, 2020. doi:10.24963/ijcai.2020/10.
- [14] Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 601–612. IEEE, 2020. doi:10.1109/FOCS46700.2020.00062.
- [15] Christian Borgs, Michael Brautbar, Jennifer T. Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *SODA 2014*, pages 946–957. SIAM, 2014. doi:10.1137/1.9781611973402.70.
- [16] Bartłomiej Bosek, Andrzej Dudek, Michał Farnik, Jarosław Grytczuk, and Przemysław Mazur. Hat chromatic number of graphs. *Discrete Mathematics*, 344(12):10, December 2021. doi:10.1016/j.disc.2021.112620.
- [17] Bartłomiej Bosek, Andrzej Dudek, Michał Farnik, Jarosław Grytczuk, and Przemysław Mazur. Hat chromatic number of graphs. *Discrete Mathematics*, 344, 2021.
- [18] Andrei Braga, Cid C. de Souza, and Orlando Lee. The eternal dominating set problem for proper interval graphs. *Information Processing Letters*, 115(6):582–587, 2015. URL: <http://www.sciencedirect.com/science/article/pii/S0020019015000216>, doi:<https://doi.org/10.1016/j.ipl.2015.02.004>.
- [19] Joseph Briggs and Christopher Cox. Restricted online Ramsey numbers of matchings and trees. *Electron. J. Comb.*, 27(3):P3.49, 2020. doi:10.37236/8649.

-
- [20] Alewyn P. Burger, Ernest J. Cockayne, W. R. Gründlingh, Christina M. Mynhardt, Jan H. van Vuuren, and Wynand Winterbach. Infinite order domination in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 50:179–194, 2004.
- [21] Steve Butler, Mohammad Taghi Hajiaghayi, Robert D. Kleinberg, and Tom Leighton. Hat guessing games. *SIAM J. Discrete Math.*, 22(2):592–605, 2008. doi:[10.1137/060652774](https://doi.org/10.1137/060652774).
- [22] Ning Chen. On the approximability of influence in social networks. *SIAM J. Discrete Math.*, 23(3):1400–1415, 2009.
- [23] Wonki Jo Cho and Biung-Ghi Ju. Multinary group identification. *Theor. Econ.*, 12(2):513–531, 2017. doi:[10.3982/TE2156](https://doi.org/10.3982/TE2156).
- [24] Morgan Chopin, André Nichterlein, Rolf Niedermeier, and Mathias Weller. Constant thresholds can make target set selection tractable. *Theory Comput. Syst.*, 55(1):61–83, 2014. doi:[10.1007/s00224-013-9499-3](https://doi.org/10.1007/s00224-013-9499-3).
- [25] David Conlon. On-line Ramsey numbers. *SIAM Journal on Discrete Mathematics*, 23:1954–1963, 2009.
- [26] David Conlon, Jacob Fox, and Benny Sudakov. On two problems in graph Ramsey theory. *Combinatorica*, 32(5):513–535, 2012.
- [27] David Conlon, Jacob Fox, and Benny Sudakov. Recent developments in graph Ramsey theory. In *Surveys in Combinatorics*, 2015.
- [28] David Conlon, Jacob Fox, and Yuval Wigderson. Three early problems on size Ramsey numbers. *arXiv*, abs/2111.05420, November 2021. arXiv:[2111.05420](https://arxiv.org/abs/2111.05420).
- [29] John H. Conway. *On numbers and games*. A K Peters/CRC Press, second edition, 2000.
- [30] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC 1971*, page 151–158, New York, NY, USA, 1971. ACM. doi:[10.1145/800157.805047](https://doi.org/10.1145/800157.805047).
- [31] Gennaro Cordasco, Luisa Gargano, Marco Mecchia, Adele Anna Rescigno, and Ugo Vaccaro. Discovering small target sets in social networks: A fast and effective algorithm. *Algorithmica*, 80(6):1804–1833, 2018. doi:[10.1007/s00453-017-0390-5](https://doi.org/10.1007/s00453-017-0390-5).
- [32] Gennaro Cordasco, Luisa Gargano, and Adele Anna Rescigno. Parameterized complexity of immunization in the threshold model. *CoRR*, abs/2102.03537, 2021. arXiv:[2102.03537](https://arxiv.org/abs/2102.03537).
- [33] Gennaro Cordasco, Luisa Gargano, Adele Anna Rescigno, and Ugo Vaccaro. Evangelism in social networks: Algorithms and complexity. *Networks*, 71(4):346–357, 2018. doi:[10.1002/net.21756](https://doi.org/10.1002/net.21756).

