Reconfiguration and combinatorial games

Marc Heinrich

N° d'ordre NNT : 2019LYSE1102

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON

opérée au sein de

l'Université Claude Bernard Lyon 1

École Doctorale N° 512, InfoMaths

Spécialité de doctorat : Informatique

Soutenue publiquement le 09/07/2019 par :

Marc Heinrich

Reconfiguration and Combinatorial Games

Devant le jury composé de :

Isabelle Guerin-Lassous, Professeure, Université Lyon 1	Présidente
Celina De Figueiredo, Professeure, Université Fédérale de Rio de Janeiro	Rapportrice
Claire Mathieu, Directrice de Recherche – CNRS, Université Paris 7	Rapportrice
Isabelle Guerin-Lassous, Professeure, Université Lyon 1	Examinatrice
Jan Van den Heuvel, Professeur, École d'économie et de science politique de Londres	Examinateur
Éric Duchêne, Maître de conférences, Université Lyon 1	Directeur de thèse
Sylvain Gravier, Directeur de Recherche – CNRS, Université Grenoble Alpes	Co-directeur de thèse
Nicolas Bousquet, Chargé de Recherche – CNRS, Université Grenoble Alpes	Co-encadrant

Contents

Remerciements

Tout d'abord, je souhaite remercier mes rapporteurs, Claire Mathieu et Celina de Figueiredo pour leur lecture attentive de ma thèse, ainsi que les autres membres du jury, Isabelle Guerin Lassous et Jan van den Heuvel, pour avoir fait le déplacement et assisté à ma soutenance.

Je souhaite également remercier mes encadrants de thèse, Eric, Sylvain, et Nicolas, qui m'ont permis de travailler sur plein de problèmes vraiment intéressants. Vous m'avez donné le goût de me casser la tête sur des maths pendant 3 ans, et j'espère pouvoir continuer à le faire pendant encore longtemps. Merci aussi pour les parties de truco, les jeux de société en tout genre, et les pique-niques à la Tête d'or.

Merci aussi aux autres permanents et doctorants de l'équipe GOAL, notamment Aline, Valentin, Gabrielle, Antoine, et Alice, pour leurs contributions à ces activités, et qui ont permis de faire de cette thèse une expérience agréable et plaisante. Je remercie aussi les autres doctorants du laboratoire, sans qui les midis ne seraient juste pas pareils.

Je pense aussi à mes amis de prépa, et notamment à ceux qui ont fait le déplacement juste pour pouvoir me supporter pendant 1h, Sylvain et Jonathan (retape toi bien). Je pense aussi à tout ceux que j'ai oublié de mentionner ici (me connaissant, il y en a forcément).

Mes pensées vont enfin à toute ma famille, qui a réussi à me supporter jusque là, et sur qui je peux toujours compter dès qu'il est question de gâteaux, de séances de ciné, ou de me rappeler que ça fait longtemps que je ne me suis pas coupé les cheveux.

Finalement merci à Alice (l'autre) qui me traîne dehors tous les week-ends, et me raconte ses journées tous les soirs jusqu'à pas-d'heure.

À tous ces gens: Merci.

Résumé (French)

Cette thèse explore des problématiques liées aux jeux. Le terme 'jeu' est assez vague et englobe beaucoup de concepts différents. Par exemple, cela peut désigner les jeux de société, qui possèdent une grande variété de règles, et qui sont souvent joués à plusieurs ; les jeux vidéos, qui ont souvent un aspect temporel important et requièrent des réflexes et de la précision ; les jeux économiques, pour lesquels il y a une notion de gains et de revenus ; ou encore les jeux sportifs qui demandent plutôt de la technique et de la force. Dans cet ouvrage, nous sommes intéressés par les jeux pour lesquels toute l'information est connue dès le début de la partie. En d'autres termes, il n'y a pas d'information cachée : tous les joueurs ont accès à toute l'information relative au jeu ; il n'y a pas non plus d'aléa, et tout est déterministe. Parmi les jeux mentionnés plus haut, seuls certains jeux de société (comme les échecs ou le go) satisfont ces propriétés et sont représentatifs des jeux que nous considérons ici. Dans la suite, nous utiliserons le terme de 'jeu' uniquement pour désigner ces jeux à information parfaite. La notion de stratégie est au centre de l'étude de ces jeux. En termes simples, une stratégie est une façon de jouer qui permet de s'assurer un certain résultat. La question centrale, à la fois quand on joue à un jeu et quand on l'étudie, consiste à trouver la 'meilleure' stratégie, qui assure la victoire du joueur qui l'applique.

Dans cette thèse, nous allons considérer à la fois des jeux à un joueur, appelés puzzles combinatoires, et des jeux à deux joueurs. Le Rubik's cube, Rush-Hour, le Sokoban, ou le taquin sont des exemples biens connus de puzzles combinatoires. D'un point de vue historique, les jeux à un et deux joueurs faisaient partie de ce qui était appelé les 'mathématiques récréatives', et peu de motivations étaient données pour étudier ces jeux, à part une curiosité naturelle pour essayer de comprendre, à l'aide des mathématiques, leur structure et leur complexité. Cependant, plus récemment un certain nombre de jeux – des jeux à un joueur notamment – ont connu un regain d'intérêt en tant qu'objets d'étude dans un domaine plus grand appelé reconfiguration. Les puzzles que l'on vient de mentionner peuvent tous être décrits de la façon suivante : il y a un ensemble de configurations, qui représente tous les états possibles du jeu ; et le joueur est autorisé à transformer une configuration en une autre via un certain nombre de règles. Le but est d'atteindre une certaine configuration cible à partir d'une configuration initiale. Dans le cas du Rubik's cube par exemple, le but est d'atteindre la configuration où tous les carrés d'une même face ont la même couleur. Le domaine de la reconfiguration étend cette définition à des problèmes de recherche : l'ensemble des configurations devient l'ensemble des solutions à une instance d'un problème donné, et l'on se demande si l'on peut transformer une solution donnée en une autre en utilisant uniquement un ensemble de règles de transformation précises.

Ainsi, en plus des puzzles combinatoires, les problèmes de reconfiguration incluent aussi un certain nombre de 'jeux' qui ne sont plus des jeux joués par le grand public, mais plutôt des problèmes mathématiques partageant un certain nombre de similarités avec les puzzles combinatoires. La

recherche sur ce type de problèmes a été guidée par plusieurs motivations. Ces motivations concernent par exemple des applications algorithmiques : ce processus peut être vu comme un moyen d'adapter une solution déjà en place pour former une nouvelle solution à l'aide de changements locaux. Il y a également des connexions avec d'autres problèmes comme la génération aléatoire, des problèmes liés au comptage du nombre de solutions à un problème, ainsi que des problèmes provenant de physique statistique. Finalement, ces problèmes de reconfiguration peuvent être un outil pour comprendre les performances d'un certain nombre d'algorithmes heuristiques qui se basent sur des modifications locales des solutions (les algorithmes de recherche locale en sont un exemple).

Les jeux à deux joueurs, qui sont aussi appelés jeux combinatoires, ont été étudiés depuis le début du vingtième siècle avec les travaux de Bouton. Ces travaux ont été continués avec le développement par Berlekamp, Conway et Guy d'une théorie assez élégante unifiant un certain nombre de jeux classiques. S'il y a quelques jeux combinatoires qui sont joués de façon active, la majeure partie des recherches sur ces jeux-là concernent le domaine de l'intelligence artificielle. Il y a eu des progrès importants dans les dernières années dans le domaine de l'intelligence artificielle pour créer des machines performantes pour jouer à ces jeux, mais ce n'est cependant pas ce qui va nous intéresser ici. Ce travail se focalise sur des joueurs parfaits, c'est-à-dire qui choisissent toujours le coup optimal. Notre but est de caractériser lequel des deux joueurs possède une stratégie qui lui assure la victoire, quels que soient les coups de son adversaire. Cette approche est au cœur de ce qui est appelé la Théorie des Jeux Combinatoires. Très peu de jeux joués par le grand public sont étudiés dans ce domaine, il y a plusieurs raisons à cela. Tout d'abord, beaucoup de ces jeux ont des règles compliquées, ce qui rend leur analyse difficile. De plus, il a déjà été montré pour un certain nombre d'entre eux qu'ils sont difficiles d'un point de vue computationnel, c'est-à-dire que trouver une stratégie optimale est difficile, même pour un ordinateur. Ainsi, une grande partie de la recherche est focalisée sur ce que l'on peut appeler des 'jeux mathématiques', qui sont des jeux inventés par des mathématiciens, souvent avec des règles très simples, et rarement connus en dehors de la recherche. Il y a plusieurs motivations pour étudier ces jeux. Tout d'abord, ces jeux produisent souvent des problèmes avec des définitions très simples et qui sont également difficiles d'un point de vue computationnel, c'est-à-dire qu'il n'existe pas d'algorithme efficace pour les résoudre. Ces problèmes sont important pour classer et comparer différents problèmes en fonction de l'efficacité des algorithmes pour les résoudre. Une deuxième motivation, et probablement la plus importante, est dûe aux liens parfois surprenants entre ces jeux et d'autres domaines des mathématiques comme entre autre la théorie des nombres, les automates, ou les systèmes dynamiques.

Notre outil principal pour étudier ces jeux est la notion de graphe. En termes simples, un graphe est un ensemble de points, avec des liens reliant ces points. Les points sont appelés des sommets, et les liens entre ces points sont appelés des arêtes. Les graphes sont un outil très puissant pour représenter des données, et ont été utilisés pour modéliser une très grande variété de structures. Par exemple, ils ont été utilisés pour représenter différents réseaux, comme des réseaux routiers, électriques, ou Internet. Ils ont été également utilisés pour décrire des réseaux sociaux, et plus généralement toute structure qui contient un grand nombre d'acteurs en interaction. Les graphes peuvent également décrire d'autres objets comme des programmes informatiques, des systèmes biologiques et physiques, ou des structures de données pour stocker des informations. Étant donné leur large éventail d'applications, il n'est pas étonnant que les graphes soient aussi reliés aux jeux.

Le premier lien évident entre les jeux et les graphes est le fait que certains jeux soient joués directement sur un graphe. Cela concerne en particulier un nombre important de problèmes de

reconfiguration étudiés dans les dernières années. En effet, un grand nombre de problèmes de reconfiguration consistent à étudier des transformations appliquées à des solutions de problèmes de graphes bien connus. C'est le cas en particulier des problèmes de reconfigurations que nous étudions dans la première partie de cette thèse. L'autre lien entre jeux et graphes est le fait qu'un jeu peut être aussi représenté comme un graphe (même si le jeu lui-même n'est pas joué sur un graphe). Plus précisément, un jeu, que ce soit un jeu à un ou deux joueurs, définit de façon implicite un graphe dont les sommets représentent toutes les configurations possibles du jeu, et les arêtes sont les coups possibles qui permettent de passer d'une configuration à une autre. Ainsi, un nombre important de questions intéressantes sur les jeux peuvent être reformulées en termes de propriétés du graphe sous-jacent. Cependant, comme le nombre de positions possibles pour un jeu est souvent extrêmement grand, on ne souhaite pas en général calculer le graphe sous-jacent explicitement (cela prendrait beaucoup trop de temps), mais plutôt utiliser la représentation implicite pour répondre directement aux questions.

Finalement, si les jeux sont les objets que nous étudions, et les graphes sont un des principaux outils que nous utilisons, les questions que nous considérons sont souvent en lien avec des problèmes de complexité. La complexité d'un problème revient à estimer le temps qu'il faudrait à un ordinateur pour résoudre le problème. Plus précisément, nous essayons de déterminer combien de temps il faudrait au meilleur algorithme afin de résoudre une instance d'un problème donné, et comment ce temps évolue en fonction de la taille de l'instance en entrée. Une partie importante de la recherche est dédiée à séparer les problèmes 'faciles', c'est-à-dire pour lesquels il existe un algorithme efficace (i.e., qui termine après un nombre d'étapes polynômial en fonction de la taille de l'entrée), des problèmes 'difficiles' pour lesquels de tels algorithmes n'existent pas. Il y a deux aspects principaux dans l'étude de la complexité d'un problème : trouver des algorithmes efficaces pour résoudre le problème, et prouver des réductions qui montrent que ce problème est au moins aussi difficile qu'un autre.

Organisation Générale

Nous allons maintenant décrire l'organisation générale de la thèse. Dans la première partie de cette thèse, nous focalisons notre attention sur les problèmes de reconfiguration. Nous donnons à la fois des résultats de complexité, et des résultats structurels liés à ces problèmes. Cette première partie est organisée en trois chapitres. Le premier de ces chapitres donne un introduction aux problèmes de reconfiguration. Nous y définissons certaines notations communes à tous les problèmes de reconfiguration et donnons également des motivations pour étudier ce type de problèmes. En particulier, ce chapitre met en avant des liens entre les problèmes de reconfiguration et d'autres domaines de l'informatique. Nous y détaillons aussi avec précision les résultats existants sur la reconfiguration d'un problème particulier : la coloration de graphes. Dans le chapitre suivant, nous étudions la reconfiguration de colorations de graphes. Ce chapitre donne plusieurs résultats, à la fois d'un point de vue complexité, et d'un point de vue structurel. Sur les aspects liés à la complexité, nous décrivons des preuves de difficulté pour deux problèmes liés à la reconfiguration de colorations. Les résultats structurels consistent à étudier le nombre d'étapes nécessaires pour transformer une solution en une autre, dans des cas où l'on est certain que ces transformations existent. Enfin, dans le dernier chapitre de cette partie, nous nous intéressons à la reconfiguration d'un autre objet combinatoire : les couplages parfaits. Nos résultats incluent à la fois des preuves montrant que ce problème est difficile en général, et des algorithmes terminant en temps polynômial

lorsque le graphe donné en entrée appartient à certaines classes de graphes.

Comme nous l'avons mentionné plus haut, une des motivations pour étudier les problèmes de reconfiguration provient des liens existant avec d'autres domaines de l'informatique. Dans la deuxième partie de cette thèse, nous nous intéresserons à deux de ces problèmes. Le premier chapitre de cette partie est dédié au problème de génération de colorations aléatoires en utilisant une méthode particulière appelée dynamique de Glauber. Ce chapitre commence par une vue d'ensemble sur les résultats existants sur cette méthode, et nous y détaillons des liens avec un certain nombre de problèmes provenant de physique statistique. Nos travaux consistent à étudier les performances de la dynamique de Glauber sous certaines hypothèses sur le graphe donné en entrée. S'il est conjecturé dans la littérature que cette méthode est efficace, sous certaines hypothèses assez faibles, ces performances ont été prouvées de façon théorique uniquement pour des contraintes plus fortes, ou pour des classes de graphes particulières. Nos résultats continuent dans la direction de prouver cette conjecture sur des classes de graphes particulières, en regardant notamment la variante du problème pour la coloration d'arêtes.

Dans le deuxième chapitre de cette partie, nous tentons d'utiliser les modifications locales provenant des problèmes de reconfiguration pour des problèmes de coloration online, c'est-à-dire lorsque les sommets du graphe en entrée sont donnés un par un. Les algorithmes classiques pour ce problème font généralement l'hypothèse que changer la couleur d'un sommet n'est plus possible une fois cette couleur attribuée. Il a été montré que cette restriction conduit à des performances assez mauvaises de ce type d'algorithme, même pour des graphes très simples. Nous considérons une extension du problème où l'algorithme est autorisé à recolorer en partie les sommets déjà colorés à des étapes précédentes de l'algorithme, et nous construisons des algorithmes pour ce problème pour plusieurs classes de graphes. Les algorithmes online peuvent aussi être vus comme une sorte de jeu à deux joueurs. L'un des joueurs (l'algorithme), essaye de trouver une solution efficace à un problème, tandis qu'un adversaire tente de l'en empêcher en choisissant un ordre des sommets mauvais pour la stratégie appliquée par l'algorithme. Ce lien nous amène naturellement à la troisième partie de cette thèse : les jeux à deux joueurs.

Dans la dernière partie, nous quittons le monde des jeux à un joueur pour nous intéresser aux jeux à deux joueurs. Le premier chapitre de cette partie donne une introduction à la théorie des jeux combinatoires, avec des définitions de certains concepts clés du domaine. Ce chapitre donne également un aperçu global des résultats connus pour une famille de jeux en particulier, ainsi que des précisions sur certains opérateurs étudiés dans la littérature qui créent de nouveaux jeux à partir de jeux existants. Les deux chapitres suivants étudient deux constructions particulières. Dans le premier de ces deux chapitres, nous nous intéressons à une construction qui autorise les joueurs à changer les règles du jeu en cours de partie. Nous commençons par étudier cette construction pour des exemples spécifiques obtenus en combinant des jeux classiques, bien connus du domaine. Nous donnons également des propriétés générales de cette construction, et étudions comment certains concepts définis dans le chapitre précédent peuvent être modifiés pour s'adapter à ces nouveaux jeux. Finalement, dans le dernier chapitre nous étudions une autre construction qui consiste à donner des règles différentes à chacun des deux joueurs. Cette construction est étudiée pour le cas d'une famille de jeux particulière appelée jeux de soustraction. Les jeux résultant de cette construction ont déjà été étudiés dans la littérature, et il existe des résultats permettant de classifier certains de ces jeux en fonction de leur comportement sur des positions très grandes. Notre étude étend ces résultats existant en affinant la classification, et en prouvant le comportement d'une large classe d'exemples.

Chapter 1

Introduction

This thesis explores problems related to games. The term 'game' is very broad and encompasses many different things. For example, it can refer to board games, which can have a very large variety of rules and are often played with several players; video games, many of which have a time component and require reflexes and precise timing; economical games, for which there are notions of gains and pay-off; or also sports games which rely more on skills, precision, and strength. In this study, we are interested in games for which everything is known from the very beginning. In other words, there is no hidden information: all the players have access to all the information related to the game; there is also no randomness and everything is deterministic. Among the games we mentioned above, only a few well-known board games (such as chess or go) fall in this category and are representative of the kinds of games that we consider here. In the following, we will use the term 'game' to denote only this type of perfect information games. Central to the study of these games is the notion of strategy, which roughly speaking, is a way of playing which ensures a certain objective. The main question of interest, when both playing and studying a game, is the problem of finding the 'best' strategy, which secures the victory for the player following it.

In this thesis, we will consider both one-player games, also called combinatorial puzzles, and two-player games. Examples of combinatorial puzzles include Rubik's cube, Rush-Hour, Sokoban, the 15 puzzle, or peg solitaire. Historically, both one- and two-players games were part of what was called 'recreational mathematics', and little motivation was given for studying these problems apart from the natural curiosity of using mathematics to understand their structure and their complexity. More recently however, some types of one-player games in particular have received a strong regain of interest as part of the larger area of reconfiguration problems. The puzzles we described above can all be described in the following way: there is a set of configurations, which represents all the possible states of the game; and the player is allowed to transform a configuration using a prescribed set of moves. Starting from an initial configuration, the goal is to reach a target configuration by a succession of valid moves. For example, the goal in Rubik's cube is to reach the target configuration with all the faces monochromatic. Reconfiguration extends this definition to any search problem: the set of configuration becomes the set of solutions to an instance of a given problem, and we ask whether we can transform one given solution to another using only a prescribed set of moves.

Hence, in addition to the combinatorial puzzles, reconfiguration problems also include many 'games' which are not played by humans anymore but are instead mathematical problems sharing a lot of similarities with combinatorial puzzles. The study of reconfiguration problems has been driven by many different motivations. It has algorithmic applications: it can be seen as a way to

adapt a current solution already in place to reach a new one by only making small local changes. It is also connected to other problems such as random sampling, approximate counting or problems coming from statistical physics. It can also be used as a tool for understanding the performance of some heuristic algorithms based on local modifications of solutions such as local search.

Two-player games, which are also called combinatorial games, have been studied since the beginning of the twentieth century, with the works of Bouton which were continued with the development of a nice theory by Berlekamp, Conway, and Guy, unifying a certain number of classical games. If there is a small number of combinatorial games which are actively played, the greater part of the research on these games is in the field of artificial intelligence and the design of competitive computer programs for playing these games. If artificial intelligence has made great progress at creating very good players for various games, this is not what we will be interested in here. Instead, we will focus on perfect strategies (i.e., players always choosing the best possible move), and try to characterize who wins under perfect play for various games. This approach is at the heart of what is called Combinatorial Game Theory. Few games played in real life are studied in this area. There are several reasons for this: many of these games have complicated rules, which makes their analysis complicated, and some of them were already shown to be computationally difficult, i.e., finding the optimal strategy is provably hard, even for a computer. Instead, most of the research in this area is focused on 'math games' which are games invented by mathematicians, often with simple rules and almost never played by humans. There are several motivations for studying these games. First, they often give simple definitions of problems which are computationally hard, i.e., for which no efficient algorithm exists. These hard problems are important for classifying and comparing problems based on how efficient the algorithms solving them are. However the main motivation comes from the nice, and sometimes unexpected connections these games have with other areas of mathematics, such as for example number theory, automatons, or dynamical systems.

Our main tool for studying all these games is the notion of graph. In simple terms, a graph is a set of points with links joining the points, as shown for example in Figure 1.1. The points are called vertices, and the links are called edges. Graphs are a very versatile way to represent data, and has been used to model a large variety of different structures. For example, they have been used to describe various networks such as roads, electrical networks, or the internet. They are also used to describe social media and more generally any structure which contain many actors in interactions with each other. Graphs can also describe other objects such as computer programs, biological/physical systems, or data structures for storing information. Given their wide range of applications, it is not surprising to see that graphs are also connected to games.

The first immediate connection between games and graphs is the fact that some games are played directly on a graph. This concerns in particular many of the reconfiguration problems which have been studied recently. Indeed, an important part of reconfiguration problems concerns the transformation of solutions to some well-known problems on graphs. This is the case of the reconfiguration problems considered in Part I of this thesis. The other connection between games and graphs is that a game can also be represented as a graph. More precisely, a game (both one-player and two-player) defines implicitly a graph, whose vertices represent all the possible configurations of the game, and whose edges are the possible moves between these configurations. Hence, many questions we would like to answer on games can be reformulated as finding properties of the underlying graph. However, since the number of possible positions for a game is usually very large, we do not wish to compute the underlying graph explicitly (it would take too much time), but instead use the implicit representation to answer the questions. For example, in the

case of reconfiguration problems, a strategy for solving a given instance corresponds to finding a path between the initial configuration and the target configuration in the underlying graph. Hence the main property of interest in this case is the connectivity of the underlying graph. Our problems often involve finding sufficient conditions for the underlying graph to be connected, or to characterize its connected components. For two player games, the structure which interests us is some partition of the underlying graph depending on which player has a winning strategy. This partition is sometimes called a kernel. If it is sometimes possible to give a simple characterisation to find a strategy for a game, there are cases where we must rely on algorithms to construct this strategy.

Finally, if games are the objects we study, and graphs are one of our major tools, the questions we consider are often related to complexity. Complexity theory consists in trying to classify problems depending on their hardness. By hardness we mean to quantify how much time it would take for a computer to solve the problem. More precisely, we are interested in studying how long the best algorithm would take to solve a given problem, and how this time increases as a function of the size of the input. An important amount of effort is dedicated to separating 'easy' problems for which there exists an 'efficient' algorithm (i.e., running in time polynomial in the size of the input), from problems which are 'hard' i.e., for which no efficient algorithm exists. There are two sides to complexity: finding efficient algorithms to solve problems, and devising reduction showing that one problem is at least as hard as another which serves as an evidence that the existence of an efficient algorithm is unlikely.

1.1 General Organisation

We now describe the general outline of this thesis. The rest of this chapter gives preliminaries for the other chapters in the form of definitions of basic results from graph theory, complexity and Markov chain theory which will be used throughout the thesis. The definitions and notations we use are quite standard, and the reader already familiar with these fields is free to skip them. Note that the introduction to Markov chain theory in Section 1.4 is a prerequisite only for Chapter 5, and is not necessary to understand the other parts. An index of all the symbols and definitions can can be found at the end of the thesis.

In the first part of this thesis, we focus our attention on reconfiguration problems. We give both complexity results as well as structural aspects related to these problems. This fist part is organized in three chapters. Chapter 2 gives an introduction to reconfiguration problems. We define in this chapter some notations common to all reconfiguration problems, and give motivations for studying them. We also describe connections between reconfiguration and other areas of discrete mathematics and computer science, and provide a more detailed survey on existing results for the reconfiguration of one particular problem: graph colouring. We then study the reconfiguration of graph colourings in Chapter 3. In this chapter, the problem is studied both from the point of view of complexity and the analysis of the structure of the reconfiguration graph. On the complexity aspects, we show the hardness of two problems related to colouring reconfiguration. Our structural result consists in studying upper bounds on the number of steps required to transform one solution into another, in a setting where this transformation is already known to exist. In Chapter 4, we consider the reconfiguration of another type of combinatorial object: perfect matchings. Our results include both hardness results for this reconfiguration problem in general, and polynomial

time algorithms when the graph is restricted to a specific class.

As we mentioned above, one motivation for studying reconfiguration problems is their connections with other areas from computer science. In Part II, we investigate two of these related problems. In Chapter 5, we consider the problem of generating random colourings using a specific method called Glauber dynamics. This chapter starts with an overview of the existing results on Glauber dynamics for colouring, as well as some details on the connections with other problems from statistical physics. Our study aims at analysing the performance of Glauber dynamics under some assumptions on the graph considered. If this algorithm is conjectured to have good performance under some mild assumptions on its input, these performance have only been verified theoretically under either stronger assumptions, or with some additional constraints on the graph. Our results continue in the direction of proving the conjecture for special cases by considering the edge colouring variant of the problem.

The second application we consider, in Chapter 6, is an attempt to use local transformations to colour graphs in an online fashion, i.e., when the input is given one vertex at a time. Standard online algorithms are usually considered in a setting where recolouring previously coloured vertices is not permitted. However, this restriction was shown to lead to poor performances even for simple graphs. We consider an extension where some amount of recolouring is allowed, and analyse several algorithms for different classes of graphs. Online algorithm can be seen as some kind of two player game, where one player (the algorithm) attempts to solve efficiently a problem, while an opponent tries to prevent this by choosing a bad ordering of the vertices. This leads us to our last part: games with two players.

In Part III, we move from one player games to two player games. We start this part with a light introduction to Combinatorial Game Theory in Chapter 7, and with definitions of some key concepts from the domain. This chapter also gives an overview of some existing results for one particular family of games, as well as operators studied in the literature to create new games from exiting ones. This chapter is followed by the study of two particular constructions in the last two chapters. In Chapter 8, we investigate a construction which allows the players to change the rules during the game. Our investigation starts with the study of specific examples which are obtained by combining several standard games. We also give some general properties of the construction, and end the chapter with by studying how some of the concepts that were introduced in Chapter 7 can be adapted to this construction. Finally, we consider in Chapter 9 another construction which gives different rules to the two players. We investigate the effects of this construction for one particular class of games called subtraction games. The games resulting from this construction have already been studied in the literature where some work has been done to classify these games based on their behaviour for very large positions. Our study continues on the existing results by refining and extending this classification for a larger class of examples.

1.2 Computational Complexity

We now give a short introduction to some aspects of computational complexity. Complexity theory aims at quantifying how difficult solving a problem is, and classifying problems depending on their complexity. In particular, it defines complexity classes, which gathers problems with similar complexities. In this section, and in the rest of this thesis, we will only discuss the complexity of decision problems, i.e., problems whose answer is either **yes** or **no**. However it should be noted

that some of the concepts can be extended to other types of problems such as function problems (i.e., algorithms which compute a particular function). There are many parameters that can be of interest to quantify the cost of an algorithm, but two of them are the most widely used: the time it takes for the algorithm to run, and how much memory it uses. We only aim at giving a light introduction to complexity, and as such our definition will not be very formal. The reader can refer to complexity textbooks such as [AB09] for more precisions.

In order to classify problems, we must be able to compare two problems and we need a tool to decide whether one is harder than the other. This is done via *polynomial-time reductions*. A decision problem B is polynomial-time reducible to another problem A if there is a polynomial time algorithm which solves B given an oracle for A (i.e., a black box solving instances of A). This means that the existence of a polynomial time algorithm for A implies the existence of a polynomial time algorithm for B, and hence A is at least as hard as B. Given a class of problems **X**, a problem is said to be **X**-hard if it is at least as hard as any problem in **X**. When a problem is both in **X** and ***X**-hard*, it is called ***X**-complete*.

polynomial-time reductions

X-hard

X-complete

The two most studied complexity classes are ***P*** which is the class of problems which can be solved in polynomial time, and ***NP*** which consists in problems which can be solved by a non-deterministic polynomial-time algorithm. Non-determinism means that the algorithm is allowed to make (non-deterministic) choices, and answer **yes** if at least one of the choices leads to answering **yes**. This can be reformulated as saying that **yes** instances to the problem have polynomial-size certificates. A certificate is in some sense a 'proof' that a given instance is a **yes** instance. Hence, an instance of the problem is a **yes** instance if and only if a certificate exists, and checking the validity of a given certificate can be done in polynomial time. Many traditional problems are problems in **NP**. Indeed, many of them are search problem, and essentially ask "Is there a substructure in the instance which satisfies some properties?". In these cases, the description of the substructure gives a certificate for the instance.

P

NP

We know that $\mathbf{P} \subseteq \mathbf{NP}$, but deciding whether this inclusion is strict is one of the main open problems in computer science. It is however widely believed that $\mathbf{P} \neq \mathbf{NP}$, and many results were derived based on the assumption that this conjecture holds. Many standard problems from computer science are in the class **NP**(some example of problems are given in the next section), and an important amount of research has been devoted to deciding whether these problems were in **P** or are **NP**-hard.

If **NP** contains most of the standard problems from computer science, many of the problems we consider in this thesis are in fact not in **NP**, but are harder than these problems. They belong to another complexity class called ***PSPACE***, which is the class of problems which can be solved using polynomial space. It is known that $\mathbf{NP} \subseteq \mathbf{PSPACE}$, but again, it is still open whether this inclusion is strict or not. In a similar way as above, we can consider problems which can be solved by a non-deterministic algorithm which uses only polynomial space. This class is called ***NPSPACE***. It was proved by Savitch that for this class non determinism does not help. In other words, $\mathbf{PSPACE} = \mathbf{NPSPACE}$.

PSPACE

NPSPACE

Showing that a problem is hard for a complexity class is usually done by a reduction from another problem whose hardness was shown using other ways. For example, boolean satisfiability, SAT, is a well-known **NP**-complete problem, and most **NP**-hardness proofs are done directly or indirectly through a reduction from SAT. The typical problem which is complete for **PSPACE** is called QBF. It is an extension of SAT with the addition of universal and existential quantifiers. More precisely, we are given as input a formula of the from $Q_1 x_1, \dots Q_n x_n,\ \phi(x_1, \dots x_n)$, where ϕ is

QBF

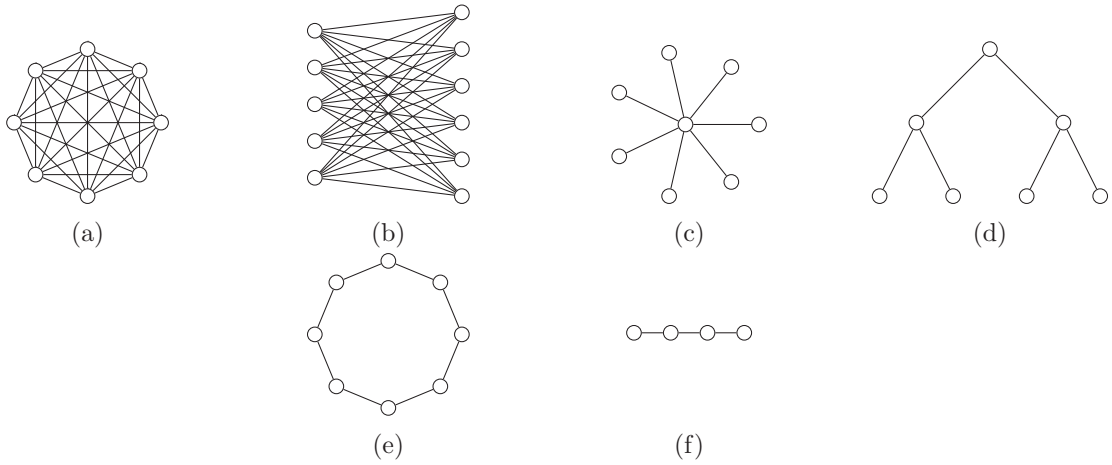


Figure 1.1: Examples of graphs. (a) The clique K_8, (b) the complete bipartite graph $K_{5,6}$, (c) the star with seven leaves, or equivalently $K_{1,7}$, (d) a complete regular tree with depth 2, (e) the cycle on 8 vertices C_8, (f) the path on four vertices P_4.

a boolean formula on n variables, and $Q_i \in \{\exists, \forall\}$ is either a universal or an existential quantifier. The problem consists in deciding whether this formula is true or false. The class **NP** corresponds to the case where all the quantifiers are existential. In a similar fashion, there is another class called **coNP** which corresponds to universal quantifiers only. There are also intermediate classes between **NP** and **PSPACE** which are defined by fixing the number of alternations between the two types of quantifiers in the formula. These classes form the *polynomial hierarchy* (see [AB09] for more details).

polynomial hierarchy

An important number of problems studied in the literature from a complexity point of view are problems on graphs; and many of the problems that we consider in this thesis are related to graphs. We will give a few examples of such problems in the section below.

1.3 Graph Theory

We give in this section some basic notations and definitions from graph theory. The interested reader can refer to [BM76] or any other textbook from graph theory for more details. A *graph* G is defined by a *set of vertices* denoted by $V(G)$, and a *set of edges* $E(G)$ which is a subset of $V(G) \times V(G)$. In the rest of this thesis, unless mentioned otherwise, all the graphs will be undirected: if $(x, y) \in E(G)$, then $(y, x) \in E(G)$; with a finite number of vertices; and simple: they do not have multiple edges or loops. For clarity, we will simplify the notations and write xy the edge (x, y).

graph
set of vertices
set of edges

Given a graph G, a graph H is a *subgraph* of G if it can be obtained from G by removing vertices and edges from G. It is an *induced subgraph* if it can be obtained only by removing vertices (and the edges incident to these vertices). Given $S \subseteq V(G)$ a subset of vertices, we will denote by $G[S]$ the subgraph induced by the vertices of S. In other words, the vertex set of $G[S]$ is S, and its set of edges is $E(G) \cap (S \times S)$.

subgraph
induced subgraph

If u and v are two vertices of G, then they will be called *neighbours* (or equivalently, they are *adjacent*) if $vu \in E(G)$. We will denote by $N_G(v)$ the set of neighbours of v (also called the

neighbours
adjacent

neighbourhood of v), formally defined by $N_G(v) = \{u \in V(G),\ uv \in E(G)\}$. When the graph G in this definition is clear from the context, we will drop the subscript and simply write $N(v)$ the neighbours of v. A graph G is *regular* if all the vertices have the same number of neighbours, i.e., $|N_G(v)|$ is the same for all vertices v.

neighbourhood

regular graph

Finally, given a graph G and two vertices u and v, a *path* between u and v is a sequence of distinct vertices $w_0 = u, w_1, \ldots, w_k = v$ such that for all $0 \leq i < k$, w_i and w_{i+1} are adjacent in G. The *length* of this path is k, the number of vertices on the path. Hence, if the two vertices u and v are adjacent, then they form a path of length 2. A graph G is said to be *connected* if there is a path (of arbitrary length) between any two vertices in G. If G is not connected, then we can find subsets $S_1, \ldots, S_k$ such that the subgraphs $G[S_i]$ are all connected, and there is no path between any two vertices in different subsets. In this case, the induced subgraphs $G[S_i]$ are called the *connected components* of G.

path

length of a path

connected graph

connected components

Finally, the *line graph* G^ℓ of G is the graph with vertex set $E(G)$ where two edges are adjacent in G^ℓ if and only if they are incident in G (i.e., they have a common endpoint). Note that many graph problems are defined in terms of vertices of G, but most of these problems admits an edge variant by considering line graphs.

line graph

1.3.1 Graph parameters

It is often convenient to look at some parameters of a graph G. These parameters allow to quantify some properties of the graph. We give here a few examples of standard parameters studied in the literature, but there are many others.

- The *maximum degree* of a graph G is the size of the largest neighbourhood: $\Delta(G) = \max\{|N_G(v)|,\ v \in V(G)\}$. Often, the graph G will be clear from the context, in which case we simply write Δ the maximum degree.

  maximum degree, $\Delta(G)$

- The *distance* between two vertices u and v is the number of edges in a shortest path between u and v. The *diameter* of G is the maximum distance between any two vertices $u, v \in V(G)$. If G is not connected, then the diameter is infinite.

  distance between vertices

  diameter

- The *degeneracy* of a graph G, denoted $\mathrm{col}(G)$, is the smallest k such that there exists an ordering $v_1, \ldots, v_n$ of the vertices of G, such that every vertex v_i has at most k neighbours v_j with a smaller index (i.e., $j < i$). It is easy to see that $\mathrm{col}(G) \leq \Delta(G)$. Moreover, this inequality is an equality if and only if G is regular[1].

  degeneracy, $\mathrm{col}(G)$

- A *clique* is a graph where all the vertices are pairwise adjacent. We denote by K_n the clique on n vertices (an example for K_8 is given in Figure 1.1a). The *clique number* of a graph G, denoted $\omega(G)$ is the size of the largest clique in G.

  clique, K_n

  clique number, $\omega(G)$

There are also other parameters that we will consider. Some of these come from particular graph problems that we introduce below such as the chromatic number $\chi(G)$, or the independence number $\alpha(G)$. Others are connected to particular classes of graphs such as the treewidth $\mathrm{tw}(G)$.

For some very common graphs, we will use special notations. We have seen the case of the clique K_n above. We will denote by P_n a *path* on n vertices, and C_n a *cycle* on n vertices as shown in Figures 1.1e and 1.1f.

path graph P_n

cycle graph, C_n

[1] Assuming G is connected.

1.3.2 Graph problems

Since graphs can be used to represent a large variety of different kinds of data, many different graph problems have been considered in the literature. We define here two of these problems that will be considered in the following chapters.

independent set
independence number, $\alpha(G)$

Independent set and matchings. Given a graph G, an *independent set* is a subset of vertices $S \subseteq V(G)$ such that no two vertices in S are adjacent. The size of the largest independent set in G is the *independence number* and is denoted by $\alpha(G)$. Deciding whether a graph contains an independent set of a given size is a well-known **NP**-complete problem. Even approximating $\alpha(G)$ up to a factor $n^{1-\varepsilon}$ for a constant ε is difficult. Note that finding an independent set in G is equivalent to finding a maximum clique in $\overline{G}$, the complement of G (obtained from G by adding all the edges not in G, and removing all the edges in G). Hence computing $\omega(G)$ or $\alpha(G)$ is equally hard in general graphs.

matching
perfect matching

Other objects related to independent sets are *matchings*. A matching is a collection of edges such that no two edges in the matching share a common endpoint. A matching in G correspond exactly to an independent set in G^ℓ, the linegraph of G. If a matching is incident to all the vertices of the graph, then it is called a *perfect matching*. Contrarily to the independent set problem, both computing a matching of maximum size, or deciding whether a graph contains a perfect matching are problems with polynomial time algorithm on general graphs.

k-colouring
proper colouring

Colouring. Given an integer $k \geq 0$, we will denote by $[k]$ the set of integers $\{1, \ldots, k\}$. A *k-colouring* α is a function $\alpha : V(G) \mapsto [k]$ which assigns to each vertex v a colour. A k-colouring is *proper* if two adjacent vertices are assigned different colours. Unless specified otherwise, all the colourings we consider are proper, and we will omit to specify it. Colourings and independent sets are related. Indeed, if we consider the set of vertices with a given colour c, then this set is an independent set. Hence, a colouring can be thought as a partition of the graph into several independent sets. It is not difficult to see that given a graph G, we can always find a colouring of G provided we have sufficiently many colours (for example, assign a different colour to each vertex).

chromatic number, $\chi(G)$

The smallest k such that a graph G admits at a k-colouring is called the *chromatic number* of G, and is denoted by $\chi(G)$. As was the case for the independence number, computing the chromatic number of a graph is **NP**-complete, and is also hard to approximate. Even deciding whether a graph is 3-colourable is **NP**-complete. However, there exist bounds on the chromatic number in terms of other graph parameters.

By colouring the vertices of the graph one by one, it can be easily seen that we can always find a colouring using at most $\Delta(G)+1$ colours. In fact we have the following inequality: $\chi(G) \leq \text{col}(G)+1 \leq \Delta(G)+1$. Note that a clique on n vertices has chromatic number n. From this observation, it follows that the chromatic number is at least the size of the largest clique: $\chi(G) \geq \omega(G)$. Graphs

perfect graphs

for which this inequality is an equality for all their induced subgraphs are known as *perfect graphs*. On standard question is to look at how much larger the chromatic number is, compared to the size of the maximum clique. In general, there is no direct relation between the two parameters (there are triangle-free graphs with arbitrarily large chromatic number), however, classes of graphs for

χ-bounded

which the chromatic number is upper bounded by a function of $\omega(G)$ are called *χ-bounded*.

Kempe chain

Given a graph G and a colouring α, a *Kempe chain* of G is a connected 2-coloured subgraph of G which is maximal inclusion-wise. Stated differently, if we consider two colours a and b, and consider the subgraph $G_{a,b}$ induced by the vertices coloured a or b, then a Kempe chain is a connected

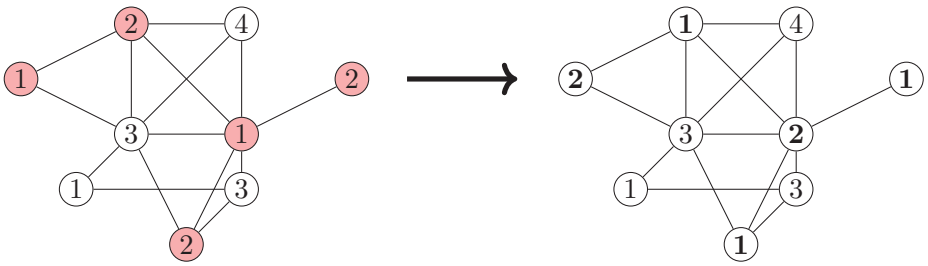


Figure 1.2: Example of a Kempe chain (vertices in red), with the corresponding Kempe exchange.

component of $G_{a,b}$. A *Kempe-exchange* consists in swapping the two colours in a Kempe chain. An example of a Kempe chain with the corresponding Kempe exchange is shown on Figure 1.2. Note that performing a Kempe exchange never creates monochromatic edges. Consequently, an initially proper colouring remains proper after applying some Kempe exchange. In some cases, a Kempe chain can be reduced to a single vertex. In this case, the corresponding Kempe exchange simply consists in changing the colour of that vertex. Given an initial colouring α, a colour c and a vertex v, we will denote by $\langle v, c\rangle$ the Kempe exchange which recolours v from $\alpha(v)$ to c. Note that a vertex w is recoloured by this exchange if and only if there is a path from v to w using only the colours $\alpha(v)$ and c.

Kempe-exchange

A very classic extension of colouring is called *list colouring*. A *list assignment* for the graph G is a function $L : V \to 2^{[k]}$ which assigns to each vertex of the graph a list of possible colours. An *L-colouring of G* is a k-colouring $\mu \in \Omega_V$ satisfying $\mu(v) \in L(v)$ for all $v \in V$. Since k-colouring is a special case of list colouring when all the lists are equal to $[k]$, the problem of deciding whether a graph admits an L-colouring for a given list assignment L is at least as hard as the k-colouring problem.

list colouring
list assignment

If the problems we just mentioned are hard in general, there are a certain number of cases where polynomial time algorithms still exists if we add some assumptions on the graph we consider. In particular, a popular approach to these problems consists in finding for which family of graph (also called graph classes) there problems are hard. We present in the next section some classical graph classes.

1.3.3 Graph classes

A *class of graph* is a family of graphs satisfying some property. Note that for all the classes below, there are polynomial time algorithms to decide whether an arbitrary graph is an element of the class. An exhaustive list of graph classes and their inclusion can be found in [dR+].

class of graph

Trees are graphs which contain no cycle. An example is given in Figure 1.1d. The *leaves* of a tree are the vertices with degree 1, and the *internal vertices* of the tree are the vertices which are not leaves. Sometimes, it is convenient to consider rooted trees, where one vertex of the tree is special and is called the *root* of the tree. If a tree is rooted, then the *parent* of a vertex v denotes the unique neighbour of v on a shortest path between v and the root of the tree. The *children* of v denotes the other neighbours of v. A tree is a *complete tree* if all the internal vertices different from the root have the same degree d (the root has degree $d-1$), and the paths from the root to a leaf all have the same length. If a tree contains a single internal vertex, then it is called a *star* (see Figure 1.1c). Trees have been used for example in biology to represent the evolution of different

tree
leaves
internal vertices
root
parent
children
complete tree
star

species, or for syntactic analysis.

bipartite graph

Bipartite graphs are graphs whose vertex set can be partitioned into two sets A and B such that every edge in the graph contains one vertex in A and one vertex in B. Equivalently, these are the graphs which admit a 2-colouring. They can also be defined as graphs which have no cycle of odd length. Note that in particular, trees are bipartite graphs. A *complete bipartite graph* is a bipartite graph where all the edges are present between the two parts. We denote by $K_{a,b}$ the complete bipartite graph with $|A| = a$ and $|B| = b$. An example is given in Figure 1.1b for $K_{5,6}$.

complete bipartite graph, $K_{a,b}$

interval graph

Interval graphs are intersection graphs of segments in a line. In other words, an interval graph can be represented as a collection of segments in a line. Each segment represents a vertex of a graph, and there is an edge between two vertices if their corresponding segments intersect. A particular subclass of interval graph is called *unit interval*, when the intervals in the representation have all the same length. Interval graphs are a particular subclass of the following. Interval graphs appear frequently for problems related to scheduling.

unit interval graph

chordal graph

Chordal graphs are a generalisation of interval graphs and trees. There are several equivalent ways to define them. They can be defined as intersection graphs of subtrees of a tree. Equivalently, they are the graphs which admits a *perfect elimination ordering*: an ordering of the vertices $v_1, \ldots, v_n$ such that for every i, $N(v_i) \cap \{v_1, \ldots v_i\}$ is a clique [Dir61]. This means that we can obtain the empty graph by removing iteratively a vertex whose neighbourhood is a clique. Chordal graphs are a subclass of perfect graphs. A special case of chordal graphs are *split graphs* which are the graphs G which can be partitioned into two sets A and B such that $G[A]$ is a clique, and $G[B]$ is an independent set.

perfect elimination ordering

split graph

Chordal graph are also related to another graph parameter called the treewidth [BB73]. The *treewidth* of a graph G is the smallest k such that G is a subgraph of a chordal graph with maximum clique at most $k+1$. The treewidth is a measure of how much a graph looks like a tree. In particular, trees have treewidth equal to 1. Many problems are known to be polynomial on graphs of bounded treewidth.

treewidth, $tw(G)$

planar graph

Planar graphs are graphs which can be drawn on the plane such that no two edges intersect. It is known that if such a drawing exists, then there exists a drawing using only straight line segments [Fár48]. A planar drawing of a planar graph defines faces which are the cells delimited by the edges of the drawing. If $F(G)$ denotes the faces of the drawing, then *Euler's formula* relates the number of vertices, edges and faces of a planar graph with the following relation: $|V(G)| - |E(G)| + |F(G)| = 2$. Using Euler's formula, it is possible to show that every planar graph has degeneracy at most 5. If this means that every planar graph can be coloured with at most 6 colours, a major result known as the four-colour theorem showed that they are in fact 4-colourable [AH89]. When all the vertices of the planar graph are incident to the same face, then the graph is called *outer-planar*. Outer-planar graphs have degeneracy 2, and consequently are always 3-colourable. Planar graphs are relevant when studying physical networks such as road networks.

Euler's formula

outer-planar

cograph

Cographs are graphs with no induced path of length 4. They were introduced in [CLSB81] and can be recognized in linear time [CPS85]. Many problems which are **NP**-hard on general graphs admit polynomial-time – often even linear-time – algorithms for cographs. For example, the treewidth and the chromatic number of a cograph can be determined in linear time [CHMW87]. Alternatively, cograph can be defined as the class of graph defined by a the recursive characterization:

- A graph consisting of a single vertex is a cograph.

- If G and H are cographs, then their disjoint union is a cograph, that is, the graph with the vertex set $V(G) \cup V(H)$ and the edge set $E(G) \cup E(H)$ is a cograph.
- If G and H are cographs, then their complete join is a cograph, that is, the graph with the vertex set $V(G) \cup V(H)$ and the edge set $E(G) \cup E(H) \cup \{vw \mid v \in V(G), w \in V(H)\}$ is a cograph.

From this characterization of cographs, we can naturally represent a cograph G by a binary (i.e., internal nodes have 2 children) tree, called a cotree of G. A *cotree* T of a cograph G is a binary tree such that each leaf of T is labelled with a single vertex in G, and each internal node of T has exactly two children and is labelled with either *"union"* or *"join"* labels. The cotree is such that two vertices of the cograph are joined by an edge if and only if their first common ancestor in T is a join-node. The cotree of a given cograph G can be constructed in linear time [CPS85]. The cotree of G is not necessarily unique but some properties do not depend on the choice of a cotree T. For example, a cograph is connected if and only if the root of T is a join-node.

cotree

union node
join node

1.4 Markov Chains

The goal of this section is to give the reader unfamiliar with Markov Chain theory a very simple introduction to some of the concepts. These notions will be used essentially in Chapter 5, and are not prerequisites for the other chapters. Note that many of the definitions given here can be generalized. However, we try to keep things as simple as possible. The reader can refer to [LP17] for a very nice and detailed introduction on the subject.

A *Markov chain* is a random walk on a graph, given by a sequence of random variables $(X_t)_{t\geq 0}$. The variable X_t represents the state of the Markov Chain after t steps. The graph associated to the Markov Chain is directed, can have loops, and has positive weights on its edges. The weights represent the probabilities of moving from one vertex to another. Since weights represent transition probabilities, we must have that at any vertex, the weights of all the out-going edges sum to one. Given a position at time t (i.e., a vertex of the graph), the position at time $t+1$ is obtained by picking one of the outgoing edges at random using their weights as the probability distribution, and moving to the vertex pointed by this edge. A simple example is given in Figure 1.3.

Markov chain

The *state space* of a Markov Chain is the set of vertices of the underlying graph, and will be denoted Ω. In all the following, we will always assume that Ω is finite, i.e., there is only a finite number of possible positions. Markov Chains can also be defined with an infinite state space, but some of the results we mention no longer hold in this setting.

state space

A Markov Chain is often represented by a *transition matrix* P, which corresponds to the (weighted) adjacency matrix of the underlying graph. The transition matrix is indexed by the

transition matrix

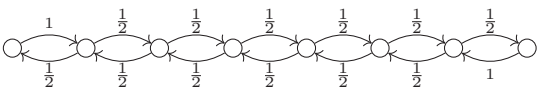


Figure 1.3: Example of a simple Markov chain. At each step, there is a $\frac{1}{2}$ probability to move either to the left, or to the right (except at the border). This chain is irreducible since the graph is strongly connected, but it is not aperiodic: after an even number of steps, we must be at an even distance from the starting position.

possible states i.e., the elements of Ω. Given $x, y \in \Omega$, $P[x \to y]$ represents the probability to be in position y at step $t+1$ conditioned on the fact that $X_t = x$, in other words:

$$\Pr(X_{t+1} = y | X_t = x) = P[x \to y] \ .$$

The matrix P is such that every row sums to 1. Given an initial distribution ν^0, we will denote by ν^t the distribution after t steps of the Markov Chain. We will also write ν_x^t the distribution after t steps when the chain starts at position x, or in other words $\nu_x^0(y) = \delta_{x,y}$, where $\delta_{x,y}$is the Kronecker symbol, equal to 1 if $x = y$, and 0 otherwise. Starting from an initial distribution ν^0, the distribution after one step is obtained by the matrix multiplication of ν^0 (seen as a vector) by the transition matrix P. More generally, we have $\nu^t = \nu^0 P^t$.

stationary distribution

A distribution π is said to be *stationary* if it is a fixpoint of the transition matrix, i.e., $\pi = \pi P$. Informally, this means that if you are initially distributed according to the stationary distribution, then you are still distributed according to the same distribution after one step of the Markov Chain. Stationary distributions play a very important role in the study of Markov Chains. Any (finite) Markov Chain always has at least one stationary distribution.

reversibility

We will say that a Markov chain is *reversible* if it satisfies the following equation for every $x, y \in \Omega$:

$$\pi(x) P[x \to y] = \pi(y) P[y \to x] \ .$$

detailed balance equations

These equations are often called the *detailed balance equations*. It is easy to check that any distribution which satisfy the detailed balance equation of a transition matrix P must be a stationary distribution of P. A particular case of reversibility is when the transition matrix is symmetric, i.e., $P[x \to y] = P[y \to x]$. In this case, the uniform distribution on Ω is a stationary distribution.

irreducibility

In order to study the stationary distributions of a Markov Chain, the two following properties are usually considered. A Markov Chain is *irreducible* if for every $x, y \in \Omega$, there exists a $t > 0$ such that $P^t[x \to y] > 0$. In terms of the underlying graph of the chain, this only means that the graph is strongly connected.

aperiodicity

A Markov Chain is *aperiodic* if for every $x \in \Omega$, there exists a $t_0 \geq 0$ such that for all $t \geq t_0$, we have $P^t[x \to x] > 0$. The notion of aperiodicity is more difficult to interpret. This condition is useful to prevent periodic behaviours which can occur such as the example in Figure 1.3. Note that in many cases, the aperiodicity of a chain is very easy to verify. For example, if for every $x \in \Omega$, $P[x \to x] > 0$, i.e., if there is a non-zero probability to stay in the same position, then the chain is aperiodic.

ergodicity

We will say that a chain is *ergodic* if it is both irreducible and aperiodic. In this case, the stationary distributions of the chain satisfy the following (see e.g. [LP17], Section 1.3):

Theorem 1. *If a Markov Chain with transition matrix P is ergodic, then:*

- *it has a unique stationary distribution π, and $\pi(x) > 0$ for any $x \in \Omega$;*
- *for any initial distribution X_0, the distribution after t steps converges to the stationary distribution, i.e.:*

$$\lim_{t \to +\infty} X_0 P^t = \pi$$

Note that if a chain is ergodic, this theorem does not give an estimation of how fast the convergence to the stationary distribution is.

1.4.1 Mixing Time

An important question on finite Markov Chains is to estimate how fast the chain converges to its stationary distribution. There are several measures which can be used to estimate this time. In order to estimate the speed of convergence, we first need a measure to quantify how far apart two distributions are. The measure usually used is the *total variation distance*. Given two probability distributions ν, μ on Ω, it is defined as

total variation distance

$$\mathrm{d_{TV}}(\nu, \mu) = \frac{1}{2} \sum_{x \in \Omega} |\nu(x) - \mu(x)| \ .$$

The most commonly used indicator for the time it takes for a Markov Chain to converge to its stationary distribution is the *mixing time* defined for any $\varepsilon > 0$ as

mixing time

$$t_{\mathrm{mix}}(\varepsilon) = \inf \left\{ t : \max_{x \in \Omega} \mathrm{d_{TV}}(\nu_x^t, \pi) < \varepsilon \right\} \ .$$

Note that often, this mixing time is only considered for $\varepsilon = \frac{1}{4}$, since it is known that below this threshold, the convergence to the stationary distribution is exponentially fast. Indeed, for all $\varepsilon < \frac{1}{4}$, we have:

$$t_{\mathrm{mix}}(\varepsilon) \leq t_{\mathrm{mix}}(1/4) \cdot \log \left(\frac{1}{\varepsilon} \right) \ .$$

We will simply write t_{mix} as a shorthand for $t_{\mathrm{mix}}(1/4)$. If the mixing time is often used due to its simple interpretation, there are cases where other quantities for estimating the convergence speed are easier to manipulate. If the Markov Chain is reversible, then we can define its *spectral gap* $\mathrm{Gap}(P)$ as the smallest non-zero eigenvalue (in absolute value) of $P - I$. The *relaxation time* is defined as

spectral gap

relaxation time

$$\tau(P) = \frac{1}{\mathrm{Gap}(P)} \ .$$

The relaxation time can be related to the mixing time by the following inequalities (see, e.g. Theorems 12.3 and 12.4 in [LP17]):

$$(\tau - 1) \cdot \log(2) \leq t_{\mathrm{mix}} \leq \tau \cdot \log \left(\frac{1}{\pi_{\min}} \right) \ , \tag{1.1}$$

where $\pi_{\min}$ is defined as $\pi_{\min} := \min_{x \in \Omega} \pi(x)$. Note that the mixing time and the relaxation time are not the only measures of the speed of convergence. Other measures include the bottleneck ratio and the log-Sobolev constant (see [LP17] for more details).

1.4.2 Continuous-time Markov Chains

We consider in this subsection a continuous-time version of Markov Chains. Informally, a *continuous time Markov Chain* can also be represented as a random walk on a directed graph with positive weights on the edges, but the update mechanism is slightly different. Instead of choosing the next position after exactly one unit of time, the time of departure from the current position is chosen according to a random variable with an exponential distribution[2] with parameter 1. As for the

continuous time Markov Chain

[2] A variable X is distributed according to an exponential distribution with parameter λ if for all $t \geq 0$, $\Pr(X \geq t) = e^{-\lambda t}$.

discrete-time version, at each transition, the next position is chosen at random with probabilities given by the outgoing edges.

A continuous time Markov Chain is usually described by its transition matrix $\mathcal{L}$. The non-diagonal values of this transition matrix correspond to the transition rates on each of the edges. The diagonal terms are chosen such that each row *sums to zero*. If $\nu(t)$ describes the probability distribution at time t, its evolution with time is given by the following differential equation:

$$\frac{d\nu}{dt}(t) = \nu(t)\mathcal{L} \ .$$

Solving this differential equation gives $\nu(t) = \nu_0 e^{t\mathcal{L}}$. All the notions we described before for discrete time Markov chains also work for their continuous-time counterparts with only few caveats. The continuous time version of Theorem 1 does not need the aperiodicity condition anymore. Indeed, since the departure time are chosen at random, the notion of aperiodicity does not make sense in the continuous-time setting. another point is that in the definition of spectral gap, we must replace $P - I$ by $\mathcal{L}$.

Given a discrete time Markov Chain with transition matrix P, we can define its continuous-time version using exactly the same transitions. In other words, we can consider the continuous time Markov Chain with transition matrix $\mathcal{L}$ such that $\mathcal{L}[x \to y] = P[x \to y]$ for any $x \neq y$ (note that the diagonal terms of $\mathcal{L}$ and P are different).

It is not difficult to see from this definition that a discrete time Markov Chain, and its continuous-time variant have exactly the same relaxation time. Moreover, their mixing time differs only by a constant factor.

Part I

Reconfiguration Problems

Chapter 2

Introduction to Reconfiguration Problems

This chapter is an introduction to reconfiguration problems. It gives notations and formal definitions used in the two next chapters, and motivates the study of these problems.

The chapter is organized as follows. In Section 2.1 formal definitions and notations on reconfiguration problems are provided. Section 2.2 gives some motivations and applications to reconfiguration. An overview of existing results for graph colouring reconfiguration is given in Section 2.3. Finally, Section 2.4 introduces Non-deterministic Constraint Logic, a tool frequently used in proving hardness results for reconfiguration problems.

2.1 Definitions and notations

reconfiguration graph, $\mathcal{G}^{\Pi}(I)$

Let Π be a problem and I be an instance of Π. The *reconfiguration graph* $\mathcal{G}^{\Pi}(I)$ is the graph whose vertices correspond to solutions of I and where there is an edge between two vertices if one can transform the first solution into the other in one step. The definition of "one step transformations" depends on the problem. Often, it consists in applying local modifications to the solution. In many cases, there is a very natural choice for the possible transitions. For example, for the reconfiguration of independent sets, it might consists in adding or removing a vertex; for reconfiguration of boolean formulas, it could be flipping the value of a single variable. One example is given in Figure 2.1 for colourings where the single step transformation consists in changing the colour of one vertex.