- [34] Danielle Cox, Erin Meger, and M. E. Messinger. Eternal k -domination on graphs. *arXiv*, abs/2104.03835, April 2021. [arXiv:2104.03835](https://arxiv.org/abs/2104.03835).
- [35] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, August 2015. [doi:10.1007/978-3-319-21275-3](https://doi.org/10.1007/978-3-319-21275-3).
- [36] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer Berlin Heidelberg, August 2005.
- [37] Dinko Dimitrov. The social choice approach to group identification. *Consensual Processes*, 267:123–134, 2011. [doi:10.1007/978-3-642-20533-0_7](https://doi.org/10.1007/978-3-642-20533-0_7).
- [38] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *SIGKDD*, pages 57–66. ACM, 2001.
- [39] Rodney G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer New York, November 1998. [doi:10.1007/978-1-4612-0515-9](https://doi.org/10.1007/978-1-4612-0515-9).
- [40] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. [doi:10.1007/978-1-4471-5559-1](https://doi.org/10.1007/978-1-4471-5559-1).
- [41] Paul A. Dreyer, Jr. and Fred S. Roberts. Irreversible k -threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627, 2009. [doi:10.1016/j.dam.2008.09.012](https://doi.org/10.1016/j.dam.2008.09.012).
- [42] Keith Driscoll, William F. Klostermeyer, Elliot Krop, Colton Magnant, and Patrick Taylor. On eternal domination and Vizing-type inequalities. *Int. J. Graphs Comb.*, 17(3):708–712, 2020. [doi:10.1016/j.akcej.2019.12.013](https://doi.org/10.1016/j.akcej.2019.12.013).
- [43] Pavel Dvořák, Dušan Knop, and Tomáš Toufar. Target set selection in dense graph classes. In *ISAAC 2018*, LIPIcs, pages 18:1–18:13. Dagstuhl, 2018. [doi:10.4230/LIPIcs.ISAAC.2018.18](https://doi.org/10.4230/LIPIcs.ISAAC.2018.18).
- [44] Vojtěch Dvořák. A note on restricted online Ramsey numbers of matchings. *Electron. J. Comb.*, 28(3):P3.16, 2021. [doi:10.37236/10025](https://doi.org/10.37236/10025).
- [45] Todd Ebert. *Applications of Recursive Operators to Randomness and Complexity*. PhD thesis, University of California, Santa Barbara, 1998.
- [46] Todd Ebert, Wolfgang Merkle, and Heribert Vollmer. On the autoreducibility of random sequences. *SIAM J. Comput.*, 32(6):1542–1569, June 2003. [doi:10.1137/S0097539702415317](https://doi.org/10.1137/S0097539702415317).
- [47] Gábor Erdélyi, Christian Reger, and Yongjie Yang. The complexity of bribery and control in group identification. *Autonomous Agents and Multi-Agent Systems*, 34(8), 2019. URL: [10.1007/s10458-019-09427-9](https://doi.org/10.1007/s10458-019-09427-9).