Recently, particular interest has been directed at the study of reconfiguration of graph problems including colourings, independent sets, vertex covers, matchings... Other problems have also been considered in the literature such as reconfiguration of boolean formulas [GKMP09] and word reconfiguration [Wro14b] just to name a few. The reader can refer to the following surveys [Heu13, Nis18] for a detailed overview of the results in the area.

Research on reconfiguration problems follows two main directions. The first direction is to study structural properties of the reconfiguration graph. The problem of finding sufficient conditions for the reconfiguration graph to be connected is one of the main questions of the area. An other widely studied property is the diameter of the components of $\mathcal{G}^{\Pi}(I)$. Other properties of the reconfiguration graph have been studied as well such as Hamiltonicity (also known as Gray codes) [CM11, Wil89], chromatic number [FMFPH$^+$12] or girth [AEH$^+$18].

A second research direction is focused on computational complexity. In particular, the following three problems have been widely considered:

- Π-Reachability: given an instance I of Π, and two solutions α and β of I, decide whether there is a transformation from α to β. In other words, decide whether α and β are in the same component of $\mathcal{G}^{\Pi}(I)$. Π-Reachability
- Π-Bound: given an instance I of Π, and two solutions of I, find the shortest transformation sequence between the two solutions. Π-Bound
- Π-Connectivity: given an instance I of Π, decide whether the the reconfiguration graph for I is connected or not (i.e., is there is a transformation sequence from any solution to any other?). Π-Connectivity

Efforts have been directed at classifying the three problems above in terms of their complexity. At first glance, it might seem that reconfiguration problems are in **NP** since they can be formulated using only existential quantifiers as follows: "Is there a valid transformation sequence from α to β?". However, the length of the transformation might not be polynomial, and because of this, reconfiguration problems can possibly be much harder. In fact, most reconfiguration problems are actually in **PSPACE**. Indeed, consider the following non-deterministic algorithm: starting from an initial configuration, guess the next step of the transformation among all the configuration adjacent to the current one until the target configuration is reached. By running this algorithm for 2^n steps (if configurations are represented with n bits) we are guaranteed to find a transformation if it exists. This shows that reachability is in **NPSPACE**, and consequently in **PSPACE** by Savitch's theorem. There are many examples of reconfiguration problems which are **PSPACE**-complete. We will mention some of them in the following sections.

Note that there are connections between the structural properties of $\mathcal{G}^{\Pi}(I)$ and the complexity of the reconfiguration problems. For example, if $\mathcal{G}^{\Pi}(I)$ is connected for any instance I, then Π-Reachability becomes trivial. In this case, the algorithm can simply always answer **yes** as a transformation always exists. If the connected components of $\mathcal{G}^{\Pi}(I)$ have polynomial diameter (in the size of I), then Π-Reachability is immediately in **NP**. Indeed, given two solutions α and β, a transformation from α to β is a polynomial size certificate that the input is a **yes** instance for Π-Reachability.

Although for many problems there is only one natural choice for the possible single step transformations, in some cases, there might also be several possible choice of transitions. For colouring reconfiguration, two types of transitions have been studied in the literature. More details on this are given in Section 2.3. Another example is the case of problems whose solutions can be represented by placing tokens on the vertices of a graph (for example independent set, or vertex cover). For these problems the following transitions have been considered:

- Token Sliding (TS): moving a token to an adjacent vertex,
- Token Jumping (TJ): moving a token to any other vertex,
- Token Addition and Removal (TAR): adding or removing one token.

In the first two cases, the size of the solution (i.e., the number of tokens) does not change. In the last case however, the size of the solution can change, and some bounds on the size of the solutions are often added to prevent the existence of a trivial transformation sequence. For example,

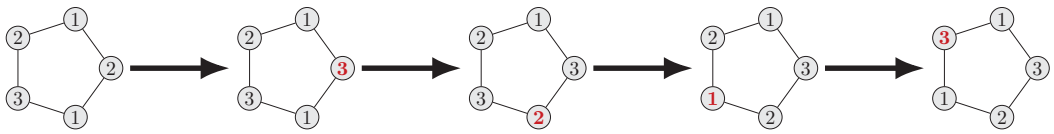


Figure 2.1: Example of reconfiguration sequence for the colouring problem. The vertices recoloured at each step appear in bold red.

consider independent set reconfiguration under TAR rule without any other constraints. A trivial transformation always exists by first removing all the tokens from the first solution, and then adding back the tokens of the second solution. Note that reconfiguration of independent sets has been considered on a variety of graph classes, and was shown to be **PSPACE**-hard under any of the three rules, even restricted to planar graphs [KMM12] or graphs of bounded bandwidth [Wro14b]. On the contrary, the edge variant of the problem, i.e., matching reconfiguration, was shown to be polynomial with any of the three rules on any graph [BKW14]. In Chapter 4 we consider the reconfiguration of perfect matching using a different kind of transformation.

Parametrized complexity has also been considered in the literature, using for example the length of the transformation as a parameter, and approximation results have been considered for the shortest transformation variant of the problem. A more detailed analysis of the existing results, both from a structural and from a complexity point of view can be found in the two surveys [Heu13, Nis18], as well as in the introduction of [Mou15]. We will give a detailed overview of the existing results for the colouring problem in Section 2.3.

2.2 Applications and motivations

Although the formalisation of the reconfiguration framework and its terminology is recent [IDH+11], the type of questions studied in this area has been considered for a long time. Many one player games (a.k.a. combinatorial puzzles) such as Rubik's cube, Rush-hour, or the 15-puzzle[1] fit in the reconfiguration setting. Indeed, in this case, the vertices of the reconfiguration graph are simply the set of possible configurations of the game, and edges are valid moves. The 15-puzzle for example has been studied since 1879 by Johnson and Story [JS79] who gave a characterization of the reachable configurations. It was then generalized as a game with tokens on arbitrary graphs, and a complete answer to the generalized version was given by Wilson [Wil74] in 1974.

Apart from the natural interest of understanding one-player games, there are many other motivations for studying these problems. Reconfiguration has a natural interpretation in terms of modifying a solution already in place in a dynamic setting. For example, colouring can be used as an abstract model for the frequency assignment problem: assigning frequencies to antennas while preventing antennas close to each other from interfering. In this case, due to evolving constraints we might want to change the existing assignment for an other more desirable one. For practical reasons such as maintaining the service running, or minimizing the costs of changing the solution, modifying the current solution in one go might not be feasible, but instead only local updates can be made. This corresponds exactly to the reconfiguration setting and was studied in [BLR06]. This type of model has also been considered for other problems such as monitoring nodes in a network [Mou15].

[1] "Jeu de takin" in French.

Reconfiguration problems are also strongly connected to Markov chains and random processes. These have applications to sampling random solutions, approximating the size of the state space (i.e., counting the number of solutions), and is linked to problems from statistical physics. The running time of some algorithms used for solving these problems is connected to some properties of the reconfiguration graph. More details on these connections are provided in Chapter 5.

All reconfiguration problems have in common that they are dealing with finding a path (i.e., a reconfiguration sequence) between two given vertices in the reconfiguration graph. The reconfiguration graph is not given as input to the problem, but instead is presented implicitly, via the instance I of the problem Π and the choice of transitions. Due to the large size of this graph, an exhaustive search is often not possible. Instead, finding a transformation sequence, or deciding that no such sequence exists must be done using the structural properties of the reconfiguration graph. This question of finding paths in very large graphs is also present in the robot motion planning problem. In this latter example, the state space is the set of all feasible positions of the robot which satisfy some constraints such as avoiding collisions with the environment. Several one player games such as Sokoban [CUL99] or token sliding problems on graphs can be viewed as simple abstract models for robot motions.

Another motivation for studying the properties of the reconfiguration graph comes from the need to analyse the performance of some heuristic algorithms, such as local search, which rely on local transformations for finding good solutions to a problem. By having a better understanding of the structure of the reconfiguration graph, it might be possible to better control or improve the performance of these algorithms. This is one of the reasons which motivated the firsts results on the reconfiguration of graph colouring [LVM81].

2.3 Colouring reconfiguration

For *k-colouring reconfiguration*, we are given as input a graph and two colourings of this graph, look whether a transformation sequence exists. In particular, the number of colours k is usually a constant and not part of the input. For this problem, two types of transitions have been studied in the literature. There is the single vertex transition that we mentioned above, for which only one vertex is recoloured at each step; and the more general Kempe exchange recolouring, for which we allow to transform a colouring by swapping the colours of an arbitrary Kempe chain. Recall from Chapter 1 that a Kempe chain is an inclusion-wise maximal 2-coloured subgraph, and a Kempe exchange consists in swapping the two colours in some Kempe chain. Kempe chains can be reduced to a single vertex, and for example, recolouring a single vertex is a valid Kempe exchange. Hence, every transition for single vertex recolouring are valid transformations for Kempe exchange recolouring.

k-colouring reconfiguration

To distinguish between the two types of transitions, we will specify explicitly when Kempe exchanges are allowed. In particular, $\mathcal{G}(k, G)$ denotes the reconfiguration graph for k-colourings of G under the single vertex update rule, while is the reconfiguration graph using Kempe-exchanges. Similarly, k-colouring-Reachability denotes the decision problem under single vertex transitions, while k-colouring-Kempe-Reachability allows for Kempe-exchanges.

$\mathcal{G}(k, G)$

$\mathcal{G}^{\mathrm{Kem}}(k, G)$

Before reviewing existing results for both types of transitions, it can be interesting to remark that the connectedness of the reconfiguration graph and its diameter are not necessarily monotone properties as a function of the number of colours. For example, there might be graphs such that the reconfiguration graph is connected with k colours, but not with $k + 1$ colours. An example of

Figure 2.2: A complete bipartite graph $K_{m,m}$ minus a matching in the case $m = 4$. With three colours, one of the side can always be recoloured with one of the three colours as the example on the left. On the right, a frozen colouring with 4 colours.

such a graph is the complete bipartite graph $K_{m,m}$ minus a perfect matching shown in Figure 2.2. If $m > 3$, the reconfiguration graph with 3 colours is connected. With m colours however, there can be frozen colourings (i.e., colourings with no possible transitions).

2.3.1 Kempe recolouring

Surprisingly, the Kempe chain transition was the first to receive attention from a reconfiguration point of view, starting with the works of Fisk [Fis77] and Meyniel [Mey78]. They showed that the reconfiguration graph $\mathcal{G}^{\text{Kem}}(k, G)$ is connected for respectively Eulerian planar triangulations with $k \geq 4$, and planar graphs with $k \geq 5$. These works were extended in [LVM81] in which it was proved that for $k \leq 5$, and all K_k-minor-free graphs G, $\mathcal{G}^{\text{Kem}}(k, G)$ is connected.

An important part of the work on transforming colourings with Kempe exchange was studied before the reconfiguration terminology was fixed. The focus has been mostly on finding sufficient conditions for the reconfiguration graph to be connected. A summary of the existing results is given in Table 2.3. An important result concerns a conjecture of Mohar [Moh06] that $\mathcal{G}^{\text{Kem}}(\Delta, G)$ is connected. This conjecture was proved recently in [FJP15, BBFJ19] for all graphs except the prism (two triangles with a perfect matching between them) for which there are two connected components in the reconfiguration graph.

Class of graphs	number of colours	Reference
Bipartite graphs	$k \geq 2$	folklore, e.g. [FS99, Moh06]
General graphs	$k \geq \text{col}(G) + 1$	[LVM81]
K_5-minor-free graphs	$k \geq 5$	[LVM81]
3-colourable planar graphs	$k \geq 4$	[Moh06]
Δ-regular graphs different from a prism	$k \geq \Delta$	[FJP15, BBFJ19]
Perfectly contractile graphs[2]	$k = \omega(G)$	[Ber90]

Table 2.3: Summary of the known sufficient conditions which ensure that the Kempe reconfiguration graph is connected.

[2]Perfectly contractile graphs are a subclass of perfect graphs which can be contracted to a clique of size $\omega(G)$ by merging two vertices with no induced odd paths between them.

The case of edge-colouring has also been considered by Mohar in [Moh06]. Using similar techniques as the proof of Vizing's theorem, it was showed that the reconfiguration graph for the edge-colourings of G is connected if $k \geq \chi'(G) + 2$, where $\chi'(G)$ is the chromatic index (equivalently the chromatic number of G^ℓ). Moreover, if G is bipartite, then the reconfiguration graph is already connected if $k \geq \Delta + 1$. The proof is constructive and provides a transformation of polynomial length. This result was improved in [MMS12] for subcubic graphs, for which it was shown that $k \geq 4$ colours was enough, and for subquadratic graphs for which $k \geq 6$ suffices to prove the connectedness of the Kempe-reconfiguration graph. Note that [MMS12] also provides examples of graphs with $\Delta > 3$ for which $\Delta + 1$ colours are not enough for the reconfiguration graph on edge-colourings to be connected. It is still open to decide whether $\Delta + 2$ colours are enough for all graphs or if $\Delta + 3$ is necessary for some graphs.

Open Problem 1. Either provide a graph for which $\Delta+3$ colours are needed for the reconfiguration graph on edge-colourings to be connected, or show that $\Delta + 2$ are enough for all graphs.

Note that if there exists an example G for which $\Delta + 3$ colours are necessary, then by the result of Mohar mentioned above [Moh06] we must have $\chi'(G) = \Delta + 1$. Other results on the number of connected components of the reconfiguration graph on edge-colourings with Kempe-exchanges were investigated in [BH14].

Apart from the results of [Moh06] on edge-colourings and bipartite graphs for which there is a simple $O(n)$ upper bound on the diameter of the $\mathcal{G}^{\text{Kem}}(k, G)$, none of the results in Table 2.3 provide any sub-exponential upper bound on the diameter of the reconfiguration graph. Upper bounds on the diameter obtained for single vertex recolouring directly give upper bounds for Kempe recolouring since recolouring one vertex is a valid Kempe move. Thus finding upper bounds on the diameter of the reconfiguration graph might be particularly interesting in cases where the reconfiguration graph is connected for Kempe recolouring, but not for single vertex recolouring.

Open Problem 2. Investigate the diameter of the reconfiguration graph for the cases in Table 2.3.

The case of d-degenerate graphs might be of particular interest, as similar work already exists for the single vertex reconfiguration. In particular, there is a conjecture on the diameter of the reconfiguration graph for d-degenerate graphs for single vertex recolouring (see Conjecture 1 below). One way to attack the conjecture could be to first try to first prove it for Kempe-chain recolouring.

Similar to the diameter, computational complexity aspects of the decision problems have not received a lot of attention. Our works in [BHI+19b, BHI+19a] (see Chapter 3) are first steps in considering both the diameter of the reconfiguration graph, and the computational complexity of the decision problems for the Kempe-chain variant. We show that 3-colouring-Kempe-Reachability is **PSPACE**-hard even restricted to bounded degree planar graphs with bounded bandwidth. We also show upper bounds on the diameter of the reconfiguration graph for chordal graphs and cographs. Finally, we show that the shortest transformation variant, k-colouring-Bound, is **NP**-hard, even on stars.

2.3.2 Single vertex recolouring

Complexity aspects

The research on k-colouring reconfiguration under the single vertex update rule was started much later. It was initiated with the works of Cereceda, van den Heuvel and Johnson [CHJ09, CHJ11,

BC09]. The computational complexity aspects of the problem have received a lot of attention. In particular, k-colouring-Reachability was proved to be **PSPACE**-hard if the number of colours is at least 4 and at most Δ [BC09, FJP14]. The problem remains **PSPACE**-complete even on planar graphs for $4 \leq k \leq 6$ and on bipartite planar graphs with $k = 4$ [BC09]. In both cases, the reconfiguration graph is connected if we have more colours, and consequently the problem becomes trivial.

The case of 3-colouring stands out compared to other reconfiguration problems. Despite the fact that the 3-colouring problem is **NP**-hard, its reconfiguration version, 3-colouring-Reachability, can be solved in polynomial time. Moreover, if a transformation exists, it has length $O(n^2)$ [CHJ11]. Additionally, 3-colouring-Connectivity is 'only' **coNP**-complete [CHJ09]. The hardness holds even for bipartite graphs but the problem becomes polynomial on planar bipartite graphs. In [CHJ08] the authors prove an alternative characterization of graphs G for which $\mathcal{G}(3, G)$ is connected.

For $k = \Delta+1$ colours, the reconfiguration graph has a simple structure. It was shown in [FJP14] that $\mathcal{G}(\Delta + 1, G)$ is composed of one single connected component with diameter $O(n^2)$, plus some isolated vertices. Additionally, the number of isolated vertices is small compared to the size of the large connected component [BBP18].

Note that although the complexity of k-colouring-Connectivity is known for 3 colours, it is surprisingly still open for $k \geq 4$. Given that Reachability is **PSPACE**-hard in this case, it would seems natural to think that Connectivity should also be difficult, but no hardness proof for the problem is known, and it can still be possible (although really surprising) that some structure of the reconfiguration graph makes the problem easy.

Open Problem 3. Prove that k-colouring-Connectivity is **PSPACE**-hard for $k \geq 4$.

In [HIZ17] the complexity of k-colouring-Reachability is studied for several graph classes. The problem is shown to be **PSPACE**-hard on chordal graphs, even with a constant number of colours. If the number of colours k is constant, the graphs constructed in the reduction also have bounded bandwidth. The problem is polynomial time solvable for 2-degenerate graphs, and has a linear time algorithm for split-graphs and trivially perfect graphs [Wro14b]. The problem is also hard on graphs of bounded bandwidth [Wro14b]. Note that for chordal graphs, k-colouring-Connectivity is easy. Indeed, it is enough to look at the size of the largest clique in the graph. If $k = \omega(G)$, then the vertices of the largest clique are frozen, and the reconfiguration graph is not connected. On the other hand, if $k > \omega(G)$, then the reconfiguration graph is connected since we must have $k \geq \omega(G)+1 = \mathrm{col}(G)+2$ (see Table 2.4). The complexity of k-colouring-Reachability is still open for interval graphs, and it is not clear what the exact complexity should be. A first step on this problem could be to consider unit interval graphs.

Open Problem 4. Study the complexity of k-colouring-Reachability on interval graphs.

Structural properties

The problem of finding properties which ensure the connectivity of the reconfiguration graph, and finding bounds on its diameter has been intensively studied. Results is this direction are summarized in Table 2.4. One interesting case is the case of d-degenerate graphs. It was proved in [DFFV06]

[3]The exponent in the polynomial depends on ε.

Class of graphs	number of colours	Diameter	Reference
d-degenerate graphs	$k \geq d+2$	$O(n^{d+1})$	Chapter 3, [BH19]
	$k \geq (1+\varepsilon)(d+1)$	$O(n^{\frac{1}{\varepsilon}})$	Chapter 3, [BH19]
	$k \geq \frac{3}{2}(d+1)$	$O(n^2)$	Chapter 3, [BH19]
	$k \geq 2d+2$	$O(kn)$	[BP16]
General graphs	$k \geq \Delta+2$	$O(\Delta n)$	[Cer07]
	$k \geq \text{mad}(G)+1+\varepsilon$	poly(n)[3]	[BP16, Feg19b]
	$k \geq \text{tw}(G)+2$	$O(n^2)$	[BB18, Feg19a]
	$k \geq \chi_g(G)+1$	$O(n\chi_g(G))$	[BB18]
Chordal-bipartite graphs	$k=3$	$O(n^2)$	[BJL+14]
Planar bipartite graphs	$k \geq 5$	$O(n^2)$	Chapter 3, [BH19]
Cographs	$k \geq \chi(G)+1$	$O(n^2)$	[BB14a]
Distance-hereditary graphs	$k \geq \chi(G)+1$	$O(n)$	[BB14a]

Table 2.4: Summary of the known sufficient conditions which ensure that the reconfiguration graph is connected under the single vertex recolouring, and the bounds on the diameter of the reconfiguration graph. χ_g is the grundy chromatic number: the worst number of colours used by a greedy algorithm to colour G.

that if $k \geq d+2$, then the reconfiguration graph is connected. A conjecture by Cereceda asserts that in this case the diameter of the reconfiguration graph is at most quadratic:

Conjecture 1 ([Cer07])**.** For any G, and any $k \geq \text{col}(G)+2$, $\mathcal{G}(k,G)$ has diameter $O(n^2)$.

The quadratic bound on the diameter is best possible. Indeed, there are graphs, for example some chordal graphs, for which the reconfiguration graph has quadratic diameter with $k = \text{col}(G)+2$ colours [BJL+14]. Note that the best known upper bound is exponential in the size of the graph, and even getting a polynomial upper bound on the diameter is still open. Several weaker versions of the conjecture were proved. Cereceda proved that the quadratic diameter holds under the stronger assumption that $k \geq 2d+1$. In [BJL+14] Cereceda's conjecture is proved for chordal graphs. This result was then generalized to graphs of bounded treewidth and $k \geq \text{tw}(G)+2$ colours in [BB18, Feg19a]. A polynomial upper bound on the diameter was shown in [BP16, Feg19b] if the number of colours is at least $k \geq \text{mad}(G)+1+\varepsilon$ for a fixed ε, and where mad(G) is the maximum of the average degree of H over all possible subgraphs H of G. This implies in particular a polynomial upper bound on the diameter for planar graphs and $k \geq 8$ colours. A quadratic diameter on planar graphs for 10 colours was shown in [Feg19c]. In [BH19] (see Chapter 3) we improve some of these results by showing that the reconfiguration graph has polynomial diameter on d-degenerate graphs if $k \geq d+2$, and d is constant. Additionally, the upper bound becomes quadratic if $k \geq \frac{3}{2}(d+1)$. For planar graphs, this implies an $O(n^6)$ bound on the reconfiguration graph with 7 colours, and a quadratic bound if $k \geq 9$.

On d-degenerate graphs, an other interesting question would be to investigate how many colours (as a function of d) are needed for the reconfiguration graph to have a linear diameter. It is known form [BP16] that $2d + 2$ are enough, but it is not clear if this number of colours is necessary, in particular for large values of d. Hence the following question:

Open Problem 5. Are there constants $\alpha < 2$ and $\beta \in \mathbb{N}$ such that for every d-degenerate graph G we have $\operatorname{diam}(\mathcal{G}(k, G)) \leq C_d \cdot n$ for some constant C_d whenever $k \geq \alpha \operatorname{col}(G) + \beta$?

Variants of the colouring problem

Reconfiguration problems have also been considered for variants of the colouring problem. The case of edge-colouring (colouring of the linegraph) for example is interesting. From a complexity point of view, it was shown in [OSIZ18] that k-edge-colouring-Reachability is **PSPACE**-complete even on planar graphs for any $k \geq 5$, while the case of 4 colours is still open.

Open Problem 6. Find the complexity of 4-edge-colouring-Reachability.

From a structural point of view, in the edge-colouring variant it is not clear how many colours are needed to make the reconfiguration graph connected. If a graph G has maximum degree Δ, its linegraph has maximum degree at most $2\Delta - 2$. Consequently, using the known results on vertex colouring reconfiguration, 2Δ colours are enough to make the reconfiguration graph on edge-colourings connected. On the other hand, the complete tripartite graph $K_{m,m,m}$ (see Figure 2.5) is an example which requires at least $\frac{3}{2}\Delta$ colours. Indeed, we can consider the colouring with $3m$ colours such that each colour forms a perfect matching between two of the three parts. This colouring is frozen, and the maximum degree of this graph is $\Delta = 2m$. Investigating how many colours are needed to make the k-edge-colouring reconfiguration graph connected seem to be an interesting problem.

Open Problem 7. What is the smallest $k = k(\Delta)$ such that the k-edge-colouring reconfiguration graph of any graph G is connected?

Other variants of the colouring problem have been considered in the literature. For example, list colouring has been considered from a complexity point of view. Since list colouring is more general than colouring, all the hardness results for k-colouring-Reachability also hold for the list-colouring variant. In general, there seems to be only very few cases for which the problem admits a polynomial time algorithm. List-colouring-Reachability was shown to be **PSPACE**-hard on

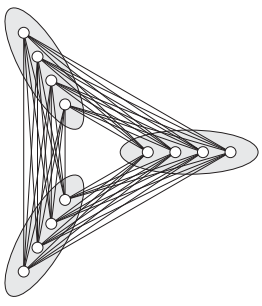


Figure 2.5: A complete tripartite graph on 4 vertices. Each of the parts are drawn in grey.

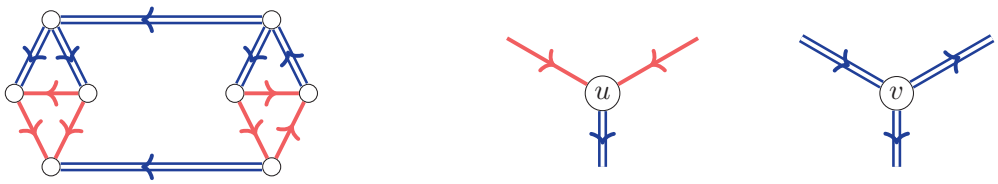


Figure 2.6: On the left, an example of an NCL machine with a valid orientation. Double edges in blue are weight 2 edges, the red ones are weight 1 edges. On the right an example of an AND node u, and an OR node v.

complete split graphs and graphs of pathwidth 2 [HIZ15]. On the other hand, it can be solved in polynomial time on graphs of pathwidth 1, i.e., caterpillars [HIZ15]. The complexity of the problem is still open on trees.

Open Problem 8. Find the complexity of List-colouring-Reachability on trees.

The problem k-colouring-Reachability and its list colouring variant have also been studied from a parametrized complexity point of view. If ℓ is the length of the transformation, the problem is W[1]-hard and in XP when parametrized by ℓ [BMNR14], and FPT when parametrized by $k + \ell$ [JKK+16]. Finally, other variations of the problem have been considered such as, reconfiguration of circular colouring [BN15, BMMN16], or homomorphism reconfiguration [BN15, Wro14a], or also reconfiguration of colourings in a distributed model of computation [BOR+18].

2.4 Hardness of reconfiguration problems

Computational complexity is often considered for reconfiguration problems. The general trend on reconfiguration is that decision problems which are **NP**-hard tend to be **PSPACE**-hard in their reconfiguration variant. There are exceptions to this trend, as we have seen for 3-colouring, however, this seem to hold for a wide variety of problems. Finding a general argument to justify this pattern is still an interesting open question. For decision problems which can be solved in polynomial time, there is no clear pattern on the complexity of their reconfiguration version. Some, like matching, are still polynomial in their reconfiguration variant [BKW14], and others like shortest path are **PSPACE**-hard [Bon13]. Due to the overall hardness of reconfiguration of graph problems, an important direction of research has been directed to restricting the graph to certain classes of graphs, trying to get dividing lines between hardness and polynomial time algorithms.

One of the most important tool for proving **PSPACE**-hardness of reconfiguration problems is called *Non-deterministic Constraint Logic*, or NCL for short. It is a reconfiguration problem which was created by Hearn and Demaine in [HD05] to prove the hardness of certain types of puzzles. Many hardness proofs of reconfiguration problems are done by reduction from NCL. We will use it in Chapters 4 and 3 in the case of reconfiguration of perfect matchings and colourings respectively.

Non-deterministic Constraint Logic (NCL)

An *NCL machine* (also called a constraint graph) is an undirected 3-regular graph together with an assignment of weights from the set $\{1, 2\}$ to each edge of the graph. We will call *node* the vertices of an NCL machine. Each node must be incident to an even number of edges with weight 1. An *NCL configuration* of this machine is an orientation of the edges such that the sum of weights of incoming arcs at each node is at least two. An illustration of an NCL machine and

NCL machine

node

NCL configuration

a valid configuration is given in Figure 2.6. With these definitions, we can see that there are two types of nodes:

- nodes incident to three edges of weight 2 are called OR nodes, because they behave like 'OR' gates from boolean circuits: at least one of the three edges must be directed inwards.
- nodes incident to one edge of weight 2, and two edges of weight 1 are called AND nodes for similar reasons. The edge with weight 2 is called the *output edge* of the AND node, the two others are the *input edge*. The output edge can be directed outwards only if the other two input edges are pointing inwards. Note that the output edge is not necessarily directed outwards even when both input edges are directed inwards.

output edge
input edge

The two types of vertices are illustrated in Figure 2.6. A reconfiguration step for NCL consists in swapping the orientation of a single edge. NCL-REACHABILITY was proved **PSPACE**-complete by Hearn and Demaine in [HD05], even if the constraint graph is planar. This result was later in improved in [Zan15] in which the following is proved:

Theorem 2 ([Zan15])**.** *NCL-*REACHABILITY *is* ***PSPACE****-complete, even if the NCL machine is restricted to planar graphs of bounded bandwidth.*

This result holds even if we allow edges with a neutral orientation [OSIZ18]. These edges are considered inwards for none of their two endpoints. The neutral orientation is useful for reductions, in particular if the gadget used in the reduction to represent edges of the NCL machines needs several steps to be reversed.

Chapter 3

Colouring Reconfiguration

This chapter presents several results on the reconfiguration of graph colouring. It considers both complexity aspects and structural properties of the reconfiguration graph. The results mentioned here appear in three articles [BHI+19b, BHI+19a, BH19].

As we have seen in Chapter 2, when considering reconfiguration of graph colourings there are two natural choices for the permitted transformations: single vertex recolourings, where it is allowed to recolour one vertex at a time; and Kempe recolourings for which we can swap the colours in a whole Kempe component. We consider both types of transitions in this chapter. We refer the reader to Section 2.3 for a detailed overview of the existing results on colouring reconfiguration. Since we consider two kinds of transitions, we will make explicit the cases where Kempe moves are allowed. Hence, Kempe-Reachability denotes the reachability problem for k-colouring using Kempe chain, while Colouring-Reachability is the same problem for the single vertex recolouring variant. Recall that $\mathcal{G}(k, G)$ denotes the k-colouring reconfiguration graph for single vertex recolouring, while $\mathcal{G}^{\text{Kem}}(k, G)$ denotes the reconfiguration graph under Kempe exchanges.

In the first part of this chapter (in Sections 3.1 and 3.2), we investigate recolouring with Kempe exchanges from a complexity point of view. We show in Section 3.1 that both Kempe-Reachability and Kempe-Connectivity are **PSPACE**-complete, even on bounded degree planar graphs and with just 3 colours. This contrasts with Colouring-Reachability which is known to be polynomial with 3 colours (and **PSPACE**-complete for $k \geq 4$) [BC09]. The results of Section 3.1 appears in [BHI+19a], in which we also provide upper bounds on the diameter of $\mathcal{G}^{\text{Kem}}(k, G)$ for several classes of graphs, such as cographs, chordal graphs and graphs of bounded treewidth, for various numbers of colours.

In Section 3.2, we consider the shortest transformation variant of the problem. We show that Kempe-Bound is already **NP**-complete, even on very simple graphs such as stars. In [BHI+19b], in addition to the hardness result on stars, we show that the problem is FPT (on stars) when parametrized by the number of colours, and give an approximation algorithm for the problem. We also show that Kempe-Bound is **NP**-complete on bipartite graphs, even with only 3 colours. In this case, it is also $W[2]$-hard when parametrized by the length of the transformation. Finally, we give an algorithm to find the shortest reconfiguration sequence on paths.

In the second part of the chapter, we study the single vertex recolouring variant of the problem from a structural point of view. We are particularly interested in the diameter of the reconfiguration graph. Motivated by Cereceda's conjecture, which states that $\mathcal{G}(k, G)$ has quadratic diameter if

$k \geq \text{col}(G) + 2$, we investigate the diameter of the reconfiguration graph for d-degenerate graphs. We show that this diameter is polynomial for $k \geq d+2$ if d is a constant. If we allow more colours, the diameter becomes quadratic if $d \geq \frac{3}{2}(d+1)$, and remains polynomial whenever $d \geq (1+\varepsilon)(d+1)$ for any constant $\varepsilon > 0$. These results appear in [BH19], where it is also proved that Cereceda's conjecture holds for planar bipartite graphs, and $k \geq 5$ colours.

3.1 Hardness of Kempe recolouring

In this section, we consider the complexity of colouring reconfiguration using Kempe exchanges. In contrast to the single vertex recolouring variant, the complexity Kempe recolouring has not received much attention up to now. We prove the following result.

Theorem 3. Kempe-Reachability *is PSPACE-complete, even with only three colours on planar graphs with maximum degree* 6.

The proof of the theorem is based on a reduction from Nondeterministic Constraint Logic (NCL for short, see definition in Section 2.4) and proceeds in two steps. First, we consider a list recolouring version of the problem and prove the hardness for this variant. The reduction from NCL is based on a construction of gadgets to simulate the different elements of an NCL machine. In the list colouring variant, the transitions between colourings are the same, but all the intermediate colourings must be proper list colourings of the graph. In particular, a Kempe exchange swapping two colours is permitted if and only if all the vertices in the chain have both colours in their lists.

A second step in the proof consists in modifying the construction made in the first step to remove the list constraints. This is done by adding some gadgets to the construction to ensure that transformations which would not be allowed in the list-colouring variant of the problem have no effect overall on the colouring of the graph. More precisely, these gadgets ensure that when attempting to perform one of these moves, it results in essentially swapping two colour classes.

3.1.1 List colouring reconfiguration

Our first step is to prove the following result which shows the hardness of recolouring with Kempe exchanges for the list colouring variant.

Lemma 4. *Both* Kempe-Reachability *and* Kempe-Connectivity *are PSPACE-complete for list colouring, even restricted to three colours, on planar graphs with maximum degree* 4.

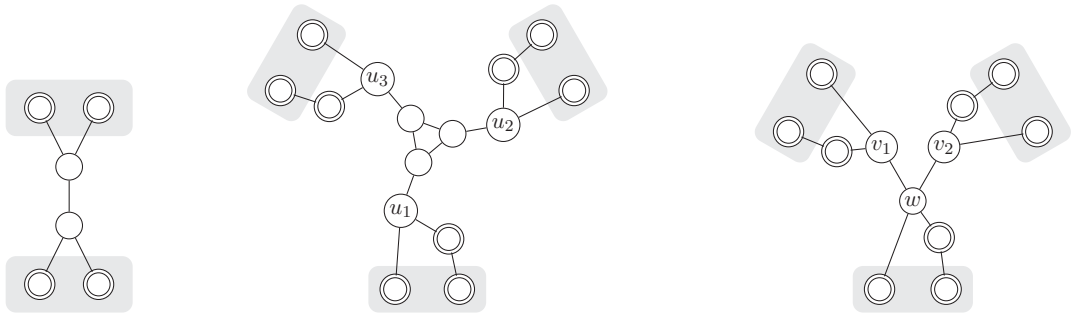
Note that the hardness result holds even if we only use two kinds of lists, i.e., vertices either can use all three colours, or can only use coluors 1 and 2. The rest of this subsection is dedicated to describing the reduction, and proving its correctness. In the following, M is an NCL machine, and G is a graph built from M using the gadgets described in Figure 3.1. The graph G is built using one gadget for each of the nodes and each of the edges of the machine M. These gadgets are glued together using the *connector vertices* (vertices in the grey areas on Figure 3.1). Hence, if one node u is incident to an edge e in the machine M, then the vertices in one of the connectors of the gadget for e are identified with the vertices in the corresponding connector of the gadget for u. A *connector pair* is a pair of vertices in the same connector, and we say that it is *monochromatic* if the two vertices have the same colour. The vertices which are not connectors will be called *internal vertices*.

connector vertices

connector pair

monochromatic connector

internal vertex



(a) (b) (c)

Figure 3.1: Gadgets used for the reduction. Vertices with double borders cannot take colour $\perp$. The grey areas are the connector. The different gadgets are assembled by identifying the connector nodes of one edge gadget and one node gadget. (a) Edge gadget (b) OR node gadget (c) AND node gadget. The bottom connector for this gadget corresponds the to the weight 2 edge.

There will be three colours. One, denoted $\perp$ will be a special colour, and plays a different role from the others. The other two, denoted 1 and 2, have symmetric roles. On Figure 3.1, the vertices represented with two concentric circles cannot be coloured $\perp$ (i.e., their lists of colours contains only the colours 1 and 2). The other vertices can take all three colours.

gate

The *gate* vertices for a node gadget are the three vertices u_1, u_2 and u_3 for an OR node, and v_1, v_2 and w for an AND node as shown in Figure 3.1. The two internal vertices of an edge gadget are also gates. Each connector is *associated* to the two gates it is adjacent to: one gate in a node gadget, and one in an edge gadget. These gates play an important role in the proof. Indeed, by colouring them with the colour $\perp$, we can 'cut' the $(1,2)$-bicoloured components in the graph G, ensuring in this way that the changes we make remain local and do not propagate throughout the graph.

associated vertices

The relation between a colouring of G and a valid orientation of the NCL machine M is done by looking at whether the connector pairs are monochromatic or not, as in Figure 3.2. If the connector pair on one end of the edge gadget is monochromatic, then the edge is oriented outwards for the corresponding node (i.e., inwards for the edge gadget). If they have different colours, then they are oriented inwards for the node (i.e., outwards for the edge gadget). Note that both extremities of the edge can be oriented inwards for the edge, in which case the edge will be said to be *neutral*. However, both extremities cannot be both oriented outwards. Indeed, if it was the case, this would force the two gates to be coloured $\perp$, which is not possible since they are adjacent. It is known that NCL reconfiguration remains **PSPACE**-complete even if we allow orientations with neutral edges [OSIZ18].

neutral edge

In the rest of this section, to remain coherent with the rest of the manuscript, α, β, γ will denote colourings, while σ and η denote orientations of an NCL machine. Given a colouring α of G, we denote by σ_α the corresponding orientation for the NCL machine M. Conversely, we denote by Ω_σ the set of colourings which correspond to the orientation σ. That is to say, Ω_σ is the set of colourings α such that $\sigma_\alpha = \sigma$). Note that σ_α only depends on the colours of the connectors in G.

We start with the two following observations:

Observation 5. *Given any colouring α of G, the internal vertices of G can be recoloured one at a time such that:*

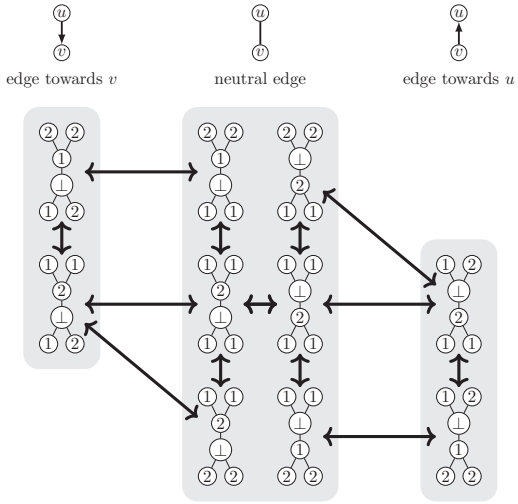


Figure 3.2: Reconfiguration graph for an edge gadget, and the corresponding orientation of the edge. Some symmetric cases were removed for clarity.

- *on each edge gadget, one of the two gates is coloured* $\perp$*,*
- *on each* OR *gadget, two of the three gates are coloured* $\perp$*,*
- *on each* AND *gadget with gates* v_1, v_2 *and* w *as in Figure 3.1c, either* w *is coloured* $\perp$ *or* v_1 *and* v_2 *are.*

Proof. On an edge gadget, if none of the two gates are coloured $\perp$, we can directly recolour one of the gates with $\perp$ since the connector vertices cannot be coloured with $\perp$.

On an OR gadget, the only neighbours of some gate which can be coloured $\perp$ are the vertices in the central triangle. Since only one of the vertices in this triangle can be coloured $\perp$, the two gates which are not adjacent to this vertex have no $\perp$ neighbour and can be directly recoloured $\perp$.

Finally, on an AND gadget, if w is not already coloured $\perp$ then the two other gates v_1 and v_2 do not have $\perp$ in their neighbourhood and can be recoloured $\perp$. □

Observation 6. *Let* α *be a colouring of* G*. Assume that there is a* $(1,2)$*-bichromatic chain intersecting a connector pair on a single vertex. Then the two gates adjacent to this connector must be coloured* $\perp$*, and the chain contains a single vertex.*

Proof. We prove the contrapositive, i.e., assuming that one of the gates is not coloured $\perp$, we will show that the two vertices in the connector pair are in the same $(1,2)$-Kempe component. Given two vertices in a connector pair, there are two paths on each side of the connector which connects the two vertices in the connector pair. On each of these two paths, only the gate vertex can take a colour different than $\perp$. Hence, if one of the two gates is not coloured $\perp$, then there is a path coloured with only 1 and 2 between the two vertices, and the two vertices are in the same $(1,2)$-Kempe component. □

The following lemma asserts that this relation between colourings of G and orientations of M preserves the constraint of the NCL machine M.

Lemma 7. *For any colouring α of G, σ_α is a valid orientation of M. Conversely, for any valid orientation σ, the set Ω_σ is not empty.*

Proof. Let α be a proper colouring of G. We need to show that the orientation σ_α satisfies the constraints of the NCL machine. Assume by contradiction that this is not the case, and let u be the node of G which does not satisfy its NCL constraints.

- If u is an OR node, then this means that all the edges incident to u are directed outwards at u and consequently, all the connectors of the node are monochromatic. Let u_1, u_2 and u_3 be the three gates in the OR gadget. Since each connector is monochromatic, u_i must be coloured $\perp$ for all $i \leq 3$. However, in this case the triangle in the center of the gadget must be coloured with only the colours 1 and 2 which is not possible.

- If u is an AND node, then the output edge (i.e., the edge with weight 2) must point outwards. This implies that the connector of the output edge is monochromatic. Let v_1, v_2 and w be the central vertices of the AND gadget as in Figure 3.1c. Then w can only be coloured $\perp$, and consequently u_1 and u_2 are both coloured with either 1 or 2. As in the previous case, this implies that the connector pairs of the input edges are not monochromatic, and the corresponding input edges point towards the node u, a contradiction of the assumption that the node u violated the constraints.

Conversely, let σ be an orientation of M. We want to show that there exists a proper colouring of G in Ω_σ. We build a colouring $\alpha \in \Omega_\sigma$ by first choosing the colour of the connectors according to the orientation σ. For example, if an edge $e = uv$ is oriented towards the node u, then the two vertices in the connector pair between e and u are coloured with 1 and 2 respectively, and the other connector pair of the edge gadget is monochromatic.

Then, for each edge gadget, we can complete the colouring by assigning a colour to the internal vertices. Indeed, for each edge gadget if the corresponding edge is oriented towards some node u, the gate of u can be coloured $\perp$, and the other gate still has one colour available since the other connector must be monochromatic.

Finally, we complete the colouring for each node gadget. In the case of an OR node, let u_1, u_2 and u_3 be the three gates. Then, since the orientation σ is valid, at least one edge is pointing outwards. This means that the corresponding connector is not monochromatic, and the vertex u_i incident to this connector can be coloured with either 1 or 2. The other u_j for $j \neq i$ can be coloured with $\perp$. Finally the central triangle can also be coloured since 2-list-colouring a triangle is always possible unless all the lists are the same which is not the case here.

In the case of an AND node, let v_1, v_2 and w be the gates of the corresponding AND gadget. If the output edge points inwards, we can colour v_1 and v_2 with $\perp$, and w with either 1 or 2. If the output edge points outwards, w can be coloured with $\perp$, and since the two other edges must point inwards, v_1 and v_2 can both be coloured with either 1 or 2 depending on the colour used for their respective connectors. □

In our construction, the orientation of the NCL machine corresponding to some colouring α only depends on the colours given by α on the connectors. The following lemma states that for the purpose of recolouring α into some other colouring, it is enough to only consider the colours of the connectors. In other words, there is always a transformation sequence between two colourings which correspond to the same orientation of M.

Lemma 8. *Let α and β be two colourings corresponding to the same orientation of the NCL machine. Then there is a transformation from α to β.*

Proof. Let α and β be two colourings in Ω_σ for some orientation σ of the NCL machine. Without loss of generality, we can assume that all three conditions of Observation 5 hold for both α and β. First, we will show that we can transform α and β such that both colourings agree on the connector nodes. Note that if they disagree on a connector, then this just means that the colours 1 and 2 in this connector are swapped between the two colourings. Given a connector where the two colourings disagree, we just apply one (or two) $(1,2)$-Kempe exchange to make the two colourings agree on the connector. Since the conditions of Observation 5 hold, the gate vertices coloured $\perp$ isolate the connectors from one another: no $(1,2)$-Kempe chain contains vertices from two different connectors. Hence, we are sure that the transformations on each connector can be performed independently. Note that after this operation, the three conditions of Observation 5 still hold for α and β.

Once the colourings agree on the connectors, we can transform them to agree on the edge gadgets. The only case where the two colourings might disagree is for neutral edges. In this case the transformation is given in Figure 3.2. Note that in some cases (for example, for the two colourings of the neutral edge at the top of Figure 3.2), we might need to temporarily recolour one connector. When doing this, the change does not propagate since the connectors of a given node gadget are isolated from each other by the $\perp$-coloured gates using Observation 5.

Finally, we can finish the transformation from α to β by making the two colourings agree on the node gadgets, without changing the colouring in the rest of the graph:

- For an AND gadget, let v_1, v_2 and w be the gates of the gadget. If the two colourings disagree on w, then the output edge must be inwards (since otherwise w has only the colour $\perp$ available). This implies that the connector associated with the output edge is not monochromatic. W.l.o.g. we can assume that the connector vertex adjacent to w is coloured 1 in both α and β. Thus, w must be coloured 2 in one of the two colourings (say, α), and $\perp$ in the other (say β). Since the conditions of Observation 5 hold, both v_1 and v_2 are coloured $\perp$ in α. Additionally, since w is coloured $\perp$ in β, v_1 and v_2 must be coloured 1 or 2 in β, which implies that the two corresponding connectors are not monochromatic. Hence, the two vertices adjacent to v_1 and different from w are either both coloured 1 or both coloured 2. If they are both coloured 2, then in α we can recolour v_1 from $\perp$ to 1. The same holds for v_2. Finally, after these operations, in α the $(\perp, 2)$-Kempe chain containing w might only contains the vertices v_1, v_2 and w. By swapping the colours in this chain, the two colourings agree on w.

  If the two colourings agree on w, then they can be made to agree on v_1 and v_2 by recolouring these two vertices individually.

- For an OR gadget, let u_1, u_2 and u_3 be the three gates. By Observation 5, in both α and β there are two gates coloured $\perp$, hence there is at least one gate, say u_3, which is coloured $\perp$ in both α and β. Let us first assume that the other gate coloured with $\perp$ is different in the two colourings. We can assume without loss of generality that u_1 is coloured $\perp$ in α but not in β, and conversely, u_2 is coloured $\perp$ in β but not in α.

  If $\beta(u_1) = \alpha(u_2) = x \in \{1, 2\}$, then after possibly applying a $(1,2)$-Kempe exchange on the two vertices in the central triangle coloured 1 and 2 in α, we can assume that u_1 and u_2 are in the same $(\perp, x)$ component in the colouring α. By swapping this Kempe chain, the two colourings agree on the the gate vertices, and the connector vertices are not modified by the operation.

If $\beta(u_1) \neq \alpha(u_2)$, then we can assume that $\beta(u_1) = 1$ and $\alpha(u_2) = 2$, the other case being symmetrical. Up to swapping the component composed of the two vertices coloured 1 and 2 in the central triangle, we can assume that the neighbours of u_1 are not coloured 1 in α. Indeed, the neighbours of u_1 are either vertices with constraints, which have the same colour in both colourings, and consequently must be different from $\beta(u_1) = 1$, or the vertex in the central triangle. If this vertex is coloured 1 in α, then the neighbour of u_3 in the central triangle must be coloured 2, and these two vertices form a Kempe component. By swapping the colours in this component, no neighbour of u_1 is coloured 1 in α. Hence, we can first recolour u_1 with $1 = \beta(u_1)$, and then swap the $(1, \perp)$-Kempe component containing u_2. After this operation, the two colourings agree on the gate vertices.

If the gates coloured $\perp$ are the same in both colourings, then the third gate must also have the same colour since at least one of 1 or 2 is used by the connector (remember that the two colouring agree on the connector vertices), and the three gates cannot be all coloured $\perp$. Finally, if the two colourings agree on the gates, then the vertex coloured $\perp$ in the central triangle is the same for both colourings since two of the gates are coloured $\perp$. The two colourings can be made to agree on the central triangle just by swapping the component composed of the two vertices coloured 1 and 2 in the central triangle.

This shows that we can transform α into β and concludes the proof. □

Proof of Lemma 4. To prove the lemma, we only need to verify that the two following properties hold:

1. If there is a transition between two colourings, then there is transformation between their corresponding orientations for the machine M.

2. If there is a transition between two orientations of the machine, then there is a recolouring sequence between their corresponding colourings.

Point 1. Let α and β be two colourings of G which differ only by a Kempe exchange. If $\sigma_\alpha = \sigma_\beta$, then there is nothing to prove, so we can assume that this is not the case. This implies that α and β differ by swapping the colours in a $(1, 2)$-Kempe chain, since no Kempe chain using the colour $\perp$ can recolour the connectors vertices. Since $\sigma_\alpha \neq \sigma_\beta$, there is at least one connector which changed state during the transition (i.e., was monochromatic before the change and is not after, or the contrary). This implies that the Kempe chains must contain only one of the two vertices of this connector. By Observation 6, the two gates associated to this connector are coloured $\perp$, and the chain is reduced to a single vertex. Consequently, σ_α and σ_β only differ at the edge e, and there is a transition between the two.

Point 2. Let σ and η be two orientations of the NCL machine M which only differ on an edge e. Without loss of generality, we may assume that e is neutral in σ, and points towards some node u in η. By Lemma 8, we only need to show that there exists $\alpha \in \Omega_\sigma$ and $\beta \in \Omega_\eta$ such that α and β only differ by a Kempe exchange. Let S be the connector linking the edge e and the node u. We take c to be a colouring of Ω_σ such that the two gates corresponding to the connector S are coloured $\perp$. We first show that such a colouring exists by building it in several steps. First, we can choose the colours of the connectors according to the orientation σ. We then extend this partial

colouring to the whole graph in such a way that the gates corresponding to S are coloured $\perp$. The proof is very similar to what we did for Lemma 8. We can extend the colouring for every gadget of an edge $e' \neq e$ and every gadget of a node $u' \neq u$ using exactly the same procedure as we did in the proof of Lemma 8. In the gadget corresponding to e, since e is neutral in σ, the connector different from S is monochromatic. We can assume w.l.o.g. that it is coloured 1. Then, the gate incident to S can be coloured $\perp$, and the other gate can be coloured 2.

Finally, if u is an OR node, then since σ is a valid orientation of M, there is an edge (different from e) incident to u oriented inwards for u. The corresponding connector is not monochromatic, and we can colour the corresponding gate with either 1 or 2. The other two gates can be coloured with $\perp$, and the colouring of the central triangle can be completed using the same argument as before.

If u is an AND node, and e is the output edge of u, then since σ is a valid orientation of M, the two other edges of e must be oriented inwards for u. Consequently, the connectors corresponding to these two edges are not monochromatic, and we can colour each of the corresponding gates with either 1 or 2. Finally, the gate for S can be coloured $\perp$.

If u is an AND node, and e is not the output edge of u, then since σ is a valid orientation of M, the output edge of u must be oriented inwards for u. This means the corresponding connector is not monochromatic, and we can colour its gate with either 1 or 2. The two other gates can be coloured $\perp$.

This concludes the existence of the colouring $\alpha \in \Omega_\sigma$ such that both gates of S are coloured $\perp$. Let β be the colouring obtained from c by applying a $(1, 2)$-Kempe exchange on one of the vertices of the connector S. Since the two gates are coloured $\perp$, the other vertex in the connector does not change colour, and the other connectors are left unmodified by the transformation. Hence $\beta \in \Omega_\eta$, which concludes the proof. □

3.1.2 Removing the constraints — Proof of Theorem 3

We now describe how to adapt the proof to remove the list constraints on the vertices in the gadgets. Remember that in the construction from previous section, every vertex can be either coloured with all three colours, or is constrained to a colour in the set $\{1, 2\}$. First observe that all the vertices with constraints in the construction (i.e., which cannot take colour $\perp$) have degree 2. The graph obtained from the construction in the previous section is modified by first replacing each of the constrained vertices by a gadget as in Figure 3.3.

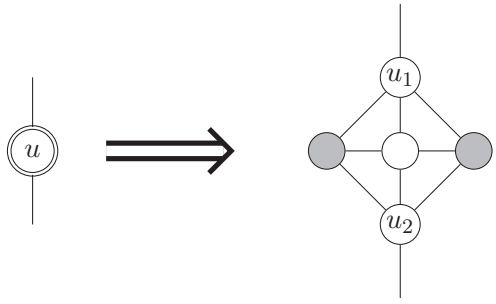


Figure 3.3: Transformation of the constrained vertices using the gadget above. The vertices in grey are here to enforce the $\perp$ constraint.

Note that in any 3-colouring, the vertices u_1 and u_2 in Figure 3.3 have the same colour. Thus, u_1 is changed by a Kempe chain, if and only if u_2 is. The vertices in grey on the figure represents our $\perp$ constraints, and will initially be coloured with colour $\perp$. We must ensure that at any time during a transformation, these vertices have the same colour across all gadgets. One way to ensure this would be simply to merge all the grey vertices together, into one single vertex. However, this operation does not preserve the planarity of the graph. Instead, we will use a chain of diamonds as shown in Figure 3.4a. Note that the extremities and the cut-vertices of the chain always have the same colour in any 3-colouring. Using these chains, we can form trees, and link all the grey vertices together as shown for example in Figure 3.4b in the case of the edge gadget. The other gadgets are treated in a similar way. Remark that this operation can be done while preserving the planarity of the graph. Additionally, the maximum degree is 6 and occurs only when three diamonds meet at a single vertex when connecting the chains together. We denote by G' the graph obtained by replacing each constrained vertex in G using the gadget from Figure 3.3, and connecting all the grey vertices together using chains of diamonds. This completes the construction, and we can now give the proof of Theorem 3.

Proof of Theorem 3. The proof is done by a reduction from the list variant of the problem that we proved to be **PSPACE**-hard in the previous subsection. Let G be the graph obtained by the construction from previous subsection with the list assignment L, and G' the graph obtained from G by applying the transformation above. Given a constrained vertex u in G, the two vertices u_1 and u_2 in G' obtained by the construction in Figure 3.3 will be called vertices *resulting* from u.

resulting vertices

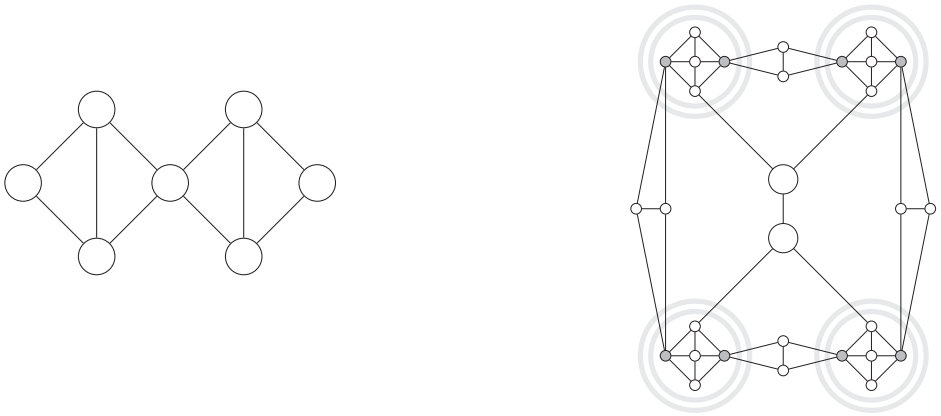
Let α' be a colouring of G'. We will assume in the following that, up to renaming the colours in α' that all the grey vertices in G' are coloured $\perp$. Given an L-colouring α of G, we will say that α' is an extension of α if the two colourings agree on the vertices in G, and for every constrained vertex u in G which was replaced by a gadget, the vertices resulting from u are both coloured $\alpha(u)$ in G'.

To prove the result, we only need to show that for any two L-colourings α and β of G, and any two extensions α' and β' of α and β respectively, then α and β are in the same component of $\mathcal{G}^{\mathrm{Kem}}(L, G)$ if and only if α' and β' are in the same component of $\mathcal{G}^{\mathrm{Kem}}(3, G')$. This follows from the following two observations:

- given a colouring α of G, all its possible extensions α' to G' are in the same connected component of $\mathcal{G}^{\mathrm{Kem}}(3, G')$;
- for any proper colouring of G', all the vertices which result from some constrained vertex in G and are coloured 1 are in the same $(1, \perp)$-Kempe chain. The same holds for these vertices which are coloured 2.

Let α and β two L-colourings of G, and α' and β' two extensions. First, assume that α and β differ by swapping the colours in a Kempe-chain. Then, by construction there is an equivalent Kempe chain in G'. By swapping the colours in this Kempe component for the colouring α', we obtain a colouring β'' which is an extension of β. By the first point above, β'' and β' are in the same Kempe component, and consequently, so is α'.

Reciprocally, assume that there is Kempe exchange which transforms α' into β'. If this Kempe exchange does not use the colour $\perp$ or does not recolour a vertex which results from a constraint vertex in G, then there is an equivalent Kempe exchange in G which transforms α into β (possibly, these two colourings are in fact equal). Otherwise, this Kempe exchange recolours at least one



(a) (b)

Figure 3.4: (a) A chain of diamonds used to connect together all the grey vertices. (b) Example in the case of an edge gadget. The area with double circles in grey represents the former vertices with constraints (here the connectors) that were replaced. The diamonds on either side can be replaced by longer chains to connect together the chains obtained from different gadgets.

vertex resulting from a constrained vertex from some colour x to $\perp$. By the second observation above, all the vertices resulting from a a constraint vertex in G which are coloured x in α' are recoloured with $\perp$. After the exchange the grey vertices in G' are coloured x. Hence, by renaming in β' the colour x in $\perp$, and $\perp$ in x, we can observe that β' is the extension of a colouring β which agrees with α on the constrained vertices. Since the two colourings agree on the constrained vertices, in particular they agree on the connector pairs, and consequently they correspond to the same orientation of the NCL machine. By Lemma 8, this implies that α and β are in the same connected component of $\mathcal{G}^{\mathrm{Kem}}(L, G)$.

Hence, α and β are in the same component of $\mathcal{G}^{\mathrm{Kem}}(L, G)$ if and only if α' and β' are in the same connected component of $\mathcal{G}^{\mathrm{Kem}}(3, G')$. This ends the reduction, and by Theorem 17, it shows that deciding whether there is a transformation between two given 3-colourings is **PSPACE**-hard. □

3.2 Shortest transformation on stars

In this section, we consider the shortest transformation version of the problem, i.e., the problem of finding a shortest reconfiguration sequence between two given colourings using Kempe exchanges. Unlike the problem we considered in the previous section, the number of colours is no longer constant, but is part of the input instead. We call Kempe-Bound this problem, formally defined as follows:

Kempe-Bound Kempe-Bound

Input: Two integers t and k, a graph G and two k-colourings α and β of G.

Output: Whether there is a transformation from α to β using at most t Kempe exchanges.

We show in this section that this problem is **NP**-complete, even on very simple graphs such as stars.

Theorem 9. Kempe-Bound *is* ***NP****-complete, even when restricted to stars.*

Note that the fact that the problem belongs in **NP** follows immediately from the existence of a transformation sequence of linear length between any two colourings of a star. Hence we only need to show the hardness part. Note that in [BHI+19b], we also considers the parametrized complexity aspects of the problem: we prove that Kempe-Bound on stars is FPT when parametrized by the number of colours. We also give an approximation algorithm for the problem when the number of colours is not bounded.

The main element to the proof of **NP**-hardness is the notion of a *sorted transformation* that we present in the next subsection. More precisely, in the next subsection we show that, for the purpose of finding a transformation of shortest length, we can assume that the transformation occurs in a particular order. Using the order of the transformation, the problem of finding a shortest transformation is equivalent to finding a special intermediate colouring. Given this intermediate colouring, computing the length of the transformation sequence can be done easily, and will follow from two lemmas below (Lemmas 12 and 13). The hardness proof consists in showing that finding this special intermediate colouring is difficult, by a reduction from the Hamiltonian Cycle problem.

3.2.1 Sorted Transformations

We start by observing that on a star, given an initial colouring α, there are two types of possible Kempe exchanges:

- Kempe exchanges which recolour the root, we will call these *root recolourings*,
- Kempe exchanges which do not recolour the root. These will be called *leaf recolourings*.

root recolourings

leaf recolourings

Let S be a star, and r be its root vertex. Note that leaf recolourings only change the colour of a single vertex. Given an initial colouring α, a sequence of bichromatic exchanges is a sequence of root recolourings (resp. leaf recolourings) if at each step of the transformation the root is recoloured (resp. is not recoloured). We will say that a sequence of Kempe exchanges is *sorted* if it is composed of first a sequence of leaf recolourings, and then a sequence of root recolourings. In other words, a sequence of bichromatic exchanges is sorted if no Kempe-exchange that alters the colour of the root precedes a Kempe-exchange that does not alter the colour of the root.

sorted transformation sequence

It is not difficult to see that there is a sequence of leaf recolourings from α to β if and only if $\alpha(r) = \beta(r)$. Similarly, there is a sequence of root recolourings from α to β if and only if the two colourings differ by a permutation of colours. In this case, we will say that α and β are *equivalent*, and we will denote $\pi_{\alpha\beta}$ the permutation of colours such that for every vertex $v \in S$, $\beta(v) = \pi_{\alpha\beta}(\alpha(v))$.

equivalent colourings

The first step in the proof consists in showing that for the purpose of finding a shortest transformation, we may restrict our attention to sorted sequences of bichromatic exchanges. Observe that for a sorted transformation between α and β, there is an intermediate colouring γ, such that the colour of the root does not change in the transformation from α to γ and it changes at each step in the transformation from γ to β. We then provide tight bounds for shortest transformations from α to γ and γ to β, respectively.

Recall that for a colouring α of the star S, and a suitable Kempe-exchange t (given by a vertex and a target colour), we denote by $t(\alpha)$ the colouring obtained by applying t to α. In the following, α and β denote two k-colourings of a star S. We start by showing that we can swap in some sense the order of a root recolouring and a leaf recolouring.

Lemma 10. *Let α be a colouring of a star S with root r. Furthermore, let t be a root recolouring when applied to α and let s be a leaf recolouring when applied to $t(\alpha)$. Starting from α, there is a leaf recolouring s', such that $s(t(\alpha)) = t(s'(\alpha))$.*

Proof. Let $t = \langle r, x\rangle$ be a Kempe-exchange recolouring the root with colour x. Furthermore, let $s = \langle u, y\rangle$ be a Kempe-exchange on a leaf vertex u of S that does not alter the colour of r in $t(\alpha)$. We only need to consider Kempe-exchanges that actually alter the colouring α, so we may assume that $x \neq \alpha(r)$. Since s does not alter the colour of r, we may also assume that $y \neq x$. Therefore two cases remain to be considered.

First, let us assume that $y \neq \alpha(r)$. In this case, we can just take $s' = s$. Indeed, by applying s on α, the root is not recoloured by assumption on s. This transformation recolours u to y. When applying t on $s(\alpha)$, u is not further recoloured since $x \neq y$.

Consider now the case $y = \alpha(r)$. In this case, we consider $s' = \langle u, x\rangle$. Applying s' on α does not recolour r since we know that $x \neq y = \alpha(r)$. By applying s' and then t on α, u is first recoloured to x, and then recoloured to $\alpha(r) = y$. Hence $s(t(\alpha)) = t(s'(\alpha))$, and the result follows. □

The lemma above implies that for any transformation between two colourings of a star, there is a sorted transformation of at most the same length.

Corollary 11. *For any two colouring α and β of a star S, if there is a transformation of length t from α to β, then there is sorted transformation sequence of length at most t between the two.*

Proof. Given a non sorted transformation sequence of length t, we can use Lemma 10 to move all the leaf recolourings at the beginning of the transformation sequence. After this operation, the transformation sequence is sorted, and its length is still t. □

For any sorted transformation from α to β there is an intermediate colouring γ, such that the colour of the center vertex does not change in the transformation from α to γ and it changes in each step in the transformation from γ to β. This implies that γ is such that $\gamma(v) = \alpha(v)$, and γ is equivalent to β. The rest of this subsection is dedicated to finding bounds on the shortest transformation sequence from α to γ, and from γ to β respectively. These bounds are such that the total length of the shortest sequence from α to β can be expressed as a function of the intermediate colouring γ. In particular, this means that finding a shortest sequence is equivalent to fining the right intermediate colouring γ. Since γ and β are equivalent, this is the same as finding a permutation on the colours such that the colouring γ which results from this permutation defines a shortest transformation. Our reduction will then proceed by showing that finding this good permutation is difficult.

Lemma 12. *Let α and γ be two colourings with $\alpha(r) = \gamma(r)$. The shortest leaf transformation sequence from α to γ has length:*

$$|\{v \in S \setminus \{r\}, \alpha(v) \neq \gamma(v)\}|$$

Proof. As we remarked before, a Kempe exchange which does not recolour the root vertex of the star recolours only one vertex. Hence, the length of the shortest leaf transformation sequence from α to γ is equal to the number of leaves which have different colours in α and in γ. □

Lemma 13. *Let γ and β be two equivalent colourings of S such that $\beta^{-1}(c) \neq \emptyset$ for any colour v. The length of the shortest root transformation sequence from γ to β is:*

$$k - |\operatorname{Fix}(\pi_{\gamma\beta})| + |\operatorname{Cyc}(\pi_{\gamma\beta})| - 2 \cdot \mathbb{1}_{\{\gamma(r)\neq\beta(r)\}} ,$$

where $\mathbb{1}_{\{\gamma(r)\neq\beta(r)\}}$ is equal to 1 if $\gamma(r) \neq \beta(r)$, and zero otherwise.

Note that the condition $\beta^{-1}(c) \neq \emptyset$ means that every colour is used at least once. This condition is only present to ensure that the permutation $\pi_{\gamma\beta}$ is uniquely defined.

Proof. Denote by $L(\gamma, \beta)$ the quantity in the statement of the lemma. To prove the result, we only need to check that the following two points holds:

1. If $\gamma \neq \beta$, then there exists a root recolouring which decreases $L(\gamma, \beta)$ by at least 1.
2. No root recolouring can decrease $L(\gamma, \beta)$ by more than 1.

In the rest of the proof, we write c_r the colour of the root in γ. Let γ' be the colouring obtained from γ by performing a root recolouring from c_r to some colour x. First observe that the permutation $\pi_{\gamma'\beta}$ can be obtained from $\pi_{\gamma\beta}$ by composing it with the transposition $(c_r\ x)$ which exchanges the colours c_r and x. In other words, we have: $\pi_{\gamma'\beta} = \pi_{\gamma\beta} \circ (c_r\ x)$.

First point. We consider several cases:

- If $\beta(r) = \gamma(r)$, then let x be any colour which is not a fixpoint in $\pi_{\gamma\beta}$ (this colour exists since by assumption, $\beta \neq \gamma$). We consider the Kempe exchange which recolour the root with x. After the transformation, c_r is no longer a fixpoint of $\pi_{\gamma'\beta}$, and x is still not a fixpoint of this permutation. Additionally, the number of cycles did not change by the operation, and finally, $\gamma'(r) = x \neq \beta(r)$. Hence, $L(\gamma', \beta) = L(\gamma, \beta) - 1$.
- If $\beta(r) \neq \gamma(r)$, and the decomposition of $\pi_{\gamma\beta}$ contains several disjoint cycles, then let x be a colour which is not in the same cycle as c_r. After recolouring the root with r, the number of cycles in the decomposition of $\pi_{\gamma'\beta}$ decreased by 1, as the operation merges the two cycles containing c_r and x respectively. Additionally, the number of fixpoints does not change. Hence, since we still have $\gamma'(r) \neq \beta(r)$, this implies that $L(\gamma', \beta)$ decreased by 1.
- Finally, if $\beta(r) \neq \gamma(r)$, and the decomposition of $\pi_{\gamma\beta}$ contains one unique cycle, let $x = \pi_{\gamma\beta}^{-1}(c_r)$ be the antecedent of c_r by the permutation $\pi_{\gamma\beta}$. Then, either the cycle had length at least 3, in which case the permutation $\pi_{\gamma'\beta}$ also contains one unique cycle. The length of this cycle decreased by 1, and $\gamma'(r) \neq \beta(r)$. Or, the cycle had length 2, and we have $\gamma' = \beta$. In this case, the number of fixpoints increased by 2, and the number of cycles decreased by 1. In both cases, we have $L(\gamma', \beta) = L(\gamma, \beta) - 1$.

Second point. We want to show that for any choice of the colour x, $L(\gamma', \beta)$ cannot decrease by more than 1. Note that the number of fixpoints can increase by at most 2, and the number of cycles can decrease by at most 1. We consider the two following cases:

- First assume that $\beta(r) = \gamma(r)$. After the transformation, the colour c_r is no longer a fixpoint of the permutation, and the number of cycles cannot decrease. This implies that $L(\gamma', \beta)$ can decrease by at most 1 since the last term of the expression $L(\gamma', \beta)$ can decrease by at most 2.

- Now, we can assume that $\beta(r) \neq \gamma(r)$. If the number of cycles decreases by 1, this means that one of the cycles in the decomposition of $\pi_{\gamma\beta}$ was just the transposition $(c_r\ x)$. After the transformation, we have $\beta(r) = \gamma'(r)$, and hence $L(\gamma', \beta)$ has decreased by at most 1. If the number of cycles does not decrease, then the number of fixpoint decreases by at most 1, and again, $L(\gamma', \beta) \geq L(\gamma, \beta) - 1$.

Hence, the two properties are verified, and the lemma holds. □

3.2.2 Hardness of Kempe-Bound on Stars

Let us now show that Kempe-Bound on stars is **NP**-hard. The proof is by reduction from the Hamiltonian Cycle problem. The Hamiltonian Cycle problem asks whether a given graph contains a simple cycle visiting each vertex exactly once. It was shown by Garey, Johnson, and Tarjan that Hamiltonian Cycle remains **NP**-complete even on planar cubic graphs [GJT76]. We now describe the construction used in the reduction to prove the **NP**-hardness of Kempe-Bound on stars.

Let G be a graph with n vertices, and K be a constant, with $K > n + 1$. We construct the star S, with two colourings α and β as follows. The number of colours will be $n + 1$. The root of the star will coloured $n + 1$ in both α and β. For each edge $ij \in G$, we add K leaves to S, with colour i in α and colour j in β. Note that we consider each edge in both direction (i.e., once in the direction ij, and once in the direction ji). In particular, S has a total of $2K|E(G)|$ leaves. The proof of Theorem 9 follows immediately from the following result:

Lemma 14. *There is a transformation sequence of length at most $K(2|E(G)| - n) + n + 1$ from α to β in S if and only if G has an Hamiltonian cycle.*

Proof. First assume that G contains an Hamiltonian cycle, and consider an arbitrary orientation of this cycle. We consider the permutation π on $[n + 1]$ such that $\pi(n + 1) = n + 1$, and $\pi(i) = j$, where j is the vertex following i on the Hamiltonian cycle of G. The permutation π contains one fixpoint, and one cycle of length n. Consider the colouring γ obtained from β by applying the permutation π on the colours. By definition of π, we have $\gamma(r) = \alpha(r) = n + 1$, and γ and β are equivalent. Additionally, since we can assume that G contains no isolated vertices, this implies that $\beta^{-1}(c) \neq \emptyset$ for any colour c.

By Lemma 12, there exists a transformation from α to γ of length at most:

$$|\{v \in S \setminus \{r\}, \gamma(v) \neq \alpha(v)\}|$$

Let $v \in S$ be a vertex which was added when considering the edge ij of G in this direction. By construction, we have $\alpha(v) = i$ and $\beta(v) = j$. Then $\alpha(v) = \gamma(v)$ if and only if $i = \pi^{-1}(\beta(v)) = \pi^{-1}(j)$. In other words, we have $\alpha(v) = \gamma(v)$ if and only if the edge ij appears with this orientation in the Hamiltonian cycle. Since there are exactly n edges in the Hamiltonian cycle of G, this implies that the there exists a transformation from α to γ of length at most $K(2|E(G)| - n)$.

Consider now a transformation from γ to β. Since γ and β are equivalent, and $\pi^{-1} = \pi_{\gamma\beta}$, Lemma 13 gives a transformation from γ to β of length $n + 1$ (recall that the number of colours here is $n + 1$). Putting together these two transformation gives a (sorted) recolouring sequence of the required length.

Reciprocally, assume that there exists a transformation sequence from α to β of length at most $L = K(2|E(G)| - n) + n + 1$. By Corollary 11, we can assume that this transformation is sorted.

Let γ be the intermediate colouring such that the sorted transformation from α to β defines a leaf transformation sequence from α to γ, and a root transformation sequence from γ to β. Consider the permutation $\pi_{\gamma\beta}$. This permutation is such that $\pi_{\gamma\beta}(n+1) = n+1$. We start by showing the following claim.

Claim 15. For all $i \in [n]$, $i\pi(i)$ is an edge of G.

Proof. Consider the shortest leaf transformation sequence from α to γ. By Lemma 12, this sequence has length

$$|\{v \in S \setminus \{r\}, \gamma(v) \neq \alpha(v)\}|$$

As we have seen just before, for a vertex $v \in S$ which was added when considering the edge ij of G in this direction, we have $\alpha(v) = \gamma(v)$ if and only if $i = \pi_{\gamma\beta}^{-1}(\beta(v)) = \pi_{\gamma\beta}^{-1}(j)$. Hence the number of vertices for which $\alpha(v) = \gamma(v)$ is equal to $K|\{i \in [n],\ i\pi_{\gamma\beta}(i) \in G\}|$. If there exists at least one index i such that $i\pi_{\gamma\beta}(i) \notin G$, then this quantity is at most $K(n-1)$, which implies that the transformation sequence from α to γ has length at least $K(2|E(G)| - n + 1) > L$, since $K > n+1$. This contradicts the assumption that the transformation sequence from α to β has length at most L. Thus, for every $i \in [n]$, $i\pi_{\gamma\beta}(i)$ is an edge of G. □

Consider the subgraph H of G containing all the edges $i\pi(i)$ for $i \in [n]$. H is composed of cycles and isolated edges, and covers all the vertices of the graph. Each component in H corresponds to a cycle the decomposition of $\pi_{\gamma\beta}$ into disjoint cycles. Since the leaf transformation sequence from α to γ has length $K(2|E(G)| - n)$, this implies that the root transformation sequence from γ to β must have length at most $L - K(2|E(G)| - n) = n + 1$. By Lemma 13, the root transformation sequence from γ to β has length:

$$n + 1 - \mathrm{Fix}(\pi_{\gamma\beta}) + \mathrm{Cyc}(\pi_{\gamma\beta}) = n + \mathrm{Cyc}(\pi_{\gamma\beta})\ ,$$

where the equality comes from the fact that $n+1$ is the only fixpoint of $\pi_{\gamma\beta}$, since for every $i \in [n]$, $i\pi_{\gamma\beta}(i) \in G$, and in particular $i \neq \pi_{\gamma\beta}(i)$. This quantity must be at most $n+1$, which implies that $\mathrm{Cyc}(\pi_{\gamma\beta}) \leq 1$. Hence this inequality must be an equality, and H contains one single component which implies that H is an Hamiltonian cycle of G. This completes the proof of the reduction. □

3.3 Single vertex recolouring

In this section, we consider single vertex recolouring. The complexity of reconfiguration with these transitions has already been considered in the literature [BC09, FJP14]. We will concentrate here on the structural aspect of the problem, and study the diameter of the reconfiguration graph. We consider this problem on the case of d-degenerate graphs. It is not very difficult to prove that $\mathcal{G}(k, G)$ is connected if G is d-degenerate, and $k \geq d+2$ (see for example [DFFV06]). However, this simple proof does not provide any non-trivial upper bound on the diameter of the reconfiguration graph. Additionally, although Cereceda conjectured that this diameter should be at most quadratic [Cer07], even finding a polynomial upper bound is currently still open. The reader can refer to Section 2.3.2 for more details on the existing results on the problem.

The main result in this section is to prove a polynomial upper bound on the diameter of $\mathcal{G}(k, G)$ when G is d-degenerate, $k \geq d+2$, and d is a constant. More precisely, we prove the following:

Theorem 16. *Let $d, k \in \mathbb{N}$ and G be a d-degenerate graph. Then $\mathcal{G}(k, G)$ has diameter at most:*

- Cn^2 *if* $k \geq \frac{3}{2}(d+1)$,
- $C_\varepsilon n^{\lceil 1/\varepsilon \rceil}$ *if* $k \geq (1+\varepsilon)(d+2)$ *and* $0 < \varepsilon < 1$,
- $(Cn)^{d+1}$ *for any* d *and* $k \geq d+2$,

where C and C_ε are constants independent of k and d.

In particular, it implies that the 7-recolouring diameter of planar graphs is polynomial (of order $O(n^6)$), answering a question of [BP16, Feg19b], and is quadratic if $k \geq 9$ (improving the recent result of Feghali giving $k \geq 10$ [Feg19c]). For general graphs, our result guarantees moreover that the diameter becomes a polynomial independent of d as long as $k \geq (1+\varepsilon)(d+2)$. We also obtain a quadratic diameter when the number of colours is at least $\frac{3}{2} \cdot (d+1)$, improving the result of Cereceda [Cer07] who obtained a similar result for $k \geq 2d+1$. Note moreover that Theorem 16 ensures that the diameter is polynomial as long as d is a fixed constant, which was open even for $d = 2$ (we get a diameter of order $\mathcal{O}(n^3)$ for $d = 2$).

In order to show Theorem 16, we need to prove a more general result which also holds for list-colourings. Indeed, we often need to consider induced subgraphs of our initial graph where the colours of some vertices are"frozen" (i.e. do not change). By considering the list colouring version, we can delete these vertices and remove their colours from the list of all their neighbours. The proofs of Theorem 16 consists in showing that, given two colourings α and β of G, there exists a transformation from α to β of the corresponding length. Since our proof is algorithmic, it also provides a polynomial time algorithm that, given two colourings α and β, outputs a transformation from α to β of length at most the bound we find on the diameter of the reconfiguration graph.

Let G be a d-degenerate graph and let $v_1, \ldots, v_n$ be a degeneracy ordering. We denote by $d^+(v)$ the out-neighbours of v (i.e., neighbours of v which appear later in the ordering). Recall that a graph is d-degenerate if there is a degeneracy ordering such that $d^+(v) \leq d$ for every vertex v of the graph.

Let us first briefly discuss the main ideas of the proof before stating formally all the results. The main idea is to proceed recursively on the degeneracy. More precisely, we want to delete a subset of vertices in order to decrease by one both the degeneracy and the number of colours. In order to do this, observe that if at some point there is one colour c such that every vertex of the graph is either coloured c or has an out-neighbour coloured c, then by removing all the vertices coloured c, we decrease the number of available colours by 1, but we also decrease the degeneracy of the graph by 1. If H is the resulting graph, by applying induction, we can recolour H however we want, and for example, we can remove completely one colour which we can then use to make the two colourings agree on a subset of vertices. If the colour c satisfies the condition above, we will say that the colour c is *full*. Our main objective will consist in finding a transformation from any colouring α of G to some colouring α' of G which has a full colour (Note that any graph can have such a colouring, for example by applying the First-Fit algorithm in the reverse order of the elimination ordering). We will build the colouring α' (and the transformation) incrementally. However in order to do this, we will need to generalise the problem to list colouring.

full colour

Recall that a list assignment L is a function which associates a list of colours to every vertex v, and an L-colouring is a (proper) colouring α of G such that for every vertex v, $\alpha(v) \in L(v)$. The total number of colours used by the assignment is $k = |\bigcup_{v \in G} L(v)|$. A list assignment L is *a-feasible*

a-feasible list assignment

feasible list assignment

if $|L(v)| \geq |d^+(v)| + a + 1$ for every vertex $v \in G$. We just say that it is *feasible* if it is 1-feasible. We denote by $\mathcal{G}(L, G)$ the reconfiguration graph of the L-colourings of G. (One can easily prove by induction, that if a list assignment is a-feasible for $a \geq 1$, then $\mathcal{G}(L, G)$ is connected). We will prove a generalisation of Theorem 16 in the case of list colourings. Namely, we will prove that:

Theorem 17. *Let G be a graph and $a \in \mathbb{N}$. Let L be an a-feasible list assignment and k be the total number of colours. Then $\mathcal{G}(L, G)$ has diameter at most:*

- *kn if $k \leq 2a$.*
- *Cn^2 if $k \leq 3a$ (C a constant independent of k, a),*
- *$C_\varepsilon n^{\lceil 1/\varepsilon \rceil}$ if $k \leq (1 + \frac{1}{\varepsilon})a$ where ε is a constant and C_ε is independent of k, a,*
- *$(Cn)^{k-1}$, if $a \geq 1$.*

The proof of Theorem 16 follows easily from this result. The only point that is not immediate is the fact that the last point of Theorem 17 implies the last point of Theorem 16. In this case, we need a small trick to guarantee that the diameter does not increase if the number of colours increases. Note that the first point of Theorem 17 implies in terms of classical colouring that the k-recolouring diameter is linear when $k \geq 2d + 2$, which is an already known result [BP16].

Proof of Theorem 16. Note that given a d-degenerate graphs and k colours, we can consider the list assignment L where $L(v) = [k]$ for every vertex v. This list assignment is a-feasible with $a = k - d - 1$.

We start with the second point of Theorem 16. If $k \geq (1 + \varepsilon)(d + 1)$ then:

$$\frac{k}{a} = \frac{k}{k - d - 1} = 1 + \frac{d + 1}{k - d - 1} \leq 1 + \frac{1}{\varepsilon}\,.$$

By applying the third case of Theorem 17, the result follows.

The first point follows immediately from the result above by taking $\varepsilon = \frac{1}{2}$.

Finally, in order to prove the last point, we need to prove that we can "replace" k by d. Let G be a d-degenerate graph and let γ be a $(d + 1)$-colouring of it. Let us prove that, if $k > d + 2$, any colouring α can be transformed into γ within $O(n^{d+1})$ steps (and not $O(n^{k-1})$ as suggested by Theorem 17). Indeed, we simply simply "forget" the vertices coloured with colour $d+3, \ldots, k$ in α. Let H be the graph without these vertices. The graph H is d-degenerate and by Theorem 17, we can transform $\alpha|_H$ into $\gamma|_H$ within $\mathcal{O}(n^{d+1})$ steps using colours in $1, \ldots, d + 2$. We finally recolour the vertices of $G \setminus H$ one by one with their colours in γ to obtain the colouring γ. □

The rest of this section is devoted to prove Theorem 17. In order to do it, we need to generalise the notion of full colour to the list-colouring setting. We also need to generalise it to sets of colours, to handle the case where $a > 1$. Given a colouring α of G, the set of colour S is *full* if for every vertex v and every colour $c \in S$ one of the following holds:

(i) $\alpha(v) \in S$,

(ii) v has at least one out-neighbour coloured c,

(iii) or $c \notin L(v)$.

We have a property similar as previously: starting from an a-feasible list assignment with a colouring α, if S is full then by removing all the vertices v with a colour $\alpha(v) \in S$ from the graph, and removing all the colours from S from all the lists, the resulting assignment is still a-feasible. Additionally, the total number of colours has decreased by $|S|$ (but the degeneracy of the graph might not have decreased as much if we had k much larger than $d+2$).

In the following, we will denote by $f_a(n,k)$ the maximum diameter of $\mathcal{G}(L,G)$ over all graphs G with n vertices, and all a-feasible assignments L with total number of colours k.

The proof of Theorem 17 is by induction on the total number of colours k. The base case, which is the first point of Theorem 17, is the following lemma. This lemma ensures that if the number of excess colours is sufficient (at least half the number of colours), then the diameter is linear[1].

Lemma 18. *Assume that $k \leq 2a$, then $f_a(n,k) \leq kn$.*

Proof. We show by induction on n that if $k \leq 2a+1$, then for any a-feasible list assignment L on a graph on n vertices, and any two L-colourings α, β, there is a transformation from α to β such that every vertex is recoloured at most k times.

The result is clearly true when $n = 1$, since in this case the unique vertex can be recoloured only once. Assume that the result holds for $n-1$. Let G be a graph on n vertices with a degeneracy ordering $v_1, \dots, v_n$, and an a-feasible list assignment L using a total of k colours. Let α and β be two L-colourings, H be the subgraph obtained after removing v_n, and define $d = |d^+(v_1)|$ the number of neighbours of v_1.

Using induction on H, there is a transformation $\mathcal{S}$ from $\alpha|_H$ to $\beta|_H$ such that every vertex is recoloured at most k times. Note that by assumption, the number of colours available to v_1 is at least $d+a+1$, hence the total number of colours k satisfies, $d+a+1 \leq k \leq 2a$, and in particular $a \geq d+1$, and $k \geq d+a+1 \geq 2d+2$. Every time one of the neighbours of v_1 is recoloured in the sequence $\mathcal{S}$, we may have to recolour v_1 beforehand so that the colouring remains proper. This happens if the neighbour wants to be recoloured with the current colour of v_1.

Every time we have to recolour v_1, we have the choice among a set S of $a \geq d+1$ colours for the new colour of v_1. We can look ahead in $\mathcal{S}$ to know which are the next $d+1$ modifications of colours of neighbours of v_1 in $\mathcal{S}$. One colour c of C does not appear in these modifications since $|C| \geq d+1$ and the first modified colour is not in C (since we need to recolour v_1 at the first step, and then the target colour is the current colour of v_1 which is not in C). We recolour v_1 with c. This way, we only need to recolour v_1 once out of every $d+1$ times its neighbours are recoloured. Finally, we may need to recolour v_1 one last time after the end of $\mathcal{S}$ to colour v_1 with its target colour. Since by induction, the neighbours of v_1 are recoloured at most k times, the total number of times v_1 is recoloured is at most:

$$\left\lceil \frac{dk}{d+1} \right\rceil + 1 \leq \frac{d}{d+1}k + 2 \leq k\,,$$

where in the last inequality, we have used the fact that $2 \leq \frac{k}{d+1}$ since $k \geq 2d+2$. This concludes the induction step and proves the result. □

In the induction step of our proof, we will build a set of full colours. For a colouring α and a set X of vertices, we denote by $\alpha(X) = \bigcup_{x \in X} \alpha(x)$. Before stating the main lemmas, let us make the following remark:

[1] The proof is similar to the linear diameter obtained in [BP16] for colourings but adapted to list colourings.

Remark 1. Let G be a graph, L be a list assignment and α be a L-list colouring. Let H be an induced subgraph of G with list assignment $L'(v) = L(v) \setminus \alpha(N(v) \setminus V(H))$. Then any recolouring sequence $\mathcal{S}$ from $\alpha|_H$ to some colouring $\beta|_H$ also is a (valid) recolouring sequence from α to β where $\beta(v) = \beta|_H(v)$ if $v \in H$ and $\alpha(v)$ otherwise.

By abuse of notation and when no confusion is possible, we will then call $\mathcal{S}$ both the recolouring sequence in H and in G.

The following lemma states that, if we already have a set of full colours, then changing it to an other given set can be done without too many additional recolouring steps.

Lemma 19. *Let α be a colouring of G, and S a set of full colours for α with $|S| = a$. For any S' with $|S'| = a$, there exists a colouring α' such that S' is full for α', and there is a transformation from α to α' of length at most $f_a(n, k-a) + (2a+2)n$.*

Proof. Let S be a set of colours with $|S| = a$ which is full for some L-colouring α. Let S' be any set of colours of size a. The main part of the proof consists in transforming α into a colouring that does not use at all any colour of S' (such a colouring exists since the list assignment is a-feasible).

Let H be the subgraph induced by the vertices not coloured S in α, and let L_H be the list assignment for H obtained from L by removing S from all the lists. Since S is full in α, L_H is a-feasible for H.

Consider the following *preference ordering* on the colours: an arbitrary ordering of $[k] \setminus (S \cup S')$, followed by an ordering of $S' \setminus S$, and finally the colours from S last. Let γ be the L-colouring of G obtained by colouring G greedily from v_n to v_1 with this preference ordering. Since L is a-feasible, and $|S| = a$, no vertex is coloured with a colour in S in γ. Indeed $|L(v)| \geq |d^+(v)| + a + 1$ and only $|d^+(v)|$ neighbours of v have been coloured when v is coloured. Since in γ no vertex has a colour in S, $\gamma|_H$ is an L_H-colouring of H. By induction hypothesis, there is a recolouring sequence that transforms the colouring $\alpha|_H$ of H into $\gamma|_H$ within at most $f_a(n, k-a)$ steps. By Remark 1, this recolouring sequence also is a recolouring sequence in G. We can then recolour the vertices of $G \setminus H$ to their target colour in γ in an additional n steps. No conflict can happen at this step since γ is a proper colouring of G.

One can easily check that, in γ, the set of colours $[k] \setminus (S \cup S')$ is full. Let K be the subgraph of G induced by all the vertices coloured S' in γ, and let L_K be a list assignment of these vertices where all the colours not in $S \cup S'$ were removed. Then since $[k] \setminus (S \cup S')$ is full, L_K is a-feasible. We will recolour K such that no vertex is coloured S' (such a colouring exists because L_K is a-feasible, and $|S'| = a$). Since the total number of colours used in L_K is $|S \cup S'| \leq |S| + |S'| = 2a$, this recolouring can be done in at most $f_a(n, 2a) = 2an$ steps by Lemma 18. By Remark 1, this recolouring sequence also is a recolouring sequence in G. The colouring γ' of G that we obtain is such that no vertex is coloured with $c \in S'$. We can finally recolour the vertices of G one by one, starting from v_n, choosing a colour of S' if it is available, or leaving it with its current colour otherwise.

Let α' be the resulting colouring. By construction α' is full for S'. The total number of steps to reach α' is at most $f_a(n, k-a) + n + 2an + n = f_a(n, k-a) + (2a+2)n$. □

Using Lemma 19, we show that we can incrementally construct a set of full colours.

Lemma 20. *Assume that $k \geq 2a$. For any colouring α, there exists a colouring β containing a set of full colours S with $|S| = a$, and there is a transformation from α to β of length at most $\frac{n}{a} f_a(n, k-a) + 4n^2$.*

Proof. Let $v_1, \dots, v_n$ be a degeneracy ordering of G. A colouring is *full up to step i* for a set of colour S if $|S| = a$, and all the vertices v_j with $j \leq i$ satisfy the conditions $(i), (ii), (iii)$ for the set S. If a colouring γ is full up to step n for the set S, then the set S is full.

Note that for any colouring α, any set of colours containing $\alpha(v_1), \dots, \alpha(v_a)$ is full up to step a. So we only need to show that given a colouring α which is full up to step i for some set S, we can reach a colouring α' full up to step $i + a$ for some (potentially different) set S' in at most $f_a(n, k-a) + 4an$ steps. Suppose now that α is full up to step i but not $i+1$ for some set S. Let S' be the colours of the vertices $v_{i+1}, \dots, v_{i+a}$. Up to adding arbitrary colours to S', we can assume that $|S'| = a$. We will then recolour the graph in order to obtain a colouring where S' is full up to step $i + a$.

Let H be the graph induced by the vertices $v_1, \dots, v_i$, and L_H be the list assignment of the vertices of H obtained from L by fixing the colours of the vertices outside H. In other words, for every vertex $v \in V(H)$, we remove from $L(v)$ the colours of all the vertices of $N(v) \cap \{v_{i+1}, \dots, v_n\}$. Note that L_H is an a-feasible assignment of H since L was an a-feasible assignment of G. Additionally, S is a set of full colours for the colouring $\alpha|_H$.

By Lemma 19, there is an L_H-colouring α'_H of H which is full for S' such that we can transform $\alpha|_H$ into $\alpha|'_H$ in at most $f_a(n, k-a) + (2a+2)n$ steps. Let α' be the colouring which agrees with α on the vertices with index larger than i, and agrees with $\alpha|'_H$ on H. By Remark 1, α' can be obtained from α into at most $f_a(n, k-a) + (2a+2)n$ steps. By construction, S' is full up to step i, and since the vertices $v_{i+1}, \dots v_{i+a}$ are coloured with colours in S', it is full up to step $i + a$.

Finally, this procedure needs to be repeated at most $\lceil \frac{n}{a} \rceil - 1 \leq \frac{n}{a}$ times (the minus one coming from the fact that at the beginning we had for free a set of colours full for $v_1, \dots, v_a$). After this many steps, we obtain a colouring full up to step n for some set S with $|S| = a$, which concludes the proof. □

Finally, we can use the two previous lemma to get the following recursive inequality.

Lemma 21. *Let $k \geq 2a$, then $f_a(n, k) \leq (\frac{2n}{a} + 3) f_a(n, k-a) + 10 \cdot n^2$.*

Proof. Let L be an a-feasible list assignment of G, and α and β be two L-colourings of G. By Lemma 20, there exists a colouring α' and a set of colours S_α with $|S_\alpha| = a$ which is full for α' such that the colouring α' can be reached from α in at most $\frac{n}{a} f_a(n, k-a) + 4n^2$ steps. Similarly, there exists a colouring β' and a set of colours S_β with $|S_\beta| = a$ such that S_β is full for β' such that the colouring β' can be reached from β in at most $\frac{n}{a} f_a(n, k-a) + 4n^2$ steps. By Lemma 19, using an additional $f_a(n, k-a) + 4n$ steps, we can get a colouring β'' from β' such that the set of full colours in β'' and α' is the same (namely S_α).

Let S be some set of colours disjoint from S_α with $|S| = a$. Let γ be a colouring of G that does not use any colour of S_α (such a colouring exists since the list assignment of G is a-feasible).

Let G_α be the graph G where the vertices coloured with colours in S_α have been deleted and the colours in S_α removed from the list assignment. Since S_α is full for α', the list assignment of G_α is a-feasible. Note that $\gamma|_{G_\alpha}$ is a proper colouring of G_α. So by induction, it is possible to recolour $\alpha'|_{G_\alpha}$ into $\gamma|_{G_\alpha}$ in at most $f_a(n, k-a)$ steps. Since the vertices of $W := V(G) \setminus V(G_\alpha)$ are coloured with colours in S_α, and since γ does not use any of these colours, one can finally recolour the vertices of W one by one from their colours in S_α to their target colours in γ.

Let $G_{\beta''}$ be the graph where the vertices coloured with $S_{\beta''} = S_\alpha$ in β'' have been deleted. One can similarly recolour $\beta''|_{G_{\beta''}}$ into $\gamma|_{G_{\beta''}}$ in at most $f_a(n, k-a)$ steps. Since vertices of

$W' := V(G) \setminus V(G_{\beta''})$ are coloured with colours in S_α, we can also can recolour the vertices of W' one by one from their colours in S_α to their target colours in γ.

The total number of steps to transform α into β is at most

$$\begin{aligned} f(n,k) \leq \quad & 2\left(\frac{n}{a} f_a(n,k-a) + 4n^2\right) + (f_a(n,k-a) + 4n) + 2f_a(n,k-a) + 2n \\ & \leq \left(\frac{2n}{a} + 3\right) f_a(n,k-a) + 10n^2 \end{aligned}$$

□

Lemma 22. *For all $k \geq 1$ and $a \geq 1$, we have $f_a(n,k) \leq Ckn\left(\frac{2n}{a}+3\right)^{\lceil \frac{k}{a} \rceil - 2}$ for some constant $C > 0$.*

Proof. We prove it by induction on the total number of colours k. The base case is when $k \leq 2a$ and simply is a consequence of Lemma 18 (note that $\lceil \frac{k}{a} \rceil = 2$ since $k > a$). The induction step is obtained using Lemma 21. □

We now have all the ingredients to prove Theorem 17, which completes the proof of Theorem 16.

Proof of Theorem 17. The first point is the result from Lemma 18. The second and last points are consequences of Lemma 22 with the corresponding values of a. Let us prove the third point. Since $k \leq (1+\frac{1}{\varepsilon})a$, we have $\lceil \frac{k}{a} \rceil - 2 \leq \lceil 1 + \frac{1}{\varepsilon} \rceil - 2 \leq \lceil \frac{1}{\varepsilon} \rceil - 1$. Consequently, the function given by Lemma 22 is $O(n^{\lceil 1/\varepsilon \rceil})$. Note that for the second and third point, the constant does not depend on k or a since k/a is a constant (but it depends on ε in the third point). □

Chapter 4

Perfect Matching Reconfiguration

This chapter studies the complexity of the reconfiguration of perfect matching. The results presented here are the outcome of a collaboration started during the first DATCoRe workshop in Lyon in July 2018. They also appear in [BBH+19].

In this chapter, we consider the reconfiguration of perfect matchings, where two perfect matchings are considered adjacent if their symmetric difference is a cycle of length 4. In Section 4.1 below, we introduce the problem and present existing results on matching reconfiguration in general. In Section 4.2 we prove that Perfect-Matching-Reachability is **PSPACE**-complete, even when restricted to (i) bipartite graphs of bounded bandwidth, and (ii) split graphs. Finally, in Section 4.3, we show that this problem is solvable in polynomial time on cographs. When a transformation exists, the algorithm also finds one of linear length. Note that in the original paper [BBH+19], polynomial time algorithms are given for two additional classes of graphs: strongly orderable graphs (and in particular interval graphs), and outerplanar graphs.

4.1 Introduction

Recall from Chapter 1 that a matching of a graph is *perfect* if it covers each vertex. Reconfiguration of (not perfect) matchings has already been considered in the literature under Token Sliding (TS) and Token Jumping (TJ) rules. Numerous algorithms and hardness results are available for finding transformations between matchings — and more generally, independent sets — using these two operations.

However, these two operations are not suitable for the reconfiguration of perfect matchings.

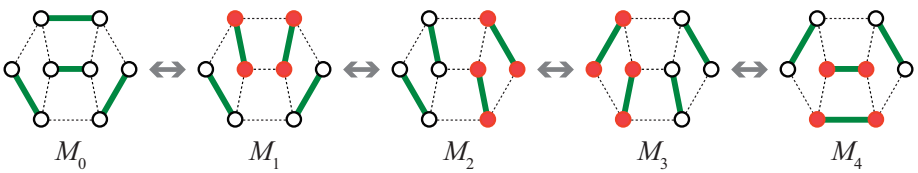


Figure 4.1: A transformation between perfect matchings M_0 and M_4 under the flip operation. For $1 \leq i \leq 4$, the matching M_i can be obtained from M_{i-1} by applying the flip operation to the cycle induced by the four painted (red) vertices in M_i.

Indeed, for both TS and TJ rules, there is no feasible transition if the starting position is a perfect matching. Since the symmetric difference of any two perfect matchings of a graph consists of even-length vertex disjoint cycles, it is natural to consider a different adjacency relation for perfect matchings. We say that two perfect matchings differ by a *flip* if their symmetric difference induces a cycle of length four. Two perfect matchings are adjacent if they differ by a flip. Intuitively, for two adjacent perfect matchings M and M', we think of a flip as an operation that exchanges edges in $M \setminus M'$ for edges in $M' \setminus M$. A flip is in some sense a minimal modification of a perfect matching. flip

An example of a transformation between two perfect matchings of a graph is given in Figure 4.1. Using the notations from previous chapter, the problem we consider here is formally defined as follows.

PM-Reachability

Input: Graph G, perfect matchings M_s and M_t of G.

Question: Is there a sequence of flips that transforms M_s into M_t?

Note that if we do not restrict the length of a cycle in the definition of a flip, the problem becomes trivial. Indeed, given two matchings M_s and M_t, let us denote by $M_s \triangle M_t$ their symmetric difference, i.e., the set of edges which are in one of the two matchings but not in both. If the two matchings are perfect, then their symmetric difference is a disjoint union of cycles, and there is a transformation from one to the other by performing a flip on each cycle in the symmetric difference $M_s \triangle M_t$. If this method always produces a reconfiguration sequence, this sequence is not necessarily the shortest. Figure 4.2 gives an example of two perfect matchings for which the symmetric difference contains 3 cycles, but there is a transformation of length 2. The problem of finding the shortest reconfiguration sequence when flips of arbitrary size are allowed can be of independent interest, and prompts the following open problem.

Open Problem 9. Investigate the complexity of Perfect-Matching-Bound with flips of arbitrary size.

4.1.1 Related work

Reconfiguration of perfect matchings using flips has been considered in the literature, in the context of random sampling. The use of flips for sampling random perfect matchings was first started in [DGH01] where it is seen as a generalisation of transpositions for permutations. Their work was later improved and generalized in [DJM17] and [DM17]. The focus of these two articles is to investigate the problem of sampling random perfect matchings using a Markov Chain called the

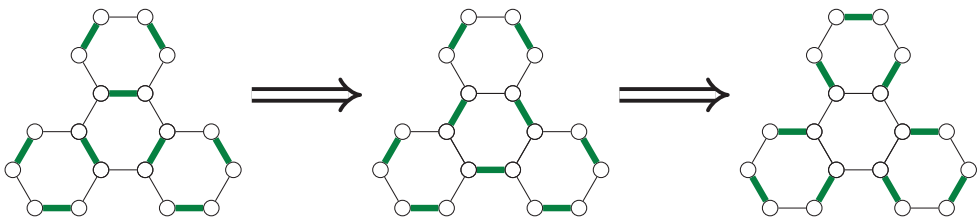


Figure 4.2: Example of transformation using flips of larger size. The symmetric difference between the matching on the left and the one on the right contains 3 cycles, however, there is a transformation of length only 2.

switch chain. Starting from an arbitrary perfect matching, the chain proceeds by applying at each step a random flip (called switch in these papers). Some of their results can be reformulated in the reconfiguration terminology. In [DJM17], it is proved that the largest hereditary class of bipartite graphs for which the reconfiguration graph of perfect matchings with flips is connected is the class of chordal bipartite graphs. This result is generalized in [DM17] where they characterize the hereditary class of general (non-bipartite) graphs for which the reconfiguration graph is connected. They call this class SWITCHABLE. Note that it is not clear whether graphs in this class can be recognized in polynomial time. The question of the complexity of PM-REACHABILITY is also mentioned in [DM17].

The problem of sampling or enumerating perfect matchings in a graph received a considerable attention (see e.g. [SVW18]). Determining the reachability, the connectivity and the diameter of the solution space formed by perfect matchings under the flip operation provide some information on the ergodicity or the mixing time of the underlying Markov chain.

As we discussed at the beginning of this chapter, transformations between matchings and independent sets have been studied in various settings, and in particular using the TS and TJ operations. Furthermore, the flip operation has been used for other problems such as reconfiguration of Hamiltonian cycles [Tak18], as well as some geometric matching problems related to finding transformations between triangulations.

Reconfiguration of Matchings and Independent Sets. Recall that matchings of a graph correspond to independent sets of its line graph. Although reconfiguration of independent sets received a considerable attention in the last decade (e.g., [BKW14, BMP17, DDF+14, HD05, IKO14, KMM12, Wro14b]), all the known results for reconfiguration of independent sets are based on the TJ or TS operations as adjacency relations. Thus, none of these results results carry over to the reconfiguration of perfect matchings.

A related problem can be found in a more general setting: The problem of determining, enumerating or randomly generating graphs with a fixed degree sequence has received a considerable attention since the fifties (see e.g. [Sen51, Hak63, Wil99]). Given two graphs with a fixed degree sequence, one might want to know if it is possible to transform one into the other via a sequence of flip operations and if yes, how many steps are needed for such a transformation; note that the host graph (i.e., the graph G in our problem) is a clique in this setting. Hakimi [Hak63] proved that such a transformation always exists. Will [Wil99] proved that the problem of finding a shortest transformation is **NP**-complete, and Bereg and Ito [BI17] provide a $\frac{3}{2}$-approximation algorithm for this problem.

Flips of Triangulations. A flip of a triangulation is similar to flipping an alternating cycle in the sense that we switch between two states of a quadrilateral. In the context of triangulations, a flip operation switches the diagonal of a quadrilateral. Transformations between triangulations of point sets and polygons using flips have been studied mostly in the plane. It is known that the flip graph of triangulations of point sets and polygons in the plane is connected and has diameter $O(n^2)$, where n is the number of points [HNU99, Law77]. Recently, **NP**-completeness has been proved for deciding the flip-distance between triangulations of a point set in the plane [LP15] and triangulations of a simple polygon [AMP15].

Houle et al. have considered triangulations of point sets in the plane that admit a perfect matching [HHNRC05]. They show that any two such triangulations are connected under the flip

operation. For this purpose they consider the graph of non-crossing perfect matchings, where two matchings are adjacent if they differ by a single non-crossing cycle (of arbitrary length). They show that the graph of non-crossing perfect matchings is connected and conclude from this that any two triangulations that admit a perfect matching must be connected. In contrast to their setting, we remove all geometric requirements, but restrict the length of the cycles allowed for the flip operation.

4.2 PSPACE-completeness

In this section, we prove that PM-Reachability is **PSPACE**-complete. Interestingly, the problem remains intractable even for bipartite graphs, even though matchings in bipartite graphs satisfy several nice properties.

Theorem 23. PM-Reachability *is* ***PSPACE****-complete for bipartite graphs whose maximum degree is five and whose bandwidth is bounded by a fixed constant.*

As we observed in Chapter 2, it is not difficult to design a non-deterministic algorithm for the problem using only polynomial space. This shows that the problem is in **NPSPACE**, and hence in **PSPACE** by Savitch's theorem [Sav70]. Thus, we only need to prove the hardness of the problem. The proof proceeds by giving a polynomial-time reduction from the Nondeterministic Constraint Logic problem (NCL for short) [HD05]. It is very similar in flavour to the proof in Section 3.1 showing the hardness of reconfiguration of graph colourings.

Recall from Section 2.4 that an instance of NCL is given by an NCL machine: a 3-regular graph with or nodes and and nodes. A configuration of the machine is an orientation of the edges which satisfies the constraints of all the nodes. The reduction is done by embedding NCL-Reachability into PM-Reachability using some gadgets to represent each part of the NCL machine. The gadgets are described in the following subsection.

4.2.1 Gadgets

Suppose that we are given an instance of NCL, that is, an NCL machine and two configurations of the machine. We will replace each NCL edge and each NCL and/or node with its corresponding gadget. If an NCL edge e is incident to an NCL node v, then we connect the corresponding gadgets for e and v by a pair of vertices, called *connectors* (*between v and e*) or (v, e)-*connectors*, as illustrated in Figure 4.3(a) and (b). Thus, each edge gadget has two pairs of connectors, and each and/or gadget has three pairs of connectors. Our gadgets are all edge-disjoint, and share only connector vertices. Moreover, there is always an edge between the two vertices of a connector pair, and this edge belongs to the corresponding node gadget.

(v, e)-connectors

In the following we will restrict our attention to perfect matchings where the two vertices in any connector are matched in the same gadget. In other words, consider the edges of the perfect matching incident to the two vertices of a connector pair. Then either these edges are all part of the edge gadget, or they are all part of the node gadget. We can remark that any flip always preserves this property if it holds initially.

The correspondence between orientations of an NCL machine and perfect matchings of the corresponding graph, is defined depending on which gadget contains the edges matching the connector vertices. The orientation of an NCL edge $e = vw$ is considered inwards for v if the two

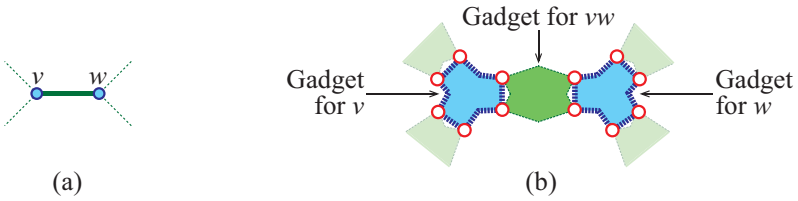
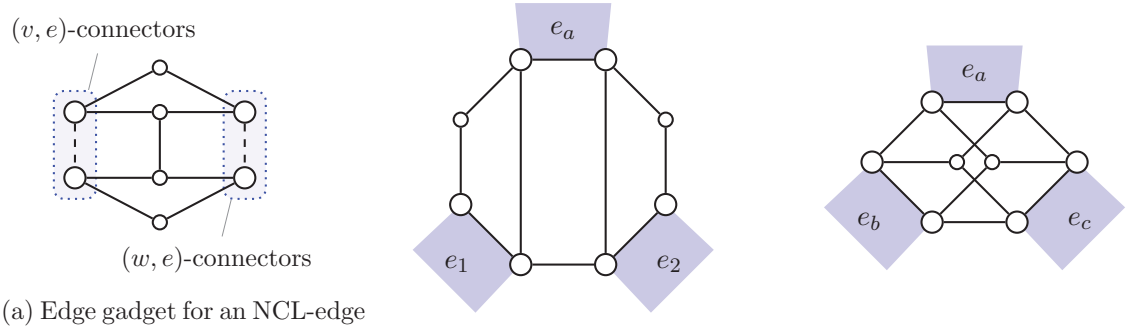


Figure 4.3: (a) An NCL edge vw, and (b) its corresponding gadgets, where the connectors are depicted by (red) circles.

(v, e)-connectors are both covered by (edges in) the AND/OR gadget for v. In other words, the edges covering the (v, e)-connectors are edges of the node gadget for v. On the other hand, the orientation of $e = vw$ is considered outwards for w if the two (w, e)-connectors are both covered by the edge gadget for e. Figure 4.4 shows our three types of gadgets which correspond to NCL edges and NCL AND/OR vertices. We explain below the behaviour of each gadget.

Edge gadget. Recall that, in a given NCL machine, two NCL vertices v and w can be joined by a single NCL edge $e = vw$. Let us explain which property is desirable for an edge gadget, and then show that the gadget in Figure 4.4a satisfy these properties. The edge gadget must satisfy the two following properties: (i) it must be consistent with its orientation with the NCL machine, and forbid the configuration where both its extremities are both directed inwards for v and w; (ii) it must be "internally connected" [OSIZ18], i.e., any configuration corresponding to an orientation σ of the NCL machine must be reachable from any other configuration corresponding σ without changing the orientation corresponding to the NCL edge; and (iii) it must respect the "external adjacency" of the NCL edge [OSIZ18], i.e., for any two adjacent orientations there must exist two adjacent perfect matchings which correspond to these orientations.

The first property is immediate: if all the connectors of e are covered by their respective node



(a) Edge gadget for an NCL-edge (v, w)

(b) AND gadget

(c) OR gadget

Figure 4.4: Illustrations of the three gadgets. In the edge gadget, note that there is an edge between the two vertices in a connector pair (dashed), but this edge belongs to the corresponding node gadget. In the AND/OR gadget, the three light blue parts represent the edge gadgets corresponding to the edges incident to the NCL node; e_1 and e_2 in the AND gadget correspond to weight-1 edges.

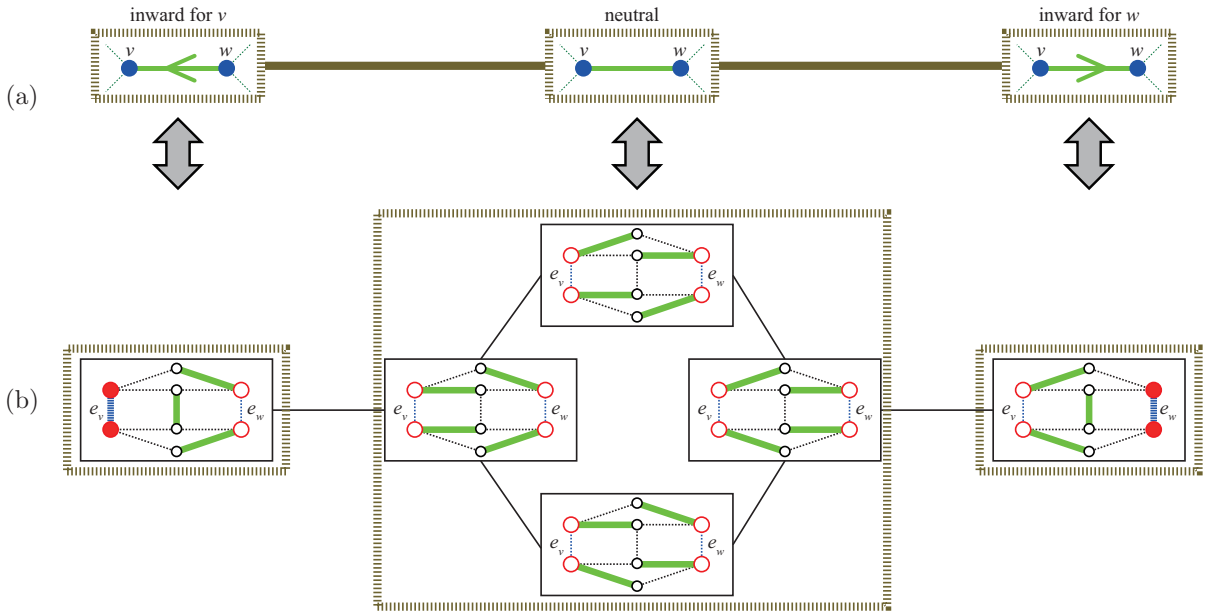


Figure 4.5: (a) A reversal of the orientation of an NCL edge vw via its neutral orientation, and (b) all configurations of the edge gadget. Note that two edges e_v and e_w joining connectors do not belong to the edge gadget, but to the AND/OR gadgets for v and w, respectively. The inside of each connector is painted in red if it is covered by the AND/OR gadget for v or w.

gadgets, then four vertices remain to be matched in the gadgets, to of which (the top and bottom vertices in Figure 4.4a) have no available edge. Hence, no perfect matching corresponds to an orientation where both extremities of the edge are oriented inwards for their respective nodes.

The second and third points are illustrated in Figure 4.5b which represents all the valid configurations of an edge gadget for $e = vw$ together with two edges e_v and e_w belonging to the AND/OR gadgets for v and w, respectively. Each (non-dotted) box represents a valid configuration. Two boxes are joined by an edge if their configurations are adjacent, that is, can be obtained by flipping a single cycle of length four. Furthermore, each large dotted box surrounds all configurations corresponding to the same orientation of an NCL edge $e = vw$. Then, the set of configurations (non-dotted boxes) in each large dotted box induces a connected component. This means that any configuration in the set can be transformed into any other without changing the orientation of the corresponding NCL edge, and the gadget is "internally connected". In addition, if we contract the configurations in the same large dotted box into a single vertex (and merge parallel edges into a single edge if necessary), then the resulting graph is exactly the graph depicted in Figure 4.5 (a), and the gadget satisfies the "external adjacency" condition. Therefore, we can conclude that our edge gadget correctly simulates the behaviour of an NCL edge.

AND and OR gadgets. Figure 4.4b illustrates our AND gadget for each NCL AND node v, where e_1 and e_2 are two weight-1 NCL edges and e_a is the weight-2 NCL edge. Figure 4.6a illustrates all valid orientations of the three edges incident to v, and Figure 4.6b illustrates valid configurations of the AND gadget together with (images of) three edge gadgets for e_1, e_2 and e_a. Then, as illustrated in Figure 4.6, our AND gadget satisfies both "internal connectedness" and "external adjacency",

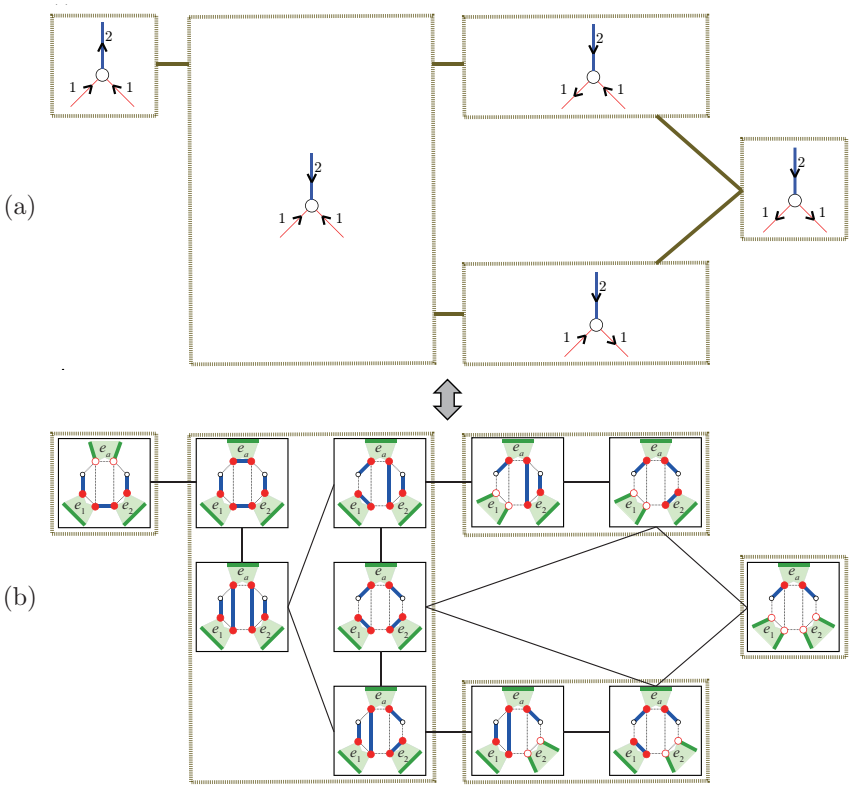


Figure 4.6: (a) All valid orientations of three edges incident to an NCL AND node v, and (b) all configurations of the AND gadget together with three incident edge gadgets, where the inside of each connector is painted (by red) if it is matched by the AND gadget.

and hence it correctly simulates an NCL AND node.

Figure 4.4c illustrates our OR gadget for each NCL OR node v, where e_a, e_b and e_c correspond to three NCL edges incident to v. For an NCL OR node, we need to forbid only one type of orientations of the three NCL edges: all NCL edges e_a, e_b and e_c are directed outward for v at the same time, that is, all six connectors are covered by the edge gadgets for e_a, e_b and e_c. Our OR gadget forbids such a case, because otherwise we cannot cover the two (white) small vertices in the center. In addition, this OR gadget satisfies both "internal connectedness" and "external adjacency", and correctly simulates an NCL OR node.

Reduction. As illustrated in Figure 4.3, we replace each of NCL edges and NCL AND/OR vertices with its corresponding gadget. Let G be the resulting graph. Notice that each of our three gadgets is of maximum degree three, and connectors in the edge gadget are of degree two; thus, G is of maximum degree five. In addition, each of our three gadgets is bipartite and such that two connectors in the same pair belong to different sides of the bipartition; therefore, G is bipartite. Furthermore, since NCL remains **PSPACE**-complete even if the input NCL machine has bounded bandwidth [Zan15], the resulting graph G also has bounded bandwidth, since each gadget consists only of a constant number of edges.

We next construct two perfect matchings of G which correspond to two given NCL configurations

C_s and C_t of the NCL machine. Note that there are (in general, exponentially) many perfect matchings which correspond to the same NCL configuration. However, by the construction of the three gadgets, no two distinct NCL configurations correspond to the same perfect matching of G. We arbitrarily choose two perfect matchings M_s and M_t of G which correspond to C_s and C_t, respectively.

This completes the construction of our corresponding instance of PERFECT MATCHING RECONFIGURATION. The construction can be done in polynomial time. Furthermore, the following lemma gives the correctness of our reduction.