-
- [48] Paul Erdős, Ralph J. Faudree, Cecil C. Rousseau, and Richard H. Schelp. The size Ramsey number. *Periodica Mathematica Hungarica*, 9:145–161, 1978.
- [49] Paul Erdős. Problems and results on finite and infinite graphs. In *Recent advances in graph theory (Proc. Second Czechoslovak Sympos., Prague)*, pages 183–192, 1975.
- [50] Michał Farnik. *A hat guessing game*. PhD thesis, Jagiellonian University, 2015.
- [51] Uriel Feige. You can leave your hat on (if you guess its color). Technical Report MCS04-03, The Weizmann Institute, Rehovot, Israel, 2004.
- [52] Stephen Finbow, Margaret-Ellen Messinger, and Martin F. van Bommel. Eternal domination on $3 \times n$ grid graphs. *Australasian Journal of Combinatorics*, 61:156–174, 2015.
- [53] Maximilien Gadouleau. Finite dynamical systems, hat games, and coding theory. *SIAM J. Discrete Math.*, 32(3):1922–1945, 2018. doi:[10.1137/15M1044758](https://doi.org/10.1137/15M1044758).
- [54] Maximilien Gadouleau and Nicholas Georgiou. New constructions and bounds for Winkler’s hat game. *SIAM J. Discrete Math.*, 29(2):823–834, 2015. doi:[10.1137/130944680](https://doi.org/10.1137/130944680).
- [55] Maximilien Gadouleau and Søren Riis. Graph-theoretical constructions for graph entropy and network coding based communications. *IEEE Trans. Inf. Theory*, 57(10):6703–6717, 2011. doi:[10.1109/TIT.2011.2155618](https://doi.org/10.1109/TIT.2011.2155618).
- [56] Kiran Garimella, Aristides Gionis, Nikos Parotsidis, and Nikolaj Tatti. Balancing information exposure in social networks. In *NIPS 2017*, pages 4663–4671, 2017.
- [57] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Comb. Theory, Series B*, 16(1):47–56, 1974. doi:[10.1016/0095-8956\(74\)90094-X](https://doi.org/10.1016/0095-8956(74)90094-X).
- [58] Wayne Goddard, Sandra M. Hedetniemi, and Stephen T. Hedetniemi. Eternal security in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 52:169–180, 2005. URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.94.2592>.
- [59] Jarosław Grytczuk, Mariusz Hałuszczak, and Hal A. Kierstead. On-line Ramsey theory. *Electronic Journal of Combinatorics*, 11(1), September 2004. doi:[10.37236/1810](https://doi.org/10.37236/1810).
- [60] Jarosław Grytczuk, Hal A. Kierstead, and Paweł Prałat. On-line Ramsey numbers for paths and stars. *Discrete Mathematics & Theoretical Computer Science*, 10, 2008. doi:[10.46298/dmtcs.427](https://doi.org/10.46298/dmtcs.427).
- [61] Frank Harary. *Graph Theory*. Addison-Wesley Publishing Company, Inc., 1969.

- [62] Penny E. Haxell, Yoshiharu Kohayakawa, and Tomasz Łuczak. The induced size-Ramsey number of cycles. *Combinatorics, Probability and Computing*, 4(03):217–239, 1995.
- [63] Xiaoyu He, Yuzu Ido, and Benjamin Przybocki. Hat guessing on books and windmills. *The Electronic Journal of Combinatorics*, 29(1), January 2022. doi:[10.37236/10098](https://doi.org/10.37236/10098).
- [64] Xiaoyu He and Ray Li. Hat guessing numbers of degenerate graphs. *Electron. J. Comb.*, 27(3):P3.58, 2020. doi:[10.37236/9449](https://doi.org/10.37236/9449).
- [65] Michael A. Henning and William F. Klostermeyer. Trees with large m-eternal domination number. *Discrete Applied Mathematics*, 211:79–85, October 2016. doi:[10.1016/j.dam.2016.04.021](https://doi.org/10.1016/j.dam.2016.04.021).
- [66] Cornelis Hoede and Xueliang Li. Clique polynomials and independent set polynomials of graphs. *Discrete Mathematics*, 125(1):219–228, 1994. doi:[10.1016/0012-365X\(94\)90163-5](https://doi.org/10.1016/0012-365X(94)90163-5).
- [67] Ian Holyer. The NP-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718–720, 1981. doi:[10.1137/0210055](https://doi.org/10.1137/0210055).
- [68] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, March 2001. doi:[10.1006/jcss.2000.1727](https://doi.org/10.1006/jcss.2000.1727).
- [69] Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k -SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.
- [70] Fionn Mc Inerney, Nicolas Nisse, and Stéphane Pérennes. Eternal domination: D-dimensional cartesian and strong grids and everything in between. *Algorithmica*, 83(5):1459–1492, February 2021. doi:[10.1007/s00453-020-00790-8](https://doi.org/10.1007/s00453-020-00790-8).
- [71] Kai Jin, Ce Jin, and Zhaoquan Gu. Cooperation via codes in restricted hat guessing games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 547–555, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems.
- [72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer US, 1972. doi:[10.1007/978-1-4684-2001-2_9](https://doi.org/10.1007/978-1-4684-2001-2_9).
- [73] Asa Kasher and Ariel Rubinfeld. On the question "who is a j?" a social choice approach. *Logique et Analyse*, 40(160):385–395, 1997. URL: <http://www.jstor.org/stable/44084614>.