Lemma 24. *There exists a reconfiguration sequence between two NCL configurations C_s and C_t if and only if there exists a reconfiguration sequence between M_s and M_t.*

Proof. We first prove the only-if direction. Suppose that there exists a reconfiguration sequence between C_s and C_t, and consider any two adjacent NCL configurations C_{i-1} and C_i in the sequence. Then, only one NCL edge vw changes its orientation between C_{i-1} and C_i. Notice that, since both C_{i-1} and C_i are valid NCL configurations, the NCL AND/OR vertices v and w have enough incoming arcs even without vw. Therefore, we can simulate this reversal by the reconfiguration sequence of perfect matchings in Figure 4.5 (b) which passes through the neutral orientation of vw as illustrated in Figure 4.5 (a). Recall that both AND and OR gadgets are internally connected, and preserve the external adjacency. Therefore, any reversal of an NCL edge can be simulated by a reconfiguration sequence of perfect matchings of G, and hence there exists a reconfiguration sequence between M_s and M_t.

We now prove the if direction. It is important to notice that any cycle of length four in G belongs to exactly one gadget (including the edge between two connectors). Therefore, even in the whole graph G, a flip of edges along a cycle of length four can happen only inside one of the gadgets. Suppose that there exists a reconfiguration sequence $M_0, \ldots, M_\ell$ from $M_0 = M_s$ to $M_\ell = M_t$. Notice that, by the construction of gadgets, any perfect matching of G corresponds to a valid NCL configuration, with possibly some NCL edges with a neutral orientation. In addition, M_s and M_t correspond to valid NCL configurations without any neutral orientation. Pick the first index i in the reconfiguration sequence $M_0, \ldots, M_\ell$ which corresponds to changing the direction of an NCL edge vw to the neutral orientation. Then, since the neutral orientation contributes to neither v nor w, we can simply ignore the change of the NCL edge vw and keep the direction of vw as the same as the previous direction. By repeating this process and deleting redundant orientations if needed, we can obtain a sequence of valid adjacent orientations between C_s and C_t such that no NCL edge takes the neutral orientation. □

This completes the proof of Theorem 23. We conclude this section by proving that the problem remains intractable even for split graphs. Recall from Section 1.3.3 that a graph is *split* if its vertex set can be partitioned into a clique and an independent set.

Corollary 25. PM-REACHABILITY *is **PSPACE**-complete for split graphs.*

Proof. By Theorem 23 the problem remains **PSPACE**-complete for bipartite graphs. Consider the graph obtained by adding new edges so that one side of the bipartition forms a clique. The resulting graph is a split graph. These new edges can never be part of any perfect matching of the graph. Indeed, since the original graph was bipartite, there must be the same number of vertices on each side of the bipartition. In a perfect matching of the split graph, all the vertices from the independent set must be matched with vertices from the clique, and no vertex from the clique remains to be matched together. Thus, the corollary follows. □

4.3 Cographs

We now consider the complexity of PM-Reachability when the input graph is a cograph. Recall from Section 1.3.3 that cograph are graphs without P_4 as an induced subgraph.

As examples concerning reconfiguration on this class of graphs, it is known that the problems Independent set-Reachability and Steiner tree-Reachability can be decided efficiently on cographs [BB14b, Bon16, MIZ17], while they are **PSPACE**-complete for general graphs [IDH+11, MIZ17]. Theorem 23 together with the following result show that the situation is similar for PM-Reachability.

Theorem 26. PM-Reachability *on cographs can be decided in polynomial time. Moreover, for a* ***yes****-instance, a reconfiguration sequence of linear length can be output in polynomial time.*

The proof of the theorem relies on the recursive construction of cographs, and the cotrees describing this construction (see Section 1.3.3). The main idea of the theorem is to decompose the graph, and apply the algorithm recursively on each of the components. In order to get a transformation of linear length using this method, we need to extend our problem to non-perfect matchings. Since the set of vertices matched by a matching does not change when performing a flip, we need to add some other operation. We will consider in this section that two matchings are adjacent if their symmetric difference is either a cycle of length four, or a path of length 3. Note that this second type of transition we added corresponds to the token sliding model: we are allowed to replace an edge of the matching by any other incident edge. We will call this operation a *sliding move*. We consider reconfiguration in this more general setting.

sliding move

GM-Reachability

Input: Graph G, two matchings M_s and M_t of G.

Question: Is there a sequence of flips and sliding moves that transforms M_s into M_t?

Note that the answer to the problem is clearly **no** when the two matchings do not have the same size. We can also remark that when the two input matchings are perfect, sliding moves become useless (since sliding requires at least one non-matched vertex), and we get back the original problem. In particular, Theorem 26 is a special case of the following more general result.

Theorem 27. GM-Reachability *on cographs can be decided in polynomial time. Moreover, for a* ***yes****-instance, a reconfiguration sequence of linear length can be output in polynomial time.*

We start by considering certain base cases for which a transformation of linear length always exists and can be computed efficiently. Let M_s and M_t be two matchings of a cograph G, and let T be a cotree of G. For the purpose of transforming M_s to M_t, we may assume that G is connected, so the root of T is a join-node (otherwise we can simply apply the algorithm to each connected component of G). We will call *root partition* the partition of the vertices of G into A and B corresponding to all the leaves in respectively the left and right subtrees of the root of T. All along this section, unless otherwise specified, A and B will denote the root partition of G, and k denotes the size of the matchings we are considering, i.e., $k = |M_s| = |M_t|$. Note that A is complete to B, because the root of T is a join-node. Without loss of generality we may assume that $|A| \geq |B|$. For a vertex subset X of G and an edge e of G, we say that $G[X]$ *contains* e if both endpoints of e are in X.

root partition

Given a connected cograph G with root partition A, B, we will consider the two following conditions:

(C1) There exists a matching M of G of size k such that $G[B]$ contains an edge of M.

(C2) There exists a matching M of G of size k such that at least one vertex of B is not matched in M.

When one of these two conditions holds, then we will be able to show directly that the reconfiguration graph on matchings of size k is connected, and has linear diameter. On the other hand, if none of the two condition holds, we will apply induction on $G[A]$ in order to conclude. The case where condition **(C1)** holds is treated in Lemma 29 and condition **(C2)** is handled in Lemma 31. The case where none of the two properties hold is treated in Lemma 32.

Note that it is possible to check in polynomial time whether one of the two conditions holds since computing a maximum matching in a graph can be done in polynomial time. Indeed, a simple algorithm to check condition **(C1)** consists in trying all possible edges in $G[B]$, removing both endpoints from the graph, and search for a matching of size $k-1$ in the remaining graph. Similarly, for condition **(C2)**, we can try to remove every possible vertex from B, and search for a matching of size k in the remaining graph.

Remark that the connected components of $M_s \triangle M_t$ can be of three types: single edges, paths of length at least 3, and even cycles. If M_s and M_t are perfect matchings, then only even cycles can occur in the symmetric difference. We start with the following observation that finding a reconfiguration sequence is easy if the symmetric difference contains no cycle.

Lemma 28. *Let G be a cograph, and M_s and M_t be two matchings of size k such that $M_s \triangle M_t$ contains no cycle. Then there is a transformation of length at most $2|M_s \triangle M_t|$ from M_s to M_t.*

Proof. The result is proved by induction on the size of the symmetric difference $M_s \triangle M_t$. If the symmetric difference is zero, then $M_s = M_t$ and the result trivially holds. Otherwise, we only need to show that we can reduce the symmetric difference by 2 in at most 4 steps. First assume that $M_s \triangle M_t$ contains a path of length $t \geq 3$. Let $x_1, \dots x_t$ be the vertices of this path, and assume without loss of generality that x_1x_2 is an edge in M_s and x_2x_3 an edge in M_t. By definition, x_1 is not matched in M_t. Consequently, starting from M_t we can slide the edge x_2x_3 to x_1x_2, and reduce the symmetric difference by 2 in one step.

Now assume that the symmetric difference does not contain any path of length at least 3. Since by assumption it does not contain any cycles either, then it must be a disjoint union of edges. Let $e_s = u_sv_s$ be an edge in $M_s \setminus M_t$, and $e_t = u_tv_t$ an edge in $M_t \setminus M_s$ (none of these sets is empty since M_t and M_s have the same size). First assume that there is an edge incident to both e_s and e_t. Without loss of generality, we can assume that this edge is u_su_t. In this case, we can simply slide u_sv_s to u_su_t and then to u_tv_t, giving a transformation of length at most 2. Otherwise, since G is a cograph, the two edges must be at distance at most 2. Hence, we can assume that there is a vertex w adjacent to both u_s and u_t. We consider the two following cases:

- w is not matched in M_s. In this case, starting from M_s, we can simply slide u_sv_s to u_sw and then to wu_t and finally to u_tv_t;
- w is matched to w' in M_s. In this case, we start by sliding ww' to wu_t and then to u_tv_t. Then, we can slide u_sv_s to u_sw and finally ww'.

In any case, we can reduce the symmetric difference by 2 in at most 4 steps, which proves the result. □

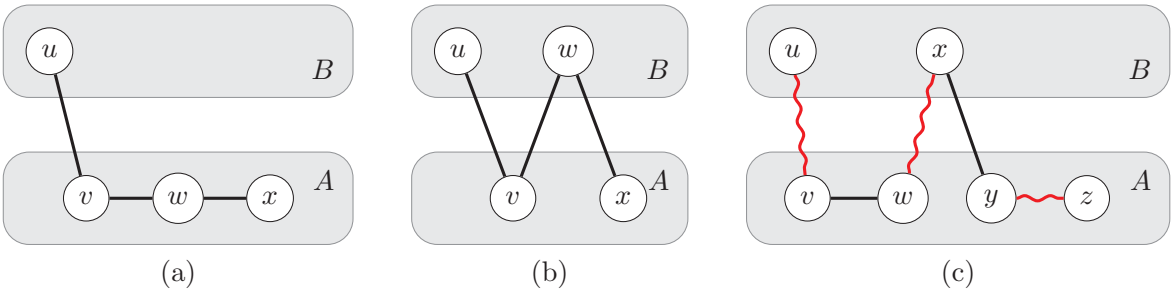


Figure 4.7: Representation of some of the cases of Claim 30. Note that all edges between A and B are present in the graph, but were removed for clarity. The edges drawn are the edges in the symmetric difference $M \triangle M_s$. (a) corresponds to Assumption 3, (b) is Assumption 4, and (c) is Assumption 5. For the last case, the red wavy edges are edges in M, the other ones are edges in M_s.

The following lemma provides a way to construct a transformation sequence of linear length when condition **(C1)** holds. Note that the proof is constructive and can be easily turned into a polynomial time algorithm computing this sequence.

Lemma 29. *Let G be a connected cograph with n vertices, and $k \geq 0$ such that condition **(C1)** holds. Then, there is a reconfiguration sequence of length $O(n)$ between any two matchings of size k in G.*

Proof. Assume that G has a perfect matching M such that $G[B]$ contains at least on edge M, and let e be such an edge. To prove the lemma, we only need to show that for any matching M_s of size k, there is a transformation sequence of linear length from M_s to M.

We first claim the following: we can assume without loss of generality that e is the only edge of M contained in B. To see this, assume that there is another edge $e' = ab$ with $a, b \in B$. Since $|A| \geq |B|$, either M also contain an edge $e'' = cd$ with both endpoints in A, or there is at least one vertex x in A not matched in M. In the first case, the edge e' can be removed from the matching by flipping ab and cd with ac and bd. In the second case, we can slide $e' = ab$ into ax.

We now prove that for every matching M_s of G of size k, there is a reconfiguration sequence of length $O(|M \triangle M_s|)$ from M to M_s. We start by proving the following claim. See Figure 4.7 for an illustration of some of the cases.

Claim 30. After a transformation of at most $O(|M \triangle M_s|)$ steps in M and M_s, we can assume[1] that M still contains an edge in $G[B]$ and M and M_s also satisfy the following statements:

1. no edge of M_s is contained in B.
2. $M \triangle M_s$ contains no cycle on A.
3. $M \triangle M_s$ contains no three edges uv, vw, wx, such that $u \in B$ and $v, w, x \in A$.
4. $M \triangle M_s$ contains no three edges uv, vw, wx, each between A and B.

[1] The resulting matchings are still denoted by M and M_s for simplicity.

5. $M \triangle M_s$ contains no five edges uv, vw, wx, xy and yz, with $uv \in M$, and $v, w, y, z \in A$ and $u, x \in B$.

Proof of Claim 30. We prove all the points in the increasing order. Let e be the edge of M contained in $G[B]$, and let x_e and y_e be its two endpoints.

Assumption 1. Let ab be an edge of M_s contained in B. Since $|A| \geq |B|$, there is either an edge cd of M_s with both endpoints in A or a vertex x in A not matched by M_s. In the first case, starting from M_s, we flip ab and cd with bc and ad. In the second case, we make a sliding move from ab to ax. Since every edge in M_s contained in B is in the symmetric difference $M_s \triangle M$ (except if $ab = e$), this operation will be repeated at most $|M_s \triangle M| + 1$ times. Each time, the symmetric difference increases by at most 1 (by 2 in the case $e = ab$).

Assumption 2. Let C be a cycle in $M \triangle M_s$, such that $V(C) \subseteq A$. We show that we can transform $M \cap E(C)$ to $M_s \cap E(C)$. The other edges of M are not modified. Let $u_1, u_2, \ldots, u_{2\ell}$ be the vertices of C, all in A. We assume without loss of generality that $u_2u_3, u_4u_5, \ldots, u_{2\ell}u_1 \in M$. We first flip x_ey_e and $u_{2\ell}u_1$ to create x_eu_1 and $y_eu_{2\ell}$. Then we flip x_eu_1 and u_2u_3 for u_1u_2 and x_eu_3. We proceed with u_4u_5 and uu_3 and so on (reducing the length of the cycle in the symmetric difference), until $x_eu_{2\ell-1}$ and $y_eu_{2\ell}$ remains. After flipping these two edges for x_ey_e and $u_{2\ell-1}u_{2\ell}$, the resulting matching still contains the edge x_ey_e with both endpoints in B and we have reconfigured $M \cap E(C)$ to $M_s \cap E(C)$, i.e., the size of the symmetric difference has decreased. The other edges in M were not modified; in particular Assumption 1 still holds.

Note that number of flips performed in this sequence is at most $|E(C)|$ and the symmetric difference decreased by $|E(C)|$.

Assumption 3. We can assume without loss of generality that uv and wx are in M and vw is in M_s. Then, we can flip uv and wx for vw and ux and reduce the size of the symmetric difference by 2. Moreover, Assumption 1 still holds in the resulting perfect matching.

Assumption 4. We can assume without loss of generality that uv and wx are in M and vw is in M_s. Then, in M we can flip uv and wx for vw and ux and reduce the size of the symmetric difference. Note moreover that Assumption 1 still holds in the resulting perfect matching.

Assumption 5. By assumption, uv, wx and yz are edges of M. Note that x_e and y_e are distinct from $\{u, v, w, x, y, z\}$ since u and x are incident to edges of M between A and B (and the other vertices are in A). We will perform a sequence of flips decreasing the symmetric difference, and preserving e at the end of the transformation (see Figure 4.8). Now, in the matching M, flip x_ey_e and yz for x_ey and y_ez. Then flip xw and x_ey for xy and x_ew. Note that at this point, the size of the symmetric difference with M_s decreased by one. Then we flip y_ez and uv for y_ev and uz. Finally, we flip x_ew and y_ev for x_ey_e and vw. By these operations, the size of the symmetric difference with M_s has decreased by at least two since we only flip edges of the symmetric difference and the resulting matching have edges vw and xy which are in M_s. Moreover, the resulting matching still contains the edge e with both endpoint in B.

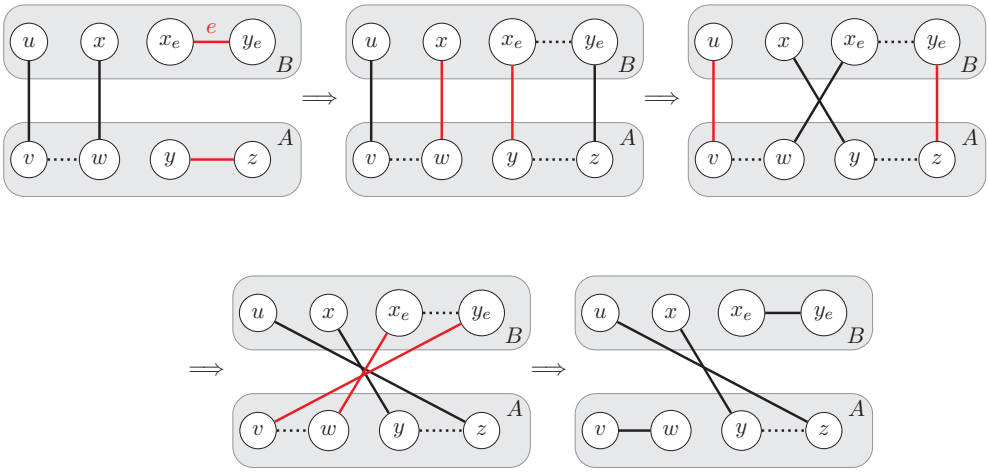


Figure 4.8: Reconfiguration sequence to reduce the symmetric difference in the case of Assumption 5. All the edges between A and B are present in the graph but were removed for clarity. The edges drawn are the edges in M. At each step, the two edges in red are the ones which are flipped to obtain the next transition.

Number of flips. In order to get Assumption 1, we may need $|M \triangle M_s| + 1$ steps and increase the symmetric difference by $|M \triangle M_s| + 2$. In all the other points, if we perform δ flips, we decrease the symmetric difference by at least $c\delta$ (where c is some constant), and we never have to apply Assumption 1 again. So the claim holds after $O(|M \triangle M_s|)$ steps and the size of the symmetric difference is still at most $O(|M \triangle M_s|)$. □

After applying Claim 30, let us still denote by M and M_s the resulting matchings. The number of steps needed to reach this point is at most $O(|M \triangle M_s|)$. Recall that e is the edge of M contained in B. We will show that the only cycle that can be in the symmetric difference is a cycle of length 4 containing the edge e. Assume by contradiction that this is not the case, and let C be a cycle in the symmetric difference $M \triangle M_s$.

Suppose first that C does not contain e. By Assumption 2, there exists a vertex $u \in C \cap B$. Let v, w and x the vertices following u on the cycle C and such that $uv \in M$. We must also have $wx \in M$ and $vw \in M_s$. Since C does not contain e, we know that $v \in A$. By Assumptions 3 and 4 and the fact that $e \notin C$, we have $w \in A$, and $x \in B$. Since both u and x are on the side B, and using Assumption 1, we have $xu \notin C$. In particular $|C| > 4$, and since C has even length this implies $|C| \geq 6$. Let y and z be the two vertices following x on the cycle C. By Assumption 1, and since $wx \in M$, we have $y \in A$. Moreover, by Assumption 4, we have $z \in A$. However, in this case the configuration of the vertices u, v, w, x, y, z contradicts Assumption 5.

Consequently, there is only one cycle C in the symmetric difference, and C must contain e. Assume by contradiction that $|C| \geq 6$. Let x_1, x_2, x_3, u, v, w be consecutive vertices on the cycle C such that x_1 is incident to e, and x_2 is not. Then $x_1x_2 \in M_s$, which implies that $x_2 \in A$. Using the Assumptions 1, 3, and 4, we also have $x_3, v, w \in A$, and $u \in B$. In particular, w is not incident to e. Let x_0 be the other endpoint of e. Since the cycle C must have even length, w and x_0 are not consecutive in C. Let x be the vertex consecutive to w in C. Then, by Assumption 3 we must have $x \in B$. Since M_s does not contain any edge in B, this implies that x and x_0 are not consecutive in

C. Let y be the vertex consecutive to x in C, and z consecutive to y. Since C has even length, we known that $z \neq x_0$. By Assumption 1 we must have $y \in A$, and by Assumption 4, we have $z \in A$. However, in this case the configuration of the vertices u, v, w, x, y, z contradicts Assumption 5.

Hence, the only possible cycle in the symmetric difference is a cycle of length 4 containing e. After flipping this cycle, the symmetric difference contains only paths and isolated edges. By Lemma 28, we can finish transforming M_s into M using an additional $O(M \Delta M_s)$ steps. □

We now handle the case where condition **(C2)** holds. As for the previous case, when this condition holds a transformation sequence can be easily found between any two matchings.

Lemma 31. *Let G be a cograph, and $k \geq 0$ such that condition **(C2)** holds. Then, there is a transformation sequence of length $O(n)$ between any two matchings of size k.*

Proof. Without loss of generality, we can assume that condition **(C1)** does not hold since otherwise we can conclude directly using Lemma 29. Consequently, no matching of size k of G uses any of the edges in $G[B]$. Hence, these edges can be removed without changing in any way the reconfiguration graph, and we can assume that $G[B]$ is an independent set.

Let M be a matching of G of size k such that there exists a vertex v in B not matched in M. Let M_s be any matching of G of size k. We will show that there is a transformation from M to M_s of length at most $O(|M \triangle M_s|)$. If the symmetric difference is zero, then the result is trivial. If the symmetric difference contains no cycle, then the result follows from Lemma 28.

Let C be a cycle in the symmetric difference $M \triangle M_s$. Since $G[B]$ is an independent set, C must contain at least one vertex in A. Let $x_1, \ldots, x_{2t}$ be the vertices in C, with $x_1 \in A$, and $x_1x_2 \in M$. We consider the following transformation starting from the matching M:

- slide x_1x_2 to x_1v,
- for i from 2 to $t-1$ slide $x_{2i+1}x_{2i+2}$ to $x_{2i}x_{2i+1}$,
- finally, slide x_1v to $x_{2t}x_1$.

This operation progressively reconfigure M such that M and M_s agree on C. Note that the edges of M outside of C are not modified by the transformation. Moreover, if M_2 is the matching obtained after the transformation, then the symmetric difference with M_s has decreased by $|C| = 2t$. Additionally, the vertex v is still not matched in M_2. Since the number of steps performed by this transformation is $t + 1 \leq |C|$, the result follows by applying induction with M_2 and M_s. □

In case none of the two conditions **(C1)** and **(C2)** holds, the following lemma states that we only need to consider what happens on the subgraph induced by A. Given a matching M, we will note M^A the matching of $G[A]$ induced by the edges of M. Remark that even if M is a perfect matching of G, M^A might not be a perfect matching of $G[A]$ since some of the vertices in A can be matched to vertices in B by M. This is the main reason we had to extend the problem to non-perfect matchings. We have the following.

Lemma 32. *Let G be a connected cograph with root partition A, B with $|A| \geq |B|$, and $k \geq 0$ such that conditions **(C1)** and **(C2)** do not hold. Let M_s and M_t be two matchings of G of size k. Then:*

- *there is a transformation sequence from M_s to M_t in G if and only if there is a transformation sequence from M_s^A to M_t^A in $G[A]$;*

- *if there is a transformation of length t from M_s^A to M_t^A in $G[A]$, then there is a transformation from M_s to M_t of length at most $t + O(|B|)$.*

Proof. Let M_s and M_t be two matchings of size k of G. First assume that there is a transformation sequence S from M_s^A to M_t^A in $G[A]$. We will build a transformation sequence from M_s to M_t in G. First, observe that any flip in S on the subgraph $G[A]$ is also a valid flip on the whole graph G. For sliding moves, there are two possibilities. Let u, v, w be three vertices in A, and consider the sliding move in $G[A]$ which replaces uv by vw. Either w is not matched to a vertex in B, and in this case this move is also a valid sliding move on the whole graph. Or w is matched to a vertex $x \in B$. In this case, consider the operation of flipping uv and wx for vw and xu. Then this transformation acts exactly as the original sliding move on A.

Hence, by eventually replacing some of the sliding moves by flips as explained above, we obtain a transformation sequence S' which transforms M_s into a matching M such that M^A and M_t^A are equal. To finalize the transformation, from M to M_t, we only need the two following observation.

- We can make M and M_t agree on the vertices $a \in A$ which are matched to vertices in B. Indeed, if there is a vertex $a \in A$ which is matched to $b \in B$ in M but not in M_t, then there must be a vertex $a' \in A$ which is matched in M_t but not in M (since $|M| = |M_t|$). Then, we can simply slide ab to $a'b$.

- Let A' the set of vertices in A which are matched to vertices in B in M_s (and by the point above, also in M_t), and G' the complete bipartite graph between A and B. Note that G' is a subgraph of G. The edges in M_s (and M_t) in G' form a perfect matching of G'. These perfect matchings can be seen as a permutation on the vertices of B. A flip in this graph consists in applying a transposition to the permutation. Hence, transforming M into M_t is equivalent to transforming one permutation into an other using transposition. It is well known that this is always possible using at most $|B|$ transpositions.

Hence, if there is a transformation of length t from M_s^A to M_t^A, then there is a transformation of length $t + O(|B|)$ from M_s to M_t.

Conversely, assume that there is a transformation sequence from M_s to M_t. We want to show that there is a transformation sequence from M_s^A to M_t^A. For this, we only need to show that if there is a one step transformation between M_s and M_t, there is a one step transformation between M_s^A and M_t^A. We consider the symmetric difference $D = M_s \Delta M_t$. If D contains no edge in $G[A]$, then M_s^A and M_t^A are equal, and there is nothing to prove. Similarly, if $D \subset G[A]$, then M_s^A and M_t^A are adjacent by definition. Thus, we can assume in the following that none of these two cases happen.

If D is a path of length 3 (i.e., the transformation is a sliding move). By the remarks above, we can assume that D contains one edge in $G[A]$, but not the other. Let u, v, w be the vertices of D, with $u, v \in A$ and $w \in B$. Then w is not matched in one of M_s or M_t. This contradicts the assumption that G does not satisfy condition **(C2)**.

If D is a cycle of length 4. There are two possible sub-cases:

- D contains exactly on edge in $G[A]$. In this case D must also contain one edge in $G[B]$, and this implies that one of G_s or G_t contains an edge in $G[B]$, a contradiction of the assumption that G does not satisfy the condition **(C1)**.

- D contains exactly two edges in $G[A]$. These two edges must be incident since otherwise $D \subset G[A]$. Then, this means that $M_s^A \Delta M_t^A$ is a path of length 3 and the two matchings are adjacent in the reconfiguration graph.

If D has three edges in $G[A]$, then we must have $D \subseteq G[A]$, and this case was already handled above.

Hence, in any case, if there is a transformation from M_s to M_t, there is also a transformation from M_s^A to M_t^A. This shows the reverse implication and ends the proof of the lemma. □

Proof of Theorem 27. Given a cograph G and two matchings M_s and M_t of G, the algorithm proceeds as follows:

1. If $|M_s| \neq |M_t|$, then return **no**, otherwise let $k = |M_s| = |M_t|$
2. If G is not connected, call recursively the algorithm on each connected component. Otherwise let A, B be the root partition of G, with $|A| \geq |B|$.
3. If G satisfies one of the conditions **(C1)** and **(C2)**, output **yes**, and produce a transformation sequence using either Lemma 29 or Lemma 31.
4. If G does not satisfy any of these conditions, call recursively the algorithm on A, and decide the instance (and produce a transformation if it exists) using Lemma 32.

Let us show that this algorithm is correct, runs in polynomial time and produces a transformation sequence of linear length.

Correctness. Whenever the algorithm answers **yes**, it also provides a certificate (i.e., a transformation sequence). Hence, the only case where the algorithm might be incorrect is when it answers **no**. Let us show by induction on G that if the algorithm returns **no** on a cograph G given two matchings M_s and M_t as input, then no transformation exists between the two matchings. If the algorithm returns **no** in step 1, then there is clearly no transformation. If one of the recursive calls returns **no** in step 2, then there is trivially no transformation either. Finally, if one of the recursive calls returns **no** in step 4, then there is also no transformation sequence by Lemma 32.

Running-time. At each recursive call, the algorithm only performs a polynomial number of steps. Indeed, checking whether G is connected, and comparing the size of M_s and M_t can be done in polynomial time. Additionally, computing cotree of a cograph can be done in polynomial time, and as we mentioned before, the two conditions **(C1)** and **(C2)** can be verified in polynomial time. Finally, all the recursive calls are made on vertex disjoint subgraphs. Consequently, it follows immediately that the algorithm runs in polynomial time.

Length of transformation. Let G be a cograph, and M_s and M_t two matchings of G such that there is a transformation from M_s to M_t. Let us show by induction on G that the algorithm produces a transformation of length at most Cn for some constant C.

If the algorithm returns at step 2, then by induction it produces a transformation of length at most Cn_i on each component G_i of G, with $n_i = |G_i|$. By combining the transformations on each component, we obtain a transformation for the whole graph of length at most $C(\sum n_i) = Cn$.

If the algorithm returns **yes** during step 3, then the induction step directly follows from Lemmas 29 and 31, provided C is chosen large enough.

Finally, if the algorithm outputs **yes** at step 4, then using the induction hypothesis on A, it produces a transformation of length at most $C|A|$ from $M_{\mathrm{s}} \cap G[A]$ to $M_{\mathrm{t}} \cap G[A]$. By Lemma 32, the total sequence produced by the algorithm has length at most $C|A| + C'|B| \leq Cn$ where C' is the constant in the big-Oh notation of Lemma 32 and assuming $C \geq C'$. □

Part II

Algorithmic Applications of Reconfiguration Aspects

Chapter 5

Random Edge-Colourings with Glauber Dynamics

This chapter covers the results obtained with Michelle Delcourt and Guillem Perrarnau started during a research visit in Birmingham. It presents results on the generation of random colourings using the Markov Chain Monte Carlo method, and Glauber dynamics. These results were published in [DHP18].

We have seen in the previous part some questions related to the computational complexity of reconfiguration problems (transforming one solution into an other) and properties of the reconfiguration graph. These properties have many applications to other problems such as local search [AEOP02] and reoptimisation [STT12]. This chapter and the next one focus on two possible ways to use the transformations between solutions and the properties of the reconfiguration graph for solving specific problem. In the next chapter, we leverage these transformations to design online algorithms for the colouring problem. The idea, which is very standard, is to adapt a partial solution to the problem into a solution for the whole input. A direct extension of a partial solution is not always possible, but the transformations can be used to modify the current solution and make the extension easier.

In this chapter, we are interested in sampling colourings uniformly at random. Throughout this chapter, we will use the notions from Markov Chain theory, which were introduced in Section 1.4. A simple and general method for approximately sampling objects according to some distribution is the Markov Chain Monte Carlo (MCMC) method. It consists in designing a Markov chain (i.e., a random walk) whose states are the objects we wish to sample, and its stationary distribution is equal to the target distribution we wish to draw the objects from. Starting from an arbitrary initial state, simulating this Markov chain for a long enough number of steps provides (under some assumptions) a random solution with a distribution which can be made arbitrarily close to the target distribution. The transformations studied in the reconfiguration setting can often be used to design very simple and natural Markov Chains for the MCMC method.

There are two conditions for this method to work. First, the Markov Chain must be ergodic, i.e., it must have a unique stationary distribution, and the chain must converge to this distribution independently of the chosen starting position. This condition is usually satisfied if the reconfiguration graph is connected. The second condition (and usually the hardest to prove) is that the convergence rate of the Markov Chain is 'fast', i.e., the time it takes for the distribution to be-

come close enough to the stationary distribution is polynomial. In other words, we must show that the mixing time of the Markov Chain is polynomial. This second condition ensures that the MCMC method is efficient: we only need to simulate the Markov Chain for a polynomial number of steps before getting a good approximation of the stationary distribution. Thus, finding good upper bounds on the mixing time of Markov chains is a crucial problem for this method to work, and it is the main question we are considering here.

We investigate the MCMC method in the case of graph colouring, and obtain a polynomial upper bound on the mixing time for edge-colouring of trees with $(\Delta + 1)$ colours. This chapter is organized as follows. In Section 5.1, we present further motivations for studying this problem. We also give an overview of existing results, and explain some connections with problems from statistical physics. Then, in Section 5.2 we present the results obtained with Michelle Delcourt and Guillem Perarnau, together with a high level description of the proof. After some preliminaries and formal definitions in Section 5.3, the main tools used in the proofs are introduced in Section 5.4. These results are standard methods (or simple generalisation of those), used to bound the mixing time of Markov chains. Finally, we prove some bounds on the mixing time of the Glauber dynamics on list vertex colourings of the clique in Section 5.5 before using these results for the proof of our main theorem in Section 5.6.

5.1 Introduction

The problem of sampling random objects is interesting for various reasons. First, a polynomial-time sampler can be used to observe experimentally typical properties of the objects. These experiments are important, as we will see below, in the field of statistical physics where the objects we try to sample are states of a dynamical system, and properties have physical interpretations. Sampling is also related to counting the number of solutions to a problem. In fact, for many problems, including colouring, randomized approximate counting and approximate sampling are polynomially equivalent: a polynomial time algorithm for one of the two problems can be adapted to produce an algorithm for the other [JVV86, SJ89]. Due to this equivalence, finding a Markov chain with fast mixing also gives a good approximation algorithm for counting the number of solutions. From a computational point of view, these counting problems are often difficult, i.e., ♯P-complete. In the case of colouring, counting the number of colourings of a graph is ♯P-complete, even with 3 colours and maximum degree 3 [BDGJ99].

We will focus here on the case of sampling colourings, but the MCMC method has been analysed for many other problems. See [JS96, Ran06] for more applications of the MCMC method, and [FV07] for more details in the case of graph colouring. The method has been considered for sampling other combinatorial objects such asindependent sets [DG00] and matchings [BB00]. Two notable applications of the methods are the problems of estimating the permanent of a matrix [JS89], and computing the volume of convex bodies [DFK91, BDJ98].

For graph colouring, we have seen in the previous chapter two possible choices to transform colourings in one step: the single vertex recolouring, where two colourings are adjacent if they differ on a single vertex, and the Kempe chain recolouring where two colourings are adjacent if they differ by swapping the colours of a maximal 2-coloured connected subgraph. Both types of transitions can be used to construct a Markov chain, but we will focus here on the chain obtained from the single vertex recolouring. In this case, the MCMC algorithm derived from the reconfiguration graph can be formulated by repeating the following a certain number of times:

1. Choose a vertex v uniformly at random.

2. Recolour v with a colour c chosen uniformly at random among the colours not used by its neighbours.

For reasons that will be explained in Section 5.1.1, we will call this process the Glauber dynamics on the (vertex) colourings of G. It is sometimes also called heat-bath Glauber dynamics or single-site Glauber dynamics in the literature. Sometimes Glauber dynamics is also used to denote the 'Metropolis' version of the chain where the colour c is chosen uniformly among all possible colours, and kept only if the colouring remains proper. For most purposes, the two chains behave similarly, and in particular they always have the same mixing time, up to a factor of k. We start with the following simple observation.

Lemma 33. *Let G be a graph, $k \geq 0$, and suppose that $\mathcal{G}(k, G)$ is connected. Then the Glauber dynamics is ergodic, reversible and its stationary distribution is uniform.*

Proof. If the reconfiguration graph is connected, this means that the Glauber dynamics is irreducible since there is a transformation sequence from any colouring to any other. It is also aperiodic since at any step, there is a non-zero probability to recolour the selected vertex with its current colour, and thus leave the colouring unmodified. Consequently, Glauber dynamics is ergodic, and by Theorem 1 it has a unique stationary distribution. To conclude the proof, we only need to check that the transition matrix P is symmetric. Indeed, if σ and τ are two colourings which differ only at vertex v, then:

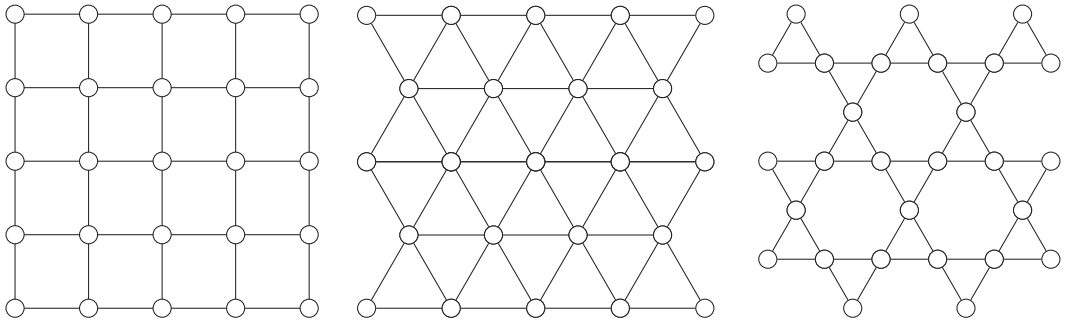
$$P[\sigma \to \tau] = \frac{1}{np_v},$$

where p_v is the number of colours available at v. In other words, p_v is equal to k minus the number of colours present in $N(v)$. In particular, this quantity does not depend on the colour of v itself, and $P[\sigma \to \tau] = P[\tau \to \sigma]$. □

If this lemma ensures that the Glauber dynamics is ergodic, it does not give any practical upper bound on the mixing time. Proving upper bounds on the mixing time is the central question we are interested in here. Note that this question is strongly related to the diameter of the reconfiguration graph $\mathcal{G}(k, G)$ studied in Chapter 3. Indeed, the diameter gives a lower bound on mixing time: after $\frac{\text{diam}(\mathcal{G}(k,G))}{2}$ steps, there are some initial starting points from which less than half of the possible colourings can be reached. For these starting points, simulating the Markov chain for $\frac{\text{diam}(\mathcal{G}(k,G))}{2}$ steps is not enough to get a distribution close to uniform.

5.1.1 Statistical Physics

This kind of mechanisms, with vertices updated one at a times, are called Glauber dynamics in the statistical physics community. Due to its simplicity, it has been used to simulate many different particle interaction systems. One of these, called the Potts model, a generalisation of the Ising model, is strongly related to the colouring problem. It consists in a network G, where each vertex represents a particle, and edges are potential interactions between them. Each particle has one state from the set $\{1, \ldots, q\}$. Its state is chosen with probabilities depending on the state of its



(a) (b) (c)

Figure 5.1: Example of lattices on which Glauber dynamics have been considered, with (a) the square lattice, (b) the triangular lattice, (c) the Kagome lattice.

neighbours. More precisely, a configuration σ (i.e., a not necessarily proper colouring) is assigned an energy as follows:

$$E(\sigma) = -J \sum_{(x,y)\in G} \delta_{\sigma(x),\sigma(y)} \ ,$$

where J is a constant, and $\delta_{a,b}$ is the Kronecker symbol, equal to 1 if $a = b$ and 0 otherwise. A configuration σ will occur with a probability proportional to $e^{\frac{-E(\sigma)}{T}}$, where T is the temperature of the system. When $J > 0$, this corresponds to the ferromagnetic case where neighbouring particles tend to choose similar states. On the contrary, for $J < 0$, this is the anti-ferromagnetic case. In the anti-ferromagnetic case, when the temperature T tends to 0, only proper colourings are feasible configurations. This corresponds to the process we described above. Some questions about these physical systems are related to the chromatic polynomial and the positions of its zeros and more details on this connection can be found in [BEMPS10].

Due to this interpretation as particle systems, problems related to Glauber dynamics have been mainly studied, in the statistical physics community, on infinite graphs, and in particular regular lattices such as the square lattice [AMMVB04, AMMVB05], the triangular lattice [GMP05, Jal12] or the Kagome lattice [MS10a, Jal09] (see Figure 5.1). The approach to the problem is also quite different, with a focus on the properties of a stationary distribution instead of the dynamical process itself[1]. The main intention is to study the macroscopic properties of the system, distinguish whether the system is in an 'ordered' or 'disordered' phase, and characterize the phase transitions (i.e., abrupt variations of some of these properties) which can occur when some of the parameters vary, like the temperature or the number of colours. This is usually done by studying the two following questions, which are presented here in the case of the colouring problems, but can be stated in a much more general setting:

- **Uniqueness of the Gibbs distribution.** On an infinite graph L, there might be more than one distribution on the set of all possible colourings of L which extend the uniform distribution on finite subgraphs of L. Stated differently, the stationary distribution of the Glauber dynamics on the infinite graph might not be unique. Distinguishing whether there is one or several such distributions is an important question. The presence of several stationary

[1]The dynamics we described above, with one vertex updated at each step do not make much sense for infinite graphs, however, it is possible to adapt them for infinite graphs, for example using their continuous-time version.

distributions is often interpreted as the system being in an ordered phase, and is related to the second question:

- **Long range correlations.** This notion is often called 'spatial mixing', and asks how fast do correlations between particles decrease as a function of the distance between them. Informally, an infinite graph L has 'strong spatial mixing' if for any finite subgraph of L, the distribution of colours for the vertices far from the boundary is almost independent from the boundary conditions, i.e., the choice of colours for the vertices on the boundary. In other words, the choice of the boundary conditions has a limited impact on the distribution of colours of the vertices far from it.

Giving a formal definition of the notions above is well outside the scope of this chapter, but the interested reader can refer to [Wei05] for a nice introduction on these concepts. These three problems, uniqueness, spatial mixing, and temporal mixing are very correlated. It is known (for some precise definitions of the notions) that fast temporal mixing and strong spatial mixing are equivalent on the integer lattice $\mathbb{Z}^d$ [DSVW04], while in general bounded degree graphs, fast temporal mixing implies some form of strong spatial mixing [BKMP05]. Additionally, some results used to prove the uniqueness of the Gibbs distribution, called Dobrushin type conditions, imply both spatial mixing and temporal mixing [Hay06]. The flavour of these conditions, based on coupling arguments, reminds of the coupling arguments often used for bounding the mixing time. More details on the relations between these notions can be found in [WS04].

The case of the square grid (see Figure 5.1a) has attracted particular attention. Despite this, some interesting questions remain open to this date. It was shown in [GMP04] that the Glauber dynamics on the square lattice with 3 colours has polynomial mixing time, while fast mixing, both in time and space was shown when the number of colours is at least 6 [AMMVB04, GJMP06]. The case of 4 and 5 colours is surprisingly still open, while numerical experiments seem to suggests that spatial mixing occurs as soon as $k \geq 4$ [FS99]. In three dimensions, fast mixing was shown for $q \geq 10$ colours [GMP05]. On the other hand, exponential lower bounds on the mixing time are known [GR07, GKRS15] for the d-dimensional case with 3 colours, for some large enough constant d.

An other interesting result is [GMP05] which shows spatial mixing for triangle-free lattices and $k > 1.763\Delta$ colours. This can be compared to the results of [DFHV04] which shows fast temporal mixing for the same number of colours, but with the stronger assumption that the girth of the graph is at least 5. Whether the results of [GMP05] can be adapted to prove a polynomial mixing time for triangle-free graphs and $k > 1.763\Delta$ does not seem obvious and is an interesting question.

5.1.2 Mixing time of the Glauber Dynamics

The problem of finding tight bounds on the mixing time of the Glauber dynamics on colourings has also attracted a lot of attention from the computer science community. The problem only makes sense if the reconfiguration graph is connected since otherwise the stationary distribution depends on the initial position. We know from Chapter 3 that this condition is satisfied if the number of colours k is at least $\Delta + 2$. Moreover it has been conjectured that this condition is enough to get a polynomial mixing time.

Conjecture 2. The Glauber dynamics on the vertex colouring of G with $k \geq \Delta(G) + 2$ colours has mixing time $O(n \log n)$.

This conjecture was proved in the particular case of 3-regular graphs with 5 colours but remains open in many other cases, even if the $O(n \log n)$ upper bound is relaxed to just polynomial. A weaker version of the conjecture was proved by Jerrum [Jer95], and independently using a completely different method by Salas and Skolal [SS97]. Both showed that if $k > 2\Delta$, then the mixing time is $O(n \log n)$. This result was later improved by Vigoda [Vig00], weakening the condition to $k \geq \frac{11}{6}\Delta$, and very recently improved to $k \geq (\frac{11}{6} - \eta)\Delta$ for some fixed $\eta \approx \frac{1}{10000}$ by both [CM18] and [DPP18] independently. Other improvements on the number of colours were found with additional restrictions on the graph. A summary of the best results in this direction are given in Table 5.2.

There are a few observations we can make from the results in Table 5.2. First, an $O(n \log n)$ upper bound on the mixing time appears for many of the cases. This bound is usually believed to be optimal, and the justification for this is to invoke a coupon collector argument. The coupon collector problem consists in repeatedly choosing an item uniformly at random from a set of n elements (the chosen item is not removed from the set). The question is how many steps it takes on average to select all the items at least once. It is known that it takes on average $n \log n$ steps to pick all the elements at least once. Hence, it seems that $n \log n$ steps should be a lower bound on the mixing time of Glauber dynamics since with less steps there are good probabilities that some vertices were never chosen and still have their initial colour. This intuition is unfortunately not necessarily correct as was shown in [HS05], and it is not obvious whether it can be turned into a formal argument. In [HS05], the authors give an example of a dynamical system on a graph for which the mixing time of the Glauber dynamics is only $O(n)$, and formally prove the $\Omega(n \log n)$ lower bound for the case of colouring in bounded degree graphs. The case of large-degree graphs is still open which suggests the following:

Open Problem 10. Show that the $\Omega(n \log n)$ lower bound on the mixing time of Glauber dynamics on colouring also holds for any graph or provide a counter example.

A second thing which can be observed by looking at the results in Table 5.2 is that a very large amount of research has been directed at studying the dynamics on graphs with sparse neighbourhoods, and in particular graphs with conditions on the girth and/or the maximum degree [DF03, HV03, Hay03, Mol04, HV07, DFHV04]. On the other hand, quite surprisingly, the case of more dense graphs has attracted very little attention, even though there is no hard evidence suggesting these graphs could be more difficult to analyse. For example, a very simple coupling argument can show that the mixing time on a clique is at most $O(n \log n)$ with $n+1$ colours. Graphs which are not sparse, but with a very simple structure, such as split graphs, interval graphs, or chordal graphs could be interesting candidates to test the conjecture. The case of powers of paths, or powers of grids would be also be very interesting, with a natural interpretation from statistical physics as particles interacting up to a fixed distance (instead of just with their neighbours). The results we obtained with Michelle Delcourt and Guillem Perarnau on edge-colourings of trees are a first step in the direction of studying graphs with dense neighbourhood. Note that prior to this work, the only known result specific to edge-colouring is [Poo16] which shows a polynomial mixing time for edge-colouring of regular trees and $k \geq 2\Delta$.

The case of sampling random edge-colouring (or equivalently vertex colourings of the line-graph), is closely related to some fundamental questions in combinatorics. For instance, $2n$-edge-colourings

[2]Locally sparse graphs are such that $N(v)$ contains few edges for every vertex v. The reader may refer to [FV06] for a formal definition.

Class of graph	Number of colours	Mixing time	Reference
General graphs	$k > (2-\eta)\Delta$	$O(n \log n)$	[DGM02]
General graphs	$k > (\frac{11}{6}-\eta)\Delta$	$O(n^2)$	[CM18, DPP18]
Locally sparse[2] graphs, $\Delta = \Omega(\log n)$	$k > (1.763+\varepsilon)\Delta$	$O(n \log n)$	[FV06]
girth$(G) \geq 5$, $\Delta = \Omega(1)$	$k > (1.763+\varepsilon)\Delta$	$O(n \log n)$	[DFHV04]
girth$(G) \geq 5$, $\Delta = \Omega(\log^3 n)$	$k > (1.645+\varepsilon)\Delta$	$O(n \log n)$	[LM06]
girth$(G) \geq 6, \Delta = \Omega(\log n)$	$k > (1.489+\varepsilon)\Delta$	$O(n \log n)$	[Hay03]
girth$(G) \geq 7$, $\Delta = \Omega(1)$	$k > (1.489+\varepsilon)\Delta$	$O(n \log n)$	[DFHV04]
girth$(G) \geq 11, \Delta = \Omega(\log n)$	$k > (1+\varepsilon)\Delta$	$O(n \log n)$	[HV03]
General graphs	$k > \Delta + \frac{\rho}{1+\varepsilon}$	$O(n \log n)$	[Hay06]
Cut-width at most $O(\frac{\log n}{\log k})$	$k \geq \Delta + 2$	poly(n)	[BKMP05]
$\rho \leq \frac{\Delta^{1-\varepsilon}}{2}$, $\Delta = \Omega(\log^{1+\varepsilon} n)$	$k = \Omega(\frac{\Delta}{\log \Delta})$	$O(n \log n)$	[HVV15]
Planar graphs	$k = \Omega(\frac{\Delta}{\log \Delta})$	$O(n^3 \log^9 n)$	[HVV15]
Trees	$k \geq 3$	$n^{O(1+\frac{\Delta}{k \log \Delta})}$	[LMP09]
$\Delta = 3$	$k = 5$	$O(n \log n)$	[BDGJ99]
Triangle-free graphs, $\Delta = 4$	$k = 7$	$O(n \log n)$	[BDGJ99]
Line-graph of trees	$k \geq \Delta + 1$	poly(n)	this chapter, [DHP18]

Table 5.2: Overview of the best upper bounds on the mixing time of the Glauber dynamics for different classes of graphs. In the table, η is a fixed, small enough constant, ε is a constant which can be taken arbitrarily small, and ρ is the principal eigenvalue of the adjacency matrix.

of the complete graph K_{2n} correspond to 1-factorisations, and n-edge-colourings of the complete bipartite graph $K_{n,n}$ are in bijection with Latin squares. Markov Chain Monte Carlo methods have been introduced to sample such combinatorial objects (see e.g. [DL17, JM96]), but it is not known if they rapidly mix. Even, if more colours are allowed, no result is known for these two graphs apart from the results on general graphs [Vig00].

Open Problem 11. Study the mixing time of the Glauber dynamics on colouring for graphs with dense neighbourhoods such as:

- split/interval/chordal graphs,
- powers of paths/grids,
- line-graphs of cliques/complete bipartite graphs.

Some of the results presented in Table 5.2 are tight. This is the case for example for vertex colouring of trees. An upper bound of $n^{O(1+\frac{\Delta}{k \log \Delta})}$ on the mixing time was shown in [LMP09],

improving on the work of [GJK10] in the case of regular trees. This result is tight due to a matching lower bound of $n^{\Omega(1+\frac{\Delta}{k \log \Delta})}$ proved in [GJK10].

On planar graphs, the polynomial mixing time has been shown in [HVV15] with a number of colours of order $\frac{\Delta}{\log \Delta}$, well below the maximum degree of the graph. On the other hand, if the number of colours is $o(\frac{\Delta}{\log \Delta})$, the mixing time is super-polynomial even for a star. This lower bound comes from the fact that the only way to recolour the central vertex of the star is if the leaves are coloured with only $k-2$ colours. By recolouring the vertices at random, if the number of colours is too small, it takes a very long time to reach such a colouring. Hence, after only a polynomial number of steps, there is still a good chance that the root was never recoloured, and the colouring is far from being uniformly distributed. The case of planar graphs is of particular interest in view of the following conjecture.

Conjecture 3 ([Wel93])**.** For any constant $k \geq 4$, there is no Fully Polynomial Randomized Approximation Scheme (FPRAS) for approximately sampling k-colourings of planar graphs.

In particular, if this conjecture holds, it implies that for any Markov Chain on the k-colourings of planar graphs for some constant k, there are graphs G for which the mixing time is super-polynomial. This holds in the case of the Glauber dynamics as mentioned above. An other natural chain to consider is the flip chain, where transitions can be arbitrary Kempe chains. However, an exponential lower bound on some planar graph was shown for this chain as well in [LV05]. If these results and the conjecture above seem to suggest little hope for finding an algorithm for sampling random colourings of planar graphs with any constant number of colours, an interesting case would be to consider the additional condition that the graph has bounded degree. The counter examples above say nothing in this case, and the results already known on trees for which the mixing time is polynomial with the bounded degree condition even if $k = 3$, suggests that it might be possible to get an upper bound of the form $n^{f(\Delta)}$ on the mixing time of the Glauber dynamics on planar graphs with bounded degree. This research direction would be a special case of the following more general consideration.

We have seen in previous chapters that the reconfiguration graph for single vertex recolouring is connected as soon as $k \geq \text{col}(G)+2$. In general, this condition is not enough to get a polynomial mixing time as shown above with the example of the star. A natural restriction would be to consider the case where the maximum degree is not much larger than the degeneracy. This suggests the following open problem:

Open Problem 12. Let G be a graph with maximum degree at most $C \cdot \text{col}(G)$ for some constant C. Does the Glauber dynamics on G with $k \geq (\text{col}(G)+2)$ colours have polynomial mixing time?

5.1.3 Methods

Apart from the methods coming from statistical physics, such as spatial mixing, there are essentially two main techniques used to prove upper bounds on the mixing time for the Glauber dynamics on colourings: coupling, and comparison methods. Our results only use the comparison techniques, but both methods have been widely used in the literature. We give here a short description of the two methods.

Coupling [Ald82] has been widely used in part due to its simplicity, even though the upper bounds on the mixing time it provides are sometimes not tight [Gur16]. The idea is to look at two

(not-independent) Markov Chains such that each chain taken independently behaves as the process we want to study, in our case the Glauber dynamic on colouring of some graph G. Looking at the average time it takes for the two chains to reach the same state gives an upper bound on the mixing time. By carefully choosing the joint evolution of the two chains, i.e., defining a coupling between the two chains, this method allows to get good upper bounds on the mixing time.

This technique gained popularity with the path coupling theorem [BD97] which simplifies the the construction of the coupling, and provides simple conditions to prove fast mixing. It is the main ingredient to some of the best current results including the proof from Vigoda [Vig00] and its following improvements [DPP18, CM18]. Some of the most recent results combine this method with local uniformity properties: properties satisfied with good probabilities by random colourings (e.g. [DF03, Mol04, Hay03]). This method is also well suited for machine assisted proof as was done in [BDGJ99] in the case of colouring or in [HLZ16] for other problems. The general idea of this approach is to define the coupling using some parameters, and use a computer for fine-tuning the choice of the parameters and prove that the conditions of the path coupling theorem holds.

Comparison methods consists in, as the name implies, comparing two Markov chains: an 'unknown' chain whose mixing time we are trying to bound, and a 'known' chain whose mixing time is easier to compute. Informally, this method can be described as simulating the 'known' chain using the transitions from the 'unknown' chain. If this simulation satisfies some property, namely that no transition of the 'unknown' chain is used too often, then we can upper bound the mixing time of the 'unknown' chain in terms of the mixing time of the 'known' chain.

The special case where the 'known' chain is just, at each step, choosing a new colouring of the whole graph uniformly at random appears in the literature (sometimes with variations in the exact statement) under the name 'canonical paths method', or 'weighted/fractional paths method' or also 'multi-commodity flow method'. Our proofs rely heavily on this technique, and a formal description of it is given in Section 5.4.1. This kind of approach was first introduced in [DSC93]. One main benefit of this method is that it allows to compare the mixing time of different dynamics. Consequently, one way to bound the mixing time for the Glauber dynamics is to compare it to other dynamics which might be easier to study such as the block dynamic that we use here and is also used in [LMP09], or a chain with the addition of Kempe moves as is done in [Vig00].

5.1.4 Related Results

Thanks to the comparison method mentioned above, studying the mixing time of variants of the initial chain might help in bounding the mixing time for the Glauber dynamics. Variants with different transitions have been considered in the literature. An example is the block dynamics, where several vertices can be updated in one single step. An other example are dynamics using Kempe chains, sometimes called flip dynamics or the Swendsen–Wang–Kotecký (WSK) algorithm [MS10a]. The case of systematic scan, updating the vertices in a specified order instead of choosing them at random, has also attracted some interest due to the simplicity of its implementation for simulating dynamical systems [Ped08, DGJ06]. Dynamics with a different state space have been also been considered, such as the single flaw dynamics from [Var17] which allows for a small number of monochromatic edges.

Variants of the colouring problem for which the Glauber dynamics has been studied include list colouring [GK07], radio frequency assignment [FNPS05] and hypergraph colouring [BDK05]. Several results were proved for the Glauber dynamics on random graphs [DF10, MS10b]. Finally,

some interesting result were found for the related problems of exact sampling. The MCMC method allows for approximate sampling: by running the chain long enough we can get arbitrarily close to the wanted distribution. Using additional tools, this method can sometimes be turned into algorithms for sampling exactly according to the wanted distribution [Hub98].

5.2 Result and Proof Overview

Our main result is a polynomial bound on the mixing time for the Glauber dynamics on edge-colourings of a tree.

Theorem 34. *Let T be a tree on n vertices and maximum degree $\Delta \geq 3$. For $k \geq \Delta + 1$ the Glauber dynamics for k-edge-colourings on T mixes in time $n^{O(1)}$.*

Note that the constant in the exponent is independent from Δ. Also, the number of colours in the statement of the theorem is optimal since the process is not ergodic with fewer colours. Indeed, the linegraph of a tree with maximum degree Δ contains a clique with Δ vertices. With only Δ colours, all the colourings of this clique are frozen, and the Markov Chain is not ergodic. We now give a brief overview of the strategy used to prove Theorem 34.

For technical reasons, instead of bounding the mixing time of the chain directly, we consider the relaxation time of a continuous-time version of the chain. As we have seen in Section 1.4, for the purpose of getting a polynomial upper bound on the mixing time, this change makes no difference.

The first step of the proof consists in showing that it suffices to bound the relaxation time for d-regular trees where $d = k - 1$, which notably simplifies the proof. To this end, we prove a monotonicity result on the relaxation time of the Glauber dynamics (see Section 5.4.3). This result can be of independent interest since it provides a way to compare two Markov Chain with similar transitions but a different state space, while standard comparison techniques usually consider Markov Chains with the same state space but different transitions.

The main idea of the proof of the theorem is to recursively decompose the tree into subtrees, and study the process restricted to each subtree. The approach by decomposition has already been used in the literature to bound the relaxation time of the Glauber dynamics for vertex-colourings of trees [BKMP05, LMP09]. To analyse the decomposition procedure we need to study the associated block dynamics, which can be informally described as follows: given a partition of the tree into subtrees, at each step we select one subtree and recolour it entirely by choosing uniformly at random a colouring which is compatible with the boundary condition. We then use a result of Martinelli [Mar99] on block dynamics to upper bound the relaxation time of the whole process in terms of the relaxation time on each block independently, and the relaxation time of the block dynamics. Our decomposition procedure will satisfy two properties,

i) the Glauber dynamics on each subtree is ergodic for every possible boundary condition;

ii) the number of recursive decompositions is small (i.e., logarithmic in the size of the tree).

Condition i) is necessary to apply Martinelli's result on block dynamics. Moreover, if ii) holds, the upper bound we obtain on the relaxation time is polynomial in n, with an exponent that is independent from Δ and k.

The splitting strategy is described precisely in Section 5.6.2. Informally speaking, to ensure that ii) holds, at every step the tree is split into subtrees of at most half the size of the original

tree. To ensure that i) holds, we make sure that every subtree has at most two edges incident to the boundary, and that these two edges are non-adjacent. The latter condition is not necessary for $k \geq \Delta + 2$ — in this case the Markov Chain remains ergodic even if this condition is not satisfied — but it is crucial for the case $k = \Delta + 1$. The strategy is implemented by splitting the tree so that all the subtrees are pending from a vertex, or an edge.

The second ingredient of the proof is a bound on the relaxation time of the block dynamics, where each block corresponds to a subtree hanging from a root vertex. This dynamic is very similar to the Glauber dynamics on the star spanned by the edges incident to the root, constrained to a boundary condition. In Section 5.6.1 we reduce the block dynamics to an analogous problem: bounding the relaxation time of list-vertex-colourings of a clique (the line-graph of a star), where the lists are given by the boundary constraints and each vertex is updated at a different rate. In Section 5.5, we use the weighted canonical paths method of Lucier and Molloy [LMP09] and a multi-commodity flow argument to bound the relaxation time of the list-colourings of the clique. These results are enough to prove the theorem for $k \geq \Delta + 2$.

To conclude the proof in the case $k = \Delta + 1$, we need to obtain a bound on the relaxation time of the block dynamics where the blocks correspond to trees hanging from a root edge instead of a vertex. In this case, it suffices to study the list-vertex-colouring dynamics on the graph formed by two cliques intersecting at a vertex, which we do in Section 5.5.3.

Our main technical results in Section 5.5 involves studying the relaxation time of the Glauber dynamics on list-colourings of a clique where some vertices are updated more often than others. Our results needs some properties on the list assignment to ensure that the process remains ergodic. Although these properties are enough to ensure ergodicity of the chain, they are not necessary. This prompts the following question.

Open Problem 13. Does the Glauber dynamics on list-colourings of a clique have polynomial mixing time as soon as it is ergodic?

Additionally, the results in Section 5.5 handle the case where the updates are not uniform: some vertices are updated more often than others. An interesting research direction would be to evaluate the impact of these non-uniformities on the mixing time of the Markov chain. Note that the non-uniformities can be of two types: updating the vertices at different rates, or choosing the colours with different probabilities. In this second case, the stationary distribution is no longer uniform.

5.3 Preliminaries

Let $G = (V, E)$ be a finite graph. Let Ω_V denote the set of k-vertex-colourings of G. The set of k-edge-colourings (i.e., k-vertex-colourings of the line-graph G^ℓ) will be denoted by Ω_E. Throughout the chapter, we use similar notation to distinguish the vertex and the edge-version of each set or parameter.

Recall from Chapter 1 that for $\mu \in \Omega_V$ and $U \subseteq V$, $\mu|_U$ stands for the restriction of μ to U. We denote by $\mu(U)$ the set of colours used by all the vertices in U. We write Ω_U^μ to denote the set of colourings $\sigma \in \Omega_V$ which agree with μ on $V \setminus U$, i.e., such that $\sigma|_{V\setminus U} = \mu|_{V\setminus U}$. Informally, we think of Ω_U^μ as colourings of U which are compatible with μ in the boundary of U.

If L is a list assignment on the vertices of G, we denote by Ω_V^L the set of all L-colourings of G. Note that if $U \subseteq V$ and H is the subgraph of G induced by U, then any $\mu \in \Omega_V$ yields a

list assignment for U where $L^\mu(u) = [k] \setminus \mu(N(u) \setminus U)$. This corresponds to fixing the colours of the vertices outside of U, and removing the colours used by these vertices from the lists of their neighbours in U. It gives a natural bijection between Ω_U^μ and $\Omega_U^{L^\mu}$.

In this chapter we will focus on k-edge-colourings of a tree G on n vertices with maximum degree at most Δ. Note that G^ℓ is a union of cliques of size at most Δ such that two cliques intersect in at most one vertex and every cycle is contained in some clique.

The vertices of G with degree 1 are called *leaves* and the vertices with degree at least 2, *internal vertices*. If T is a subtree of G, we define the *exterior and interior (edge) boundary of* T respectively as

leaves

internal vertices

exterior and interior (edge) boundary of T

$$\partial_G T = \{e \in E \setminus E(T) : N(e) \cap E(T) \neq \emptyset\} ,$$
$$\overline{\partial}_G T = \{e \in E(T) : N(e) \cap \partial_G T \neq \emptyset\} ,$$

where $N(e)$ is the set of edges in G incident to e. If G is clear from the context, we will denote them by ∂T and $\overline{\partial} T$. A subtree T has a *fringe boundary* if all edges in $\overline{\partial} T$ have an endpoint that is a leaf of T. We will use t to denote the size of $\overline{\partial} T$.

fringe boundary

In this chapter, we will always consider t to be a constant with respect to n, Δ, k. In this sense, for functions f and g we use $f = O_t(g)$ if there exists $c = c(t)$ such that $\limsup f/g \leq c(t)$. We also use $f = \Theta_t(g)$ if $f = O_t(g)$ and $g = O_t(f)$.

5.3.1 Glauber dynamics

For a graph $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ and a positive integer k, the *Glauber dynamics for k-vertex-colourings of G* is a discrete-time Markov chain X_t on Ω_V, where X_{t+1} is obtained from X_t by choosing a $v \in V$ and $c \in [k]$ uniformly at random, and updating v with c if this colour does not appear in $N(v)$. Note that this definition differs slightly from the one given in the introduction where the colour c is chosen uniformly among the colours not used by the neighbours of v. However, it is known that the two chains have the same mixing time up to a factor of k.

In all the rest of this chapter, we will work with the continuous-time version of this chain. The *continuous-time Glauber dynamics for k-vertex-colourings of G with parameters* $(p_1, \dots, p_n)$ is a continuous-time Markov chain on Ω_V with generator matrix given by

$$\mathcal{L}_V[\sigma \to \eta] = \begin{cases} p_i & \text{if } \sigma, \eta \text{ differ at } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

We call $\mathcal{L}_V$ *uniform* if $p_i = 1/k$ for every $i \in [n]$. As $\mathcal{L}_V$ is symmetric for any set of parameters (i.e. $\mathcal{L}_V[\sigma \to \eta] = \mathcal{L}_V[\eta \to \sigma]$), if $\mathcal{L}_V$ is ergodic then its (unique) stationary distribution π is uniform on Ω_V.

Note that the continuous version updates vertices faster than the discrete one. Indeed, in the continuous version, at time 1, an average of n updates have already been made. In particular, we have $\mathcal{L}_V = n(P - I)$, where P is the transition matrix of X_t and I is the identity matrix. It follows from standard Markov chain comparison results (see e.g. [RT00]) that

$$t_{\text{mix}}(X_t) = O(n\, t_{\text{mix}}(\mathcal{L}_V)) . \tag{5.1}$$

If $G = (V, E)$ is a tree with maximum degree Δ its line graph G^ℓ is $(\Delta - 1)$-degenerate (i.e. every subgraph has a vertex of degree at most $\Delta - 1$). It is known [DFFV06] that in this case the Glauber dynamics $\mathcal{L}_E$ for k-edge-colourings of G is ergodic provided $k \geq \Delta + 1$.

For $U \subseteq V$ and $\mu \in \Omega_V$, denote by $\mathcal{L}_U^\mu$ the dynamics defined by $\mathcal{L}_V$ restricted to the set Ω_U^μ. Similarly, for a list assignment L of U, let $\mathcal{L}_U^L$ be the dynamics defined on the state space Ω_U^L. All these chains are symmetric but their ergodicity will depend on the size of the boundary of μ in U, and the size of the lists $L(u)$ for $u \in U$. If ergodic, then their stationary distribution will be uniform in the corresponding state space.

5.4 Comparison of Markov chains

This section introduces methods for the analysis of the relaxation time of reversible Markov chains that we will use later in the proofs. The first result is a unified framework of two well-known Markov chain comparison techniques. We then define the (reduced) block dynamics and present known results on how this can be used to bound the original chain. Finally, we provide a result on the monotonicity of the Glauber dynamics that will allow us to notably simplify the proof.

5.4.1 The weighted multi-commodity flow method

In this subsection we present a unified framework for two similar techniques used to compare the relaxation time of two different Markov chains $\mathcal{L}$ and $\mathcal{L}'$ on the same state space. These two techniques are (1) the *fractional paths* (or *multi-commodity flows*) method [Sin92] and (2) the *weighted canonical paths* method [LMP09]. Both methods are generalisations of the *canonical paths method* (see e.g. [DSC93]). The main idea of this method is to simulate the transitions of $\mathcal{L}'$ using transitions of $\mathcal{L}$ in such a way that no transition of $\mathcal{L}$ is used too often, to obtain an upper bound on the ratio between $\tau(\mathcal{L})$ and $\tau(\mathcal{L}')$.

Consider two continuous-time ergodic reversible Markov chains $\mathcal{L}$ and $\mathcal{L}'$ on Ω with stationary distributions π and π', respectively. In the following, we denote by $\omega\colon \Omega \times \Omega \to \mathbb{R}$ a weight function on the transitions of $\mathcal{L}$.

To each $\alpha, \beta \in \Omega$ with $(\alpha, \beta) \in \mathcal{L}'$, we associate a set of paths $\Gamma_{\alpha,\beta}$ where every $\gamma \in \Gamma_{\alpha,\beta}$ is a sequence $\alpha = \xi^0, \ldots, \xi^m = \beta$, for some $m \geq 1$ with $(\xi^{i-1}, \xi^i) \in \mathcal{L}$ for every $i \in [m]$. We define a *flow* to be a function $g\colon \Gamma_{\alpha,\beta} \to [0,1]$ such that $\sum_{\gamma \in \Gamma_{\alpha,\beta}} g(\gamma) = 1$. The *weight* of γ with respect to ω is defined as

$$|\gamma|_\omega := \sum_{i=1}^{m} \omega(\xi^{i-1}, \xi^i) \,.$$

If the weight function ω is a constant equal to 1, then the weight of γ is simply denoted by $|\gamma|$, and corresponds to the length of the transformation sequence. Define for every transition $(\sigma, \eta) \in \mathcal{L}$:

$$b := \max_{\alpha \in \Omega} \frac{\pi(\alpha)}{\pi'(\alpha)} \,,$$

$$\rho_{\sigma,\eta} := \frac{1}{\pi(\sigma)\mathcal{L}[\sigma \to \eta]\omega(\sigma,\eta)} \sum_{(\alpha,\beta)\in\mathcal{L}'} \sum_{\substack{\gamma \in \Gamma_{\alpha,\beta} \\ \gamma \ni (\sigma,\eta)}} g(\gamma)\pi'(\alpha)\mathcal{L}'[\alpha \to \beta] \cdot |\gamma|_\omega \,.$$

The quantity $\rho_{\sigma,\eta}$ corresponds to the *congestion* on the transition (σ, η). Let $\rho_{\max} = \max\{\rho_{\sigma,\eta} : (\sigma,\eta) \in \mathcal{L}\}$ be the maximum congestion over all transitions of $\mathcal{L}$. We are now ready to state the main lemma.

Proposition 35 (Weighted multi-commodity flows method)**.** *We have* $\tau(\mathcal{L}) \leq b^2 \rho_{\max} \cdot \tau(\mathcal{L}')$.

The proof of this result is based on the same ideas as the proofs of both the weighted canonical paths method and the fractional paths method, providing a common framework for them. It is included in the Appendix for sake of completeness, and follows from standard computations using the variational characterisation of $\mathrm{Gap}(\mathcal{L})$ that involves the variance and the Dirichlet form (see Appendix A.1). We will not need this result in full generality in our proofs, but we believe it is interesting in its own right and will use it in two particular cases.

The first case is when the stationary distribution of the two chains is uniform over Ω, and for every pair $(\alpha, \beta) \in \mathcal{L}'$ the set of paths $\Gamma_{\alpha,\beta}$ consists of a single path $\gamma_{\alpha,\beta}$, with $g(\gamma_{\alpha,\beta}) = 1$. In this case, Proposition 35 implies the result in [LMP09].

Proposition 36 (Weighted canonical paths method)**.** *If π and π' are uniform, then*

$$\tau(\mathcal{L}) \leq \tau(\mathcal{L}') \cdot \max_{(\sigma,\eta)\in\mathcal{L}} \left(\frac{1}{\mathcal{L}[\sigma \to \eta]\omega(\sigma,\eta)} \sum_{\substack{(\alpha,\beta)\in\mathcal{L}' \\ \gamma_{\alpha,\beta} \ni (\sigma,\eta)}} \mathcal{L}'[\alpha \to \beta] \cdot |\gamma_{\alpha,\beta}|_\omega \right) .$$

In the second case, all transitions have weight $\omega(\sigma, \eta) = 1$, and again both chains have uniform stationary distributions. In this case, we obtain the following.

Proposition 37 (Fractional paths method)**.** *If π and π' are uniform, then*

$$\tau(\mathcal{L}) \leq \tau(\mathcal{L}') \cdot \max_{(\sigma,\eta)\in\mathcal{L}} \left(\frac{1}{\mathcal{L}[\sigma \to \eta]} \sum_{(\alpha,\beta)\in\mathcal{L}'} \sum_{\substack{\gamma\in\Gamma_{\alpha,\beta} \\ \gamma \ni (\sigma,\eta)}} g(\gamma)\mathcal{L}'[\alpha \to \beta] \cdot |\gamma| \right) .$$

Finally, we state a very simple corollary of Proposition 35.

Corollary 38. *Suppose there exists a constant $c > 1$ such that for every $\alpha, \beta \in \Omega$ it holds that*

- $\frac{1}{c}\pi'(\alpha) \leq \pi(\alpha) \leq c\pi'(\alpha)$;
- $\frac{1}{c}\mathcal{L}'[\alpha \to \beta] \leq \mathcal{L}[\alpha \to \beta] \leq c\mathcal{L}'[\alpha \to \beta]$.

Then, there is a constant $K = c^4$ such that

$$\frac{1}{K}\tau(\mathcal{L}') \leq \tau(\mathcal{L}) \leq K\tau(\mathcal{L}') .$$

This result simply states that if two Markov chains have the same transitions with similar transition rates, and similar stationary distributions, then their relaxation time is also the same up to a constant factor.

5.4.2 Weighted and Reduced Block Dynamics

This subsection describes block dynamics and its reduced version. Informally speaking, block dynamics is a generalisation of Glauber dynamics where one splits the set of vertices into "blocks" (usually with few intersections/interactions between them), and updates each block at a time, according to some probabilities (see [Mar99]). In this chapter we will only consider disjoint blocks

partitioning the vertex set. Note that if all blocks are singletons, we recover Glauber dynamics. The method is presented for vertex-colourings, but it works the same for edge-colourings by taking an edge partition instead. We will present some known results for bounding the relaxation time of Glauber dynamics in terms of the relaxation time of its (reduced) block dynamics.

Let $G = (V_G, E_G)$ be a graph, $T = (V, E)$ be an induced subgraph and let $\mathcal{V} = \{V_1, \dots, V_r\}$ be a *partition* of V. Let $\mu \in \Omega_{V_G}$. Suppose $\mathcal{L}_s^\sigma$ is ergodic for every $i \in [r]$ and every $\sigma \in \Omega_V^\mu$ and let $\pi_{V_i}^\sigma$ denote its stationary distribution, which is also the uniform distribution on $\Omega_{V_i}^\sigma$. The *weighted block dynamics on $\mathcal{V}$ with boundary condition μ* is a continuous-time Markov chain with state space Ω_V^μ and generator matrix $\mathcal{B}_\mathcal{V}^\mu$ given for any $\sigma \neq \eta$ by

$$\mathcal{B}_\mathcal{V}^\mu[\sigma \to \eta] = \begin{cases} g_i \pi_{V_i}^\sigma(\eta) & \text{if there exists } i \in [r] \text{ such that } \eta \in \Omega_{V_i}^\sigma, \\ 0 & \text{otherwise,} \end{cases}$$

where $g_i = \min_{\sigma \in \Omega_V^\mu} \mathrm{Gap}(\mathcal{L}_{V_i}^\sigma)$ is the minimum gap for the Glauber dynamics on the block V_i, where the minimum is taken over all possible boundary conditions. Note that $\mathcal{B}_\mathcal{V}^\mu$ can be understood as the dynamics where each block V_i updates its entire colouring at times given by an independent Poisson clock of rate g_i. The new colouring of V_i is chosen uniformly among all the possible colourings which are compatible with the current boundary conditions. It is clear that the block dynamics is ergodic if the Glauber dynamics is, since each transition of the Glauber dynamics is a valid transition of the block dynamics. Moreover, both dynamics have the same stationary distribution.

An unweighted version of the block dynamics, corresponding to $g_i = 1$, was used by Martinelli in [Mar99]. Lucier and Molloy generalised this result for weighted block dynamics:

Proposition 39 (Proposition 3.2 in [LMP09])**.** *For every $\mu \in \Omega_{V_G}$ and partition $\mathcal{V}$ of V, we have*

$$\tau(\mathcal{L}_V^\mu) \leq \tau(\mathcal{B}_\mathcal{V}^\mu) \,.$$

Given a block partition $\mathcal{V}$, let $H = (U, F)$ be the subgraph of T composed of the vertices adjacent to vertices in other blocks. Let $\Omega_\mathcal{R}$ be the set of colourings of U induced by the colourings in Ω_V^μ; we will use $\hat{\sigma}, \hat{\eta}, \dots$ to denote the elements of $\Omega_\mathcal{R}$. It is convenient to see $\Omega_\mathcal{R}$ as a set of colouring classes of Ω_V^μ where two colourings are equivalent if and only if they coincide on U. More precisely, for $\hat{\sigma} \in \Omega_\mathcal{R}$, let $\Omega_*^{\hat{\sigma}}$ be the set of colourings $\sigma \in \Omega_V^\mu$ with $\sigma|_U = \hat{\sigma}$. In a slight abuse of notation, we write $\Omega_{V_i}^{\hat{\sigma}}$ to denote the set $\{\eta \in \Omega_{V_i}^\sigma : \sigma \in \Omega_*^{\hat{\sigma}}\}$, and we write $\pi_{V_i}^{\hat{\sigma}}$ for $\pi_{V_i}^\sigma$ where σ is an arbitrary colouring in $\Omega_*^{\hat{\sigma}}$. Note that the projection of $\pi_{V_i}^{\hat{\sigma}}$ onto U is well-defined and independent of the choice of $\sigma \in \Omega_*^{\hat{\sigma}}$.

The *reduced* version of $\mathcal{B}_\mathcal{V}^\mu$, is a dynamics with state space $\Omega_\mathcal{R}$ and generator matrix $\mathcal{R}_\mathcal{V}^\mu$ given for any $\hat{\sigma} \neq \hat{\eta}$ by

$$\mathcal{R}_\mathcal{V}^\mu[\hat{\sigma} \to \hat{\eta}] = \begin{cases} g_i \pi_{\mathrm{proj}}^{i,\hat{\sigma}}(\eta) & \text{if there exists } i \in [r] \text{ such that } \hat{\eta} = \eta|_U \text{ for some } \eta \in \Omega_{V_i}^{\hat{\sigma}}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

where $\pi_{\mathrm{proj}}^{i,\hat{\sigma}}$ is the projection of $\pi_{V_i}^{\hat{\sigma}}$ onto U; that is, for an arbitrary $\sigma \in \Omega_*^{\hat{\sigma}}$, the probability distribution on the colourings of U defined for any $\hat{\eta} \in \Omega_\mathcal{R}$ by

$$\pi_{\mathrm{proj}}^{i,\hat{\sigma}}(\hat{\eta}) = \pi_{V_i}^\sigma(\{\eta \in \Omega_{V_i}^\sigma,\, \eta|_U = \hat{\eta}\}) \,. \tag{5.3}$$

In the particular case where each block contains only one vertex in H, only one vertex of H changes colour during a transition of $\mathcal{R}^\mu_\mathcal{V}$. In this case, the reduced block dynamic is very similar to the Glauber dynamics on H with some parameters p_i determined by the Glauber dynamics on V_i, only with slightly different transition rates.

We will use another result of Lucier and Molloy that shows that the weighted block dynamics and the reduced block dynamics have the same relaxation time.

Proposition 40 (Proposition 3.3 in [LMP09])**.** *For every $\mu \in \Omega_{V_G}$ and partition $\mathcal{V}$ of V, we have*

$$\tau(\mathcal{B}^\mu_\mathcal{V}) = \tau(\mathcal{R}^\mu_\mathcal{V}) \,.$$

Finally, we will use the following property. If the reduced block dynamics is ergodic, then the projection of π^μ_V onto U is its stationary distribution.

Lemma 41. *The reduced block dynamics $\mathcal{R}^\mu_\mathcal{V}$ is reversible for the projection of π^μ_V onto U.*

Proof. Recall that $H = (U, F)$ is the subgraph of T induced by the vertices adjacent to vertices in other blocks. Let π_{proj} be projection of π^μ_V onto U; that is, the probability distribution on the colourings of U defined for any $\hat{\sigma} \in \Omega_\mathcal{R}$ by

$$\pi_{\text{proj}}(\hat{\sigma}) = \pi^\mu_V(\{\sigma \in \Omega^\mu_V,\, \sigma|_U = \hat{\sigma}\}) \,. \tag{5.4}$$

First observe that, since H consists of all the vertices in each block which are adjacent to vertices of other blocks, for any $\hat{\sigma} \in \Omega_\mathcal{R}$, any extension of $\hat{\sigma}$ to V is obtained by computing an extension on each of the blocks separately. As π^μ_V is uniform, we have

$$\pi_{\text{proj}}(\hat{\sigma}) = \frac{|\{\sigma \in \Omega^\mu_V,\, \sigma|_U = \hat{\sigma}\}|}{|\Omega^\mu_V|} = \frac{\prod_{j=1}^r |\{\sigma \in \Omega^{\hat{\sigma}}_{V_j},\, \sigma|_U = \hat{\sigma}\}|}{|\Omega^\mu_V|} \,.$$

Thus, it follows that, for any two colourings $\hat{\sigma}, \hat{\eta} \in \Omega_\mathcal{R}$ which differ only on the block V_i, we have

$$\begin{aligned}
\pi_{\text{proj}}(\hat{\sigma})\mathcal{R}^\mu_\mathcal{V}[\hat{\sigma} \to \hat{\eta}] &= \frac{\prod_{j=1}^r |\{\sigma \in \Omega^{\hat{\sigma}}_{V_j},\, \sigma|_U = \hat{\sigma}\}|}{|\Omega^\mu_V|} \cdot g_i \frac{|\{\eta \in \Omega^{\hat{\sigma}}_{V_i},\ \eta|_U = \hat{\eta}\}|}{|\Omega^{\hat{\sigma}}_{V_i}|} \\
&= \frac{g_i \cdot |\{\sigma \in \Omega^{\hat{\sigma}}_{V_i},\, \sigma|_U = \hat{\sigma}\}| \cdot |\{\eta \in \Omega^{\hat{\sigma}}_{V_i},\, \eta|_U = \hat{\eta}\}|}{|\Omega^\mu_V||\Omega^{\hat{\sigma}}_{V_i}|} \cdot \prod_{j \neq i} |\{\sigma \in \Omega^{\hat{\sigma}}_{V_j},\, \sigma|_U = \hat{\sigma}\}| \,.
\end{aligned}$$

This quantity is symmetric in $\hat{\sigma}$ and $\hat{\eta}$. Indeed, since $\hat{\sigma}$ and $\hat{\eta}$ only differ on V_i, we have $\Omega^{\hat{\sigma}}_{V_i} = \Omega^{\hat{\eta}}_{V_i}$. Moreover, for every $j \neq i$, $\hat{\sigma}$ and $\hat{\eta}$ agree on $V_j \cap U$. As a consequence, we have

$$|\{\sigma \in \Omega^{\hat{\sigma}}_{V_j},\, \sigma|_U = \hat{\sigma}\}| = |\{\eta \in \Omega^{\hat{\eta}}_{V_j},\, \eta|_U = \hat{\eta}\}| \,.$$

Thus, it follows that $\pi_{\text{proj}}(\hat{\sigma})\mathcal{R}^\mu_\mathcal{V}[\hat{\sigma} \to \hat{\eta}] = \pi_{\text{proj}}(\hat{\eta})\mathcal{R}^\mu_\mathcal{V}[\hat{\eta} \to \hat{\sigma}]$, and $\mathcal{R}^\mu_\mathcal{V}$ is reversible for π_{proj}. □

5.4.3 Monotonicity of Glauber dynamics

Finally, we introduce a monotonicity statement that will allow us to simplify some of our proofs. The previous subsections gave tools to compare the relaxation time of two Markov chains with the same state space but different transitions. Here we are interested in comparing Markov chains with

similar transitions but different state spaces. A natural example is comparing the relaxation time of the Glauber dynamics on G and H, where H is a subgraph of G. In general, it is not clear which of the two relaxation times should be smaller, however, if H and G have a particular structure, we will be able to derive a monotonicity bound.

Proposition 42. *Let $G = (V, E)$ be a graph on n vertices, and k be a positive integer. Let $v \in V$ such that $N(v)$ induces a clique of size at most $k-2$. For any choice of parameters $(p_1, \dots, p_n)$, the Glauber dynamics $\mathcal{L}_V$ and $\mathcal{L}_{V\setminus\{v\}}$ for k-colourings of V and $V \setminus \{v\}$ respectively and defined with the same parameters satisfy,*

$$\tau\left(\mathcal{L}_{V\setminus\{v\}}\right) \leq \tau(\mathcal{L}_V)\ .$$

The proof follows from standard computations and is included in Appendix A.2.

5.5 Glauber dynamics for list-colourings of a clique

In this section we analyse the relaxation time of the Glauber dynamics for list-vertex-colourings of a clique. This will be used later in the proof of our main result for edge-colourings of trees.

Consider the clique with vertex set $U = \{u_1, \dots, u_d\}$ with $d \geq 1$. Throughout this section we fix the number of colours to $k = d + 1$. Recall that given a list assignment of U, we can define Ω_U^L and the dynamics $\mathcal{L}_U^L$, governed by the parameters $(p_1, \dots, p_d)$, where p_i is the rate at which vertex u_i changes to $c \in [k]$.

For a positive integer t, a list assignment L is *t-feasible* if it satisfies

- $|L(u_i)| \geq t + 1$ for every $i \in [t]$;
- $|L(u_i)| = d + 1$ for every $i \in [d] \setminus [t]$.

We say $u \in U$ is *free* if $|L(u)| = d + 1$, and *constrained* otherwise. We should stress here that all the lists are subsets of $[d + 1]$, although the results in this section can be generalised to arbitrary lists of size $k \geq d + 1$ using a variant of Proposition 42 for list colouring. If L is t-feasible, we have

$$\left(\prod_{i=1}^{t} |L(u_i)| - i + 1\right)(d + 1 - t)! \leq |\Omega_U^L| \leq \left(\prod_{i=1}^{t} |L(u_i)|\right)(d + 1 - t)!\ . \tag{5.5}$$

The chain $\mathcal{L}_U^L$ is symmetric and it will follow from the results below that, if L is t-feasible, for some t, then the chain is also ergodic. So its stationary distribution is uniform on Ω_U^L. The main goal of this section is to prove the following bound on its relaxation time.

Lemma 43. *If L is t-feasible and $k = d + 1$, then we have*

$$\tau(\mathcal{L}_U^L) = O_t\left(\sum_{i=1}^{t} \frac{d}{p_i} + \sum_{i=t+1}^{d} \frac{1}{p_i}\right)\ .$$

In order to prove the lemma we will use the comparison techniques introduced in Section 5.4.1. Consider the dynamics $\mathcal{L}_{\text{unif}}^L$ on Ω_U^L with generator matrix given for any $\sigma \neq \eta$ by

$$\mathcal{L}_{\text{unif}}^L[\sigma \to \eta] = \frac{1}{|\Omega_U^L|}\ . \tag{5.6}$$

Clearly $\tau(\mathcal{L}^L_{\text{unif}}) = 1$. The main idea will be to compare $\tau(\mathcal{L}^L_U)$ with $\tau(\mathcal{L}^L_{\text{unif}})$. For technical reasons, it will be easier to introduce an intermediate chain and compare both to it. Define $\mathcal{L}^L_{\text{int}}$ to be the dynamics on Ω^L_U with generator matrix given for any $\sigma \neq \eta$ by

$$\mathcal{L}^L_{\text{int}}[\sigma \to \eta] = \begin{cases} \frac{1}{|\Omega^L_U|} & \text{if } (\sigma, \eta) \text{ is a good pair,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

Speaking informally, the dynamics $\mathcal{L}^L_{\text{int}}$ can be seen as $\mathcal{L}^L_{\text{unif}}$ where only moves between "good" pairs of colourings are allowed. We defer the formal definition of a good pair to later in the section.

Lemma 43 follows from the combining these two lemmas (proved in the next two subsections) with the fact that $\tau(\mathcal{L}^L_{\text{unif}}) = 1$.

Lemma 44. *If $k = d + 1$, then we have*

$$\frac{\tau(\mathcal{L}^L_U)}{\tau(\mathcal{L}^L_{int})} = O_t\left(\sum_{i=1}^{t} \frac{d}{p_i} + \sum_{i=t+1}^{d} \frac{1}{p_i}\right) .$$

Lemma 45. *If L is t-feasible and $k = d + 1$, then we have*

$$\frac{\tau(\mathcal{L}^L_{int})}{\tau(\mathcal{L}^L_{unif})} = O_t(1) .$$

5.5.1 Comparing $\mathcal{L}^L_U$ with $\mathcal{L}^L_{\text{int}}$: the proof of Lemma 44

Let w be an additional vertex with $L(w) = [k]$, so w is free. Consider an order on $U \cup \{w\}$ where w is the smallest vertex. We can extend $\alpha \in \Omega^L_U$ to $U \cup \{w\}$, by letting $\alpha(w)$ be the unique colour not in $\alpha(U)$. To every pair $\alpha, \beta \in \Omega^L_U$, we can assign a permutation $f = f(\alpha, \beta)$ on $U \cup \{w\}$ such that $f(x) = y$ if and only if $\alpha(x) = \beta(y)$. In particular, if $\alpha = \beta$ then f is the identity permutation. One can see the permutation f as a blocking permutation: if $v = f(u) \in U$ for some $u \in U$, then u blocks v from being directly recoloured from $\alpha(v)$ to $\beta(v)$. However, if $v = f(w) \in U$, then v can be directly changed to $\beta(v)$ in the colouring α.

It is useful to think about f using its representation as a union of directed cycles. Throughout this section, by cycle in the permutation we mean a cycle of order at least 2. If a cycle in f contains w, we will be able to recolour every vertex in the cycle by successively recolouring the vertices whose preimage is w. The main difficulty will arise from handling the other cycles that do not contain w, which correspond to circular blockings of vertices in U (e.g. $u, v \in U$, u blocks v and v blocks u). In this case, we will need to insert w into this cycle, and then process it. The merging operation corresponds to recolouring a vertex in α with the only available colour, which must be in its list. This motivates the following classification: a cycle in f is a *1-cycle* if it contains w, a *2-cycle* if it does not contain w but has at least one free vertex and a *3-cycle* if it does not contain w and all its vertices are constrained.

Recolouring a 3-cycle C might be hard because the only colours available in $L(C)$ might already be taken by other vertices in the clique. This motivates the definition of good pairs which governs $\mathcal{L}^L_{\text{int}}$.

Definition 1. A pair of colourings (α, β) is *good* if $f(\alpha, \beta)$ contains no 3-cycles.

Let (α, β) be a good pair and $f = f(\alpha, \beta)$. A *v-swap* is an operation on f that gives the permutation $\hat{f}$ obtained from f by reassigning $\hat{f}(w) = f(v)$ and $\hat{f}(v) = f(w)$. These operations are in bijection with the valid transitions of $\mathcal{L}_U^L$.

One can define recolouring sequences between α and β using swaps. The order on $U \cup \{w\}$ gives a canonical way to deal with the cycles in $f(\alpha, \beta)$, processing the cycle with the smallest free vertex, at a time. Precisely, while there is a free vertex in a cycle of length at least 2 of f, let v be the smallest one and

- if v is in a 1-cycle, then $v = w$. While $f(w) \neq w$, update f by performing an $f(w)$-swap.
- if v is in a 2-cycle, then $v \in U$. Update f by performing a v-swap. Then, while $f(w) \neq w$, update f by performing an $f(w)$-swap.

As there are no 3-cycles, after termination the procedure produces the identity permutation. As swaps correspond to valid recolouring moves, it gives a recolouring path $\gamma_{\alpha,\beta}$ from α to β that uses transitions from $\mathcal{L}_U^L$. Note that any vertex in U is recoloured at most twice. In fact, a vertex is only recoloured twice if it is the smallest free vertex in a 2-cycle.

For every transition $(\sigma, \eta) \in \mathcal{L}_U^L$, define

$$\Lambda_{\sigma,\eta} = \{(\alpha, \beta) \in \mathcal{L}_{\text{int}}^L : (\sigma, \eta) \in \gamma_{\alpha,\beta}\} .$$

Our goal is to prove Lemma 44 using the weighted canonical paths method from Section 5.4.1. To this end, the two following lemmas will help us analyse the congestion resulting from this construction.

Lemma 46. *Given a transition $(\sigma, \eta) \in \mathcal{L}_U^L$ and a permutation f, there are at most two good pairs (α, β) such that $f = f(\alpha, \beta)$ and $(\alpha, \beta) \in \Lambda_{\sigma,\eta}$.*

Proof. Let v be the vertex at which σ and η differ. The order in which we recolour the vertices is fixed by the permutation f. Let $v_1, \ldots, v_\ell$ be the vertices in the order they are recoloured with repetitions, where v_i is the vertex recoloured at the i-th step. Given this sequence, at every step the only valid transition is to perform a v_i-swap. If the recolouring done by the transition (σ, η) corresponds to the i^*-th step in the sequence, then α and β are fully determined. Indeed, β can be recovered from η by sequentially performing v_i-swaps for every $i > i^*$. Symmetrically, α can be recovered from σ by performing v_i-swaps for every $i < i^*$ (recall that the u-swap operation is an involution).

Since every vertex is recoloured at most twice in a recolouring path, v appears at most twice in the sequence $v_1, \ldots, v_\ell$. So there are at most 2 choices for $i^* \in [\ell]$ with $v_{i^*} = v$, and by the argument above, there are at most two pairs (α, β), with $f(\alpha, \beta) = f$, and $(\alpha, \beta) \in \Lambda_{\sigma,\eta}$. □

Using the previous lemma, we can bound the size of $\Lambda_{\sigma,\eta}$

Lemma 47. *Suppose that σ and η differ at $v \in U$. If v is free, then $|\Lambda_{\sigma,\eta}| = O_t\left(|\Omega_U^L|\right)$. If v is constrained, then $|\Lambda_{\sigma,\eta}| = O_t(d|\Omega_U^L|)$.*

Proof. By Lemma 46, it suffices to bound the number of permutations f which are compatible with (σ, η). The key idea is the following claim that not all candidates for f are compatible, as constrained vertices can only block and be blocked by vertices with colours in their lists.

Claim 48. Let $(\alpha, \beta) \in \Lambda_{\sigma,\eta}$ with $f(\alpha, \beta) = f$. For every constrained vertex $u \in U$ with $u \neq v$, one of the following holds:

i) $\eta(f(u)) = \beta(f(u))$;

ii) $\sigma(f^{-1}(u)) = \alpha(f^{-1}(u))$;

Moreover, i) holds if and only if u is recoloured in $\gamma_{\alpha,\beta}$ before the transition (σ, η).

Proof. By construction of the recolouring paths, constrained vertices are only recoloured once. Let C be the cycle containing u. If $v \notin V(C)$, then the vertices in C are recoloured either all before v, or all after v. In particular we have $\sigma|_{V(C)} = \eta|_{V(C)}$, and it is equal to $\beta|_{V(C)}$ if u is recoloured before (σ, η) and to $\alpha|_{V(C)}$ otherwise. Since $f(u), f^{-1}(u) \in V(C)$, either i) or ii) holds.

If $v \in V(C)$, then

- If $f(u)$ is not the smallest free vertex in C, then assume that i) does not hold. Since $f(u)$ is recoloured only once, this means that $\eta(f(u)) = \alpha(f(u))$. Consequently, $f(u)$ is recoloured after the transition (σ, η), and since $u \neq v$, this is also the case for u. This means that $f^{-1}(u)$ is recoloured either during or after the transition (σ, η), and in both cases ii) holds.

- Symmetrically, if $f^{-1}(u)$ is not the smallest free vertex in C, then assuming that ii) does not hold we conclude that u has been coloured before the transition (σ, η) and that i) holds.

- Finally, if $f(u) = f^{-1}(u)$ is the smallest free vertex in C, then $v = f(u)$. In this case, either this is the first time v is recoloured, so u is recoloured after (σ, η) and $\sigma(v) = \alpha(v)$, and ii) holds, or it is the second time v is recoloured, so u is recoloured before (σ, η) and $\eta(v) = \beta(v)$ and i) holds.

□

Suppose that v is free. As there are at most t constrained vertices, there are at most 2^t choices to decide which of items i) or ii) holds for them. For $s \in [t]$, let $x_1, \dots, x_s$ be the constraint vertices that satisfy i) and $y_1, \dots, y_{t-s}$ be the ones that satisfy ii). For every $i \in [s]$, by definition of f we have $\eta(f(x_i)) = \beta(f(x_i)) = \alpha(x_i) \in L(x_i)$ and there are at most $|L(x_i))|$ choices $z \in U \cup \{w\}$ for $f(x_i) = z$, namely the ones with $\eta(z) \in L(x_i)$. Analogously, for $i \in [t-s]$, by definition of f we have $\sigma(f^{-1}(y_i)) = \alpha(f^{-1}(y_i)) = \beta(y_i) \in L(y_i)$ and there are at most $|L(y_i)|$ choices $z \in U \cup \{w\}$ for $f(z) = y_i$. Thus, there are at most $\prod_{i=1}^{s} |L(x_i)| \prod_{i=1}^{t-s} |L(y_i)| = \prod_{i=1}^{t} |L(u_i)|$ choices for the images of x_i and the preimages of y_i. Note that $f(x_i) \neq y_j$ for every i, j as i) holds if and only if u is recoloured before (σ, η) and v is a free vertex. So these choices fix exactly t images in f. Finally, there are at most $(d+1-t)!$ ways to complete f by choosing successively the image of the remaining elements. Using Lemma 46 and (5.5), we have

$$|\Lambda_{\sigma,\eta}| \leq 2 \cdot 2^t (d+1-t)! \prod_{i=1}^{t} (|L(u_i)|) \leq 2^{t+1} \left(\prod_{i=1}^{t} \frac{|L(u_i)|}{|L(u_i)| - i + 1} \right) |\Omega_U^L| = O_t(|\Omega_U^L|) \, .$$

Assume now that v is constrained. There are at most 2^{t-1} ways to choose a configuration of i) and ii) for the remaining constrained vertices. In comparison to the case where v is free, as neither

i) nor ii) holds for $u = v$, there is an extra factor $|L(v)| \leq d + 1$ to choose either the image or the preimage of v in f. Similarly as before, it follows that

$$|\Lambda_{\sigma,\eta}| = O_t(d|\Omega_U^L|) .$$

□

We are now in a good situation to prove Lemma 44.

Proof of Lemma 44. In order to apply the weighted canonical paths theorem to $\mathcal{L}_U^L$ and $\mathcal{L}_{\text{int}}^L$, we need to choose a weight function ω for all $(\sigma, \eta) \in \mathcal{L}_U^L$. Let v be the vertex where σ and η differ. We define ω as follows

$$\omega(\sigma, \eta) := \begin{cases} d/p_i & \text{if } v = u_i \text{ for } i \in [t], \\ 1/p_i & \text{if } v = u_i \text{ for } i \in [d] \setminus [t]. \end{cases} \tag{5.8}$$

As for every recolouring path γ, each element of U is recoloured at most twice, it follows that

$$|\gamma|_\omega \leq 2 \left(\sum_{i=1}^{t} \frac{d}{p_i} + \sum_{i=t+1}^{d} \frac{1}{p_i} \right) . \tag{5.9}$$

Both stationary distributions of $\mathcal{L}_U^L$ and $\mathcal{L}_{\text{int}}^L$ are uniform on Ω_U^L. Also recall that $\mathcal{L}_{\text{int}}^L[\alpha \to \beta] = 1/|\Omega_U^L|$ for any good pair (α, β).

Using Lemma 47, regardless of whether the vertex where σ and η differ is free or constrained, we can bound the congestion of the transition (σ, η) as follows

$$\begin{aligned} \rho_{\sigma,\eta} &= \frac{1}{\mathcal{L}_U^L[\sigma \to \eta]\omega(\sigma, \eta)} \sum_{(\alpha,\beta)\in\Lambda_{\sigma,\eta}} \frac{|\gamma_{\alpha,\beta}|_\omega}{|\Omega_U^L|} \\ &\leq \frac{|\Lambda_{\sigma,\eta}|}{\mathcal{L}_U^L[\sigma \to \eta]\omega(\sigma, \eta)|\Omega_U^L|} \cdot \max_{\alpha,\beta} |\gamma_{\alpha,\beta}|_\omega \\ &= O_t(\max_{\alpha,\beta} |\gamma_{\alpha,\beta}|_\omega) = O_t\left(\sum_{i=1}^{t} \frac{d}{p_i} + \sum_{i=t+1}^{d} \frac{1}{p_i} \right) . \end{aligned}$$

The desired result follows from Proposition 36. □

5.5.2 Comparing $\mathcal{L}_{\text{int}}^L$ with $\mathcal{L}_{\text{unif}}^L$: the proof of Lemma 45

The proof of this lemma uses the fractional paths method with uniform weights. To define the paths, we first split the set of constrained vertices into two subsets. Let A be the set of constrained vertices u satisfying $|L(u)| \leq 2(3t + 2)$, and let B be the remaining ones. Before we define the paths, we will need the following result.

Lemma 49. *Let $\alpha, \beta \in \Omega_U^L$ and let ξ_A be an L-colouring of the vertices in A. Assume that $\alpha|_A$ and ξ_A differ on at most one vertex, and similarly for ξ_A and $\beta|_A$. There exists a constant $c(t) > 0$ such that there are at least $c(t)|\Omega_U^L|$ colourings $\xi \in \Omega_U^L$ satisfying that $\xi|_A = \xi_A$, and both (α, ξ) and (ξ, β) are good pairs.*

Proof. Assume that $A = \{u_1, \dots, u_a\}$, so $|B| = t - a$. Construct the colouring ξ by setting $\xi|_A = \xi_A$ and then choosing the colours in $U \setminus A$ one by one, starting from the vertices in B, as follows:

- for each $v \in B$, choose $\xi(v) \notin \xi(A) \cup \alpha(A \cup B) \cup \beta(A \cup B)$ that has not been used already;
- for each free vertex, choose a colour not used by the vertices already coloured in ξ.

When we choose the colour for $v \in B$, there are at most $3t+2$ forbidden colours. Indeed, there are at most $a + 2(t-a) + 2$ colours in $\xi(A) \cup \alpha(A \cup B) \cup \beta(A \cup B)$, and at most $t - a$ colours used by the previous vertices in B that have already been coloured by ξ. Thus, for $v \in B$ there are at least $|L(v)| - (3t+2) \geq |L(v)|/2$ choices for $\xi(v)$. Using (5.5), the total number of colourings extending ξ_A is at least

$$\begin{aligned}\prod_{v \in B} (|L(v)| - (3t+2)) \cdot (d+1-t)! &\geq \frac{(d+1-t)!}{2^{t-a}} \prod_{v \in B} |L(v)| \\ &\geq \frac{(d+1-t)!}{2^t (3t+2)^a} \prod_{i=1}^{t} |L(u_i)| \\ &\geq \frac{|\Omega_U^L|}{(2(3t+2))^t} \,.\end{aligned}$$

It suffices to show that for any such extension ξ of ξ_A, (α, ξ) and (ξ, β) are good pairs. We only prove it for (α, ξ) as the other case is symmetric. Assume by contradiction that $f(\alpha, \xi)$ contains a 3-cycle, and let C be this cycle. Then $V(C) \cap B = \emptyset$. Indeed, if $v \in V(C) \cap B$, then there exists a vertex $u \in A \cup B$ with $\alpha(u) = \xi(v)$, but this contradicts the fact that $\xi(v) \notin \alpha(A \cup B)$. Thus, $V(C) \subseteq A$, but this is not possible since $\alpha|_A$ and $\xi|_A$ differ by at most one vertex. □

We can now compare the relaxation times of $\mathcal{L}^L_{\text{int}}$ and $\mathcal{L}^L_{\text{unif}}$.

Proof of Lemma 45. We use the fractional paths method. Note that both $\mathcal{L}^L_{\text{int}}$ and $\mathcal{L}^L_{\text{unif}}$ are ergodic, reversible and symmetric and that their stationary distributions are uniform on Ω^L_U.

It suffices to define a collection of fractional paths $\Gamma_{\alpha,\beta}$ between any two colourings α and β in Ω^L_U. Since there are at most t constrained vertices and their lists have size at least $t+1$, we can find a sequence of L-colourings of A, $\alpha|_A = \xi^0_A, \xi^1_A, \dots, \xi^m_A = \beta|_A$, such that any two consecutive colourings differ by one vertex. Let M be an upper bound on the length of these paths for every α, β, which only depends on t.

For $m \in [M]$, let $\Gamma^m_{\alpha,\beta}$ be the collection of all the paths of the form $\alpha = \xi^0, \xi^1, \dots, \xi^m = \beta$ where $\xi^i|_A = \xi^i_A$, and (ξ^{i-1}, ξ^i) is a good pair for $i \in [m]$. By Lemma 49, for each $i \in [m-1]$ there are at least $c(t)|\Omega^L_U|$ choices for ξ^i, independently of the choices of the other ξ^j for $j \neq i$. Thus, $|\Gamma^m_{\alpha,\beta}| \geq (c(t)|\Omega^L_U|)^{m-1}$, and if g is the uniform flow, then each $\gamma \in \Gamma^m_{\alpha,\beta}$ satisfies $g(\gamma) \leq (c(t)|\Omega^L_U|)^{-(m-1)}$. Let $\Gamma^m = \{\Gamma^m_{\alpha,\beta} : \alpha, \beta \in \Omega^L_U\}$ and $\Gamma = \cup_{m=1}^M \Gamma^m$.

We need to bound the congestion of any good pair (σ, η). We fix $m \in [M]$ and will bound the contribution of Γ^m to it. Let $\gamma = \xi^0, \xi^1, \dots, \xi^m \in \Gamma^m$ containing (σ, η). There are at most m choices for $i \in [m]$ such that $\sigma = \xi^{i-1}$ and $\eta = \xi^i$. Then, there are at most $|\Omega^L_U|^{m-1}$ choices for ξ^j with $j \notin \{i-1, i\}$. Each such path satisfies $g(\gamma) \leq (c(t)|\Omega^L_U|)^{-(m-1)}$. Thus,

$$\sum_{\substack{\gamma \in \Gamma^m \\ \gamma \ni (\sigma,\eta)}} g(\gamma)|\gamma| \leq m c(t)^{-(m-1)} \,.$$

Hence,

$$\rho_{\sigma,\eta} \leq \sum_{m=1}^{M} mc(t)^{-(m-1)} = O_t(1) \ ,$$

and, by Proposition 37, we obtain the desired result. □

5.5.3 Dynamics of two cliques intersecting at a vertex

In this section we study a similar dynamics, that we will also use in the main proof. Let z be a vertex. For $d \geq 1$, let $X = \{z, x_1, \ldots, x_{d-1}\}$ and $Y = \{z, y_1, \ldots, y_{d-1}\}$ two sets of vertices and consider the graph with vertex set $Z = X \cup Y$ where each set X and Y induces a clique. As before, we fix the number of colours $k = d + 1$. For a list assignment L of Z, recall the definition of Ω_Z^L and the dynamics $\mathcal{L}_Z^L$, where we denote by $p_z, p_1, \ldots, p_{d-1}, q_1, \ldots, q_{d-1}$ the parameters for $z, x_1, \ldots, x_{d-1}, y_1, \ldots, y_{d-1}$, respectively.

Let t, t_X, t_Y be non-negative integers with $t \geq t_X + t_Y$ and $1 \leq t_X, t_Y \leq d-1$. Without loss of generality, we will assume that d is sufficiently large with respect to t. If this is not the case, then $|\Omega_Z^L|$ is a constant depending on t, and the relaxation time is $O_t(1)$. A list assignment L is (t, t_X, t_Y)-*feasible* if

- $|L(z)| = d + 1$;
- $|L(x_i)|, |L(y_j)| \geq t$ for every $i \in [t_X]$ and $j \in [t_Y]$;
- $|L(x_i)|, |L(y_j)| = d + 1$ for every $i \in [d-1] \setminus [t_X]$ and $j \in [d-1] \setminus [t_Y]$;

We define free and constrained vertices as before, with the exception of z which is considered a constrained vertex. If L is (t, t_X, t_Y)-feasible, then

$$|\Omega_Z^L| \geq \Big(\prod_{i=1}^{t_X} |L(x_i)| - i + 1\Big)\Big(\prod_{j=1}^{t_Y} |L(y_j)| - j + 1\Big) \cdot (d + 1 - t)(d - t_X)!(d - t_Y)! \tag{5.10}$$

$$|\Omega_Z^L| \leq \Big(\prod_{i=1}^{t_X} |L(x_i)|\Big)\Big(\prod_{j=1}^{t_Y} |L(y_j)|\Big) \cdot d(d - t_X)!(d - t_Y)! \ . \tag{5.11}$$

As before, the chain is symmetric and, if L is (t, t_X, t_Y)-feasible for some t, t_X, t_Y satisfying the conditions stated above, it follows from results below that it is ergodic, so its stationary distribution is uniform.

We will prove a bound analogous to the one in Lemma 43 on the relaxation time of $\mathcal{L}_Z^L$.

Lemma 50. *If L is (t, t_X, t_Y)-feasible and $k = d + 1$, then we have*

$$\tau(\mathcal{L}_Z^L) = O_t\left(\frac{d^2}{p_z} + \sum_{i=1}^{t_X} \frac{d}{p_i} + \sum_{i=t_X+1}^{d-1} \frac{1}{p_i} + \sum_{j=1}^{t_Y} \frac{d}{q_j} + \sum_{j=t_Y+1}^{d-1} \frac{1}{q_j}\right) \ .$$

The proof follows the same lines as the proof of Lemma 43, so we will only sketch it, stressing the parts where the two differ. For $\alpha \in \Omega_Z^L$, denote by α_X and α_Y the restrictions of α onto X and Y, respectively. As we did for the clique, we extend α by adding two artificial vertices: w_X

in X and w_Y in Y, and by assigning to them the only available colour in each set. Define the permutations f_X and f_Y as before.

We say that (α, β) is good if and only if both (α_X, β_X) and (α_Y, β_Y) are good (i.e., there is no 3-cycle in the permutations f_X and f_Y). Remember that z is a constrained vertex, and as a consequence if (α, β) are good, every cycle in the permutations contains a free vertex different from z. Redefine the dynamics $\mathcal{L}^L_{\text{unif}}$ on Ω^L_Z as in (5.6), and, using the new definition of good pairs, redefine $\mathcal{L}^L_{\text{int}}$ on Ω^L_Z as in (5.7). We proceed in two steps by bounding the ratios of the relaxation times of $\mathcal{L}^L_Z$ and $\mathcal{L}^L_{\text{int}}$ and of $\mathcal{L}^L_{\text{int}}$ and $\mathcal{L}^L_{\text{unif}}$.

Lemma 51. *If $k = d + 1$, then we have*

$$\frac{\tau(\mathcal{L}^L_Z)}{\tau(\mathcal{L}^L_{unif})} = O_t\left(\frac{d^2}{p_z} + \sum_{i=0}^{t_X} \frac{d}{p_i} + \sum_{i=t_X+1}^{d-1} \frac{1}{p_i} + \sum_{j=0}^{t_Y} \frac{d}{q_j} + \sum_{j=t_Y+1}^{d-1} \frac{1}{q_j}\right) .$$

Sketch of the proof. We reuse the recolouring paths defined in Lemma 44 for the clique. Given two colourings α and β, denote by $\gamma^X_{\alpha_X,\beta_X}$ and $\gamma^Y_{\alpha_Y,\beta_Y}$ the recolouring paths constructed for each of the two sets X and Y independently. Observe that for each path in these sets, each constrained vertex is recoloured at most once. In particular, z changes its colour at most once. Construct the recolouring path $\gamma_{\alpha,\beta}$ in the following way:

- apply the recolourings in $\gamma^X_{\alpha_X,\beta_X}$ until z needs to be recoloured;
- apply the recolourings in $\gamma^Y_{\alpha_Y,\beta_Y}$;
- apply the remaining recolourings in $\gamma^X_{\alpha_X,\beta_X}$.

Note that in the second step, z can be safely recoloured with $\beta(z)$ because its target colour is available in X, since the next move according to $\gamma^X_{\alpha_X,\beta_X}$ would be to recolour z with colour $\beta(z)$. We need to bound the congestion of each transition for this collection of paths.

For a transition (σ, η), let $\Lambda_{\sigma,\eta}$ be the set of good pairs α, β such that $\gamma_{\alpha,\beta}$ contains (σ, η). The analogues of Lemmas 46 and 47 hold in this setting. In particular, for every (σ, η) differing at a vertex v, if v is free then $|\Lambda_{\sigma,\eta}| = O_t\left(|\Omega^L_Z|\right)$, if $v \neq z$ is constrained then $|\Lambda_{\sigma,\eta}| = O_t(d|\Omega^L_Z|)$, and if $v = z$ then $|\Lambda_{\sigma,\eta}| = O_t(d^2|\Omega^L_Z|)$ as we get an extra factor d for each permutation. This allows us to bound the congestion of a transition as in the previous section, and so Lemma 51 follows. □

Lemma 52. *If L is (t, t_X, t_Y)-feasible and $k = d + 1$, then we have*

$$\frac{\tau(\mathcal{L}^L_{int})}{\tau(\mathcal{L}^L_{unif})} = O_t(1) .$$

Sketch of the proof. The lemma can be proved using the same steps as in the proof of Lemma 45. Let A_X and A_Y be the set of constrained vertices with lists of size at most $2(3t + 2)$ in X and Y, respectively. Let $A = A_X \cup A_Y$ and let B be the set of constrained vertices that are not in A. As z is a constrained vertex with $|L(z)| = d+1 > 2(3t+2)$, we have $z \in B$. The analogous of Lemma 49 still holds in this setting.

For any $\alpha, \beta \in \Omega^L_Z$, we would like to find a sequence $\alpha|_A = \xi^0_A, \ldots, \xi^m_A = \beta|_A$ of colourings of A such that each consecutive pair differs only at one vertex. As $z \notin A$ and as all constrained vertices in A have a list of size at least $t + 1$, this sequence can be found by independently recolouring A_X

and A_Y. Arguing as in the proof of Lemma 45, we can construct many recolouring paths between α and β with transitions that correspond to good pairs. Then, Proposition 37 implies the desired result. □

5.6 Glauber dynamics on edge-colourings of trees

In this section we prove our main theorem. We follow a similar approach to the one of Lucier, Molloy and Peres in [LMP09] for vertex-colourings, by recursively splitting the tree into smaller subtrees using block dynamics. However, there are several points where our strategies differ.

5.6.1 Relaxation time of block dynamics

In all this section, we will assume that $G = (V_G, E_G)$ is a d-regular tree, that is every internal vertex has degree *exactly* d. We also assume that $k = d + 1$.

Definition 2. A subtree T of G is splitting if one of the following holds.

- T is a single edge,
- T has fringe interior boundary $|\overline{\partial} T| \leq 2$. If $\overline{\partial} T = \{e, f\}$, then e and f are not incident.

Fix $\mu \in \Omega_{E_G}$, and fix $T = (V, E)$ a splitting subtree of G. Note that since T has fringe boundary, it is also d-regular. The central point of our proof is to study $\mathcal{L}_E^\mu$ by decomposing it into the dynamics of its subtrees using the block dynamics defined in Section 5.4.2. We will assume that T is rooted in one of the two following ways.

Vertex-rooted trees: The root of T is $r \in V$, an internal vertex of T. Let $e_1, \dots, e_d$ be the edges incident to r. For each $i \in [d]$, we consider the block formed by the edges of the subtree hanging from e_i.

Edge-rooted trees: The root of T is an edge $e = xy$ where x and y are internal vertices of T (so, $e \notin \overline{\partial} T$). Let $e_1, \dots, e_{2d-2}$ be the edges incident to e. We let $\{e\}$ be a block, and for every $i \in [2d-2]$, we consider the block formed by the edges of the subtree hanging from e_i.

In both cases, we denote by $\mathcal{E} = \{E_1, \dots, E_r\}$ the block partition described above. Note that each block in $\mathcal{E}$ contains at most one edge incident to edges in other blocks. Let $H = (U, F)$ be the subgraph induced by these edges. Note that in the case of a vertex rooted tree, H is a star, and in the case of an edge rooted tree, H is a bi-star: two stars joined by an edge. For each $i \in [r]$, let T_i be the subtree with edge set E_i.

Throughout this section we will make the following two assumptions on $\mathcal{E}$.

(A1) T_i is splitting for every $i \in [r]$;

(A2) $\mathcal{L}_{E_i}^\sigma$ is ergodic for every $i \in [r]$ and every $\sigma \in \Omega_E^\mu$.

Thus, we can define the reduced block dynamics on $\mathcal{E}$ with boundary condition μ. Recall from Section 5.4.2 that its state space is $\Omega_{\mathcal{R}}$, the restriction to H of the colourings in Ω_E^μ. Moreover,

its transition matrix is given for any $\hat{\sigma} \neq \hat{\eta}$:

$$\mathcal{R}^{\mu}_{\mathcal{E}}[\hat{\sigma} \to \hat{\eta}] = \begin{cases} g_i \pi^{i,\hat{\sigma}}_{\text{proj}}(\hat{\eta}) & \text{if there exists } i \in [r] \text{ such that } \hat{\eta} = \eta|_U \text{ for some } \eta \in \Omega^{\hat{\sigma}}_{E_i}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.12}$$

where $\pi^{i,\hat{\sigma}}_{\text{proj}}$ is the projection of $\pi^{\hat{\sigma}}_{E_i}$ onto F (see (5.3)). To bound the relaxation time on T, we will proceed in two steps. First we compare the original dynamics $\mathcal{L}^{\mu}_{\mathcal{E}}$ to the reduced block dynamics on $\mathcal{E}$ using Propositions 39 and 40 from Section 5.4.2. Then, we bound the relaxation time for the reduced block dynamics using the results from Section 5.5. The three following lemmas will help us for the second step. They show that the transitions rates and the stationary distribution of the reduced block dynamics are close to uniform. We will first prove bounds on the stationary distribution for the reduced block dynamics, and then proceed to bound the relaxation time for the reduced block dynamics.

Lemma 53. *Assuming (A1)–(A2), the reduced block dynamics $\mathcal{R}^{\mu}_{\mathcal{E}}$ is ergodic and reversible, and its stationary distribution $\pi_{\mathcal{R}}$ is the projection of π^{μ}_{E} onto F.*

Proof. Let H^{ℓ} be the line graph with vertex set F, the set of edges of H. Consider the Glauber dynamics $\mathcal{L}^{L}_{F}$ with the following list constraints on $e \in F$

- $L(e) = [k] \setminus \mu(N(e) \cap \partial T)$ if $e \in \overline{\partial} T$,
- $L(e) = [k]$ for every other edge.

Then, since each block E_i contains only one edge in H, the reduced block dynamics $\mathcal{R}^{\mu}_{\mathcal{E}}$ has exactly the same transitions as $\mathcal{L}^{L}_{F}$, but with possibly different probability transitions. Thus, $\mathcal{R}^{\mu}_{\mathcal{E}}$ is ergodic if and only if $\mathcal{L}^{L}_{F}$ is.

If T is vertex-rooted, then H is a star, and H^{ℓ} is a clique. Additionally, since T is splitting, the two edges in $\overline{\partial} T$ are not adjacent, and in particular only one is in H. This edge, if it exists, is assigned a list of length 2, so L is 1-feasible. The ergodicity of $\mathcal{L}^{L}_{F}$ follows from Lemma 43 with $t \leq 1$.

If T is edge-rooted, then H is a bi-star, and H^{ℓ} is composed of two cliques intersecting at one vertex. Moreover, the two edges in $\overline{\partial} T$ cannot be in the same side of the bi-clique, and each has a list of length at least 2. So L is $(2, 1, 1)$-feasible. The ergodicity of $\mathcal{L}^{L}_{F}$ follows from Lemma 50 with $t_X, t_Y \leq 1$ and $t \leq 2$.

Lemma 41 implies that the reduced block dynamics is reversible for the projection of π^{μ}_{E} onto F, concluding the proof. □

Before giving a bound on the relaxation time of the reduced block dynamics, we will prove some bounds on its stationary distribution $\pi_{\mathcal{R}}$ to show that it deviates from a uniform distribution by at most a constant factor. To this end, the following lemma is a technical tool that we will reuse later. It shows that, given a boundary configuration, under the uniform distribution the probability that an edge e_i is assigned an available colour is close to uniform.

Lemma 54. *For any subtrees T_i with edge-set E_i, any $e_i \in \overline{\partial} T_i$ and any $\sigma \in \Omega^{\mu}_{E}$, let $C = \sigma(N(e_i) \cap \partial T_i)$. Assuming (A1)–(A2), for every $c \in [k] \setminus C$, we have*

$$\pi^{\sigma}_{E_i}(\{\xi \in \Omega^{\sigma}_{E_i} : \xi(e_i) = c\}) = \frac{1}{2}(1 + O(1/d)) .$$

Moreover, if $|\overline{\partial} T_i| = 1$, *then*

$$\pi^{\sigma}_{E_i}(\{\xi \in \Omega^{\sigma}_{E_i} : \xi(e_i) = c\}) = 1/2 \,.$$

Proof. First assume that $\overline{\partial} T_i = \{e_i, f_i\}$. As G is d-regular, and T_i is splitting, in particular its boundary is fringe. Since $k = d + 1$, this implies that there are exactly two colours available for e_i and two colours for f_i.