-
- [74] David Kempe, Jon M. Kleinberg, and Éva Tardos. Influential nodes in a diffusion model for social networks. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 1127–1138. Springer, 2005. doi:[10.1007/11523468_91](https://doi.org/10.1007/11523468_91).
- [75] William F. Klostermeyer and Gary MacGillivray. Eternal dominating sets in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 68:97–111, February 2009. URL: https://www.researchgate.net/publication/283935997_Eternal_dominating_sets_in_graphs.
- [76] William F. Klostermeyer and Gary MacGillivray. Eternal domination in trees. *CoRR*, 2021. arXiv preprint arXiv:2112.03107.
- [77] William F. Klostermeyer and Christina M. Mynhardt. Protecting a graph with mobile guards. *Applicable Analysis and Discrete Mathematics*, 10, 07 2014. doi:[10.2298/AADM151109021K](https://doi.org/10.2298/AADM151109021K).
- [78] William F. Klostermeyer and Christina M. Mynhardt. Domination, eternal domination and clique covering. *Discussiones Mathematicae Graph Theory*, 35(2):283, 2015. doi:[10.7151/dmgt.1799](https://doi.org/10.7151/dmgt.1799).
- [79] Dušan Knop, Šimon Schierreich, and Ondřej Suchý. Balancing the spread of two opinions in sparse social networks. *CoRR*, abs/2105.10184, 2021. URL: <https://arxiv.org/abs/2105.10184>, arXiv:2105.10184.
- [80] K. P. Kokhas, A. S. Latyshev, and V. I. Retinskiy. Cliques and constructors in “Hats” game. II. *Journal of Mathematical Sciences*, 255(1):58–70, April 2021. doi:[10.1007/s10958-021-05349-8](https://doi.org/10.1007/s10958-021-05349-8).
- [81] Konstantin Kokhas, Aleksei Latyshev, and Vadim Retinsky. Cliques and constructors in “hats” game. April 2020. arXiv:2004.09605.
- [82] Konstantin P. Kokhas and Aleksei S. Latyshev. Cliques and constructors in “Hats” game. I. *Journal of Mathematical Sciences*, 255(1):39–57, May 2021. doi:[10.1007/s10958-021-05348-9](https://doi.org/10.1007/s10958-021-05348-9).
- [83] Konstantin P. Kokhas, Aleksei S. Latyshev, and Vadim I. Retinskiy. Cliques and constructors in “Hats” game. II. *Journal of Mathematical Sciences*, 255(1):58–70, May 2021. doi:[10.1007/s10958-021-05349-8](https://doi.org/10.1007/s10958-021-05349-8).
- [84] Aaron Krim-Yee, Ben Seamone, and Virgélot Virgile. Eternal domination on prisms of graphs. *Discret. Appl. Math.*, 283:734–736, 2020. doi:[10.1016/j.dam.2020.01.032](https://doi.org/10.1016/j.dam.2020.01.032).
- [85] Andrzej Kurek and Andrzej Rucinski. Two variants of the size Ramsey number. *Discussiones Mathematicae Graph Theory*, 25:141–149, 2005.