We will bound $|\Omega^{\sigma}_{E_i}|$ and $|\Omega^{\sigma}_{E_i}(c)|$, the number of colourings in $\Omega^{\sigma}_{E_i}$ that assign c to e_i. Let $P = (V_P, E_P)$ be the unique path in T_i that connects e_i and f_i and let $s = |V_P|$. As T is splitting and $|\overline{\partial} T_i| = 2$, we have $s \geq 4$. If we fix a colouring ξ_P of E_P, observe that the number of colourings of $\xi \in \Omega^{\sigma}_{E_i}$, such that $\xi|_P = \xi_P$ is independent of ξ_P. Indeed, if we remove E_P, we obtain a collection of rooted subtrees $T'_1, \ldots, T'_s$ with root $v_i \in V_P$. Given ξ_P, there are exactly $(d-1)!$ ways to colour the edges of T'_i incident to v_i, and for each internal vertex, there are exactly $d!$ ways to choose a colouring of the edges hanging from it.

Therefore, in order to bound the ratio $|\Omega^{\sigma}_{E_i}(c)|/|\Omega^{\sigma}_{E_i}|$, we only need to compute $|\Omega^{\sigma}_{E_P}|$, and $|\Omega^{\sigma}_{E_P}(c)|$, respectively the number of colourings of P compatible with σ, and the number of these colourings ξ_P for which $\xi_P(e_i) = c$. We can obtain a colouring of P by first colouring e_i and f_i, and then choosing the colour of the other edges in P in the order they appear on the path from e_i to f_i. As $s \geq 4$, there is at least one edge in $E_P \setminus \{e_i, f_i\}$. For each of these edges except for the last one, there are d choices of colours. For the last edge there are either d or $d-1$ choices. It follows that

$$\begin{aligned} 4d^{s-1}(d-1) \leq |\Omega^{\sigma}_{E_P}| \leq 4d^s \,, \\ 2d^{s-1}(d-1) \leq |\Omega^{\sigma}_{E_P}(c)| \leq 2d^s \,. \end{aligned}$$

We conclude that

$$\pi^{\sigma}_{E_i}(\{\xi \in \Omega^{\sigma}_{E_i} : \xi(e) = c\}) = \frac{|\Omega^{\sigma}_{E_i}(c)|}{|\Omega^{\sigma}_{E_i}|} = \frac{|\Omega^{\sigma}_{E_P}(c)|}{|\Omega^{\sigma}_{E_P}|} = \frac{1}{2}(1 + O(1/d)) \,.$$

The second statement follows by a simple symmetry argument. □

Lemma 55. *Assuming (A1)–(A2), for every* $\hat{\sigma} \in \Omega_{\mathcal{R}}$, *we have*

$$\pi_{\mathcal{R}}(\hat{\sigma}) = \frac{1 + O(1/d)}{|\Omega_{\mathcal{R}}|} \,.$$

Proof. Recall that $H = (U, F)$ and that $e_i \in F \cap E_i$ is unique edge of E_i in H. Let $\hat{\sigma} \in \Omega_{\mathcal{R}}$, we will compute $\pi_{\mathcal{R}}(\hat{\sigma})$ by using Lemma 53 and by bounding the number of $\sigma \in \Omega^{\mu}_{E}$ such that $\sigma|_F = \hat{\sigma}$. If $t_i = 1$, then e_i is the only boundary edge and, by symmetry, the number of extensions of $\hat{\sigma}$ in E_i does not depend on $\hat{\sigma}(e_i)$. If $t_i = 2$, then there exists $f \in E_i$ with $f \neq e_i$ such that $f \in \overline{\partial} T_i$. As in the proof of Lemma 54, in this case the number of extensions is the same, up to a $1 + O(1/d)$ multiplicative factor. Since T_i is splitting, there are at most two values of $i \in [r]$ with $t_i = 2$. It follows that, up to a $1 + O(1/d)$ multiplicative factor, each $\hat{\sigma}$ has the same number of extensions. This concludes the proof. □

We will also need the following simple bound on the gap of the dynamics of a single edge.

Lemma 56. *Let $e \in E_G$ be an edge of G. Then,*

$$\min_{\mu \in \Omega_{E_G}} \mathrm{Gap}(\mathcal{L}^{\mu}_{\{e\}}) \geq \frac{2}{d+1} .$$

Proof. Observe that $\Omega^{\mu}_{\{e\}}$ is the set of colourings of a single edge with $k_0 \geq 1$ colours, where each transition happens at rate $1/(d+1)$. If $k_0 = 1$, then the relaxation time is 1. If $k_0 \geq 2$, all positive eigenvalues of $-\mathcal{L}^{\mu}_{\{e\}}$ are equal to $k_0/(d+1)$, so $\mathrm{Gap}(\mathcal{L}^{\mu}_{\{e\}}) \geq 2/(d+1)$. □

We can finally obtain a bound on the relaxation time of $\mathcal{L}^{\mu}_{E}$.

Lemma 57. *Assuming (A1)–(A2), the following holds,*

$$\tau(\mathcal{L}^{\mu}_{E}) = O\left(d^3 + \sum_{i=1}^{r} \tau_i\right) ,$$

where $\tau_i := \max_{\sigma \in \Omega^{\mu}_{E}} \tau(\mathcal{L}^{\sigma}_{E_i})$.

Proof. Using Propositions 39 and 40, we know that the relaxation time of $\mathcal{L}^{\mu}_{E}$ satisfies

$$\tau(\mathcal{L}^{\mu}_{E}) \leq \tau(\mathcal{R}^{\mu}_{\mathcal{E}}) . \tag{5.13}$$

Thus, to get the result, we only need to bound the relaxation time of the reduced block dynamics with partition $\mathcal{E}$. We define an alternative dynamics. Let $\mathcal{R}_{\mathrm{const}}$ be the continuous-time Markov chain with state space $\Omega_{\mathcal{R}}$ and generator matrix given for any $\hat{\sigma} \neq \hat{\eta}$ by

$$\mathcal{R}_{\mathrm{const}}[\hat{\sigma} \to \hat{\eta}] = \begin{cases} g_i & \text{if } \hat{\sigma} \text{ and } \hat{\eta} \text{ differ only at } e_i, \\ 0 & \text{otherwise,} \end{cases}$$

where $g_i = 1/\tau_i$. Observe that $\mathcal{R}^{\mu}_{\mathcal{E}}$ and $\mathcal{R}_{\mathrm{const}}$ have the same state space and transitions (but different transition probabilities). For every $\hat{\sigma}, \hat{\eta} \in \Omega_{\mathcal{R}}$, Lemma 54 implies that $\pi^{\hat{\sigma},i}_{\mathrm{proj}}(\hat{\eta}) = \pi^{\sigma}_{E_i}(\{\xi \in \Omega^{\sigma}_{E_i} : \xi(e_i) = \hat{\eta}(e_i)\}) = \Theta(1)$, where σ is an arbitrary colouring in $\Omega^{\hat{\sigma}}_{*}$. So $\mathcal{R}^{\mu}_{\mathcal{E}}[\hat{\sigma} \to \hat{\eta}] = \Theta(\mathcal{R}_{\mathrm{const}}[\hat{\sigma} \to \hat{\eta}])$. Moreover, the stationary distribution π_{const} of $\mathcal{R}_{\mathrm{const}}$ is uniform on $\Omega_{\mathcal{R}}$, and by Lemma 55 we have $\pi_{\mathcal{R}}(\hat{\sigma}) = \Theta(\pi_{\mathrm{const}}(\hat{\sigma}))$ for every $\hat{\sigma} \in \Omega_{\mathcal{R}}$. Thus, it follows from Corollary 38 that

$$\tau(\mathcal{R}^{\mu}_{\mathcal{E}}) = \Theta(\tau(\mathcal{R}_{\mathrm{const}})) \tag{5.14}$$

We will bound the relaxation time of $\mathcal{R}_{\mathrm{const}}$ using the results in Section 5.5, and conclude using (5.13) and (5.14).

Suppose first that T is vertex-rooted. This implies that H induces a star with edges $e_1, \ldots, e_d$. Consider the clique with vertex-set $U = \{u_1, \ldots, u_d\}$, where u_i is identified with the edge e_i, and the list assignment L of U defined by $L(u_i) = [k] \setminus \mu(N(e_i) \cap \partial T)$. Up to relabelling of the edges, as $k = d+1$, the list assignment L is 1-feasible. Consider the dynamics $\mathcal{L}^{L}_{U}$ with probabilities $p_i := g_i$. We can identify $\Omega_{\mathcal{R}}$ and $\mathcal{R}_{\mathrm{const}}$ with Ω^{L}_{U} and $\mathcal{L}^{L}_{U}$. By applying Lemma 43 with $t \leq 1$ we have,

$$\tau(\mathcal{R}_{\mathrm{const}}) = \tau(\mathcal{L}^{L}_{U}) = O\left(\sum_{i=1}^{t} \frac{d}{g_i} + \sum_{i=t+1}^{d} \frac{1}{g_i}\right) = O\left(d^2 + \sum_{i=1}^{d} \frac{1}{g_i}\right) ,$$

where we used Lemma 56 in the last equality.

If T is edge rooted, then H is a bi-star. We can do the same proof replacing Lemma 43 by Lemma 50 obtaining a similar bound with an additive factor of order d^3. □

5.6.2 Proof of Theorem 34

Let $k \geq \Delta + 1$ and $G = (V_G, E_G)$ be a tree on n vertices with maximum degree at most Δ. From (1.1) and (5.1) to prove Theorem 34 it suffices to bound $\tau(\mathcal{L}_{E_G})$, the relaxation time of the continuous-time Glauber dynamics.

Let $d := k - 1 \geq \Delta$. Construct the *d-regularisation* $G^d = (V_G^d, E_G^d)$ of G by adding $d - |N(u)|$ leaves adjacent to each internal vertex $u \in V_G$. Note that G^d is d-regular as a tree and it has at most dn vertices. By Proposition 42, we have $\tau(\mathcal{L}_{E_G}) \leq \tau(\mathcal{L}_{E_G^d})$. So, to prove Theorem 34 it suffices to bound the relaxation time of d-regular trees with at most dn vertices, and in the following we assume that G is d-regular.

We will prove the following result by induction on the size of the subtree T:

Claim 58. There is a constant C such that, for every splitting subtree $T = (V, E)$ of G with m edges, and every edge-colouring $\mu \in \Omega_{E_G}$, the Glauber dynamics $\mathcal{L}_E^\mu$ with parameters $p_i = 1/k$ is ergodic and

$$\tau(\mathcal{L}_E^\mu) \leq d^3 m^C \ .$$

From this claim, the theorem is obtained immediately by taking $T = G$. If T is composed of a single edge, the Claim 58 follows from Lemma 56. Let m be the number of edges in T and assume $m > 1$. Let $v \in V$ be the vertex such that each subtree T_i hanging from T rooted at v has at most $\lceil \frac{m}{2} \rceil$ edges; this vertex always exists and it is internal. Let e_i be the edge from T_i incident to v. We are going to split T into several subtrees by applying Lemma 57, possibly several times. Note that in order to apply this lemma, we must ensure that each of the subtrees is splitting. This gives constraints on how we can split T. Precisely, while splitting the tree into subtrees, none of the subtrees can have an internal boundary of size at least 3, and for the subtrees with an internal boundary of size 2, the two edges in the boundary must be non-incident.

The splitting procedure is done according to different cases:

1. If all the T_i are splitting, then simply root T at v, and apply Lemma 57. In the following we will assume that not all the T_i are splitting.

2. If $|\overline{\partial} T| = 1$, then there is exactly one subtree, say T_1, which is not splitting. All other subtrees T_i for $i \neq 1$ are splitting and have fringe boundary of size 1. Root T at e_1. Now, all subtrees pending from e_1 are splitting and have at most $\lceil \frac{m}{2} \rceil$ edges, and we can apply Lemma 57.

3. If $|\overline{\partial} T| = 2$, with e and f the two edges on the internal boundary, and v is on the path between e and f. Without loss of generality, assume that T_1 and T_2 contain e and f respectively.

   - If exactly one of T_1 or T_2 is not splitting, w.l.o.g. we can assume that it is T_1. By rooting T at e_1, all subtrees pending from e_1 are splitting and contain at most $\lceil \frac{m}{2} \rceil$ edges, and we can apply Lemma 57.
   - If both T_1 and T_2 are not splitting, write $e_1 = (v_1, v)$. Since T_1 is not splitting, v_1 must be incident to e. Moreover, since T_2 is not splitting, e_1 and f are not incident, and all the subtrees pending at v_1 are splitting. We apply Lemma 57 a first time by rooting T at v_1. Let T' be the subtree hanging from v_1 that contains v and root T' at e_2. Then, all subtrees pending from e_2 are splitting, and we can apply Lemma 57 a second time. The resulting subtrees all have at most $\lceil \frac{m}{2} \rceil$ edges.

4. Finally, if v is not on the path between e and f, let v' be the vertex on this path which is closest to v. We can first split at v' by applying the procedure from the case 3. After this splitting, all the resulting subtrees have at most $\lceil \frac{m}{2} \rceil$ edges except maybe the subtree T' containing v. However T' has a fringe boundary of size 1, and by splitting one more time according to either case 1 or case 2, all resulting subtrees are splitting, and have at most $\lceil \frac{m}{2} \rceil$ edges. In the worst case, we needed to use Lemma 57 three times in this case.

Let $T'_1, \dots, T'_s$ be the subtrees into which T is split by applying the procedure above, and let E'_i be the set of edges of T'_i. Since T'_i is splitting, by the induction hypothesis, $\mathcal{L}^{\sigma}_{E'_i}$ is ergodic for every $\sigma \in \Omega^{\mu}_E$, so $\mathcal{L}^{\mu}_E$ is also ergodic. Recall that $\tau_i := \max_{\sigma \in \Omega^{\mu}_E} \tau(\mathcal{L}^{\sigma}_{E'_i})$. Lemma 57 shows that there exists K such that if T is split at a vertex/edge into r subtrees $T'_1, \dots, T'_r$, then $\tau(\mathcal{L}^{\mu}_E) \leq K(d^3 + \sum_{i=1}^{r} \tau_i)$. As we use Lemma 57 at most three times in each step of the splitting procedure described above, we have

$$\tau(\mathcal{L}^{\mu}_E) \leq K^3 \Big(3d^3 + \sum_{i=1}^{s} \tau_i \Big) ,$$

Using the induction hypothesis on T'_i, if we denote by $m_i \leq \lceil \frac{m}{2} \rceil$ the number of edges in T'_i, we have:

$$\begin{aligned} \tau(\mathcal{L}^{\mu}_E) &\leq K^3 \Big(3d^3 + \sum_{i=1}^{s} d^3 m_i^C \Big) \leq K^3 \Big(3d^3 + d^3 \Big\lceil \frac{m}{2} \Big\rceil^{C-1} \sum_{i=1}^{s} m_i \Big) \\ &\leq K^3 d^3 \left(3 + \frac{m^C}{2^{C-2}} \right) \leq d^3 m^C \cdot \frac{K^3}{2^{C-3}} \cdot \end{aligned}$$

So letting $C \geq 3(1 + \log K)$ gives the desired inequality, and we conclude the proof of Claim 58.

Chapter 6

Online Colouring with Kempe Chains

In this chapter, we are interested in using local reconfiguration to colour graphs in an online setting. These results were obtained with Sylvain Gravier and appear in [GH18].

Usual reconfiguration problems study the question of transforming one solution into an other. However there are settings which are not directly captured by these questions. For example, the target solution might not be known in advance, but instead we want to reach any solution which satisfies some properties. This case appears naturally if we try to use the transformations from the reconfiguration setting to adapt a partial solution to the rest of the input. In the case of graph colouring for example, we might want to adapt a partial solution to colour one additional vertex. Hence, the goal is to modify the current colouring to remove one colour from the neighbourhood of the new vertex we wish to colour. An other notion not directly captured in the reconfiguration setting is locality: in some cases, we might want to find a transformation that changes only a small part of the solution. These two restrictions are interesting in particular in the context of online algorithms.

6.1 Introduction

online algorithm

Online algorithms are a class of algorithms reading their input sequentially. In the case of graph problems, this usually means that the vertices of the graph arrive one by one. As the formal definition of greedy, online, and sequential algorithms is not completely fixed, we start by fixing the convention we use here. In an online algorithm, for each new vertex, the algorithm must adapt a partial solution of the problem on the graph without the new vertex into a solution for the whole graph. For the graph colouring problem, this means that, if G is the graph and v the newly added vertex, the algorithm must transform a colouring of $G-v$ into a colouring of G. The study of online algorithms is motivated by situations where the whole input is not available to the algorithm, for example if the input is too large to fit into the main memory of the computer.

In the online setting, recolouring the whole graph at each step is too expensive. Instead, we wish to keep the changes local to the new vertex which was just added to the graph. A special case of such algorithms is if we allow no change at all on the current solution. In the case of colouring, this means that changing the colour of previously coloured vertices is not allowed. Such algorithms are called greedy colouring algorithms. Each time a new vertex is read from the input, a new colour different from its neighbours must be immediately assigned to it, and vertices coloured at an earlier

step cannot be recoloured. The most common greedy algorithm for graph colouring is First-Fit, which consists in taking the smallest colour not already present in the neighbourhood of the new vertex. These online algorithms are usually studied in a setting where the order on the vertices of the graph is arbitrary. The performance of online colouring algorithms is measured in terms of the number of colours used for the worst case ordering. First-Fit

A similar type of algorithms are sequential algorithms. These algorithms first decide on an ordering of the vertices, and then, apply an online algorithm to colour the graph according to this ordering. In this case the ordering is chosen by the algorithm.

The greedy graph colouring problem is a widely studied subject. For general graphs, a randomized greedy algorithm finding an $O(\frac{n}{\log n})$ approximation was devised in [Hal97] while there is a lower bound of $\Omega(\frac{n}{\log^2 n})$ on the approximation ratio of any greedy algorithm [HS92]. This lower bound holds even if the algorithm knows in advance the graph being coloured, but not the sequence of vertices in which it is given [Hal00]. An important effort has been directed at studying the performance on usual graph classes. A greedy algorithm with an approximation ratio of $O(\log n)$ was shown for trees [GL88], bipartite graphs [LST89], planar and chordal graphs [Ira94]. On the other hand, this approximation ratio was shown to be optimal for these graph classes [Bea76, AS17, LST89]. The lower bounds even hold for randomized algorithms, algorithms using a small reordering buffer, or algorithm allowed to look at a few future inputs before making a choice [AS17]. Online algorithms with constant approximation ratio exists for other classes of graphs such as interval graphs [KT81, LV98, Smi10, NB08], co-interval graphs [GL88, KQ95, ZZ07, ZZC09] and disk intersection graphs [CFKP07, EF02, AS17].

Online algorithms are closely related to an other model of computation: the dynamic graph model. In this setting, the input is an online sequence of updates to the graph (usually addition/deletion of edges), and for each of these updates, the algorithm must maintain a feasible solution to the problem. This is very similar to the online setting where only addition (of edges or vertices) are permitted, while deletions can also be performed in the dynamic graph model. Several problems have been considered in this setting, see [DEGI09, DFI04] for a survey on the existing results on the problem. The case of colouring in the dynamic graph model has been considered only recently. In [BCK+17], two algorithms are presented for dynamic graph colouring with different trade-off between two parameters: the number of colours used by the algorithm, and the number of vertices which are updated at each step. These results were subsequently improved in [SW18] for some range of the parameters. The problem has also been considered in [BCG19] for various models of computation.

Since the approximation ratio of greedy algorithms is quite large even for some simple classes of graphs such as trees, it is natural to look at more general algorithms. In this chapter, we consider online algorithms which are allowed to change the colour of previous vertices only by making local Kempe exchanges. Recall from Chapter 1 that a Kempe exchange consists in swapping the colours of the vertices of a maximal connected 2-coloured sub-graph of G (called Kempe chain). Applying this transformation creates a new proper colouring of the graph. We call these algorithms online algorithms with Kempe exchanges. For each new vertex v, the algorithm can perform Kempe exchanges to remove one colour from the neighbours of v, and then use this colour for v.

The choice of Kempe exchanges as an operation to recolour the graph is quite natural. It was first considered in Kempe's failed attempt at proving the four colour theorem, and was used later to prove Vizing's theorem [Viz64]. More recently, sequential algorithm using Kempe exchange were considered in [MP99] to colour a special subclass of perfect graphs, and in [HG96, HGM98, HM97,

Tuc87] using more complex recolouring operations. In this context, Kempe exchanges are used to locally modify an existing colouring in order to colour a new vertex.

We have also seen in previous chapters that Kempe exchanges can be used in the context of reconfiguration, and sampling random colourings. In the reconfiguration setting, the problem considered is global: the transformations can be usually applied anywhere in the graph. On the contrary, we consider here local transformations. This means that, as in sequential algorithms, the Kempe exchanges are restricted to the neighbourhood of the newly added vertex.

Overview. The chapter is organized as follows. In the next section, we give a formal definitions of online algorithm with Kempe exchanges. Then, we describe such algorithms for several classes of graph: bipartite graphs in Section 6.1.2, chordal graphs in Section 6.2, outer-planar and planar graphs in Section 6.3. An extension of the algorithm on planar graphs to the case of graphs with bounded genus is presented in Section 6.4. Apart from the cases of planar graph and graphs with bounded genus, these algorithm are optimal and robust: they either find an optimal colouring or exhibit a forbidden substructure for this class of graphs. In the case of planar graph, the algorithm only provides a $O(\log(\Delta))$-colouring. The question of whether we can achieve a constant number of colours is still open. Finally, in Section 6.5 we give an example of 3-colourable graphs for which any online algorithm using Kempe exchanges (with some additional restrictions) performs badly.

6.1.1 Definitions and notations

We recall some of the notations defined in Chapter 1. A Kempe exchange consists in swapping the two colours in a maximal 2-coloured connected subgraph of G. We will write $\langle v, i\rangle$ the Kempe exchange that, given a colouring c produces a colouring $c' = \langle v, i\rangle(c)$ obtained by swapping the $(i, c(v))$-Kempe chain containing v. A vertex u is changed by the Kempe exchange $\langle v, i\rangle$ if and only if there is a path between u and v coloured using only the colours i and $c(v)$. Such a path will be called a bicoloured $(i, c(v))$-path.

The algorithms that we consider are a special case of online algorithms where the algorithm is allowed to recolour previously coloured vertices only by performing Kempe exchanges. Allowing the algorithm to perform any Kempe exchange would be too powerful as it would allow, in many cases, to recolour the whole graph. To prevent this, the algorithm is only allowed to perform Kempe exchanges which are local to the newly added vertex in the following sense:

local Kempe exchange

Definition 3. Let G be a graph, c a colouring of G, and v, x two vertices of G. The Kempe exchange $\langle i, x\rangle$ is *local to* v if $x \in N(v)$.

We will always consider Kempe exchanges which are local to v, the (not yet coloured) vertex that was just added to the graph. Consequently, we will omit to specify v, and just write that the Kempe exchanges are local as a shorthand for local to v. Note that vertices outside the neighbourhood of v might still be recoloured by a local Kempe exchange. Indeed, we only require that the starting point of a local Kempe exchange is in the neighbourhood of v, but this transformation can still recolour a large part of the graph. We can now give a formal definition of online algorithms with Kempe exchanges.

online colouring algorithm with Kempe exchanges

Definition 4. An *online colouring algorithm with Kempe exchanges* is an online algorithms such that, for each new vertex v:

- it applies a sequence of local Kempe exchanges,

- and then selects a colour for v not used by any vertex in its neighbourhood.

This definition is quite general, however the algorithms considered here are more simple: they only perform Kempe exchanges when necessary. If a colour is already available to the new vertex without any recolouring, then the algorithm immediately selects that colour. Additionally, the algorithms we describe in the following sections always choose the smallest colour available, and make no assumption on eventual properties of the existing colouring. In other words, with these restrictions an algorithm with Kempe exchanges is given by a procedure taking as input a graph G with a vertex v and any colouring c of $G - v$. The algorithm must find a sequence of Kempe exchanges such that applying these transformations removes one colour from the neighbourhood of v.

In terms of graph colouring reconfiguration, the question this type of algorithms try to answer can be formulated in the following way. Given a graph G, a vertex v, and a k-colouring c of $G - v$, is there a transformation of c using local Kempe exchanges into a colouring c' for which $N(v)$ is $(k-1)$-coloured. Thus, the problem is similar to a colouring reconfiguration problem with two main variations:

- the Kempe exchanges must be local,
- the target colouring is not given, but can be any colouring with the desired properties.

An other important parameter for this kind of algorithm is the length of the reconfiguration path: the number of Kempe exchanges which are applied for each new vertex in the worst case. Since we are interested in only making local changes to the colouring, we want this number to be polynomial in Δ.

The question we are interested in is to determine for which classes of graphs does such an algorithm exist and how many colours it requires. In the following we will present several algorithms for different classes of graphs.

6.1.2 Bipartite graphs

The first simple class of graph that we consider are bipartite graphs. For these graphs, we prove the following result:

Theorem 59. *There is an online colouring algorithm with Kempe exchange finding a 2-colouring of bipartite graphs using at most Δ operations at each step.*

Proof. Let G be a bipartite graph, v a vertex of G, and c the colouring of $G - v$ obtained from the previous steps of the algorithm. We now describe how to recolour G using local Kempe exchanges in order to make $N(v)$ monochromatic. If one of the two colours is not present in the neighbourhood of v, then this colour can be used to colour v. Otherwise, the recolouring procedure is the following: while there is a vertex $u \in N(v)$ coloured 2, apply the Kempe exchange $\langle u, 1\rangle$. None of the vertices in $N(v)$ coloured 1 can change colour back to 2. Indeed, assume by contradiction that at some step a vertex $w \in N(v)$ changes colour from 1 to 2 during the Kempe exchange $\langle u, 1\rangle$. This implies that there is a path p of odd length from u to w in $G - v$. Consequently, $p \cup \{v\}$ is an odd cycle, a contradiction of the assumption that G is bipartite.

When the procedure ends, after at most Δ Kempe exchanges, all the vertices in $N(v)$ are coloured 1, and v can be coloured 2. □

In particular, this algorithm colours optimally any tree, while the best greedy online algorithm needs $\Omega(\log n)$ colours in the worst case.

6.2 Chordal graphs

In this section, we will exhibit an online algorithm with Kempe exchanges that can colour optimally any chordal graph. A chordal graph is a graph with no induced cycles of length larger than 3. These graphs satisfy the following property:

Theorem 60 (Dirac, 1961)**.** *A graph G is chordal if and only if there is an ordering $u_1, \ldots, u_n$ of its vertices such that for all $i \geq 1$, $N(u_i) \cap \{u_1, \ldots, u_{i-1}\}$ is a clique. Such an ordering is called an simplicial ordering.*

The ordering in the characterisation above can be computed in polynomial time by iteratively removing simplicial vertices from the graph (i.e., vertices whose neighbourhood forms a clique). Chordal graphs are an important subclass of perfect graphs for which the chromatic number is equal to the largest clique in the graph. The idea of the online algorithm is the following. Let v be the new vertex that was added to the graph. The algorithm will recolour $N(v)$ using one less colour. Since the subgraph induced by $N(v)$ is also chordal, we can find an simplicial ordering of the vertices in $N(v)$ and recolour these vertices one by one according to this ordering, using local Kempe exchanges. The property that the whole graph is chordal will ensure that the colours of previously recoloured vertices do not change during the successive operations. This property will be proved using the following Lemma.

Lemma 61. *Let G be a graph, v a vertex of G, and c a colouring of $G - v$. Assume that there are two vertices $u_1, u_2 \in N(v)$, such that there is a bi-coloured (i, j)-path between u_1 and u_2 in $G - v$. If p is the shortest such path, then $p \subseteq N(v)$.*

Proof. Let u_1 and u_2 be two neighbours of v such that there is a bi-coloured (i, j)-path between u_1 and u_2 in $G - v$. Take p to be the shortest bi-coloured (i, j)-path from u_1 to u_2 for some colours i and j. We will show that $p \subseteq N(v)$. Assume by contradiction that this is not the case, and there is at least one vertex on the path p which is not a neighbour of v. We can find a subpath p' of p such that p' has length at least 2, and the only vertices of p' in $N(v)$ are its two endpoints.

Since p is the shortest bi-coloured (i, j)-path from u_1 to u_2, the graph induced by p' is a path, and consequently, $G[v \cup p']$ is a cycle of length at least 4. This is a contradiction of the assumption that G is chordal. □

Theorem 62. *There is an online colouring algorithm with Kempe exchanges that colours optimally every chordal graph. The algorithm performs at most Δ Kempe exchanges for each new vertex.*

Proof. Given a graph H, we denote by $\omega(H)$ the size of the largest clique in H. Let G be a graph, v a vertex of G, and c a colouring of $G - v$ using $\omega(G - v)$ colours. If $\omega(G) = \omega(G - v) + 1$, then we can directly colour v using the new additional colour. Otherwise, let $\omega = \omega(G)$. We will describe an algorithm finding a sequence of Kempe exchanges such that the resulting colouring uses at most $\omega - 1$ colours on $N(v)$. The transformation is made using at most Δ Kempe exchanges.

Since G is chordal, the induced subgraph $G[N(v)]$ is also chordal. Let $k = |N(v)|$ be the number of neighbours of v, and let $v_1, \ldots, v_k$ be an simplicial ordering of the vertices of $N(v)$. We write G_i

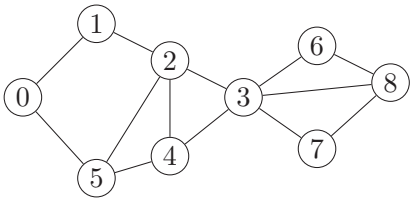


Figure 6.1: Example of outer-planar graph. The vertex 3 appears twice on the outer face of the graph.

the subgraph induced by the vertices $v_1, \ldots, v_i$. The recolouring procedure is, for each i from 1 to k, if v_i is coloured ω, we apply the Kempe exchange $\langle v_i, x_i \rangle$ where x_i is the smallest colour not present in $N_{G_i}(v_i)$. There is always such a colour x_i. Indeed, by definition of the order of the vertices, $N_{G_i}(v_i)$ is a clique, and $N_{G_i}(v_i) \cup \{v_i, v\}$ is also a clique of G. Consequently, $|N_{G_i}(v_i)| \leq \omega - 2$.

During step i, none of the vertices v_j with $j < i$ is recoloured. Indeed, suppose by contradiction that this is not the case, and let i be the first step at which a vertex v_j, with $j < i$ is recoloured. Before the exchange, we have $c(v_j) \neq \omega$. After the exchange, the colour of v_j changes to ω. Consequently, there is bi-coloured (ω, x_i)-path from v_i to v_j in $G - v$. By Lemma 61, this implies that there is a bi-coloured (ω, x_i)-path $p \subseteq N(v)$ from v_i to v_j.

Let v_k be the vertex of the path p with the largest index k. By choice of the colour x_i, we know that $v_k \neq v_i$, since the (only) neighbour of v_i in p has an index larger than i. Consequently, since $j < i$, the vertex v_k has two neighbours v_a and v_b in p. Since p is 2-coloured, both v_a and v_b have the same colour. Additionally, by choice of k, both a and b are smaller than k. Since the ordering of the vertices is an simplicial ordering, the neighbours of v_k in G_{k+1} form a clique. In particular, there is an edge between v_a and v_b, and this edge is monochromatic, a contradiction.

After the recolouring is done, none of the vertices of $N(v)$ is coloured ω, and the colour ω can be assigned to v. □

6.3 Planar graphs

The goal of this section is to show two online algorithm with Kempe exchanges. The first one allows to colour outer-planar graphs using at most 3 colours. The second one can colour any planar graph, but only computes a $O(\log \Delta)$-colouring. We start by the algorithm on outer-planar graphs.

6.3.1 Outer-planar graphs

An outerplanar graph is a graph which admits a planar drawing such that there is a face adjacent to all the vertices of the graph. Such a drawing of an outerplanar graph can be computed in linear time [Bre77]. The face containing all the vertices of the graph is the outer-face. Given an outerplanar graph G, and an outer-planar drawing of G, we can consider the sequence of vertices in the order they appear on the outer face. Note that the same vertex can appear several times in this sequence as in the example in Figure 6.1. We will prove the following Theorem:

Theorem 63. *There is an online colouring algorithm with Kempe exchange that colours optimally any outer-planar graph. The algorithm performs at most Δ Kempe exchanges at each round.*

Proof. Let G be an outer-planar graph, v a vertex of G, and c an optimal colouring of $G - v$. If G is bipartite, then we can use the procedure from Theorem 59 to remove one colour from $N(v)$, and obtain a 2-colouring of G. Otherwise, we can assume that the colours $1, 2$ and 3 are present in $N(v)$.

We will describe a procedure recolouring $N(v)$ with only two colours using at most Δ local Kempe exchanges.

Let $k = |N(v)|$, and let $v_0 = v, v_1, \ldots, v_k$ be an ordering of the vertices of $N(v) \cup v$, in the order they appear on the outer face in an outerplanar drawing of G. Note that there is no ambiguity in the choice of this ordering. Indeed, suppose that there is a vertex, say v_i, that appears several times on the outer-face. Then none of the vertices that appear between the first and last occurrence of v_i on the outer face can be neighbours of v, since it would contradict the fact that G is outer-planar. We will recolour the vertices $v_1, \ldots, v_k$ successively using the colours 1 and 2.

The algorithm recolours the vertices $v_1, \ldots, v_k$ in this order by doing the following. For i from 1 to k, if $c(v_i) = 3$, let $x \in \{1, 2\} \setminus \{c(v_{i-1})\}$ (if $i = 1$, x can be either 1 or 2). We apply the Kempe exchange $\langle v_i, x \rangle$.

During step i, none of the vertices $v_1, \ldots, v_{i-1}$ are recoloured. Indeed, since G is outer-planar, any path in $G - v$ from v_i to some vertex v_j with $j < i$ necessarily goes through v_{i-1}. Additionally, the colour of v_{i-1} is different from x and 3 by choice of x. Consequently, v_{i-1} is not contained in any bi-coloured $(x, 3)$-path, and there is no bi-coloured $(x, 3)$-path between v_i and v_j. At the end of this procedure, $N(v)$ is coloured with colours 1 and 2, and v can be coloured 3. □

Note that the algorithm we described above needs to compute the outer-planar embedding of the graph to work. It could be interesting to see if it is possible to devise an algorithm without it.

6.3.2 General planar graphs

The case of general planar graph is more complicated, and we will only prove the following weaker theorem.

Theorem 64. *There is an online algorithm with Kempe exchanges colouring any planar graph with $O(\log(\Delta))$ colours. The algorithm performs at most $\frac{\Delta^2}{2}$ Kempe exchanges for each new vertex.*

The question of whether there exists an algorithm using only a constant number of colours is still open. The algorithm is quite simple but the analysis is more complex than previous cases. The idea is to repeatedly apply a greedy procedure trying to remove one colour from the the neighbourhood of the new vertex v. We will show that when this greedy procedure cannot be applied any more, $N(v)$ is coloured using at most $O(\log \Delta)$ colours.

size of a colour

We start by a few definitions. Let G be a graph, v a vertex of G, and c a colouring of $G - v$. Given a colour i, the *size* of i in the colouring c is the number of neighbours of v coloured i: $\text{size}_c(i) = |N(v) \cap c_i|$. Let $\langle u, j \rangle$ be a local Kempe exchange, and let $c' = \langle u, j \rangle(c)$ and $i = c(u)$. Suppose that $\text{size}_c(i) \leq \text{size}_c(j)$. The Kempe exchange $\langle u, j \rangle$ is said to *increase inequalities* in c if we have $\text{size}_{c'}(i) < \text{size}_c(i)$ and $\text{size}_{c'}(j) > \text{size}(j)$. Intuitively, a Kempe exchange increases inequalities if it decreases the size of colours with smaller sizes, and increases the size of colour with larger sizes. The greedy recolouring procedure that we apply is the following.

Procedure (GreedyRecolor). While there is a local Kempe exchange $\langle u, i \rangle$ that increases inequalities, apply $\langle u, i \rangle$ to the current colouring.

To prove the theorem, we only need to prove two things: (i) the procedure **GreedyRecolor** ends in a polynomial in Δ number of rounds, and (ii) if c is a colouring such that there is no local Kempe exchange increasing inequalities, then $N(v)$ is coloured with at most $O(\log \Delta)$ colours. The first point is proved in the following Lemma 65. The second point will be proved in Lemma 67 in the next subsection.

Lemma 65. *The procedure **GreedyRecolor** ends after at most $\frac{\Delta^2}{2}$ rounds.*

Proof. To show this result, we will exhibit a potential function Ψ such that:

- Ψ increases at every iteration of **GreedyRecolor**,
- Ψ is upper bounded.

Let $k \leq \Delta + 1$ be the number of colours, and c be a k-colouring, we define the potential $\Psi(c)$ by:

$$\Psi(c) = \sum_{i \in [k]} (\text{size}_c(i))^2.$$

We will show that Ψ increases by at least 2 at each iteration of **GreedyRecolor**. Let $\langle u, j \rangle$ be a local Kempe exchange increasing inequalities in c, and let $i = c(u)$ and $c' = \langle u, j \rangle(c)$. By assumption on $\langle u, j \rangle$, we know that $\text{size}_c(i) \leq \text{size}_c(j)$. Moreover, there is an integer x such that:

- $\text{size}_{c'}(i) = \text{size}_c(i) - x$
- $\text{size}_{c'}(j) = \text{size}_c(j) + x$

Since $\langle u, j \rangle$ increases inequalities, we have $x \geq 1$. Additionally, the following holds:

$$\begin{aligned}
\Psi(c') - \Psi(c) &= (\text{size}_{c'}(i))^2 + (\text{size}_{c'}(j))^2 - (\text{size}_c(i))^2 - (\text{size}_c(j))^2 \\
&= (\text{size}_c(i) - x)^2 + (\text{size}_c(j) + x)^2 - (\text{size}_c(i))^2 - (\text{size}_c(j))^2 \\
&= 2x^2 + 2x(\text{size}_c(j) - \text{size}_c(i)) \geq 2
\end{aligned}$$

Moreover $\Psi(c)$ is upper bounded by Δ^2 for any colouring c. Indeed, since we know that for any colouring c, $\sum_i \text{size}_c(i) = \Delta$, $\Psi(c)$ is maximum when $\text{size}_c(i)$ is zero for all but one colour. The potential Ψ is positive, upper bounded by Δ^2, and increases by 2 at each iteration of **GreedyRecolor**. Consequently, the procedure **GreedyRecolor** must end after at most $\frac{\Delta^2}{2}$ rounds. □

6.3.3 Grundy colouring of circular graphs

To prove the correctness of the algorithm described in previous subsection, we only need to show that if a colouring has no local Kempe exchange that increases inequalities then $N(v)$ is coloured using at most $O(\log \Delta)$ colours. To prove this property, we will use a result on a particular drawing of a graph which could be of independent interest. We will first describe this construction, and prove that any colouring satisfying some properties related to this drawing uses a small number of colours. This will then allow us to complete the proof of Theorem 64.

Let G be a graph, with G not necessarily planar. A drawing of G is a function f that associates to each vertex v of the graph a point $f(v)$ in the plane, and to each edge (u, v) of the graph a path from $f(u)$ to $f(v)$. We call a *circular drawing* of G a drawing of G such that all the vertices are represented by points on a circle, and the paths representing the edges are straight line segments. We are interested in colourings which satisfy the following property.

Definition 5. Let $G = (V, E)$ be a graph, with a drawing of G. A proper colouring c of G is *intersection compatible* if for any two crossing edges (u_1, v_1) and (u_2, v_2), the two sets $\{c(u_1), c(v_1)\}$ and $\{c(u_2), c(v_2)\}$ intersect.

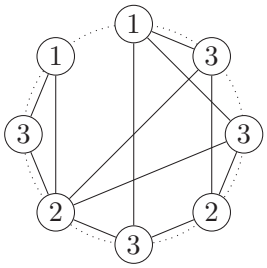


Figure 6.2: Example of circular drawing of a graph with an intersection compatible colouring.

An example of a circular drawing of a graph with an intersection compatible colouring is given in Figure 6.2. Note that not all graphs G have a drawing which admits an intersection compatible colouring. Determining which graphs have one could be an interesting question on its own. For example, there is no intersection compatible colouring of K_5, the clique on 5 vertices, for any of its drawing on a plane. Indeed, since K_5 is not planar, there is at least two edges crossing in any drawing of K_5 on the plane. The endpoints of these two edges have all different colours. On the contrary, for any 3-colourable graph, there is a drawing on the plane which has an intersection compatible colouring. In fact any 3-colouring of a 3-colourable graph is intersection compatible for any drawing of the graph, since in this case any two edges share at least one colour.

The idea behind this notion, is that, given an initial planar graph G with a colouring c, we can build an auxiliary graph G' whose edges represents Kempe chains. More precisely, if two vertices with different colours are adjacent in G', then there exists a Kempe chain containing both vertices. By a considering a drawing of G' such that if two edges of G' intersect, then the corresponding Kempe chain also intersect, then c is a colouring of G' which must be intersection compatible since the initial graph G is planar.

The upper bound on the number of colours in $N(v)$ after applying the procedure **GreedyRecolor** will be obtained in two step. First, we show that when the procedure ends, the colouring of $N(v)$ satisfies some property. Then, we show that a colouring of G' which satisfy this property and is intersection compatible cannot have too many colours. We start by proving this second point.

circular graph

Given a graph G with a colouring c, the coloured graph (G, c) will be called *circular* if there is a circular drawing of G that makes the colouring c intersection compatible. A k-colouring c of a graph is a *Grundy k-colouring* if for every vertex u, and for every colour $j < c(u)$, there is a neighbour $w \in N(u)$ such that $c(w) = j$, and at least one vertex is coloured k. In other words, every vertex is assigned the smallest colour that is not present in its neighbourhood. The Grundy chromatic number of a graph G is the largest number of colours k such that G has a Grundy k-colouring. To put it differently, a colouring is a Grundy colouring if it can be obtained from the algorithm First-Fit, with a certain ordering of the vertices. The Grundy chromatic number is the largest number of colours that the algorithm First-Fit might need for the worst case ordering. We will prove the following result.

Grundy k-colouring

Lemma 66. *There is a constant K_0, such that for any circular (intersection compatible) coloured graph (G, c) with n vertices, if c is a Grundy k-colouring of G, then $k \leq K_0 \log n$.*

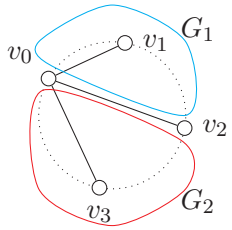


Figure 6.3: Any edge crossing the edge (v_0, v_2) must have either $c(v_0)$ or $c(v_2)$ as the colour of one of its endpoints. Since G' contains no vertex with colour $c(v_0)$ or $c(v_2)$, G' is split into two connected components, one on each side of the edge (v_0, v_2).

Note that the lemma does not prove that the Grundy chromatic number of a circular graph is at most $O(\log n)$. Indeed, a circular graph could have a Grundy colouring using more colours. However, the lemma states that in this case, this colouring is not intersection compatible.

Proof. We denote by $T(k)$ the smallest number of vertices of a circular graph G having an intersection compatible Grundy k-colouring c. To prove the lemma, it is enough to show that there is a constant $\alpha > 1$ such that $T(k) \geq \alpha^{k-1}$ for any $k \geq 1$. We take $\alpha = 2^{\frac{1}{4}}$, and show this inequality by induction on k. If $k \leq 4$, we know that $T(k) \geq k \geq \alpha^{k-1}$.

Let us assume that $k > 4$, and let (G, c) be a coloured circular graph with $T(k)$ vertices such that c is a Grundy k-colouring. Let u_0 be a vertex coloured k. Since c is a Grundy colouring, there are three vertices u_1, u_2, and u_3 adjacent to u_0 such that for each $i \in \{1, 2, 3\}$, u_i is coloured $k - i$. Let $v_0 = u_0, v_1, v_2, v_3$ be the vertices $\{u_0, u_1, u_2, u_3\}$ in the order they appear on the circle in the corresponding circular drawing of G. We denote by G' the subgraph induced by the vertices with a colour at most $k-4$. We will show that G' is composed of at least 2 connected components, each of which contains all the colours from 1 to $k-4$. By applying the induction hypothesis on each of these components, this gives as needed:

$$T(k) = |G| \geq |G'| \geq 2 \cdot T(k-4) \geq 2 \cdot \alpha^{k-5} = \alpha^{k-1}.$$

Thus, we only need to show that G' contains at least two connected components, each of which contains all the colours from 1 to $k-4$. Since c is a Grundy colouring and $c(v_1) > k-4$, there is a vertex w_1 adjacent to v_1 with $c(w_1) = k-4$. Similarly, there is a vertex w_2 adjacent to v_3 with colour $c(w_2) = k-4$. See Figure 6.3 for an illustration. Let G_1 and G_2 be the connected components of G' containing w_1 and w_2 respectively. Then $G_1 \neq G_2$. Indeed, if w_1 and w_2 were in the same connected component of G', then there would be a path from w_1 to w_2 using only colours less than or equal to $k-4$. By adding the two edges (v_1, w_1) and (v_3, w_2), we would obtain a path from u_1 to u_3 which does not use the colours $k = c(v_0)$ and $c(v_2)$. However, this path must necessarily cross the edge (v_0, v_2), a contradiction of the assumption that the colouring c is intersection compatible.

Consequently we have $T(k) \geq \alpha^{k-1}$ which can be rewritten as $k \leq 1 + \frac{\log(T(k))}{\log(\alpha)} \leq K_0 \log(n)$ with $K_0 = 1 + \frac{1}{\log(\alpha)}$. □

We can now see how to use the previous result to complete the proof of the algorithm on planar graphs. The following lemma is the only remaining missing part to prove Theorem 64.

Lemma 67. *Let G be a planar graph, v a vertex of G, and c a colouring of $G - v$. If there is no local Kempe exchange that increases inequalities, $N(v)$ is coloured using at most $K_0 \log \Delta$ colours for some constant K_0.*

Proof. Consider a graph G' whose vertex set is $N(v)$, the neighbours of v. For every pair of vertices $u_1, u_2 \in N(v)$ such that $c(u_1) \neq c(u_2)$, we add the edge (u_1, u_2) in G' if there is a bi-coloured path from u_1 to u_2 in G. We consider a circular drawing of G' where the vertices appear on the circle in the order they appear around the vertex v in a planar drawing of G.

The colouring c of G induces a colouring c' of G' such that:

- The colouring c' is proper. Indeed, we only added edges in G' between vertices with different colours.
- The colouring c' is intersection compatible. Let (u_1, u_3) and (u_2, u_4) be two crossing edges in G'. Then, the vertices u_i appear with the order u_1, u_2, u_3, u_4 around the vertex v in the planar drawing of G. Additionally, there is a bi-coloured $(c(u_1), c(u_3))$-path p_1 in G from u_1 to u_3, and a bi-coloured $(c(u_2), c(u_4))$-path p_2 from u_2 to u_4. Since G is planar, p_1 and p_2 must cross at some vertex w. This implies that $c(w) \in \{c(u_1), c(u_3)\} \cap \{c(u_2), c(u_4)\}$, and consequently, this intersection is not empty.
- Finally, assume that the colours are ordered $1, \ldots, k$ such that $\text{size}_c(1) \geq \text{size}_c(2) \geq \ldots \geq \text{size}_c(k)$. Then c is a Grundy colouring of G' for this ordering of the colours. To prove this, assume by contradiction that there is a vertex u and a colour $i < c'(u)$ such that u has no neighbour coloured i. We consider the local Kempe exchange $\langle u, i \rangle$ in the graph G. This Kempe exchange does not change the colour of any neighbour of v coloured i. Additionally, since $i < c(u)$, this implies $\text{size}(i) \geq \text{size}(u)$. Performing the Kempe exchange $\langle u, i \rangle$ would increase the size of i, and decrease the size of $c(u)$. Thus $\langle u, i \rangle$ increases inequalities, a contradiction of the assumption that c did not contain any local Kempe exchange increasing inequalities.

By applying Lemma 66 on the graph G' with the colouring c', we obtain that there is a constant K_0 such that c' uses at most $K_0 \log \Delta$ different colours. Consequently, the colouring c uses at most $K_0 \log \Delta$ colours on $N(v)$. □

We now have all the ingredients to prove the theorem.

Proof of Theorem 64. Let K_0 be the constant in Lemma 67. The algorithm uses $K_0 \log \Delta + 1$ colours. For each new vertex v, the procedure **GreedyRecolor** is applied on $N(v)$. By Lemma 65, the procedure performs at most $\frac{\Delta^2}{2}$ of Kempe exchanges. After the procedure has ended, we obtain a colouring c' such that there is no local Kempe exchange increasing inequalities. By Lemma 67, this implies that c' colours $N(v)$ using at most $K_0 \log \Delta$ colours. The vertex v can then be coloured using one additional colour and consequently the algorithm colours any planar graph with at most $K_0 \log \Delta + 1$ colours. □

6.4 Graphs of bounded genus

The algorithm described in previous section can be adapted to the more general case of graphs with bounded genus, by only paying a small (depending on the genus) number of additional colours. We

will assume that the reader is familiar with basic notions of topology and surfaces with boundaries. An introduction to these notions can be found for example in chapter 4 from [Ada04]. We start by defining some notation.

Given a triangulation T of a surface S, we denote by $V(T)$, $E(T)$ and $F(T)$ the set of vertices, edges and faces respectively of the triangulation T. Let S be a surface with a triangulation T. The *Euler characteristic* of S is the quantity $|V(T)| - |E(T)| + |F(T)|$. This quantity is an invariant of the surface S that is preserved by homeomorphism, and is independent of the chosen triangulation. The Euler characteristic can be negative, and is at most 2 for a connected closed surface, and at most 1 for a connected surface with boundary. For a closed orientable surface S, the genus g of S is related to its Euler characteristic h by the formula $h = 2 - 2g$.

A drawing of a graph G on a surface S is a mapping f that associates to every vertex v of G, a point $f(v)$ on S, and to every edge (u, v) of G, a path on S from $f(u)$ to $f(v)$. An embedding of a graph G on S is a drawing f of G on the surface S such that for any two edges e_1 and e_2, the paths $f(e_1)$ and $f(e_2)$ do not intersect. In the following, the Euler characteristic (resp. genus) of a graph G will denote the largest number h (resp. smallest number g) such that there exists an embedding of G on a connected surface S with Euler characteristic h (resp. genus g). Planar graphs have Euler characteristic 2 and genus 0. The goal of this section is to show the following theorem:

Theorem 68. *There is an online algorithm with Kempe exchanges colouring any graph with Euler characteristic h using $5 - 2h + O(\log \Delta)$ colours.*

The algorithm is exactly the same as for the planar case. To remove one colour from the neighbourhood of a vertex v, we just apply the procedure **GreedyRecolor**. We already know from Lemma 65 that the procedure ends after at most $\frac{\Delta^2}{2}$ steps. Thus, to prove the correctness of the algorithm, we only need to show that, when there is no more Kempe exchanges increasing inequalities, the neighbourhood of the v is coloured using at most $4 - 2h + O(\log \Delta)$ colours.

To prove this result, we generalize the definition of circular graph from previous section to handle graphs drawn on surfaces with higher genus. Let S be a surface with boundary, and G be a graph. A boundary drawing of G on S is a drawing f of G on the surface S such that for every vertex v, $f(v)$ is on the boundary of S. In a boundary drawing, paths corresponding to different edges are allowed to intersect. In particular, if S is a disk, a boundary drawing of a graph G on S is a circular drawing of G.

The definition of intersection compatible colouring extends to boundary drawings in a natural way. If G is a graph with a boundary drawing f on a surface S, a colouring c of G is intersection compatible if for every pair of edges $e = (u, v)$ and $e' = (u', v')$, if the paths $f(e)$ and $f(e')$ intersect, then the two edges have at least one colour in common, i.e., $\{c(u), c(v)\} \cap \{c(u'), c(v')\} \neq \emptyset$. We will need the following simple result describing how the Euler characteristic of a surface changes when we cut this surface along a path.

Lemma 69. *Let S be a surface with a boundary, and p be a simple path on S between two boundary points. Cutting the surface S along the path p yields a surface with Euler characteristic increased by 1.*

Proof. Let S be a surface with boundary, and h the Euler characteristic of S. Let p be a path between two boundary points of S, and S' be the surface obtained by cutting S along the path p. Let T be a triangulation of S. Without loss of generality, we can assume that the path p is along

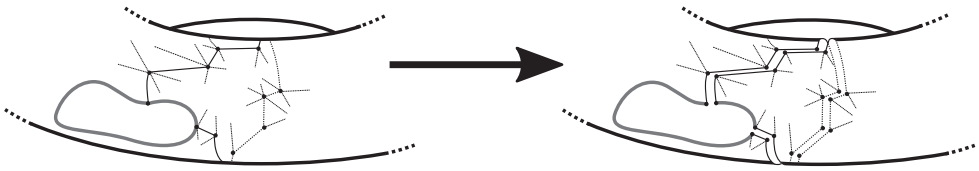


Figure 6.4: Cutting a surface (here a torus with a boundary) along a path. The bold grey line is the boundary of the surface.

the edges of the triangulation T. Consider the triangulation T' where each edge and each vertex of the path p were duplicated as in Figure 6.4. Let n be the number of vertices in the path p. Then, T' is a triangulation for S', and the Euler characteristic of S' is:

$$|V(T')| - |E(T')| + |F(T')| = (|V(T)| + n) - (|E(T)| + n - 1) + |F(T)| = h + 1$$

□

Note that in some cases, cutting a surface along a path can disconnect the surface, or increase the number of boundary components as in Figure 6.4. To prove the theorem, the key ingredient is to prove an extension of Lemma 66 for graphs with a boundary drawing on an arbitrary surface. This is done in the following lemma.

Lemma 70. *There exists a constant K_0 such that, if G is a graph with n vertices with a boundary drawing on a surface S with Euler characteristic h, and c is a Grundy k-colouring of G which is also intersection compatible, then $k \leq K_0 \log n + 2(1 - h)$.*

Proof. We show the result by induction on h, the Euler characteristic of the surface S. If $h = 1$, then the surface S is a disk, and the result follows from Lemma 66. Suppose that $h < 1$, and let G be a graph with a boundary drawing f on a surface S with Euler characteristic h. Let c be a Grundy colouring of G that is intersection compatible for this drawing. We distinguish two cases:

Case 1. There is an edge e of G such that cutting the surface S along the path $f(e)$ leaves the surface connected. Let a and b be the two colours of the endpoints of this edge. We consider S', the surface obtained by cutting S along $f(e)$, and G' the graph induced by the vertices with a colour different from a or b. Then, the boundary drawing of G on S induces a boundary drawing of G' on S'. Indeed, if (u, v) is an edge of G', then the path corresponding to (u, v) cannot cross $f(e)$ since otherwise one of the endpoints u or v would be coloured either a or b since the colouring is intersection compatible.

Moreover, by Lemma 69, the surface S' has Euler characteristic $h + 1$. Using the induction hypothesis on G', with the colouring induced by c on G', we know that c colours G' with at most $K_0 \log n + 2(1 - (h + 1))$ colours. Since the vertices we removed were the ones coloured a or b, this implies that the colouring c of G uses at most $K_0 \log n + 2(1 - h)$ colours, which proves the induction step.

Case 2. For any edge e, cutting S along the path $f(e)$ disconnects the surface. In this case, we claim that the graph G has a boundary drawing on a disk such that c is intersection compatible for this drawing. To prove this, we first remark that we can assume that there is only one boundary

component on the surface S. Indeed, suppose that this is not the case. Consider the boundary that contains a vertex v with colour k, the largest colour used by c. Let G' be the graph induced by all the vertices on this boundary. Then all the colours are present in G'. Indeed, G' contains the vertex v with colour k. Since c is a Grundy colouring, there are vertices $v_1, \ldots, v_{k-1}$ adjacent to v, with colours $1, \ldots, k-1$ respectively. For all i, the vertex v_i is on the same boundary as v since otherwise cutting along the path corresponding to the edge (v, v_i) would leave the surface connected. Let c' be the colouring induced by c on G'. If the result holds for G' with colouring c', it also holds for G with colouring c since c and c' use the same number of colours. So we now assume that S has one unique boundary.

Let $u_1, \ldots, u_n$ be the vertices of G in the order they appear on the boundary of S, and consider the circular drawing of G using the same ordering $u_1, \ldots, u_n$ around the circle. To prove the result, we only need to show that c is also intersection compatible for this new drawing. Indeed, if this is the case, then the result immediately follows from Lemma 66. Let e_1 and e_2 be two edges of G that intersect in the circular drawing of G. Without loss of generality, we can assume that $e_1 = (u_1, u_i)$, and $e_2 = (u_j, u_l)$, with $1 < j < i$, and $i < l \leq n$. Let p_1 and p_2 be the paths on S corresponding to these two edges. Assume by contradiction that in the drawing of G on S, the two paths p_1 and p_2 do not intersect, we will show that cutting along p_1 does not disconnect the surface S. For this, it is enough to show that there is a path not intersecting p_1 going from one side of p_1 to the other side. We will construct a path from u_{i-1} to u_{i+1} that does not intersect p_1. The path is as follows. Go from u_{i-1} to u_j by following the border of S, then follow the path p_2 from u_j to u_l, and finally go from u_l to u_{i+1} by following again the border of S. This path does not intersect p_1. Hence cutting along p_1 leaves the surface S connected, a contradiction.

Thus, we know that for every pair of edges e_1 and e_2, if the segments corresponding to e_1 and e_2 in the circular drawing of G intersect, then the paths corresponding to these two edges on the drawing of G on S also intersect. Consequently, c is intersection compatible for this circular drawing of G, and the result follows from Lemma 66.

□

We now have everything we need to prove the theorem.

(proof of Theorem 68). The proof follows the same argument as the proof of Theorem 64. Let G be a graph with Euler characteristic h, v a vertex of G, and c a colouring of $G - v$. We want to show that after applying procedure **GreedyRecolor**, the neighbourhood of v is coloured using at most $K_0 \log \Delta + 2(2-h)$ colours. Let S be a closed surface with Euler characteristic h such that there is an embedding of G on S. Let G' be the graph whose vertices are the neighbours of v, and such that there is an edge between u_1 and u_2 if and only if $c(u_1) \neq c(u_2)$ and there is a bi-coloured path from u_1 to u_2. Without loss of generality, we can assume that the colours are ordered by decreasing size: $\text{size}_c(1) \geq \text{size}_c(2) \geq \ldots$. Then, as before, c is a Grundy colouring of G'. Indeed, if there is a vertex v and a colour $i < c(v)$ such that v has no neighbour coloured i, then the Kempe exchange $\langle v, i \rangle$ increases inequality.

Moreover, if S' is the surface S where a small disk around vertex v was removed, then from the embedding of G on S, we can construct a boundary drawing of G' on S'. The colouring c induces an intersection compatible drawing of G' for this drawing. Since the Euler characteristic of S' is $h-1$, by Lemma 70, at most $K_0 \log \Delta + 2(2-h)$ are used by c on G'. Consequently, at least one colour is not present in the neighbourhood of v. □

6.5 Bad graphs for online algorithms with Kempe exchanges

The goal of this section is to exhibit concrete example of graphs with small chromatic number that need a large number of colours to be coloured by an online algorithm using Kempe exchanges. Unfortunately, this example does not work for general algorithms with Kempe exchanges, but only to more special cases with the following restrictions on the algorithm:

- The algorithm is specified as input the number k of colours it is allowed to use to colour the graph.
- If one of the k colours is available, the algorithm must select it, without performing any Kempe exchange.
- The algorithm always chooses the smallest colour available.

These assumptions might seem a bit restrictive, but all the algorithms we described above satisfy these constraints. Getting a counter example that would work in the general case is more complicated as it is difficult to quantify how much the graph can be recoloured by Kempe exchanges performed by the algorithm. We show the following theorem:

Theorem 71. *There is a sequence of graph $(G_k)_{k\geq 0}$ such that for all $k \geq 4$, G_k is 3 colourable, and no online algorithm with Kempe exchanges with the restrictions above can colour G_k with k colours or less.*

Note that the graphs built in the proof of the theorem are similar in flavour to the Fibonacci trees used to lower bound the performance of greedy colouring algorithms.

Proof. Let A be an online algorithm with Kempe exchanges with the restrictions above. First observe that as long as one colour is available, the algorithm A behaves exactly as First-Fit: it assigns the smallest colour available without changing the colour of previously coloured vertices. We denote by T_k the fibonacci tree with the following definition. The tree T_1 is a single vertex, and for $k > 0$, T_k is composed of a root vertex u, to which we attach the trees $T_1, \ldots, T_{k-1}$.

It was proved in [Bea76], that if the vertices of T_k are presented starting from the leaves, and going up to the root, then First-Fit needs k colours to colour T_k, and the root of T_k is coloured with colour k. We build the graph H_k in the following way:

- for each $1 \leq i \leq k$, we add a copy of T_i, with vertex u_i as the root,
- for each $i < j$, and each $i' \neq i$ and $j' \neq j$ with $i' \neq j'$, we add one copy of $T_{i'}$ and one copy of $T_{j'}$ with roots $w^1_{i,j,i',j'}$ and $w^2_{i,j,i',j'}$ respectively,
- we add all the following edges: $(u_i, w^1_{i,j,i',j'})$, $(u_j, w^2_{i,j,i',j'})$, and $(w^1_{i,j,i',j'}, w^2_{i,j,i',j'})$,
- finally, a vertex u is added, with u adjacent to all the u_i.

The graph H_k is presented starting from the leaves of all the copies of T_i, and going up to the roots. The vertex u is presented last. Since H_k is 2-degenerate, it is 3 colourable. We prove that the algorithm A can't colour H_k if it is given exactly k colours. Indeed, since the algorithm always colour immediately a vertex if one colour is available, it will be able to colour all the vertices except for u, by applying the rules of First-Fit. When it tries to colour u, all the colours are

already present in the neighbourhood of u, so the algorithm might try to remove one colour using local Kempe exchanges. However, we will show that at this point, no local Kempe exchange can remove one colour from the neighbourhood of u. More precisely, we will show that any local Kempe exchange will preserve the following invariants:

1. The vertices u_i all have different colours.
2. For every indices i and j, with $i \neq j$, and every pair of colours $a \neq b$ with $a \neq c(u_i)$, and $b \neq c(u_j)$, there is a path on four vertices (u_i, w^1, w^2, u_j) such that $c(w_1) = a$ and $c(w_2) = b$.

By construction, these invariants are satisfied before any Kempe exchange is made. Suppose now that there is a colouring c that satisfies these invariants. Consider the Kempe exchange $\langle u_i, x\rangle$, and let $j \neq i$ such that $x = c(u_j)$. We will show that the colouring $c' = \langle u_i, x\rangle(c)$ still satisfy the two invariants above.

Using the fact that c satisfies the second invariant, we know that there is a Kempe path from u_i to u_j. Consequently, applying the Kempe exchange $\langle u_i, x\rangle$, swaps the colours of the vertices u_i and u_j. Consequently, c' still satisfies the invariant 1. We only need to show that c' also satisfies the second invariant. Consider two indices i' and j', and two colours $a \neq c'(i')$ and $b \neq c'(j')$. We consider the following three cases:

- If $i' \neq i$ and $j' \neq j$, then $c'(u_{i'}) = c(u_{i'})$ and $c'(u_{j'}) = c(u_{j'})$. Since c satisfies the second property, there are two vertices w^1 and w^2 such that $c(w^1) = a$ and $c(w^2) = b$ such that $(u_{i'}, w^1, w^2, u_{j'})$ is a path on four vertices. Since the colours of w^1 and w^2 also do not change during the Kempe exchange, we also have $c'(w^1) = a$ and $c'(w^2) = b$.
- If $i' = i$ and $j' \neq j$, then $c'(u_i) = c(u_j)$, and $c'(u_{j'}) = c(u_{j'})$. If $a \neq c(u_j)$, then in the colouring c there was a path on four vertices with successive colours $(c(u_i), a, b, c(u_{j'}))$. This path now has colours $(c(u_j), a, b, c(u_{j'}))$. If $a = c(u_i)$ and $b \neq c(u_j)$, then with the colouring c there was a path from u_i to $u_{j'}$ with colours $(c(u_i), c(u_j), b, c(u_{j'}))$. This path is now coloured $(c(u_j), a = c(u_i), b, c(u_{j'}))$. Finally, if $a = c(u_i)$ and $b = c(u_j)$, the path with colours $(c(u_i), c(u_j), c(u_i), c(u_{j'}))$ now has the necessary colours.

  The argument is symmetrical if $i' \neq i$ and $j = j'$.
- If $i' = i$ and $j' = j$, the same argument as above can be used to prove that there is still a path between u_i and u_j with colours $(c'(u_i), a, b, c'(u_j))$. The idea is that colours different than $c(u_i)$ and $c(u_j)$ are unchanged, and vertices coloured $c(u_i)$ or $c(u_j)$ gets their colour swapped.

Consequently, the invariant is preserved when we perform any local Kempe exchange. Consequently, the algorithm A won't be able to colour H_k if it is given exactly k colours. However, the argument relies on the fact that the algorithm does not perform Kempe exchanges before considering the vertex u. Consequently, the algorithm might still manage to colour H_k if it is given less than k colours. This issue can be solved by considering $G_k = \bigcup_{i \leq k} H_i$. For any $k' \leq k$, the algorithm A will fail to colour H'_k, and consequently G_k, and the theorem holds. □

6.6 Conclusion

We studied a variation of online algorithms where the algorithm is allowed to recolour some of the previously coloured vertices and constructed algorithms for several classes of graphs. In the case of planar graphs, it remains open whether the $O(\log \Delta)$ bound can be improved or not.

Open Problem 14. Is there an online algorithm with Kempe exchanges which colours any planar graph with a constant number of colours?

Our algorithm on planar graph relies on the notion of intersection compatible colourings, which is an interesting combination of graph drawing with colouring constraints. It could be worth investigating this notion on its own. For example, one could try to characterize the graphs which have a drawing on a given surface S with an intersection compatible colouring. An other approach could be to find the smallest genus of S such that G admits an intersection compatible colouring for some drawing on S. The particular case where S is a disk is also interesting. Indeed, in this case we know that any 3-colouring is intersection compatible for any boundary drawing of G on S. However, we were not able to find examples of graphs with larger chromatic numbers. This prompts the following question:

Open Problem 15. Is there a graph G with $\chi(G) = 4$ such that G admits a boundary drawing on a disk with an intersection compatible colouring?

Finally, this work has also some similarities with the reconfiguration problems we studied in Part I. More precisely, the work we did here consisted in studying a recolouring problem with some locality constraints on the permitted transformations. It could be interesting to consider standard reconfiguration questions with the addition of these constraints. An example of such problem could be deciding if a given colouring can be transformed into an other using only local Kempe exchanges, or given two vertices u and v, deciding if there is a sequence of Kempe exchanges local to u changing the colour of v.

Part III

Adding a Second Player

Chapter 7

Combinatorial Game Theory

This chapter serves as a light introduction to Combinatorial Game Theory (CGT), a field dedicated to the study two players perfect information games. In this chapter, we give basic definitions of CGT terminology, and illustrate these definitions with examples. The notions defined here will be used in the last two chapters.

In the previous chapter, we considered online algorithms for graph colouring. Online algorithms are strongly related to two player games. For example, finding the smallest number of colours for which an online colouring algorithm exists can be seen as a game between two players: one of the players decides which vertex will be coloured next, while the other (the algorithm) decides how to colour the vertex selected by his opponent. The first player tries to force the algorithm to use many different colours. In the case of an online algorithm, the algorithm has no knowledge of the whole graph, but can only see the vertices selected up to now. On the contrary, in this part we consider games with perfect information: the two players know all the information related to the game, and there is no random event. Additionally, the players play alternately.

The games we consider in this part are a subclass of perfect information games called combinatorial games. In these games, the winner is determined by the player who makes the last move. In particular, these games are often studied with the convention that the player making the last move wins. With these conditions, for any game at least one of the two players has a winning strategy, i.e., a way of playing which ensures that this player wins no matter what his opponent plays. Indeed, saying that one player has no winning strategy means that no matter what strategy this player chooses, his opponent has a way to counter it and win. In other words, his opponent has a winning strategy. Although a winning strategy always exists for one of the players, deciding which of the players has a winning strategy can be difficult. Additionally, even if the winner is known, describing the winning strategy might not be easy. For example, there are games (such as Chomp [Gal74] or Hex [Maa05]) for which the player with the winning strategy is known, but the first move of this winning strategy is still open.

Apart from the interest in understanding traditional games such as Chess or Go, combinatorial games have been considered partly because of surprising connections with other areas of mathematics such as algebras [Con00], automatons [Niv05, Lar13, CLN17] and dynamical systems [MFL11]. Interested readers can refer to the following books [Sie13, ANW07, BCG04] for a more complete introduction and presentation of combinatorial game theory.

Combinatorial games are strongly related to reconfiguration problems which were studied in

Part I. Many one-player games, also known as combinatorial puzzles, are in fact reconfiguration problems, and there are similarities between reconfiguration problems and two player games. As in reconfiguration problems, any combinatorial game defines an underlying graph, the *game graph*[1], which is the equivalent of the reconfiguration graph. The game graphs represents all the possible positions of the game, together with the valid moves the players can make between them. In both reconfiguration and games, we are dealing with a graph which is too large to be computed entirely, but for which we try to find some properties. In the case of reconfiguration, the focus is made on problems related to the connectivity of the reconfiguration graph, and finding transformation sequences between solutions. For games, we are concerned with the existence of winning strategies, which correspond in terms of graphs to the notion of kernel (see [Fra97]). In both cases, we have to rely on structural properties of this graph in order to solve the problems.

game graph

Note that combinatorial games and reconfiguration problems also share similarities in terms of complexity. For combinatorial games, the problem of deciding which of the two players has a winning strategy is often **PSPACE**-hard. Hence, both reconfiguration problems and combinatorial games are a natural source of computationally hard problems. Although there are many interesting questions related to the complexity and algorithmic aspects of combinatorial games, we will focus more on algebraic and structural properties of combinatorial games.

The rest of the section is organized as follows. In Section 7.1, we give a basic definitions of combinatorial games, and illustrate these with examples. In Section 7.2 we introduce basic terminology related to game values and disjunctive sums. In Section 7.3, we consider a particular class of games, called subtraction games, and give an short overview of known results and problems related to these games. Finally, Section 7.4 presents some existing results on composition of games, a notion which is related to the construction we study in the next chapter.

7.1 Introduction

The definition of combinatorial games is not completely fixed, and the exact limit of which games are combinatorial games, and which are not is blurry. We adopt here a very conservative approach for our definitions [Sie13, ANW07, BCG04], but it should be noted that in the literature they are sometimes extended to cover games with different winning conditions or outcomes such as games with draws or scoring games, games with more than two players, or games with an infinite number of positions.

In the following, a *combinatorial game* is a two player games with no randomness, no hidden information, where the two players play alternately. The winner is determined depending on which of the players makes the last move. The two players will be denoted by $\mathcal{L}$eft and $\mathcal{R}$ight. A game is described by a *ruleset*, and a *position*. The ruleset gives the permitted moves, while the position describes the current state of the game. More formally, we will denote by Ω the set of all the possible positions of a game. A *ruleset* $\mathcal{R}_0$ on Ω gives for each player p and for each position g in Ω the set of possible positions which can result from this player making a move on position g. In the following, we will denote by $(g)_{\mathcal{R}_0}$ the game with ruleset $\mathcal{R}_0$ and starting at position $g \in \Omega$, and use upper case letters $G, H, \ldots$ to denote games, and lower case $g, h \ldots$ to denote positions. We use $\mathcal{R}_0, \mathcal{R}_1, \ldots$ to denote rulesets, and specific instances of rulesets will be written using small capitals. The *$\mathcal{L}$eft-options* (resp. *$\mathcal{R}$ight-options*) of a game $(g)_{\mathcal{R}_0}$ are all the games of the form

combinatorial game

ruleset

game position

$\mathcal{L}$eft-options

$\mathcal{R}$ight-options

[1] Note that the game graph is a directed graph.

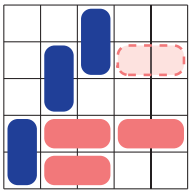


Figure 7.1: Example of a domineering board with already placed dominoes. The dashed transparent domino is an example of a valid move for $\mathcal{R}$ight.

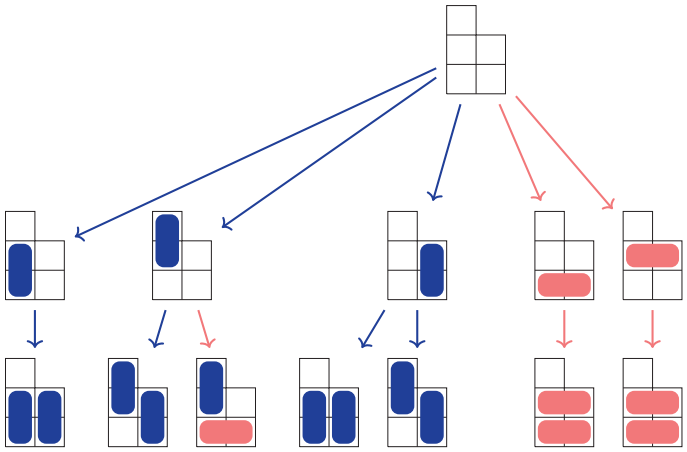


Figure 7.2: Representation of a game of DOMINEERING as a tree. Red edges corresponds to moves made by $\mathcal{R}$ight. Blue edges are moves made by $\mathcal{L}$eft.

$(g')_{\mathcal{R}_0}$, where g' can be reached from g after one move of $\mathcal{L}$eft (resp. $\mathcal{R}$ight). Before continuing with the definitions, we start by giving an example.

The game DOMINEERING is played on a rectangular board. $\mathcal{L}$eft places vertical 1×2 dominoes on the board, while $\mathcal{R}$ight places horizontal 2×1 dominoes. The dominoes cannot overlap. The first player with no move available loses. Note that the board may have any shape. An example of a position is given in Figure 7.1. On this example, the move shown in dashed is a winning move for $\mathcal{R}$ight. Indeed, after making this move, $\mathcal{L}$eft can play only one more domino, while $\mathcal{R}$ight has secured three additional moves. If $\mathcal{L}$eft was playing the next move instead of $\mathcal{R}$ight, then the reader can check that playing in the top right corner would have been a winning move to $\mathcal{L}$eft.

It is often convenient to represent a game as a tree as in Figure 7.2. The root of the tree corresponds to the current position of the game, and its children are its $\mathcal{L}$eft- and $\mathcal{R}$ight-options. Any path down the tree corresponds to a sequence of moves by the players. The leaves corresponds to positions for which there is no move available for any of the two players. The edges are coloured blue and red in Figure 7.2 to distinguish the moves made by $\mathcal{L}$eft (in blue) and those made by $\mathcal{R}$ight(in red). In all the rest of this part, we will only consider games which have finite game trees (i.e., the game ends after a finite constant number of moves, and there is always a finite number of options). Note that the same position can appear several times in this tree if there are more than one sequence of moves resulting in this position. Also note that in the tree, sequences of moves where the same player plays twice in a row can appear (such as the left-most branch of Figure 7.2).

If this might seem strange at first glance to include these moves, the reason for this will become clear in Section 7.2 when considering combinations of games.

A *play* of a game is a sequence of alternating moves from both players until one player has no available moves left. The winner of a game is given by looking at which player made the last move. Note that the information of which player has to play next is not encoded in the position. Hence, there are two possible ways to play a game: $\mathcal{L}$eft make the first move, or $\mathcal{R}$ight is the first player. play of a game

There are two main conventions which were studied in the literature: *normal-play*, where the player making the last move wins, and *misère-play* where the player making the last move loses. Although it would seem at first that there is not much difference between the two conventions, it turns out that the theory for misère play is much more complicated than normal play. Understanding difficulties which arise from misère play is a topic of ongoing research (see [MO07] for example). Unless specified otherwise, we will always consider games under the normal-play convention. Note however that the definitions below can be made similarly for the misère-play convention. normal-play misère-play

In the rest of the part, we will assume that the two players play optimally. Hence, we will say that one of the player *wins* if he has a winning strategy for the game. Conversely, this player *loses* if his opponent has a winning strategy. The *outcome* of a game G, denoted $o(G)$ is the result of the game if the two players play optimally. There are four possible types of outcomes: outcome

- the outcome is $\mathcal{L}$ if $\mathcal{L}$eft wins regardless of who the first player is,
- the outcome is $\mathcal{R}$ if $\mathcal{R}$ight wins regardless of who the first player is,
- the outcome is $\mathcal{N}$ (for $\mathcal{N}$ext player wins) if the starting player always wins,
- the outcome is $\mathcal{P}$ (for $\mathcal{P}$revious player wins) if the starting player never wins (i.e., the second to play always wins).

Note that the outcome can be easily computed from the game tree. Indeed, given a game G, $\mathcal{R}$ight wins playing first on G if and only if there exists one $\mathcal{R}$ight-option of G such that $\mathcal{R}$ight wins playing second on it. Similarly, $\mathcal{R}$ight wins playing second on G if and only if $\mathcal{R}$ight wins playing first on every $\mathcal{L}$eft-option of G. Once we have determined who wins when either $\mathcal{R}$ight or $\mathcal{L}$eft plays first, the outcome can be computed according to Table 7.3. Hence, the outcome of a game can be computed from the outcome of its options. For example, the position of Domineering at the root of the tree in Figure 7.2 has outcome $\mathcal{N}$ since both players have a winning strategy if they start. Given a ruleset $\mathcal{R}_0$ on Ω, we will say that a position $g \in \Omega$ is a *$\mathcal{P}$-position* if $(g)_{\mathcal{R}_0}$ has outcome $\mathcal{P}$. We use the same notation for the other outcomes. $\mathcal{P}$-position

Characterizing the outcome of the positions of a given game, or finding an efficient algorithm computing this outcome, is the major problem considered in combinatorial game theory. As we just saw, this outcome can be computed in time linear in the size of the game tree. However, the game tree is usually too large to make this computation efficient, and we must rely on properties of the specific games we consider in order to answer this question.

7.1.1 Impartial games

We will say that a ruleset is *impartial* if both player always have the same available moves on any position. If a ruleset is not impartial, we will say that it is *partizan*. For example, Domineering is partizan since the two players place different types of dominoes. On the contrary, if we consider impartial game partizan game

$\mathcal{L}$eft plays first \ $\mathcal{R}$ight plays first	$\mathcal{R}$ight wins	$\mathcal{R}$ight loses
$\mathcal{L}$eft wins	$\mathcal{N}$	$\mathcal{L}$
$\mathcal{L}$eft loses	$\mathcal{R}$	$\mathcal{P}$

Table 7.3: Table showing the four different possible outcomes depending on which player wins if they play first or second.

the version of DOMINEERING where both player can place both types of dominoes, then this ruleset (called CRAM) is impartial. Note that impartial and partizan are properties of the ruleset, and not of the game, so the starting position of a game has no influence on this property. Since the two players play symmetric roles in impartial rulesets, not all outcomes are possible. Indeed, if $\mathcal{R}$ight wins playing first on G, then $\mathcal{L}$eft also wins playing first on G, and similarly if $\mathcal{R}$ight plays second. Hence, from Table 7.3 we can see that only the outcomes $\mathcal{N}$ and $\mathcal{P}$ can occur for these games. If a ruleset $\mathcal{R}_0$ is impartial, we can view it as a function $\Omega \to 2^\Omega$, which describes which moves are possible from any given position. In particular, for $g \in \Omega$ we can write $\mathcal{R}_0(g)$ the positions which can be reached from g. Since both players have symmetric roles, given a game $G = (g)_{\mathcal{R}_0}$, its options denotes the games which can be reached from G after making one move. In other words, it corresponds to the set $\{(g')_{\mathcal{R}_0},\ g' \in \mathcal{R}_0(g)\}$.

Additionally, when considering the game tree for an impartial ruleset, we no longer need colours to distinguish moves available for $\mathcal{L}$eft or $\mathcal{R}$ight since the two players always have the same available moves. We will denote by $\mathfrak{G}$ the set of all possible impartial games (in other words, the set of all rooted trees). Note that sometimes, it is convenient to also represent an impartial game by a (directed) graph. This graph is obtained from the game tree by identifying (i.e., merging together) the vertices of the game tree which corresponds to the same position. Note that the graph obtained in this way is directed and contains no oriented cycles.

set of $\mathcal{P}$-positions

Sometimes, instead of trying to give the outcome of particular games, it is easier to describe the set of all the positions with outcome $\mathcal{P}$. This set is called the *set of $\mathcal{P}$-positions*. The following lemma is a standard result which gives sufficient conditions for a subset $P \subset \Omega$ of position to be the set of $\mathcal{P}$-positions of $\mathcal{R}_0$. The first condition states that there is no move from a position in P to another position in P. The second condition asserts that for every position not in P, there is a move to a position in P. This result will be used to characterize the $\mathcal{P}$-positions of several games.

Lemma 72. *Let $\mathcal{R}_0$ be a ruleset over a set of positions Ω, and let $P \subset \Omega$. P is the subset of $\mathcal{P}$-positions if and only if:*

- *there are no two positions $s, s' \in P$, with $s' \in \mathcal{R}_0(s)$,*
- *for every position $s \in \Omega \setminus P$, we have $\mathcal{R}_0(s) \cap P \neq \emptyset$*

Proof. Let P be a set satisfying the two conditions above, and let p be a position in Ω. We will show by induction on the size of the game tree of $(p)_{\mathcal{R}_0}$ that $o_{\mathcal{R}_0}(p) = \mathcal{P}$ if and only if $p \in P$. If $(p)_{\mathcal{R}_0}$ has no option, then $o_{\mathcal{R}_0}(p) = \mathcal{P}$, and $p \in P$ by the second property of P.

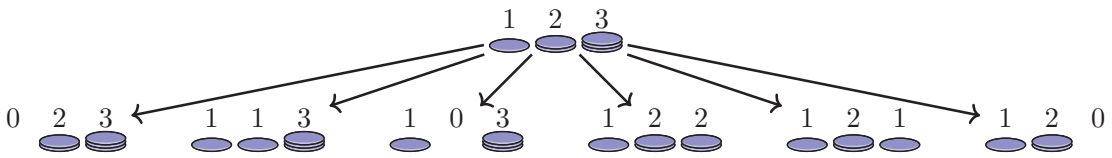


Figure 7.4: Example of a NIM position with its available options.

Assume now that there is at least one option on p, and the induction hypothesis holds for all the options of p. If $p \in P$, then for all $p' \in \mathcal{R}_0(p)$, we have $p' \notin P$ by the first condition, and consequently, $o_{\mathcal{R}_0}(p') = \mathcal{N}$ using the induction hypothesis. This implies that p is a $\mathcal{P}$-position. If $p \notin P$, then by the second condition there is a move from p to p', with $p' \in P$. Using the induction hypothesis, we have $o_{\mathcal{R}_0}(p') = \mathcal{P}$, and consequently p is a $\mathcal{N}$-position.

This concludes the induction step and proves the result. □

7.1.2 Examples of games

In this section, we give several examples of games which are well known in the CGT community. We already mentioned DOMINEERING which is played on a board, with $\mathcal{L}$eft placing vertical dominoes, and $\mathcal{R}$ight horizontal ones. DOMINEERING was introduced by Göran Anderson around 1973 [BCG82, p.119]. An important interest has been focused on solving the game on small boards, as well as finding positions with interesting properties [Ber88, BUH00, Bul02, Uit16]. The game has also been studied on special boards called 'snakes', in relation to some periodic behaviour of the game [Wol93]. In [LMR02], polynomial time algorithms for computing a winning strategy was found for rectangular boards of size $k \times n$, with $k \leq 11$. DOMINEERING

The impartial version of DOMINEERING, where both player can place the two types of dominoes, is called CRAM. The outcome of CRAM on $1 \times n$ strips is known. On this particular type of board, the game is equivalent to an octal game denoted 0.07, and is also called DAWSON'S KAYLES. The sequence of outcomes of CRAM on strips of size $1 \times n$ for $n = 1, 2, 3 \ldots$ is known to be ultimately periodic with a period of length 34 and a preperiod of length 52. On strips of even size, the first player has a simple winning strategy by placing a first domino in the center of the strip, and then play the move symmetrical to his opponent. Using similar argument, it is possible to describe winning strategies for CRAM on rectangular boards of size $2 \times n$ for any $n \geq 0$, and more generally for any board with at least one even dimension. The outcome was also computed on small square boards [LV12] up to 7×7. The complexity of determining the winner on an arbitrary position is still open for both DOMINEERING and CRAM, even for generalisations of the games played on arbitrary graphs. CRAM

Another type of game which has been intensively studied are heap games, i.e., games played on (one or several) heaps of tokens. The most famous of these games is NIM where the players can remove any number of tokens from one single heap. The player taking the last token is the winner. An example of a position for NIM, together with its valid options is shown in Figure 7.4. Played on a single heap, this game is not very interesting since the first player always has a winning move by removing all the tokens at once (assuming the heap is not empty). In the case of multiples heaps, the game was solved by Bouton in 1901 [Bou01] who gave the following characterization of the $\mathcal{P}$-positions. NIM

(a) (b)

Figure 7.5: Alternative definition for the games NIM (on the Left) and WYTHOFF (on the right) as a game of moving a piece on a semi-infinite board. The possible moves are shown in orange. The games end when the piece reaches the bottom left corner.

Theorem 73 ([Bou01])**.** *The game of* NIM *played on heaps of size* $n_1, \dots, n_k$ *has outcome* $\mathcal{P}$ *if and only if* $n_1 \oplus \cdots \oplus n_k = 0$*, where* $\oplus$ *is the bitwise XOR.*

The game of NIM plays a special role in combinatorial game theory. In fact the theorem above is a particular case of a more general result related to sums of games and game values. Due to this, there is a special notation for NIM positions: we will denote by $*n$ the game of NIM played on a single heap of size n.

On two heaps, there is another way to describe NIM. Consider a semi-infinite grid with a rook placed on this grid. The game where the two players can move the rook vertically or horizontally, only towards the bottom left corner is exactly NIM on two heaps (see Figure 7.5a). Indeed, there is a bijection between the coordinates of the rook and the number of tokens on each heap. Removing k tokens from one of the two heaps corresponds to moving the rook by k squares in the corresponding direction. On two heaps, the $\mathcal{P}$-positions are the positions on the diagonal. NIM is also related to other kinds of games called SUBTRACTION GAMES that we consider in more detail in Section 7.3.

WYTHOFF

A variant of NIM, called WYTHOFF consists in moving a queen instead of a rook in the game described above. In other words, in addition to moving the piece vertically or horizontally, the players can also move it diagonally as in Figure 7.5b. With the formulation with heaps of tokens, this means that the players are also allowed to remove the same number of tokens from both heaps. The $\mathcal{P}$-positions of WYTHOFF were characterized in [Wyt07] where the following result is proved:

Theorem 74. *The* $\mathcal{P}$*-positions for* WYTHOFF *are the positions of the form* $(\lfloor \Phi n \rfloor, \lfloor \Phi^2 n \rfloor)$ *and their symmetric, for* $n \geq 0$*, and where* Φ *is the golden ratio.*

Many variations of the game WYTHOFF have been considered in the literature [Con59, FB73, DR08, Lar11]. In many cases, the $\mathcal{P}$-positions of these games are related to Beatty sequences: sequences of integer of the form $(\lfloor \alpha n \rfloor)_{n \geq 0}$, where α is irrational. Note also that due to the properties of the golden ratio we have $\lfloor \Phi^2 n \rfloor = \lfloor \Phi n \rfloor + n$.

Finally, the last game we would like to introduce is a game called EUCLID. As the name implies, EUCLID is related to the euclidian division. It is played on two non-empty heaps of tokens. At their turn, the players can remove from the largest heap a number of tokens which is a multiple of the smallest heap. The game ends when the two heaps have the same size. If it has probably attracted

less attention than Wythoff and Nim, this ruleset has the particularity that the moves available depends on the current position of the game. This property of the ruleset is sometimes called *non-invariant* (or just 'variant') [DR10], *self-referential* [Mul16], *pilesize dynamic* [HRR03, HRR04] or also *time and size dependent game* [Fla82]. The $\mathcal{P}$-positions for Euclid were given in [CD69]:

non-invariant game

Theorem 75 ([CD69])**.** *The position (a, b), with $b \geq a$ is a $\mathcal{P}$-position for* Euclid *if and only if $\frac{b}{a} < \Phi$, where Φ is the golden ratio.*

The outcomes for Euclid under misère-play convention were considered in [Gur07], and a partizan version of the game was also considered in [MN13].

7.1.3 Complexity

Although the results in this part are less focused on algorithmic complexity compared to our results on reconfiguration in Part I, we would like to mention a couple of results concerning the complexity of combinatorial games, in comparison with reconfiguration problems. For combinatorial games, the problem often considered from a complexity point of view is the following. For a fixed ruleset $\mathcal{R}_0$, given as input a position g, compute the outcome of $(g)_{\mathcal{R}_0}$. When we mention the complexity of a combinatorial game, this is the problem we consider.

As it was the case for reconfiguration problems, two player games tend to be much harder than traditional problems: many are **PSPACE**-hard, and some like some generalisation of Chess are even **EXPTIME**-hard [FL81]. Although this seems to imply some similarities between the two problems, there are also important differences. Indeed, the two types of problems are usually hard for different reasons. As we saw in Part I, reconfiguration problems are hard because of the existence of configurations for which the shortest transformation has exponential length. On the contrary, combinatorial games usually end after only a polynomial number of rounds (nobody wants to play a game that takes exponential time to even finish). The high complexity comes from the alternation between players, which acts as an alternation between existential and universal quantifiers. Indeed, saying that the first player has a winning strategy can be reformulated as this player having a move, such that for every move of his opponent he can reply another move, such that ... such that the the first player wins. Remark that when the game ends in a polynomial number of moves, this formulation effectively expresses the problem as an instance of QBF (Quantified Boolean Formula, see Section 1.2), which implies immediately that deciding the outcome of the game is in **PSPACE**. Note however that there are games may take an exponential time to play. All the heap games that we described in the previous section are example of these. Indeed, if we consider Nim for example, a position on two heaps of size n_1 and n_2 can be represented using only $\log(n_1) + \log(n_2)$ bits, but playing the game can take a number of rounds linear in $n_1 + n_2$. More details and discussions about these aspects can be found in [Fra04].

The complexity landscape of reconfiguration is pretty well known: many problems have a known complexity, and the interesting questions are more related to what happens on restricted inputs. The situation is different for combinatorial games. The complexity of many games is not settled, even though many of them are believed to be hard. Even for those games for which hardness reduction were proved, few results are known when there are restrictions on the input (for example on specific classes of graphs for games played on graphs).

There are several reasons for this disparity. First, reconfiguration problems tend to be much more resilient to perturbations than combinatorial games: adding or removing one single edge to the reconfiguration graph is usually not going to change much about to the complexity of the problem.

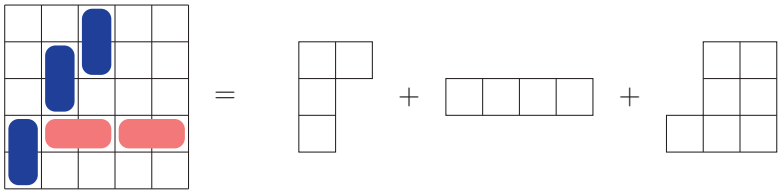


Figure 7.6: Decomposition of a position for DOMINEERING into a disjunctive sum of several components.

On the contrary, for combinatorial games, the slightest modification of the configuration space can have dramatic effects and completely change the overall behaviour. Due to this sensitivity, combinatorial games often have very chaotic behaviours which can be very difficult to characterize.

7.2 Game Values

disjunctive sum

The notion of value of a game emerges from the need to understand compound games, i.e., games containing several sub-games as components. Many different types of compound games have been considered in the literature, but one, called *disjunctive sum* received most of the attention, both due to the simplicity of the theory it creates, and its natural occurrence in many games. The disjunctive sum of two games G_1 and G_2, denoted $G_1 + G_2$, can be informally described as putting G_1 and G_2 side by side. At each turn, one player can choose to play in either component (but not on both). Remark that this implies that a player might be able to make two consecutive moves in the same component, if his opponent plays in the other component. This explains why in the representation of the game as a tree in Section 7.1, we allowed consecutive moves. If these consecutive moves do not occur when then game is played in isolation, they can be played when we consider compounds of games.

More formally, the set of options of $G_1 + G_2$ for player $\mathcal{L}$eft are: (i) games of the form $G_1^{\mathcal{L}} + G_2$, where $G_1^{\mathcal{L}}$ is a $\mathcal{L}$eft-option of G_1; (ii) games of the form $G_1 + G_2^{\mathcal{L}}$, where $G_2^{\mathcal{L}}$ is a $\mathcal{L}$eft-option of G_2. The options for right can be defined symmetrically. The disjunctive sum provides $\mathfrak{G}$ with a structure of monoid. The neutral element is 0, the game with no option available for either $\mathcal{L}$eft or $\mathcal{R}$ight.

The disjunctive sum appears naturally, and can be seen in the examples we gave in Section 7.1.2. For example, the case of NIM played on several heaps of tokens is a disjunctive sum, where each component of the sum is NIM played on one single heap. Indeed, at each round, players have to choose one of the heap, and make a NIM move on this heap. This decomposition of heap games on several heaps as a sum on games on one heap holds in general for any game played on heaps provided that: (i) each move of the players affects one single heap, (ii) the moves available on one heap do not depend on the number of tokens in the other heaps.

Similarly, for positional games such as DOMINEERING or CRAM, if at some point during the game the grid is separated into several connected components, then the game can be decomposed into the disjunctive sum on each of the connected components as in Figure 7.6. This decomposition into disjunctive sums of games is particularly interesting for computing the outcome of end-game position for various games as these positions often tend to decompose into several components.

The main idea behind game values is to try to answer the following question: "Can we compute the outcome of a game compound, only by knowing the outcomes of its individual components?".

In other words, we would like to be able to decide which of the player has a winning strategy on a compound game, only by considering each of the component independently. When the compound operation is the disjunctive sum, in general, it is not possible to give the outcome of a disjunctive sum, knowing only the outcome of the components. For example, the outcome of NIM played on one heap of size 1 or 2 is $\mathcal{N}$, but the outcome on two heaps (the disjoint sum of each individual heap) is $\mathcal{N}$ on $(1,2)$, but $\mathcal{P}$ on $(1,1)$. There are some cases however, where the outcome of a disjunctive sum can be decided from the outcome of its components.

Lemma 76. *Let G_1 and G_2 be two games, with $o(G_1) = \mathcal{P}$, then $o(G_1 + G_2) = o(G_2)$.*

Proof. Assume that one of the player has a winning strategy on G_2 playing first (the case where he plays second is similar). Then, this player can start by playing according to his strategy on G_2. Then, whenever his opponent plays on G_2, he continues with his strategy on G_2. If his opponent plays on G_1, then he can answer by applying the second player's strategy on G_1. In any cases, this player always has an available move, and his opponent will eventually lose the game. The argument when the player has a winning strategy playing second is similar. □

If this lemma allows us to get the outcome of the compound in a special case, this is not always possible in general. The notion of game value is a refinement of the outcome, and attempts to solve this problem. The main idea behind game values is that sometimes, some components in a sum can be replaced by more simple games. More precisely, given two games G_1 and G_2, they are said to be *equivalent*, and we write $G_1 \equiv G_2$ if: equivalent

$$\forall X \in \mathfrak{G},\ o(G_1 + X) = o(G_2 + X)\ .$$

In other words, two games are equivalent if we can replace one by the other in any disjunctive sum without changing the outcome. This relation is an equivalence relation (i.e., it is symmetric, reflexive, and transitive). The equivalence classes of this relation are the *game values*. Note that, game value
although the quantification is over all possible games $\mathfrak{G}$, which is infinite, it was shown in [BCG82], that given a game G, it is possible to compute a canonical representative for the equivalence class containing G. Moreover, this can be done in time which is polynomial in the size of the game tree of G. In the following, the canonical form of G will denote this particular representative of the equivalence class containing G. A lot more could be said about canonical forms, and their algebraic properties. The reader can refer to [Sie13] for example for an extensive overview of these results. We will focus here on the case of impartial games.

7.2.1 Values of impartial games

In the case of impartial games, the canonical forms are in fact quite simple. More precisely, the following theorem was proved by Sprague and Grundy independently [Spr35, Gru39]:

Theorem 77 (Sprague-Grundy Theorem)**.** *For any impartial game G, there exists a unique $n \geq 0$ such that $G \equiv *n$.*

Recall that $*n$ denotes the game of NIM played on a single heap of size n. We already know from the solution for NIM (see Theorem 73) that $*i$ and $*j$ are not equivalent if $i \neq j$. The theorem above states that any impartial game is equivalent to a Nim heap of a certain size. The size of the heap G is equivalent to is called the Sprague-Grundy value of G, or $\mathcal{SG}$-value for short, and will be denoted $\mathcal{SG}(G)$. When $G = (g)_{\mathcal{R}}$ for some position g and ruleset $\mathcal{R}$, we will denote $\mathcal{SG}_{\mathcal{R}}(g)$ the Grundy value of $(g)_{\mathcal{R}}$. The $\mathcal{SG}$-value satisfies the following properties:

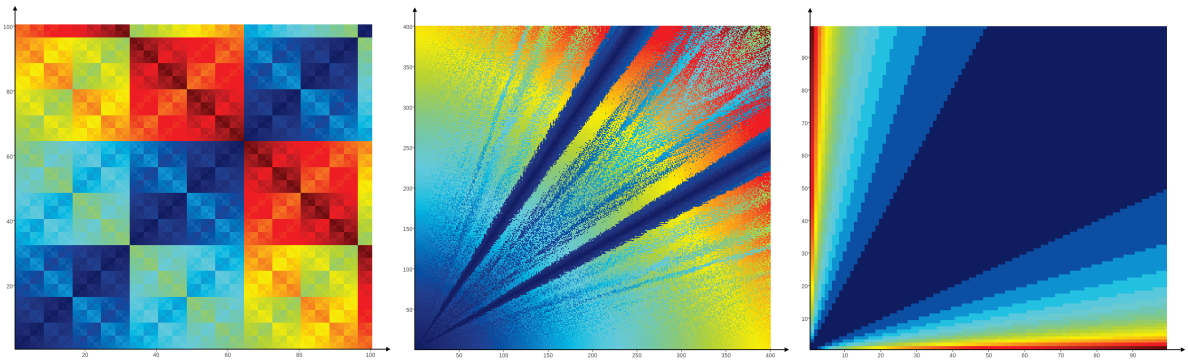
Theorem 78. *For any impartial games G, G_1 and G_2, we have*

- $o(G) = \mathcal{P}$ *if and only if* $\mathcal{SG}(G) = 0$,
- $\mathcal{SG}(G_1 + G_2) = \mathcal{SG}(G_1) \oplus \mathcal{SG}(G_2)$,
- $\mathcal{SG}(G) = \text{mex}(\{\mathcal{SG}(G'),\ G' \text{ option of } G\})$,

where in the last item, $\text{mex}(S)$ *denotes the smallest non-negative integer which is not an element of* S.

The first item only states that the $\mathcal{SG}$-value is a refinement of the outcome of a game. The second item states that $\mathcal{SG}$-values indeed solve the problem they were built for: studying a disjunctive sum by only looking at the components independently. Finally, the third item describes explicitly how the $\mathcal{SG}$-value of a game can be computed if we know the $\mathcal{SG}$-values of the options of the game. In particular, the third item implies that the $\mathcal{SG}$-value of a game can be computed in time which is linear in the size of its game tree. As an example, the $\mathcal{SG}$-values of several games are shown in Figure 7.7. The $\mathcal{SG}$-values for Nim and Euclid are completely characterized. On the contrary, the $\mathcal{SG}$-values for Wythoff appear to be very chaotic, and no characterization is known. A polynomial time algorithm for computing the positions of value 1 was devised in [BF90]. This algorithm was improved in [Niv05] for finding values with position g for some constant g. However, both algorithms rely on some (unproved) conjectures on these positions.

Remark 2. As we have said in the introduction, all this theory works for the normal convention when the last to make a move wins. In the misère convention, things are significantly more difficult: there are many more different canonical forms and fewer games are equivalent, even in the impartial setting. Thus, using values for misère games has proved more complicated than in the normal convention, and a certain number of techniques have been devised to get around these difficulties [Pla05, Pla06, PS08, Sie15].



(a) (b) (c)

Figure 7.7: $\mathcal{SG}$-values for two different ruleset. The point at position x, y on the picture represents the $\mathcal{SG}$-value on position (x, y). Higher $\mathcal{SG}$-values are shown in red, and smaller ones are shown in blue. The two rulesets are: (a) Nim played on two heaps, (b) Wythoff, and (c) Euclid.

7.3 Subtraction Games

We now give more details to a certain class of games called Subtraction *games*. Subtraction games are played with one heap of tokens. The ruleset for a subtraction game is parametrized by a (possibly infinite) set S which provides the available moves. At their respective turns, the players must remove a number x of tokens from the heap, with $x \in S$. The game ends when the number of tokens is strictly smaller than $\min(S)$. We denote by Subtraction(S) the ruleset for the subtraction game with set S. For example, Nim played on a single heap is exactly equivalent to Subtraction$(\mathbb{N}^+)$. In the following, we will assume that S is finite, unless mentioned otherwise.

Subtraction games

Subtraction games are interesting in part because they form some of the simplest possible games. Despite this simplicity, there are only few results known about them. Given a subtraction ruleset $\mathcal{R}_0$, the sequence $(\mathcal{SG}_{\mathcal{R}_0}(n))_{n\geq 0}$ is called the $\mathcal{SG}$*-sequence* of the game. For subtraction games, this sequence is known to be ultimately periodic. Since $\mathcal{SG}$-values are a refinement of outcomes, this implies that the sequence of outcomes is ultimately periodic.

$\mathcal{SG}$-sequence

Theorem 79. *Let S be a finite set. The $\mathcal{SG}$-sequence of* Subtraction(S) *is ultimately periodic with period and preperiod at most* $(1 + |S|)^{\max(S)}$.

Proof. Let S be a finite set of positive integers. The proof of periodicity follows from two facts. First, the $\mathcal{SG}$-values of positions for Subtraction(S) are upper bounded by $|S|$. This follows immediately from the characterisation of $\mathcal{SG}$-values using the mex (the third item of Theorem 78) and the fact that $\text{mex}(X) \leq |X|$ for any subset X. The second point is the simple observation that the $\mathcal{SG}$-value of a position n only depends on the $\mathcal{SG}$-values of all the positions $n - x$ for $0 < x \leq \max(S)$.

Hence it follows that if there are $n_0 \geq 0$ and $p > 0$ such that $n_0 + x$ and $n_0 + p + x$ have all the same $\mathcal{SG}$-value for $0 \leq x < \max(S)$, then for all $n \geq n_0$, n and $n + p$ also have the same $\mathcal{SG}$-values, and the sequence is periodic with period at most p, and perperiod at most n_0. Additionally, there are at most $(1+|S|)^{\max(S)}$ sequences of length $\max(S)$ of non-negative integers smaller or equal than $|S|$. Hence, if we look at the sub-sequences of $\mathcal{SG}$-values for the positions $n_0, \dots n_0 + \max(S) - 1$, for all possible choices of $n_0 \leq (1 + |S|)^{\max(S)}$, then two of these sub-sequences are equal, and the result follows. □

Note that it is very unclear how far from the truth the upper bound on the length of the period and preperiod is. The upper bound in the theorem above can be slightly improved using more complicated arguments [San10]. It was shown in [ANW07] that the grundy sequence is purely periodic with a period of linear length if $|S| = 2$. Moreover, it was conjectured by Guy [GN96] that a tighter upper bound on the period length could be $O(\max(S)^{\binom{|S|}{2}})$, in particular, polynomial in $\max(S)$ if the subtraction set contains a bounded number of elements. Examples of subtraction games with $|S| = 3$ and quadratic period (in terms of $\max(S)$) were discovered in [ANW07, p. 529], and with $|S| = 4$ and period of cubic length in [AB95]. Subtraction games with $|S| = 3$ were studied in [Ho12a], and some conjecture on the length of their period were made in [War16].

Open Problem 16. Investigate the length of the period and preperiod of subtraction games.

A version of subtraction games played on graphs was introduced in [BCD+18], and studied for simple graphs such as subdivided stars. Subtraction games are in fact a particular case of a more general family of rulesets called octal games. In octal games, in addition to removing a certain

number of tokens, the players are allowed to split the remaining tokens from the heap into two heaps. The exact rules depend on a parameter represented by a string of numbers between 0 and 7 (hence the name octal game). A famous question on these games is whether all octal games have an ultimately periodic $\mathcal{SG}$-sequence. If this periodicity was shown for some octal games, the behaviour of the $\mathcal{SG}$-sequence of others remain elusive, despite a very large number of computed values (see a list of such results in [Fla00]).

Finally, this subtraction games can be considered from a complexity point of view. Due to Theorem 79, if the subtraction set is fixed, deciding the outcome of a given position can be done in linear time. Indeed, since the $\mathcal{SG}$-sequence, and consequently the outcome-sequence, is ultimately periodic, this period can be hard-coded into the algorithm, and used to determine the outcome of an arbitrary position.

However, when the subtraction set is not fixed, but part of the input as well, things are very different. Consider the problem where we are given as input a subtraction set S, together with a starting position n, and ask which of the player has a winning strategy for Subtraction(S) starting on n. No hardness result is known for the problem. Moreover, it is not even clear that the problem is in **PSPACE**. Indeed, since at each round, the players might only remove a small number of tokens, the total number of rounds in a play might not be polynomial. Due to this, the only upper bound on the complexity we can give is that the problem is in **EXPTIME**.

Open Problem 17. Study the complexity of deciding the outcome of a position for subtraction games when the subtraction set is part of the input.

7.4 Compounds of games

The notion of compounds of games follows from a natural approach of studying games by splitting them into smaller, more manageable, components and analysing each component individually. Two kinds of compounds have appeared in the literature: compounds which combine individual games, such as the disjunctive sum we saw above; and compounds which combine rulesets. Concerning compounds combining games, the disjunctive sum is by far the most studied kind of compound, but other operations have been considered in the literature. One such example is the *ordinal sum* operator [CNS18], which behaves like a disjunctive sum, with the additional constraint that any move in G annihilates the game H. Another example is the *sequential compound*. Given two games two games G and H, their sequential compound $G \to H$ consists in starting by playing G, and when it is finished continue with H. It was studied in [SU93] where it was shown to be related to misère play. This construction was considered in conjunction to the disjunctive sums, and properties of games of the form $(*n + *m) \to H$ have been investigated in [ACM10a, ACM14]. The sequential compound is also related to the game Euclid [Len03].

ordinal sum

sequential compound

There are many result which can be viewed as studying a variation on the disjunctive sum. Six different compounds were considered in [Con00] (twelve if we account for the choice of normal- or misère-play convention), including the disjunctive sums. For some of these constructions, the outcome of the compound, for impartial games, can be decided by only looking at the outcome of the components. For others, quantities called *remoteness* and *suspense* were defined and play a similar role as the $\mathcal{SG}$-values for the disjunctive sum. These different compounds were studied for the game Node-Kayles in [GS09]. Several games (including Node-Kayles) were studied on subdivided stars [FT04, BCD+18, FH18]. These games can be described as almost disjunctive sums of each of the branches of the star, where only moves close to the center might affect several

branches at the same time. In [BCD+18] in particular, the game is solved by assigning values to each branch of the star, and computing tables describing how these values combine.

More recently, operators have been considered to combine several rulesets to build new ones. For example, *conjoined rulesets*, which have some similarities with the sequential compound, were introduced in [HN19]. As for the sequential compound, the players start the game according to a first set of rules, and when the game has ended, continue with a second set of rules. The main difference with the sequential compound is that the ending position of the first part of the game is used as a starting position for the second part. In [HN19] this construction was used to create the games Go-Cut and Sno-Go which are compounds of the rulesets NoGo, Cut-Throat and Snort. Other games such as Building Nim [DDHL16], Three Men's Morris, Picaria [LR17] (first place then slide to neighbours) and Nine men's morris are similar to these conjoined rulesets since they are played in several phases, with the ending position of the previous phase used as a starting position for the next one.

Note that sometimes, considering combinations, or variations or rulesets also leads to natural variations of usual operators on games such as the disjunctive sum. One such example is the case of games with a pass, where players can pass once during the game (but not if there are already no other moves available). In this setting, it is natural to consider almost disjunctive sums where the pass move is common to all the components (in other words, the pass move can be played only once, even if there are several components). Games with a pass were considered in [MFL11, CLLW18, HN03] for various games including Octal games and Nim. Note that the the $\mathcal{P}$-positions of Nim with a pass on three heaps are very different from classical Nim. Finding a characterization of these positions is still an open problem. The push-button compound that we consider in the next chapter follows this line of research of studying compounds which almost behave like a disjunctive sum.

Chapter 8

Composition of Combinatorial Games: the Push-Compound

In this chapter, we study operators that combine rulesets of combinatorial games. We introduce a construction called the 'push-compound', and study the composition of several standard rulesets. We also give some general properties of this construction, and consider almost disjoint sums of games related to this compound. Some of the results presented in this chapter were published in [DHLP18].

As we saw in the previous chapter, an important part of Combinatorial Game Theory (CGT) focuses on the study of combinations of individual games. The most famous example is the *disjunctive game sum* operator, and several variations of this compound were defined by Conway in his reference book [Con76], and Berlekamp, Conway, and Guy in [BCG82].

More recently, the so-called *sequential compound* operator was introduced [SU93]. It consists in playing successively the ordered pair of games (G, H), where H starts only when G is exhausted. These constructions are in a sense static: the starting position of the second game is fixed until the first game is exhausted. In this chapter, we study dynamic compounds, where the starting position of the second game depends on the moves that were made in the first game. As a consequence, our construction requires the full rulesets of the two games to build the compound.

compatible rulesets

In order to define such compounds, we will say that two rulesets $\mathcal{R}_1$ and $\mathcal{R}_2$ are *compatible* if they have the same set of positions. Note that the construction also works with the weaker condition that, for any position of $\mathcal{R}_1$ there is a description of how to move in $\mathcal{R}_2$. This is the case for example if there is a function that maps any position of $\mathcal{R}_1$ into a position of $\mathcal{R}_2$. In our case, the two rulesets $\mathcal{R}_1$ and $\mathcal{R}_2$ will always be defined on the same set of positions. Given a compatible pair of rulesets $(\mathcal{R}_1, \mathcal{R}_2)$ and a specially designed *switch procedure* '$\Rightarrow$' (it can be a game in itself or anything else that declares a shift of rules), players start the game $\mathcal{R}_1 \Rightarrow \mathcal{R}_2$ with the rules $\mathcal{R}_1$. At their turn, a player can choose, instead of playing according to $\mathcal{R}_1$, to play in the switch procedure, and when this procedure has terminated, the rules are switched to $\mathcal{R}_2$ (using the current game configuration). Thus, we call this class of operators *switch operators* (or switch procedures). Note that the games we mentioned in Section 7.4, such as Go-Cut and Sno-Go can also be described with this kind of construction, where the switch procedure can be started only when the first game has ended.

In the current chapter, the switch procedure is called the *push-the-button* operator (for short *push operator*). It consists in a button that must be pushed once and only once, by either player.

push operator

Pushing the button counts as a move, hence after a player pushes the button, his opponent makes the next move. The button can be pushed any time, even before playing any move in $\mathcal{R}_1$, or when there is no move left in either game. In particular, if there is no move left in $\mathcal{R}_1$ and the button has not been pushed, then it has to be pushed before playing in $\mathcal{R}_2$ (the only way to invoke $\mathcal{R}_2$ is via the push-the-button move). In all cases, the game always ends according to the rules of $\mathcal{R}_2$. We will concentrate our attention to the combination of impartial rulesets, but the construction could be considered for partizan games as well. In the rest of the chapter, we will assume implicitly that all rulesets are impartial.

We first give in Section 8.1 a formal definition of the push operator. In Section 8.2, we motivate this operator by the resolution of a particular game called ZERUCLID, which is proved to be a push compound of the classical games of two-heaps NIM [Bou01] and EUCLID [CD69]. Moreover we show that the second player's winning positions are similar to those of another classical ruleset, WYTHOFF [Wyt07]. Then, in Section 8.3 we continue by studying various push compounds of these three classical rulesets, as well as a variation of the game DOMINEERING built using this compound. Finally, in Section 8.4 we consider a sum of games which is compatible with the push-compound, i.e., pushing the button affects all the components of the sum at the same time. We describe how to compute canonical forms for this operation, and study sums of a push-variant of subtraction games.

8.1 The switch compound: push-the-button

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two rulsets over the same set of positions Ω. We consider combining $\mathcal{R}_1$ and $\mathcal{R}_2$ to form a new compound ruleset. In the compound, players start the game according to $\mathcal{R}_1$, and at their turn, a player can decide, instead of playing according to $\mathcal{R}_1$, to push a single allocated button, which switches the rules to the second ruleset $\mathcal{R}_2$. The button must be pushed exactly once, at some point during play, by either player. It can be pushed before any move has been made, at some intermediate stage, or even if there is no other move available. We call this operator, the *push operator*, denoted by '$\circledcirc$'. By pushing the button, the current position, which is a position of the first ruleset, becomes the starting position for the second ruleset.

Definition 6 (The push operator)**.** Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two rulesets over the same set of positions Ω. The push-the-button ruleset (for short *push ruleset*) $\mathcal{R}_1 \circledcirc \mathcal{R}_2$ is defined over $\{1,2\} \times \Omega$ by:

$$(\mathcal{R}_1 \circledcirc \mathcal{R}_2): \begin{array}{rcl} (1,g) & \longmapsto & \{(1,g'), g' \in \mathcal{R}_1(g)\} \cup (2,g) \\ (2,g) & \longmapsto & \{(2,g'), g' \in \mathcal{R}_2(g)\}. \end{array}$$

Once the button is pressed, the game is played according to the rules $\mathcal{R}_2$. In other words, we have $(2,g)_{\mathcal{R}_1 \circledcirc \mathcal{R}_2} = (g)_{\mathcal{R}_2}$. Consequently, when the ruleset is of the form $\mathcal{R}_1 \circledcirc \mathcal{R}_2$, the interesting positions are those of the form $(1,g)$ for $g \in \Omega$. Thus, to simplify notation, we write $(g)_{\mathcal{R}_1 \circledcirc \mathcal{R}_2}$ instead of $(1,g)_{\mathcal{R}_1 \circledcirc \mathcal{R}_2}$.

In the following, we will say that a game G is a *push-game* if its ruleset is a push-compound. Since pushing the button is a 'special' move, it makes sense to differentiate it from its other possible moves. Hence, we will denote by $\mathrm{push}(G)$ the game obtained after pushing the button. In other words, if $G = (g)_{\mathcal{R}_1 \circledcirc \mathcal{R}_2}$ for some compatible rulesets $\mathcal{R}_1$ and $\mathcal{R}_2$, then $\mathrm{push}(G) = (2,g)_{\mathcal{R}_1 \circledcirc \mathcal{R}_2} = (g)_{\mathcal{R}_2}$. The *normal options* of G denotes the games which can be obtained by playing according to $\mathcal{R}_1$. Hence, the normal options of G contain all the options of G, except the move which consists in pushing the button.

push-game

normal options of a game

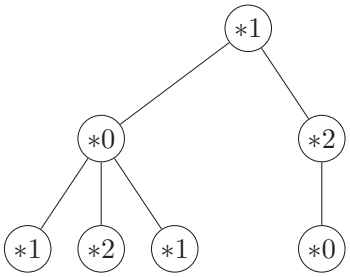


Figure 8.1: Example of the tree representation of a push-game. After the button is pushed, the game continues with the rules of Nim, with a number of tokens which depends on the position before pushing the button. The first player has a winning move by playing on the right branch.

We have seen in Chapter 7 that impartial games can be represented as a tree. In the same manner, we will adapt this tree representation for push-games. Since pushing the button plays a special role, it makes sense to distinguish it from the other moves. Hence, we will represent a push-game by a labelled rooted tree. Each node of the tree represents a position of the game, and its children corresponds to the normal moves from this position. Each node is labelled by the game which will follow if we push the button from this position. An example of a tree representation of a push-game is given in Figure 8.1, where after pushing the button the rules are those of Nim. We denote by $\mathfrak{G}^{\circledcirc}$ the set of all push-games (equivalently, trees with nodes labelled by impartial games).

8.1.1 General properties of the push compound

Before studying concrete examples of push-compounds, we state some general properties of the push operator. If $G = (g)_{\mathcal{R}_1 \circledcirc \mathcal{R}_2}$ is a push game, the possible options of G are either pushing the button, or playing according to the rules $\mathcal{R}_1$. Hence, it follows that the outcome of G is $\mathcal{P}$ if and only if pushing the button is a losing move, i.e., $o_{\mathcal{R}_1}(g) = \mathcal{N}$, and or every option $g' \in \mathcal{R}_1(g)$, $o_{\mathcal{R}_1 \circledcirc \mathcal{R}_2}(g') = \mathcal{N}$. In particular, with this observation we can reformulate Lemma 72 to characterize the $\mathcal{P}$-positions for push-rulesets.

Lemma 80. *Let $\mathcal{R}_1 \circledcirc \mathcal{R}_2$ be a push ruleset over Ω. A subset $P \subset \Omega$ is the set of $\mathcal{P}$-positions of $\mathcal{R}_1 \circledcirc \mathcal{R}_2$ if and only if:*

(i) $\forall g \in P,\ o_{\mathcal{R}_2}(g) = \mathcal{N}$,

(ii) $\forall g \in P,\ \mathcal{R}_1(g) \subset \Omega \setminus P$,

(iii) $\forall g \in \Omega \setminus P$, *either* $o_{\mathcal{R}_2}(g) = \mathcal{P}$ *or* $\mathcal{R}_1(g) \cap P \neq \emptyset$.

Proof. Obvious, according to Proposition 72 and the definition of push-ruleset. □

In some cases, the $\mathcal{P}$-positions of $\mathcal{R}_1 \circledcirc \mathcal{R}_2$ can be determined directly from the $\mathcal{P}$-positions of $\mathcal{R}_1$. Indeed, adding the possibility to change the rules can be seen as a way to disrupt the game $\mathcal{R}_1$ using an additional move (pushing the button). The following results give sufficient conditions for which such a disruption preserves the $\mathcal{P}$-positions of $\mathcal{R}_1$.

Proposition 81. *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two rulesets over Ω and let g in Ω be a game position. We denote by P_1, P_2 and P_1^- the set of $\mathcal{P}$-positions for respectively $\mathcal{R}_1$ and $\mathcal{R}_2$ under normal-play convention, and $\mathcal{R}_1$ under misère-play convention.*

The ruleset $\mathcal{R}_1 \circledcirc \mathcal{R}_2$ satisfies the following properties:

(i) If $o_{\mathcal{R}_2}(g) = \mathcal{P}$, then $o_{\mathcal{R}_1 \circledcirc \mathcal{R}_2}(g) = \mathcal{N}$.