- [86] Ron Larson. *Calculus: An Applied Approach*. Brooks Cole, 8 edition, 2007.
- [87] Aleksei Latyshev and Konstantin Kokhas. The hats game. On max degree and diameter, 2021. [arXiv:2108.08065](https://arxiv.org/abs/2108.08065).
- [88] Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010. [doi:10.4086/toc.2010.v006a005](https://doi.org/10.4086/toc.2010.v006a005).
- [89] Margaret-Ellen Messinger. Closing the gap: Eternal domination on 3 x n grids. *Contributions to Discrete Mathematics*, Vol 12:No 1 (2017), 2017. [doi:10.11575/CDM.V12I1.62531](https://doi.org/10.11575/CDM.V12I1.62531).
- [90] M. E. J. Newman, Stephanie Forrest, and Justin Balthrop. Email networks and the spread of computer viruses. *Phys. Rev. E*, 66:035101, Sep 2002. [doi:10.1103/PhysRevE.66.035101](https://doi.org/10.1103/PhysRevE.66.035101).
- [91] André Nichterlein, Rolf Niedermeier, Johannes Uhlmann, and Mathias Weller. On tractable cases of target set selection. *Social Netw. Analys. Mining*, 3(2):233–256, 2013. [doi:10.1007/s13278-012-0067-7](https://doi.org/10.1007/s13278-012-0067-7).
- [92] Wesley Pegden. The lefthanded local lemma characterizes chordal dependency graphs. *Random Struct. Algorithms*, 41(4):546–556, 2012. [doi:10.1002/rsa.20439](https://doi.org/10.1002/rsa.20439).
- [93] Šárka Petříčková. Online Ramsey theory for planar graphs. *Electron. J. Comb.*, 21(1):P1.64, 2014. [doi:10.37236/2940](https://doi.org/10.37236/2940).
- [94] Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. Syst. Sci.*, 67(4):757–771, 2003. [doi:10.1016/S0022-0000\(03\)00078-3](https://doi.org/10.1016/S0022-0000(03)00078-3).
- [95] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *SIGKDD*, pages 61–70. ACM, 2002. [doi:10.1145/775047.775057](https://doi.org/10.1145/775047.775057).
- [96] Søren Riis. Information flows, graphs and their guessing numbers. *Electron. J. Comb.*, 14(1), 2007. [doi:10.37236/962](https://doi.org/10.37236/962).
- [97] Sara Robinson. Why mathematicians now care about their hat color. *New York Times*, April 10, 2001. URL: <https://www.nytimes.com/2001/04/10/science/why-mathematicians-now-care-about-their-hat-color.html>.
- [98] Dov Samet and David Schmeidler. Between liberalism and democracy. *J. Econ. Theory*, 110(2):213–233, 2003. [doi:10.1016/S0022-0531\(03\)00080-2](https://doi.org/10.1016/S0022-0531(03)00080-2).
- [99] Alexander D. Scott and Alan D. Sokal. On dependency graphs and the lattice gas. *Comb. Probab. Comput.*, 15(1-2):253–279, 2006. [doi:10.1017/S0963548305007182](https://doi.org/10.1017/S0963548305007182).
- [100] Ramy Shaheen, Mohammad Assaad, and Ali Kassem. On eternal domination of generalized $J_{s,m}$. *J. Appl. Math.*, 2021:1–7, 2021. [doi:10.1155/2021/8882598](https://doi.org/10.1155/2021/8882598).

-
- [101] Ramy Shaheen and Ali Kassem. Eternal domination of generalized Petersen graph. *J. Appl. Math.*, 2021:1–10, 2021. doi:10.1155/2021/6627272.
- [102] James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985. doi:10.1007/BF02579368.
- [103] Manuel Sorge. The graph parameter hierarchy, 2019. online, accessed 25. March 2022. URL: <https://manyu.pro/assets/parameter-hierarchy.pdf>.
- [104] Witold Szczechla. The three colour hat guessing game on cycle graphs. *Electron. J. Comb.*, 24, 2017. doi:10.37236/5135.
- [105] Craig A. Tovey. A simplified np-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. URL: <https://www.sciencedirect.com/science/article/pii/0166218X84900817>, doi:[https://doi.org/10.1016/0166-218X\(84\)90081-7](https://doi.org/10.1016/0166-218X(84)90081-7).
- [106] Christopher M. van Bommel and Martin F. van Bommel. Eternal domination numbers of $5 \times n$ grid graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 97:83–102, 2016. URL: <http://people.stfx.ca/mvanbomm/publicat/eternal5xn.pdf>.
- [107] Peter Winkler. Games people don't play. In D. Wolfe and T. Rodgers, editors, *Puzzlers' Tribute: A Feast for the Mind*, pages 301–313. 2002. doi:10.1201/9781439864104-50.
- [108] Taoyang Wu, Peter Cameron, and Søren Riis. On the guessing number of shift graphs. *Journal of Discrete Algorithms*, 7(2):220–226, 2009. doi:10.1016/j.jda.2008.09.009.
- [109] Yongjie Yang and Dinko Dimitrov. How hard is it to control a group? *Auton. Agent Multi. Agent Syst.*, 32(5):672–692, Sep 2018. doi:10.1007/s10458-018-9392-1.

Reviewed Publications of the Author Relevant to the Thesis

- [A.1] Václav Blažej (33.33%), Jan Matyáš Kříšťan (33.33%), and Tomáš Valla (33.33%). On the m-eternal domination number of cactus graphs. *Reachability Problems - 13th International Conference, RP 2019*, volume 11674 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2019. [doi:10.1007/978-3-030-30806-3_4](https://doi.org/10.1007/978-3-030-30806-3_4).

The paper has been cited in:

- Toshihiro Fujito, Tomoya Nakamura. Eternal Connected Vertex Cover Problem. *16th Annual Conference on Theory and Applications of Models of Computation, TAMC 2020*, volume 12337 of *Lecture Notes in Computer Science*, pages 181–192, 2020. [doi:10.1007/978-3-030-59267-7_16](https://doi.org/10.1007/978-3-030-59267-7_16)
- [A.2] Václav Blažej (33.33%), Pavel Dvořák (33.33%), and Michal Opler (33.33%). Bears with hats and independence polynomials. *Graph-Theoretic Concepts in Computer Science - 47th International Workshop, WG 2021*, volume 12911 of *Lecture Notes in Computer Science*, pages 283–295. Springer, 2021. [doi:10.1007/978-3-030-86838-3_22](https://doi.org/10.1007/978-3-030-86838-3_22).

The paper has been cited in:

- Aleksei Latyshev, Konstantin Kokhas. The Hats game. On maximum degree and diameter. *Discrete Mathematics*, 345(7), July 2022. [10.1016/j.disc.2022.112868](https://doi.org/10.1016/j.disc.2022.112868)
- [A.3] Václav Blažej (33.33%), Pavel Dvořák (33.33%), and Tomáš Valla (33.33%). On induced online Ramsey number of paths, cycles, and trees. *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019*, volume 11532 of *Lecture Notes in Computer Science*, pages 60–69. Springer, 2019. [doi:10.1007/978-3-030-19955-5_6](https://doi.org/10.1007/978-3-030-19955-5_6).

- [A.4] Václav Blažej (33.33%), Dušan Knop (33.33%), and Šimon Schierreich (33.33%).
Controlling the spread of two secrets in diverse social networks (student abstract).
In *Computer Science - Theory and Applications - 36th Conference on Artificial Intelligence, AAAI 2022, (to appear)*, 2022.

Remaining Publications of the Author

- [A.5] V. Blažej, O. Suchý, T. Valla *A Simple Streaming Bit-Parallel Algorithm for Swap Pattern Matching*. Mathematical Aspects of Computer and Information Sciences. MACIS 2017. Lecture Notes in Computer Science, vol 10693. Springer, Cham
- [A.6] V. Blažej, J. Fiala, G. Liotta *On the edge-length ratio of 2-trees*. Graph Drawing and Network Visualization - 28th International Symposium, GD 2020
- [A.7] V. Blažej, M. Opler, M. Šileikis, and P. Valtr *On the Intersections of Non-homotopic Loops*. 7th Annual International Conference on Algorithms and Discrete Applied Mathematics, CALDAM 2021
- [A.8] V. Blažej, M. Opler, M. Šileikis, and P. Valtr *Non-homotopic Loops with a Bounded Number of Pairwise Intersections*. Graph Drawing and Network Visualization - 29th International Symposium, GD 2021
- [A.9] V. Blažej, P. Choudhary, D. Knop, J. M. Kříšťan, O. Suchý, and T. Valla *Constant Factor Approximation for Tracking Paths and Fault Tolerant Feedback Vertex Set*. Approximation and Online Algorithms - 19th International Workshop, WAOA 2021