(ii) If $P_1 \cap P_2 = \emptyset$, then $o_{\mathcal{R}_1 \circledcirc \mathcal{R}_2}(g) = \mathcal{P} \iff g \in P_1$.

(iii) If $P_1^- \cap P_2 = \emptyset$ and $\mathcal{R}_1(g) = \emptyset$ implies $g \in P_2$, then $o_{\mathcal{R}_1 \circledcirc \mathcal{R}_2}(g) = \mathcal{P} \iff g \in P_1^-$.

Proof. (i) Pushing the button is a winning move.

(ii) We check that P_1 satisfies the three conditions $(i), (ii)$ and (iii) of Lemma 80. The condition (i): for all $g \in P_1$, $o_{\mathcal{R}_2}(g) = \mathcal{N}$, corresponds to the assumption we made that $P_1 \cap P_2 = \emptyset$. Since P_1 is the set of $\mathcal{P}$-positions for the ruleset $\mathcal{R}_1$, there is no move, according to $\mathcal{R}_1$, from a position in P_1 to another one, hence (ii) holds. Moreover for any position not in P_1, there is a move, according to $\mathcal{R}_1$, to a position in P_1 and (iii) holds.

(iii) The argument is essentially the same as above. If a position g has no move for $\mathcal{R}_1$, then by hypothesis, pushing the button is a winning move. Otherwise, g has at least one option. If $g \in P_1^-$, then all moves, according to $\mathcal{R}_1$, are out of P_1^-, and by hypothesis, pushing the button is a losing move. If $g \notin P_1^-$, then there exists at least one move, according to $\mathcal{R}_1$, landing in $g' \in P_1^-$.

□

This result gives sufficient conditions for which the $\mathcal{P}$-positions of $\mathcal{R}_1$ remain unchanged by adding the possibility to switch the rules to $\mathcal{R}_2$. For some of the games we study in Section 8.3, we will use this property to characterize their set of $\mathcal{P}$-positions.

8.2 Zeruclid

We now present the ruleset which is the source of the push compound, and motivated the initial research on this construction. It is called Zeruclid (a.k.a. Ruthlein), and is a variant of the ruleset Susen [Mul16]. Both are part of a larger family of non-invariant rulesets [DR10].

Definition 7 (Zeruclid)**.** The game Zeruclid is played on several heaps of tokens. At each turn, a player can remove from any heap a number of tokens which is a positive multiple of the smallest non-zero heap. The heap sizes must remain non-negative.

Zeruclid on two heaps is very similar to the well-known game Euclid. Recall from Section 7.1.2 that in Euclid, the players can remove a positive multiple of the smallest heap from the largest heap. On two heaps, Zeruclid has only two differences with Euclid:

- In Zeruclid, the game ends when both heaps are zero, whereas in Euclid it ends when they are equal.
- In Zeruclid, it is possible to remove the smallest heap in a single move.

One can easily check that these two differences do not modify the $\mathcal{P}$-positions. Indeed, in the 2-heap case, removing the smallest heap is always a losing move in ZERUCLID. Hence the $\mathcal{P}$-positions of 2-heap ZERUCLID with non-zero coordinates, and the $\mathcal{P}$-positions of EUCLID are the same. Using the known results on EUCLID, these are the positions (a, b) such that $\frac{1}{\Phi} < \frac{b}{a} < \Phi$, where Φ is the golden ratio (see Theorem 75).

Played on three heaps, our game is similar to the game 3-EUCLID introduced in [CL08], and to some of its variations studied in [Ho12b]. The main difference with the game ZERUCLID is that 3-EUCLID requires the heap sizes to remain positive. In particular, the number of heaps in 3-EUCLID does not vary during the game. In our version, it is allowed to completely remove the smallest heap, and thus to decrease the total number of heaps. An example is given in Figure 8.2 describing the possible move from the position $(2, 3, 5)$.

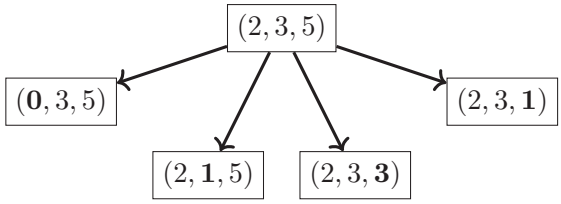


Figure 8.2: Example of the possible moves for ZERUCLID on position $(2, 3, 5)$.

8.2.1 ZERUCLID$(1, a, b)$

We call ZERUCLID$(1, a, b)$ the game ZERUCLID played on three heaps of tokens where the smallest heap has size 1. The link between this game and the push-the-button operator is given in the following proposition:

Proposition 82. *The game* ZERUCLID *played on position* $(1, a, b)$ *has the same outcome as the push ruleset* NIM $\circledcirc$ EUCLID *on position* (a, b).

First, remark that EUCLID and NIM are not played exactly on the same set of positions: the position $(0, i)$ is not a valid position for EUCLID, while it is a valid position for NIM. Consequently, what we call EUCLID here, and in all the rest of Section 8.2, is the variant where there is a single move from $(0, i)$ to $(0, 0)$ if $i > 0$. In particular, the position $(0, i)$ is an $\mathcal{N}$-position.

Proof. Before the heap with 1 token is taken, it is possible to remove any number of tokens from any of the two other heaps, hence the rules are those of NIM on two heaps. Once the heap with size one is removed, the rules are essentially those of EUCLID, up to some minor differences, which, as mentioned before, do not modify the set of $\mathcal{P}$-positions. Removing the heap of size one in ZERUCLID is equivalent to pushing the button in NIM $\circledcirc$ EUCLID. □

The rest of this section will be dedicated to finding characterizations of the $\mathcal{P}$-positions of NIM $\circledcirc$ EUCLID. By the previous proposition, this also gives a characterization of the $\mathcal{P}$-positions for ZERUCLID$(1, a, b)$.

Surprisingly, the $\mathcal{P}$-positions of NIM$\circledcirc$EUCLID are very similar to the $\mathcal{P}$-positions of WYTHOFF's game. In the following, we denote $a_n = \lfloor \Phi n \rfloor$ and $b_n = \lfloor \Phi n \rfloor + n$. Recall from Theorem 74 that the $\mathcal{P}$-positions for Wythoff are exactly the (a_n, b_n) and (b_n, a_n) for $n \geq 0$. The pairs (a_n, b_n) for $n \geq 0$ are generally called WYTHOFF *pairs*.

WYTHOFF pairs

The characterization of the $\mathcal{P}$-positions of NIM $\circledcirc$ EUCLID is based on the WYTHOFF pairs, as well as a sequence related to the Fibonacci numbers and defined as follows.

Definition 8. Denote by $(F_n)_{n\geq 0}$ the Fibonacci sequence starting with $F_0 = 0$ and $F_1 = 1$. The sequence $(u_n)_{n\geq 0}$ is defined by $u_n = F_{n+1} - 1$.

Before showing the theorem which gives a characterization of the $\mathcal{P}$-positions of NIM $\circledcirc$ EUCLID in terms of the sequences a_n, b_n and u_n, we need the following result.

Proposition 83. *For all $n \geq 0$, the pair (u_{2n}, u_{2n+1}) is a* WYTHOFF *pair.*

This is a well-known result. Since we will reuse the proof afterwards, we put the details of it below.

Proof. Let Φ denote the golden ratio. Using the closed form formula for the Fibonacci numbers, we can write that:

$$\begin{aligned} F_{n+1} - \Phi F_n &= \frac{\Phi^{n+1} - (-\Phi)^{-n-1}}{2\Phi - 1} - \Phi\frac{\Phi^n - (-\Phi)^{-n}}{2\Phi - 1} \\ &= \frac{1}{(-\Phi)^n}\frac{(\Phi - \frac{1}{\Phi})}{2\Phi - 1} = \frac{1}{(-\Phi)^n(2\Phi - 1)}. \end{aligned}$$

This gives $F_{2n+1} - \frac{1}{\Phi^{2n}(2\Phi-1)} = \Phi F_{2n}$, and by taking the integer part on both sides of the equality, we obtain $u_{2n} = \lfloor \Phi F_{2n} \rfloor$. Additionally we have

$$u_{2n+1} = F_{2n+2} - 1 = F_{2n+1} - 1 + F_{2n} = \lfloor \Phi F_{2n} \rfloor + F_{2n} = \lfloor \Phi^2 F_{2n} \rfloor.$$

Consequently, the equality $(u_{2n}, u_{2n+1}) = (\lfloor \Phi F_{2n} \rfloor, \lfloor \Phi^2 F_{2n} \rfloor)$ proves that it is a WYTHOFF pair. □

We now have all the ingredients we need to characterize the set of $\mathcal{P}$-positions of NIM $\circledcirc$ EUCLID. As we can see in the theorem below, this set coincides with the $\mathcal{P}$-positions of WYTHOFF, except for a very small fraction of the positions. The first $\mathcal{P}$-positions of the two games are given in Table 8.3.

Theorem 84. *The set of $\mathcal{P}$-positions of* NIM $\circledcirc$ EUCLID *is given by:*

$$P = \{(a_n, b_n), n \geq 0\} \setminus \{(u_{2n}, u_{2n+1}),\ n \geq 0\} \bigcup \{(u_{2n+1}, u_{2n+2}),\ n \geq 0\}.$$

Proof. Denote by (a'_n, b'_n) the positions of the set P above, with $b'_n \geq a'_n$, reordered by increasing a'_n. Denote by A and B the two sets defined by $A = \{a'_n,\ n \geq 0\}$ and $B = \{b'_n,\ n \geq 0\}$. We know from [Wyt07] that the sets $\{a_n,\ n \geq 0\}$ and $\{b_n,\ n \geq 0\}$ are complementary. Additionally, using Proposition 83 and the fact that the u_i for $i \geq 1$ are all distinct, we can deduce that A and B are complementary. This implies that there is no move, according to NIM from a position of P to another position of P. Indeed, if there were a move between two position in P, then these two positions have one coordinate in common. Since a'_n is strictly increasing, this implies that one of the position is of the form (a'_n, b'_n), while the other is (b'_m, a'_m), and in particular, this contradicts the assumption that A and B are disjoint.

In order to apply Lemma 80, we only need to show that: (i) pushing the button on a position (a'_n, b'_n) is a losing move, and (ii) from any position $(a, b) \notin P$, there is either a move, according to NIM, to a position in P, or pushing the button is a winning move.

Wythoff	(0,0)	(1,2)	(3,5)	(4,7)	(6,10)	(8,13)	(9,15)	(11,18)	(12,20)	(14,23)	(16,26)	(17,28)
Nim ⊙ Euclid	(0,1)	(2,4)	(3,5)	(6,10)	(7,12)	(8,13)	(9,15)	(11,18)	(14,23)	(16,26)	(17,28)	

Table 8.3: Sequence of the first $\mathcal{P}$-positions for the games Wythoff and Nim ⊙ Euclid. Some blanks were inserted to highlight the similarities between the two sequences.

We start by showing that, for $n \geq 1$, the position (a'_n, b'_n) satisfies the equality $b'_n = \lceil \Phi a'_n \rceil$. If (a'_n, b'_n) is a Wythoff pair, this relation is a well-known result (this is proved for example in Lemma 5 from [Fra07] or in [Sil76]). Consequently, we only need to prove it in the case $(a'_n, b'_n) = (u_{2n+1}, u_{2n+2})$. Reusing the computations from the proof of Proposition 83, we know that:

$$\begin{aligned} u_{2n+2} - \Phi u_{2n+1} &= F_{2n+3} - 1 - \Phi F_{2n+2} + \Phi \\ &= \frac{1}{\Phi^{2n+2}(2\Phi - 1)} - 1 + \Phi. \end{aligned}$$

This shows that $u_{2n+2} - \Phi u_{2n+1} \leq \Phi - 1 + \frac{1}{\Phi^2(2\Phi-1)} < 1$ and additionally, $u_{2n+2} - \Phi u_{2n+1} > 0$, which proves that $u_{2n+2} = \lceil \Phi u_{2n+1} \rceil$.

We have $(a'_0, b'_0) = (0, 1)$, and pushing the button from this position is a losing move. If $n \geq 1$, then $\frac{b'_n}{a'_n} > \Phi$, and consequently, by Theorem 75 pushing the button is also a losing move.

Now suppose that we have $(a, b) \notin P$. We want to show that either there is a move, according to Nim, to a position in P, or pushing the button is a winning move. If $a = 0$, then either $b = 0$ and pushing the button is a winning move, or $b > 0$, and there is a move to $(0, 1) \in P$. Consequently, we can assume $a > 0$. Suppose $a = b'_n$ for some n. Thus we have $b \geq a = b'_n > a'_n$, and there is a move, according to Nim, to the position (b'_n, a'_n). Otherwise, since A and B are complementary, we have $a = a'_n$ for some n. If $b > b'_n$, then again, there exists a move, according to Nim, to (a'_n, b'_n). Otherwise, we have $b < b'_n$. Since $b'_n = \lceil \Phi a'_n \rceil$, we must have $\frac{b}{a} < \Phi$, which implies by Theorem 75 that pushing the button is a winning move. □

In [DHLP18], we also give alternative characterizations of the $\mathcal{P}$-positions for Zeruclid. These other results rely on similar characterizations for the $\mathcal{P}$-positions of Wythoff, using for example another numeration system based on Fibonacci numbers. Although the $\mathcal{P}$-positions of Zeruclid with the smallest heap of size 1 are well characterized, the $\mathcal{SG}$-values seem to have a much more complicated structure as shown in Figure 8.4. Note that the patterns which can be seen on the figure seem to have the same shape as those obtained in [ACM10b] about variations of the game Nim. This general shape seems to be

In [DHLP18], we also look at properties of the $\mathcal{P}$-positions when the first heap is larger than 1. However, we do not have a characterization in the general case.

8.3 Other push compounds

In this section we investigate the push operator on several other well-known impartial rulesets. We start by considering rulesets played on two heaps of tokens, and study all the other possible compounds of the rulesets Nim, Euclid, and Wythoff. We give characterizations of the $\mathcal{P}$-positions of the resulting games. Then in Section 8.3.2 we consider push-compounds for positional games, and study a variant of the ruleset Cram.

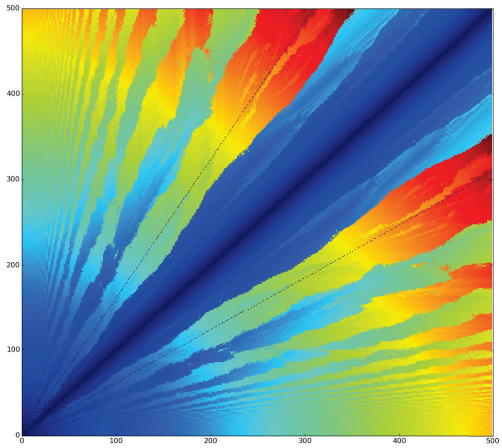


Figure 8.4: $\mathcal{SG}$-values for positions $(1, a, b)$ of 3-heaps ZERUCLIDfor different values of a and b. colour blue corresponds to small values while yellow and red correspond to high values. The $\mathcal{P}$-positions can be seen along the line $y = \Phi x$ and its symmetric. Other small positive $\mathcal{SG}$-values seem to be close to the diagonal $y = x$.

8.3.1 Push compounds of NIM, WYTHOFF and EUCLID

The set of positions of these three games can be represented as the integer lattice $\mathbb{N}^2$, where each coordinates corresponds to the number of tokens in one of the two heaps. Recall that the case NIM ⊙ EUCLID has already been considered in the previous section and corresponds to 3-heap ZERUCLID with the smallest heap of size 1. The three results below cover the other compounds of NIM with EUCLID and WYTHOFF.

Proposition 85. *The $\mathcal{P}$-positions of* NIM ⊙ WYTHOFF *is the set $P_{N\odot W}$ defined as:*

$$P_{N\odot W} = \{(0,1);(1,0);(k,k),\ k \geq 2\}.$$

Proof. The set $P_{N\odot W}$ is known as the set of $\mathcal{P}$-positions of NIM under misère convention [Bou01]. Using Proposition 81 (iii), one only needs to show that:

- If g is a position with no option for NIM, then it is a $\mathcal{P}$-position for WYTHOFF. This is immediate because the only position with no option for NIM is $(0, 0)$.
- All the positions of $P_{N\odot W}$ are winning for WYTHOFF. This also holds since the $\mathcal{P}$-positions of WYTHOFF are of the form $(\lfloor \Phi n \rfloor, \lfloor \Phi n \rfloor + n)$ (see Theorem 74), and none of these positions is in $P_{N\odot W}$.

□

Proposition 86. *A position (a, b) with $a \leq b$ is a $\mathcal{P}$-positions of* EUCLID ⊙ NIM *if and only if one of the following assumptions holds:*

- $\frac{b}{a} < \Phi$ *and* $\frac{b}{a} \neq \frac{F_{2i+2}}{F_{2i+1}}$ *for any i,*

- $\frac{b}{a} = \frac{F_{2i+1}}{F_{2i}}$ *for some* i,

where the F_i *are the Fibonacci numbers.*

Proof. If we note P the set of positions defined by the union of the two above conditions, then P is the set of $\mathcal{P}$-positions of Euclid in misere convention [Gur07]. To show that this also corresponds to the set of $\mathcal{P}$-positions of Euclid ⊛ Nim, using Property 81, we only need to show that for any position in P, this position is winning for Nim, and that all the terminal positions of Euclid are $\mathcal{P}$-positions of Nim. This is straightforward since the $\mathcal{P}$-positions of Nim are of the form (a, a), and $a/a = 1 = \frac{F_2}{F_1}$, thus $(a, a) \notin P$. In addition, the terminal positions of Euclid are exactly these positions. □

Proposition 87. *The* $\mathcal{P}$*-positions of* Wythoff ⊛ Nim *is the set* $P_{W\circledcirc N}$ *defined as follows:*

$$P_{W\circledcirc N} = \{(a_n - 1, b_n - 1) \quad n \geq 1\}.$$

where (a_n, b_n) *are the* $\mathcal{P}$*-positions of* Wythoff.

This is an example where the $\mathcal{P}$-positions of the compound ruleset differs from the $\mathcal{P}$-positions of the original ruleset (both in normal or misere convention). Therefore, Proposition 81 cannot be applied.

Proof. According to Proposition 74, the sets $A = \{a_n - 1\}_{n\geq 1}$ and $B = \{b_n - 1\}_{n\geq 1}$ are complementary sets and $(a_n - 1) - (b_n - 1) = a_n - b_n = n$. As a consequence, there is no move, according to Wythoff, from one position in $\mathcal{P}_{W\circledcirc N}$ to another. Moreover, since $a_n - b_n = n > 0$ for any $n \geq 1$, none of these elements are $\mathcal{P}$-positions of Nim. Hence, to prove that this set is the set of $\mathcal{P}$-positions of Wythoff ⊛ Nim, it only remains to show that for any position not in $\mathcal{P}_{W\circledcirc N}$, either pushing the button is a winning move, or there is a move according to Wythoff ruleset to a position in $\mathcal{P}_{W\circledcirc N}$. The proof is essentially the same as for the classical Wythoff.

Take a position $(a, b) \notin \mathcal{P}_{W\circledcirc N}$ with $0 \leq a \leq b$. If $a = b$, changing the rules to Nim (by pushing the button) is a winning move. Hence, we can assume that $a < b$. Since A and B are complementary, there are two possibilities:

- $a = b_n - 1$ for some $n \geq 1$, then $b > a = b_n - 1 \geq a_n - 1$. In this case there is a move from (a, b) to $(b_n - 1, a_n - 1)$ by playing on the second heap.
- $a = a_n - 1$ for some $n \geq 1$. If $b > b_n - 1$, then there is a move to $(a_n - 1, b_n - 1)$ by playing on the second heap. Otherwise $b < b_n - 1$, and let $m = b - a$. We have $m < b_n - 1 - a_n - 1 = n$. Consequently, by choosing $k = a_m - a_n$, the move to $(a - k, b - k) = (a_m - 1, b_m - 1)$ is a winning move.

□

8.3.2 Push compounds of placement games: the Cram ruleset

In previous examples, we studied several games played with heaps of tokens. We now move to a different kind of game: placement games. We look in particular at the game Domineering and its impartial variant Cram, and study a push compound of this game. We here study a variation of Cram where the players first play only vertical dominoes, then, when the button is pushed, switch

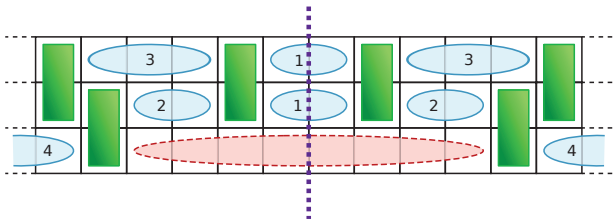


Figure 8.5: Case (*iii*). The blue dotted vertical line is the symmetry axis. All the rows go by pair except one (in red with a dashed border) which has even size. Since 0.07 played on a row of even size has a non-zero $\mathcal{SG}$-value, pushing the button is a losing move on these symmetric positions.

to horizontal ones. Since the two rulesets are defined on the same game board they are compatible. We call Push Cram this new game, and present a solution for some of its non trivial sub-cases.

Note that Push Cram is a special case of a more general construction which consists in building a push-ruleset from a partizan game. More precisely, given an arbitrary partizan game, we can consider a push-variant of the game where the players start with the rules for $\mathcal{L}$eft and then, after pressing the button continue with the rules for $\mathcal{R}$ight. In a similar way, the push compound of Subtraction games that we will consider in Section 8.4.3 can be obtained by applying this construction to the Partizan Subtraction games we study in Chapter 9.

In this subsection, the notation $m \times n$ will denote the grid with m rows and n columns. The game of Cram played on a $1 \times n$ row has the same rules as the octal game 0.07. Its $\mathcal{SG}$-values are periodic, with period 34 and pre-period 52 [BCG04]. In particular, if we have $\mathcal{SG}_{0.07}(n) = 0$, then n is odd (i.e.: positions $1 \times (2k)$ have outcome $\mathcal{N}$, see Section 7.1.2). In Push Cram on an $m \times n$ board, when the button is pushed the game decomposes into a disjunctive sum of m distinct 0.07 positions. Indeed, after the button is pushed the players can only place horizontal dominoes, and each domino placed on a row does not affect the other rows. In this case, the outcome can be computed easily using the properties of the $\mathcal{SG}$-values (see Theorem 78) and the periodicity of the $\mathcal{SG}$-values of 0.07.

Proposition 88. *Let k and n be two non-negative integers. We have the following outcomes for* Push Cram*:*

(i) $(2k) \times n$ *is an $\mathcal{N}$-position,*

(ii) $(2k+1) \times n$ *is an $\mathcal{N}$-position if* $\mathcal{SG}_{0.07}(n) = 0$,

(iii) $3 \times (2k)$ *is a $\mathcal{P}$-position,*

(iv) $n \times 3$ *is a $\mathcal{P}$-position if and only if* $\mathcal{SG}_{0.07}(n) = 0$,

(v) $(2k+1) \times 4$ *is a $\mathcal{P}$-position.*

Proof. For each of these cases, we describe a winning strategy for one of the players.

(*i*) The first player directly wins by pushing the button. Indeed, once the button is pushed, the game decomposes into a disjunctive sum of $2k$ games, each game corresponding to a row. Since all rows have the same length, the value (obtained by the XOR of the values of each game) is zero.

(*ii*) Again, the first player can push the button and win since once the button is pushed, the games corresponding to each row have $\mathcal{SG}$-value 0.

(*iii*) We give a strategy for the second player: play the move symmetrically to the first player relatively to the vertical line cutting the grid in half. Since such a move is always possible, we

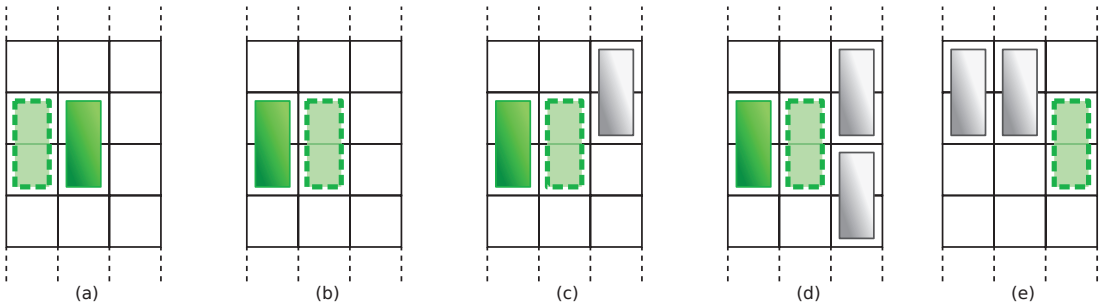


Figure 8.6: Subcases $(a) - (d)$ consist in playing symmetric moves on the first two columns. (e) is the case where the player plays on the last column. Grey tiles correspond to previous moves. The green tile is the last move played by the opponent, and the dotted one is the move that the player wants to play by applying the strategy.

only have to show that for the first player pushing the button is always a losing move. Each move of the players cuts the horizontal rows. Looking at the connected horizontal pieces of rows, they go in pairs with their symmetric except for three of them in the center which are their own symmetric. Among these three, two of them have the same length and can be paired together, and the remaining one has even length (see Figure 8.5). Moreover, since 0.07 played on a row of even length has a value different from zero, pushing the button on this kind of symmetric position is a losing move for the first player.

(iv) The second player can play his optimal strategy for 0.07 on the last column and play symmetrically on the two other ones. At some point, the opponent will run out of moves and will be forced to push the button, giving a winning position to the second player (see Figure 8.6). At any point during the game, when it's the first player's turn, there is an odd number of rows with 2 or 3 empty cells. Indeed, if the first player plays on one of the two first columns, by playing symmetrically, two rows with either 2 or 3 empty cells are transformed into rows with 0 or 1 empty cells. Additionally, playing into the last column changes the number of free cells of the two rows from 3 to 2 or from 1 to 0. After pushing the button, the value of the game is the number modulo 2 of rows with 2 or 3 empty cells. Consequently, for the first player, pushing the button is a losing move.

(v) Group the columns in two consecutive pairs as in Figure 8.7. Either the second player can push the button and win, or, by playing the symmetric in the paired column, the value of the game for 0.07 does not change. If he plays on a side column (case (a) in Figure 8.7), his move removes two rows of size 1, and does not change the value for 0.07 on these rows. If he plays on a middle column (case (b) in Figure 8.7), then he changes rows of length 1 to 0, and rows of length 3 to 2. Since $\mathcal{SG}_{0.07}(1) = \mathcal{SG}_{0.07}(0)$ and $\mathcal{SG}_{0.07}(3) = \mathcal{SG}_{0.07}(2)$, the value for 0.07 is not affected either by the move.

□

With the current tools, it seems difficult to completely characterize the outcome for Push Cram on any board size. However, using numeric computations, we were able to formulate the following conjecture. A game is a *bluff* game [BDKK17] if it is a first player win, and any first move is a winning move.

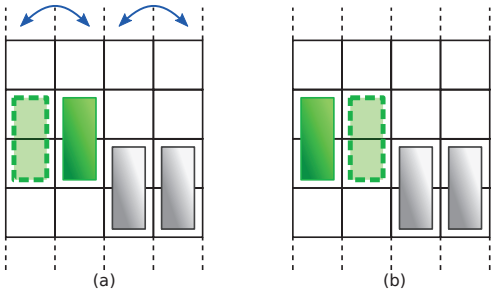


Figure 8.7: The second player can play symmetrically to the other player on each pair of column.

Conjecture 4. For all $k \geq 0$, the grid of size $3 \times (2k+1)$ is an $\mathcal{N}$-position for the game Push-Cram. In addition, it is a bluff game.

This conjecture was verified up to grids of size 3×25 for the outcome, and 3×13 for the bluff property. The remaining unsolved case for Push Cram on rectangular grids is when there is an odd number of rows, and a number of columns m such that $\mathcal{SG}_{0.07}(m) \neq 0$. The complexity of computing the outcome of an arbitrary position (with already placed dominoes) is also unknown.

For Cram, when the board is not connected, we can split the game into the disjunctive sum of each connected component, and consider each component independently. By computing the $\mathcal{SG}$-values of each component, we can deduce the outcome for the whole game. In the push-variant of the game, we would like to apply a similar argument. However, for this variant, when the board is not connected the game is not a disjunctive sum. Indeed, if one player pushes the button, the rules change for all components at the same time, and not just for one of the components. Hence, in the next section we consider an alternative version of the disjunctive sum for push-games.

8.4 Push-sums and push-canonical forms

Given two push-games G and H, we define the *push-sum* of these two games, denoted $G \oplus H$, as the game where the players can either play in G or in H, but pushing the button changes the rules for both components at the same time. More formally, the sum $G \oplus H$ is such that $\text{push}(G \oplus H) = \text{push}(G) + \text{push}(H)$ (i.e., after pushing the button, we get a disjunctive sum), and the normal options of $G \oplus H$ are exactly the games of the form: push-sum

- $G' \oplus H$, where G' is a normal option of G,
- $G \oplus H'$, where H' is a normal option of H.

As we did in Section 7.2 when we introduced values for the disjunctive sum, we wish to define values of games such that the value of a push-sum can be computed from the values of the individual components. Hence, we define an equivalence relation between push-games, called the *push-equivalence* and denoted $\overset{\circledcirc}{\equiv}$, where for any two push-games G and H we have: push-equivalence

$$G \overset{\circledcirc}{\equiv} H \iff \forall X \in \mathfrak{G}^{\circledcirc},\ o(G \oplus X) = o(H \oplus X)\ .$$

We call *push-values* the equivalence classes for this relation. If the push-values theoretically solve the problem, this definition is not necessarily practical from a computational point of view. push-values

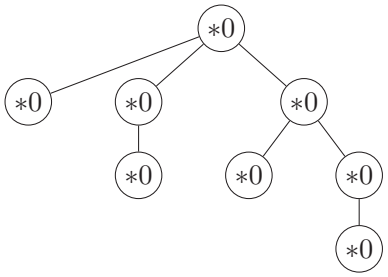


Figure 8.8: Tree representation of the push-game ⊛3.

Indeed, it does not provide an algorithm to compute the push-value of a game since the definition relies on quantifying over the (infinite) set of all possible push-games. The rest of this section is dedicated to proving properties of the push-equivalence which allows us to compute canonical representative for the equivalence class of an arbitrary push-game.

To simplify some statements, we will adopt special notations for some of the games. Given an impartial game G, we denote by $\textcircled{G}$ the push-game where the only possible move is to push the button to create the game G. In other words, the game tree of $\textcircled{G}$ contains a single node labelled by G. For example, if Empty denotes the ruleset where the players have no move on any position, then $\textcircled{*n}$ corresponds to Empty $\circledcirc$ Nim on a heap of size n. For this game, the only possible move is pushing the button. After this, the players continue with the rules of Nim on a heap of size n. In particular, $\textcircled{0}$ is a neutral element for the push-sum: $\forall X \in \mathfrak{G}^{\circledcirc}$, $X \oplus \textcircled{0} = X$. Indeed, there is no move in the game $\textcircled{0}$ both before and after pushing the button. We also denote by $\circledast n$ the game with rules Nim $\circledcirc$ Empty played on a heap of size n. This game is played by starting with the rules of Nim, and at any time the players can push the button and win. The game tree of $\circledast 3$ is illustrated in Figure 8.8.

8.4.1 Push-values

Note that when considering the push-value of a game G, the exact shape of $\mathrm{push}(G)$, the game obtained after pushing the button on G, does not matter, but only its $\mathcal{SG}$-value is important. As such, we will assume in the rest of this section that for every push-game G that we consider, $\mathrm{push}(G)$ is in canonical form, i.e., $\mathrm{push}(G) = *i$ for some $i \geq 0$.

Before showing how push-canonical forms can be computed, we need a few technical lemmas. The first of these lemmas, shown below gives some criterion to decide which games are push-equivalent to $\textcircled{0}$.

Lemma 89. *Let G be a push-game, then $G \overset{\circledcirc}{\equiv} \textcircled{0}$ if and only if:*

- *$o(\mathrm{push}(G)) = \mathcal{P}$, i.e., pushing the button on G is a winning move,*
- *and for every normal option G' of G, there exists a normal option G'' of G' such that $G'' \overset{\circledcirc}{\equiv} \textcircled{0}$.*

Proof. Let G be a push-game which satisfies the two conditions above. We want to show that G is push-equivalent to $\textcircled{0}$. Let X be a push-game. We will show by induction on the size of the trees of G and X that $o(G \oplus X) = o(X)$. If $G = \textcircled{0}$, then the result follows the fact that $\textcircled{0}$ is a neutral element for $\oplus$. Hence we can assume that G has at least one normal option. Let us first consider the case $o(X) = \mathcal{N}$. There are two possible cases:

- pushing the button on X is a winning move. Then pushing the button on $G \oplus X$ is also a winning move. Indeed, we have $o(\text{push}(G \oplus X)) = o(\text{push}(G) + \text{push}(X)) = o(\text{push}(X))$ where the last equality comes from the fact that $o(\text{push}(G)) = \mathcal{P}$ and using Lemma 76. Since pushing the button on X is a winning move, it follows that $o(\text{push}(X)) = \mathcal{P}$.

- There is a normal option X' of X such that $o(X') = \mathcal{P}$. Then, playing from $G \oplus X$ to $G \oplus X'$ is a winning move since by the induction hypothesis on G and X' , we have $o(G \oplus X') = o(X') = \mathcal{P}$.

Consider now the case $o(X) = \mathcal{P}$. We wish to show that the second player has a winning strategy on $G \oplus X$. Since by assumption pushing the button on X is a losing move, it is also a losing move on $G \oplus X$ for the same reasons as above. If the first player plays to $G \oplus X'$ for some normal option X' of X, then using the induction hypothesis we have $o(G \oplus X') = o(X') = \mathcal{N}$. Finally, if the first player moves to $G' \oplus X$ for some normal option G' of G, then by assumption on G, there exists a normal option G'' of G such that $G'' \overset{\circledcirc}{\equiv} \circled{0}$. Consequently, playing to $G'' \oplus X$ is a winning move for the second player since $o(G'' \oplus X) = o(X) = \mathcal{P}$.

Let us prove now the reverse implication. Let G be a push-game such that $G \overset{\circledcirc}{\equiv} 0$. For the first point, assume by contradiction that pushing the button on G is not a winning move. Let n be the $\mathcal{SG}$-value of $\text{push}(G)$. By assumption, we have $n \neq 0$. Hence, we must have $o(G \oplus \circled{*n}) = o(\circled{*n}) = \mathcal{P}$. However, the first player has a winning move on $G \oplus \circled{*n}$ by pushing the button, since $\text{push}(G \oplus \circled{*n}) = \text{push}(G) + *n \equiv 0$, a contradiction. This implies that G satisfies the first condition.

Consider now the second point, and again assume by contradiction that there exists a normal option G' of G such that there is no normal option of G'' of G' such that $G'' \overset{\circledcirc}{\equiv} \circled{0}$. We will build a push-game X such that $o(X) = \mathcal{P}$, but $o(G \oplus X) = \mathcal{N}$. Let $G''_1, \dots, G''_k$ be the normal options of G'' for $k \geq 0$. By assumption, for each $1 \leq i \leq k$, there exists a push-game X_i such that $o(G''_i + X_i) \neq o(X_i)$. Up to replacing X_i by $G_i + X_i$, we can assume that $o(X_i) = \mathcal{N}$, and $o(G_i + X_i) = \mathcal{P}$. Consider the push-game X as follows: $\text{push}(X) = *m$ for some sufficiently large m, and the normal options of X are all the X_i, for $1 \leq i \leq k$.

Then, by construction, we have $o(X) = \mathcal{P}$ since pushing the button or playing to one of the X_i is a losing move. However, the first player has a winning strategy on $G \oplus X$ by playing to $G' \oplus X$. Indeed, on $G' \oplus X$

- pushing the button is a losing move since $\text{push}(G') + *m$ has outcome $\mathcal{N}$, if we take m large enough,

- and for any move to either $G' \oplus X_i$ or $G''_i \oplus X$ the first player can answer to $G''_i \oplus X_i$ which has outcome $\mathcal{P}$.

This shows that $o(G + X) \neq o(X)$, which is the necessary contradiction. Hence, the game G also satisfies the second property and the equivalence holds. □

Lemma 90. *For all push-game $G \in \mathfrak{G}^{\circledcirc}$, we have $G \oplus G \overset{\circledcirc}{\equiv} \circled{0}$.*

Proof. Let G be a push-game. We prove by induction on G that $G \oplus G$ satisfies all the conditions of Lemma 89. If $G = \circled{0}$, the result is trivial. Assume by induction that G has at least one normal option, and the result holds for all the normal options of G. The first condition is straightforward as

$\mathrm{push}(G \oplus G) = \mathrm{push}(G) + \mathrm{push}(G) \equiv 0$. The second point follows immediately from the induction hypothesis using symmetric moves: any normal option of $G \oplus G$ is of the form $G \oplus G'$ for some normal option G' of G, and from this game, there is a normal option to $G' \oplus G'$ which is push-equivalent to $\circled{0}$ by the induction hypothesis. □

Using the two results above,

Corollary 91. *For any two push-games G and H, $G \overset{\circledcirc}{\equiv} H$ if and only if $G \oplus H \overset{\circledcirc}{\equiv} \circled{0}$.*

Proof. The proof follows from the following claim:

Claim 92. Let G, H and X be three push-games, then $G \overset{\circledcirc}{\equiv} H$ implies $G \oplus X \overset{\circledcirc}{\equiv} H \oplus X$.

Proof. Assume that $G \overset{\circledcirc}{\equiv} H$, and let X be a push-game. The proof follows from the associativity of the operator '$\oplus$'. For every $Y \in \mathfrak{G}^{\circledcirc}$, we have: $o((G \oplus X) \oplus Y) = o(G \oplus (X \oplus Y)) = o(H \oplus (X \oplus Y)) = o((H \oplus X) \oplus Y)$. Since this holds for any push-game Y, this implies as wanted $G \oplus X \overset{\circledcirc}{\equiv} H \oplus X$. □

Let G and H be two push-games. First assume that $G \overset{\circledcirc}{\equiv} H$, then by the claim above, we have $G \oplus H \overset{\circledcirc}{\equiv} H \oplus H \overset{\circledcirc}{\equiv} \circled{0}$, where the second equivalence comes from Lemma 90.

Now, assume that $G \oplus H \overset{\circledcirc}{\equiv} \circled{0}$, then using again the claim above, we have $G \oplus H \oplus H \overset{\circledcirc}{\equiv} H$. Since $H \oplus H \overset{\circledcirc}{\equiv} \circled{0}$ by Lemma 90 we have $G \overset{\circledcirc}{\equiv} G \oplus H \oplus H \overset{\circledcirc}{\equiv} H$, which completes the proof. □

From the results above, we can see that the operator $\oplus$ defines a structure of group on $\mathfrak{G}^{\circledcirc}$ (more precisely on the quotient of $\mathfrak{G}^{\circledcirc}$ by the equivalence relation $\overset{\circledcirc}{\equiv}$), in a similar fashion as the disjunctive sum defines a group on the set of $\mathcal{SG}$-values of traditional impartial games. The group defined by $\oplus$ preserves some of the nice properties of the $\mathcal{SG}$-values, but is also at the same time more complicated. For example, every push-game is its own inverse, as was the case with the $\mathcal{SG}$-values, but there are also now several equivalence classes which correspond to the outcome $\mathcal{P}$.

Note that the two conditions in Lemma 89 can be checked in polynomial time according to the size of the game tree. Hence, with the corollary above, it is possible to check whether two push-games are push-equivalent in time which is polynomial in the size of the game tree. In addition to being able to decide push-equivalence, we can also compute canonical representatives for all the equivalence classes. The method for computing these representative is detailed in the next subsection.

8.4.2 Computing push canonical forms

The operations to compute the representative are inspired from the case of disjunctive canonical forms (see [Sie13] for example). The canonical representative is computed by iteratively simplifying the game tree of a push game. This simplification must preserve the value of the game. We first describe how the simplification is done, and then prove in Lemma 93 that it indeed computes canonical representative for the equivalence classes of the push-equivalence relation. As was the case for the canonical forms of the disjunctive sum, there are two kinds of simplification:

- Let G be a push-game, and G' and G'' be two normal options of G. We say that G' is *push-dominated*[1] by G'' if $G' \overset{\circledcirc}{\equiv} G''$. In this case, we can consider the simplified game H obtained from G by removing the move to G'.

push-dominated option

- If G' is an option of G, and G'' an option of G', we say that G' is *push-reversible* through G'' if $G'' \overset{\circledcirc}{\equiv} G$. If G has an option which is push-reversible through G'', we can construct the game H by removing the option to G' and adding all the options of G''.

push-reversible option

A push-game is said to be in *push-canonical form* if all its normal options are in push-canonical forms, and it contains no push-dominated nor push-reversible options. Let us denote by $\mathrm{Can}(G)$ the game obtained from G by replacing each of its normal options by its canonical form, and simplifying all the push-reversible and push-dominated options as we described above. The following result shows that $\mathrm{Can}(G)$ gives a canonical representative of the equivalence class of a game. The first point in the following lemma states that a game and its simplified version remain push-equivalent. The second point means that there is only one push-canonical form for each push-equivalence class.

push-canonical form

Lemma 93. *Let G and H be two push-games. Then:*

- $G \overset{\circledcirc}{\equiv} \mathrm{Can}(G)$,
- $H \overset{\circledcirc}{\equiv} G$ *if and only if* $\mathrm{Can}(G) = \mathrm{Can}(H)$.

Proof. For the first point, we only need to show that if G is a push-game, and H was obtained by one step of the simplification, then $G \overset{\circledcirc}{\equiv} H$. Assume that H was obtained from G by removing an option G' which was push-dominated by another option G''. To prove the result, we only need to show that $G \oplus H \overset{\circledcirc}{\equiv} \circled{0}$. By construction of H, pushing the button on $G \oplus H$ is a winning move. Moreover, consider a normal option of $G \oplus H$. This option can be of three possible types.

- It can be an option of the form $G \oplus H_2$ for some normal option H_2 of H. Since the normal options of H are included in the normal options of G, H_2 is also a normal option of G. Hence there is a move from $G \oplus H_2$ to $H_2 \oplus H_2 \overset{\circledcirc}{\equiv} \circled{0}$.
- It can be an option of the form $G_2 \oplus H$ for some normal option G_2 of G different from G'. By construction of H, G_2 is also a normal option of H. Hence there is a move from $G_2 \oplus H$ to $G_2 \oplus G_2 \overset{\circledcirc}{\equiv} \circled{0}$
- Finally, it can be the option $G' \oplus H$. In this case, G'' is an normal option of H, hence there is a move from $G' \oplus H$ to $G' \oplus G'' \overset{\circledcirc}{\equiv} 0$ using the assumption that $G' \overset{\circledcirc}{\equiv} G''$ and by Corollary 91.

Hence, pushing the button on $G \oplus H$ is a winning move, and for every normal option of $G \oplus H$, there exists a move to a game push-equivalent to $\circled{0}$. By Lemma 89, this implies $G \oplus H \overset{\circledcirc}{\equiv} 0$, and by Corollary 91, we have $G \overset{\circledcirc}{\equiv} H$.

Assume now that H was obtained from G by simplifying a normal option G' which is push-reversible through G''. Since by assumption $G \overset{\circledcirc}{\equiv} G''$, to prove that $G \overset{\circledcirc}{\equiv} H$, it is enough to show

[1] Note that even though the term 'push dominated' suggests an asymmetry between the two games we compare, the relation is symmetric. The choice for this term comes from a simplification rule for the normal disjunctive sum which is very similar to this one (see [Sie13, p.64]).

that $G'' \oplus H \overset{\circledcirc}{\equiv} \textcircled{0}$. Clearly, pushing the button on $G'' \oplus H$ is a winning move since $\mathrm{push}(G'' \oplus H) = \mathrm{push}(G'') + \oplus H \equiv \mathrm{push}(G'') + \mathrm{push}(G) \equiv 0$. Additionally, the normal options of $G'' \oplus H$ can be:

- either of the form $G_2'' \oplus H$ or $G'' \oplus G_2''$, where G_2'' is a normal option of G''. In this case, there is a move from this option to $G_2'' \oplus G_2''$ which is push-equivalent to $\textcircled{0}$ by Lemma 90;
- or of the form $G'' \oplus G_2$, where G_2 is a normal option of G different from G'. However, using the fact that $G \oplus G'' \overset{\circledcirc}{\equiv} 0$, and the fact that $G_2 \oplus G''$ is also a normal option of $G \oplus G''$, by Lemma 89 there exists a normal option X of $G'' \oplus G_2$ such that $X \overset{\circledcirc}{\equiv} \textcircled{0}$.

This shows that from every normal option of $G'' \oplus H$ there is a move to a game push-equivalent to 0, and shows by Lemmas 89 and 91 that $H \overset{\circledcirc}{\equiv} G'' \overset{\circledcirc}{\equiv} G$. This ends the proof for the first point of the lemma.

Consider now the second point, and first assume that $\mathrm{Can}(G) = \mathrm{Can}(H)$, then using the first point we have $G \overset{\circledcirc}{\equiv} \mathrm{Can}(G) = \mathrm{Can}(H) \overset{\circledcirc}{\equiv} H$. Hence, we only need to prove the direct implication.

Let G and H be two push-games in canonical form such that $G \overset{\circledcirc}{\equiv} H$. We want to show that G and H are equal. This is shown by induction on the size of the game trees of G and H. First, since $G \overset{\circledcirc}{\equiv} H$, pushing the button on $G \oplus H$ is a winning move. Consequently, since both $\mathrm{push}(G)$ and $\mathrm{push}(H)$ are in canonical form, it follows that $\mathrm{push}(G) = \mathrm{push}(H)$. Let us show that for every normal option G' of G there exists a normal option H' of H such that $G' = H'$.

Let G' be a normal option of G. Then $G' \oplus H$ is a normal option of $G \oplus H$. Since $G \oplus H \overset{\circledcirc}{\equiv} \textcircled{0}$, this implies that there exists a normal option X of $G' \oplus H$ such that $X \overset{\circledcirc}{\equiv} \textcircled{0}$. If X is of the form $X = G'' \oplus H$ for some normal option G'' of G', then this implies $G'' \overset{\circledcirc}{\equiv} H \overset{\circledcirc}{\equiv} G$. This contradicts the assumption that G has no reversible options. Hence, X must be of the form $X = G' \oplus H'$ for some normal option H' of H, and consequently $G' \overset{\circledcirc}{\equiv} H'$. Using the induction hypothesis, this implies that $G' = H'$.

Symmetrically, for every option H' of H, there exists an option G' of G such that $H' = G'$. Since there is no two normal options of G (or H) which are push-equivalent, this implies that there is a bijection between the normal options of G and H, and consequently $G = H$. □

We have shown how to compute the push canonical forms for an arbitrary push game. If this method is very general, there are some games for which the canonical form is quite simple, and can be easily computed. The following theorem is an example of such games.

Theorem 94. *Let G be a push-game such that all the nodes of the game tree of G are labelled with the same game H. Then there exists $i \geq 0$, such that $G \overset{\circledcirc}{\equiv} \circledast i \oplus \textcircled{H}$.*

Proof. If all the nodes in the game tree of G are labelled with H, then all the nodes in the game tree of $G \oplus \textcircled{H}$ are labelled with 0. Hence, to prove the result, we only need to show that games of the form $\circledast i$ are the only games in push-canonical form for which all the nodes in the game tree are labelled with 0. Let G be a game in push-canonical form, where all the nodes in the game tree of G are labelled with 0. We prove that $G \overset{\circledcirc}{\equiv} \circledast i$ for some i by induction on the size of the game-tree of G. If the game tree of G contains a single node, then $G = \textcircled{0} = \circledast 0$, and there is nothing to prove. Assume now that the induction hypothesis holds, and consider the game G such that all the nodes in the game tree of G are labelled with 0. Since G is in push-canonical form, using the induction

hypothesis, all the normal options of G are of the form $\circledast i$ for some $i \geq 0$. Consider the set A of all integers i such that $\circledast i$ is a normal option of G. Let $j = \text{mex}(A)$, the smallest non-negative integer not in A. We will show that $G \oplus \circledast j \overset{\circledcirc}{\equiv} \circled{0}$, which implies $G \overset{\circledcirc}{\equiv} \circledast j$ by Corollary 91.

Clearly, pushing the button on $G \oplus \circledast j$ is a wining move since both components are equal to 0 after pushing the button. Additionally, consider a normal option of $G \oplus \circledast j$. This option can be either:

- $G \oplus \circledast j'$ for some $j' < j$. By definition of j, $\circledast j'$ is also a normal option of G, and there is a move from $G \oplus \circledast j'$ to $\circledast j' \oplus \circledast j' \overset{\circledcirc}{\equiv} \circled{0}$;
- or $\circledast i \oplus \circledast j$, where $i \in A$. Then by definition of j, we have $i \neq j$. Hence, either $i < j$ and there is a move from this game to $\circledast i \oplus \circledast i$, or $i > j$, and there is a move to $\circledast j \oplus \circledast j$. In any case, there is a move to a game which is push-equivalent to $\circled{0}$ by Lemma 90.

Using Lemma 89, this implies that $G \oplus \circledast j \overset{\circledcirc}{\equiv} \circled{0}$, as required. □

Before trying to apply these canonical forms to several games, we highlight the similarities and the differences between the $\mathcal{SG}$-values and the push-canonical forms of games. The push-canonical forms are their own inverse, i.e., $X \oplus X \overset{\circledcirc}{\equiv} \circled{0}$, as it was the case for the $\mathcal{SG}$-values. Additionally, the push-canonical form can be computed by applying a simplification procedure on the tree of a game. This simplification procedure is very similar to what is done for the $\mathcal{SG}$-values of impartial games. However, unlike the $\mathcal{SG}$-values, the number of push-canonical games seem to grow much faster (as a function of the depth of the tree) than the number of $\mathcal{SG}$-values. This will be visible in particular in the examples in the next section. Finally, for any $x \in \{\mathcal{N}, \mathcal{P}\}$, the number of push-canonical games with outcome x is infinite, which is another difference with the $\mathcal{SG}$-values for which there is only one canonical form with outcome $\mathcal{P}$.

We now attempt to apply these results to push-compounds of Subtraction games. Even though Subtraction rulesets produce some of the most simple games we can consider, even in this case it seems that the canonical forms for push-compounds of Subtraction games become complicated very quickly.

8.4.3 Push-Subtraction games

We now consider the push operator applied to Subtraction rulesets. Recall from Section 7.3 that the ruleset Sub(S) is played with one heap of tokens. At his turn a player can remove $v \in S$ tokens from the heap, provided there are at least v tokens in the heap. Before looking at push-values and push-canonical forms of subtraction games, we consider the outcomes of these games on a single heap. It is known that the sequences of $\mathcal{SG}$-values (and consequently also the sequences of outcomes) for usual subtraction games are periodic (see Theorem 79). We show in the following theorem that the periodicity of outcomes still holds if Subtraction rulesets are composed using the push-the-button construction. In the following, we call Push-Subtraction the rulesets of the form $\mathcal{R}_1 \circledcirc \mathcal{R}_2$, where $\mathcal{R}_1$ and $\mathcal{R}_2$ are both Subtraction games.

Theorem 95. *Suppose that $\mathcal{R}_2$ is a ruleset over $\mathbb{N}$ with purely periodic $\mathcal{P}$-positions of period k, and $\mathcal{R}_1 = \text{Sub}(S)$ for a finite set S. The ruleset $\mathcal{R}_1 \circledcirc \mathcal{R}_2$ has ultimately periodic $\mathcal{P}$-positions. The lengths of the period and pre-period are at most $k \cdot 2^{\max(S)}$.*

Proof. Let $\mathcal{R} = \mathcal{R}_1 \circledcirc \mathcal{R}_2$. The proof follows essentially the same lines as the proof of Theorem 79. We denote $M = \max(S)$ the largest element in the subtraction set S. Remark that the function $n \mapsto o_{\mathcal{R}}(n)$ is such that its value on an integer n only depends on two things:

- the value of $o_{\mathcal{R}_2}(n)$, which only depends on the congruence class of n modulo k,
- and the value of $o_{\mathcal{R}}(n-i)$ for $1 \leq i \leq M$.

As a consequence, we can write:

$$o_{\mathcal{R}}(n) = f(o_{\mathcal{R}}(n-1), \ldots, o_{\mathcal{R}}(n-M), n \bmod k)$$

for some function $f : \{\mathcal{P},\mathcal{N}\}^M \times \mathbb{Z}_k \longrightarrow \{\mathcal{P},\mathcal{N}\}$. If we note X the set $\{\mathcal{P},\mathcal{N}\}^M \times \mathbb{Z}_k$, we can define the function $g : X \to X$ as:

$$g(a_1, \ldots a_M, n) = (f(a_1, \ldots a_M, n), a_1, \ldots a_{M-1}, n+1 \bmod k).$$

We denote by F_0 the initial vector $(o_{\mathcal{R}}(M-1), \ldots o_{\mathcal{R}}(0), M)$, and $g^{(m)}$ the function g composed m times. We can see that the first element of the tuple $g^{(m)}(F_0)$ is $o_{\mathcal{R}}(M+m-1)$. Now, since $|X| = k2^M$, there exists n_0 and p smaller than $|X|$ such that $g^{(n_0+p)}(F_0) = g^{(n_0)}(F_0)$, and consequently, for all $m \geq n_0$, $o_{\mathcal{R}}(m+p) = o_{\mathcal{R}}(m)$. □

Note that the theorem above assumes that the $\mathcal{P}$-positions of $\mathcal{R}_2$ are purely periodic (i.e., there is no preperiod). However, the result can be easily adapted if the $\mathcal{P}$-positions of $\mathcal{R}_2$ are only ultimately periodic by just 'forgetting' the first positions which corresponds to the preperiod.

In particular, since the $\mathcal{SG}$-sequence of Subtraction games is ultimately periodic, the result above implies that the outcome sequence of Push-Subtraction games is also ultimately periodic. As a consequence deciding the outcome of a push-subtraction game on a single heap can be done in polynomial time. On the other hand, this does not prove anything in the case of multiple heaps. For usual subtraction games, the periodicity of the $\mathcal{SG}$-values (see Theorem 79) implies that computing the outcome on multiple heaps can be done in polynomial time as well. For this, we only need to compute the $\mathcal{SG}$-value of each of the heaps individually, and combine them using the XOR-rule. Ideally, we would want this type of argument to work for Push-Subtraction games as well. In other words, ideally, we would want the push-canonical forms of Push-Subtraction games to be periodic as well. However the few examples below suggest that this is usually not the case, even for very simple subtraction sets. As a consequence, it seems unclear whether the canonical forms of Push-Subtraction games are simple enough to allow computing the outcomes on multiple heaps in polynomial time.

Note that for usual heap games, the periodicity of the $\mathcal{P}$-positions, and the periodicity of the $\mathcal{SG}$-values often occur together. The Push-Subtraction games are an example where this does not hold. Note that this is also true for the Partizan Subtraction games that we will study in the next chapter.

Theorem 96. *The sequence of canonical forms for the ruleset* $\mathcal{R} = \text{Sub}(\{1,2\}) \circledcirc \text{Sub}(\{1\})$ *contains infinitely many values.*

Proof. We will show by induction on i that for all $j \leq i$, the games $(j)_{\mathcal{R}}$ are already in canonical form, and in particular they are pair-wise non-equivalent. The result clearly holds if $i = 1$. Assume now that $i \geq 2$, and consider the game $(i)_{\mathcal{R}}$. Using the induction hypothesis, we know that all the options of $(i)_{\mathcal{R}}$ are already in canonical form. To prove that $(i)_{\mathcal{R}}$ is in canonical form, we only need to show that it has no push-reversible nor push-dominated options.

- To prove that $(i)_{\mathcal{R}}$ has no push-reversible options, it is enough to show that $(i)_{\mathcal{R}}$ is not push-equivalent to $(j)_{\mathcal{R}}$ for $j \in \{i-2, i-3, i-4\}$. Clearly $(i)_{\mathcal{R}}$ and $(i-3)_{\mathcal{R}}$ are not push-equivalent since i and $i-3$ do not have the same parity, and consequently, $\text{push}((i)_{\mathcal{R}}) \not\equiv \text{push}((i-3)_{\mathcal{R}})$. Moreover, $(i)_{\mathcal{R}} \oplus (i-2)_{\mathcal{R}}$ is not push-equivalent to $\circled{0}$. Indeed one of its normal options is $(i-2)_{\mathcal{R}} \oplus (i-2)_{\mathcal{R}}$ which is push-equivalent to $\circled{0}$ by Lemma 90. This normal option will never be removed by any further simplification (it cannot be push-reversible since it has no option), and will remain in the push-canonical form of $(i)_{\mathcal{R}} \oplus (i-2)_{\mathcal{R}}$ which consequently won't be push-equivalent to $\circled{0}$. Finally, consider the game $(i)_{\mathcal{R}} \oplus (i-4)_{\mathcal{R}}$. This game has a normal option to $(i-1)_{\mathcal{R}} \oplus (i-4)_{\mathcal{R}}$, for which (using the induction hypothesis) there is no normal option which is push-equivalent to $\circled{0}$.

  This concludes the proof showing that $(i)_{\mathcal{R}}$ is not push-equivalent to $(j)_{\mathcal{R}}$ for $j \in \{i-2, i-3, i-4\}$, and consequently, $(i)_{\mathcal{R}}$ has no push-reversible option.

- The fact that $(i)_{\mathcal{R}}$ has no push-dominated option follows immediately from the fact that $(i-1)_{\mathcal{R}}$ and $(i-2)_{\mathcal{R}}$ are not push equivalent using the induction hypothesis.

Hence, none of the two simplification cases apply, and consequently $(i)_{\mathcal{R}}$ is already in canonical form, and consequently not push-equivalent to $(j)_{\mathcal{R}}$ for any $j < i$. □

This result implies that using the push-canonical forms to compute the outcome of $\textsc{Sub}(\{1,2\}) \circledcirc \textsc{Sub}(\{1\})$ on multiple heaps is not practical (at least not directly, but we could still have a behaviour similar to Nim with the bit-wise XOR). However, we will see in the next subsection that for this particular game, the push-canonical forms can be simplified further if we do not care about computing push-sums of this game with other games, but only consider push-sums of several positions of this game. In particular, we will be able to compute in polynomial time the outcome on several heaps using simplified canonical forms. The result above showed that push-canonical forms can become complicated, even if the second ruleset in the compound is very simple (here it is just $\textsc{Sub}(\{1\})$). The next theorem completes this result by showing that the values can also be complicated if the first ruleset is only $\textsc{Sub}(\{1\})$. Unlike the other example above, we do not have a general method for computing the outcome on several heaps for these games.

Theorem 97. *Let S be a finite subset of positive integers, with $2 \in S$. The sequence of push-canonical forms for $\textsc{Sub}(\{1\}) \circledcirc \textsc{Sub}(S)$ contains infinitely many values.*

Proof. Consider the ruleset $\mathcal{R} = \textsc{Sub}(\{1\}) \circledcirc \textsc{Sub}(S)$. We will show by induction on i that the game $(i)_{\mathcal{R}}$ (i.e., the game with rules $\mathcal{R}$ played on a heap of size i) is already in canonical form. This clearly holds if $i = 0$, since the game cannot be simplified further. Assume by induction that $(i)_{\mathcal{R}}$ is in canonical form, and consider the game $(i+1)_{\mathcal{R}}$. The only normal option of $(i+1)_{\mathcal{R}}$ is $(i)_{\mathcal{R}}$. To prove that $(i+1)_{\mathcal{R}}$ is already in canonical form, we only need to prove that this option is not reversible. This follows immediately from the fact that $(i)_{\mathcal{R}}$ and $(i-2)_{\mathcal{R}}$ are not push-equivalent. Indeed, since $2 \in S$, the two games $(i)_{\textsc{Sub}(S)}$ and $(i-2)_{\textsc{Sub}(S)}$ do not have the same $\mathcal{SG}$-values since one is an option of the other. □

Even though using the push-canonical forms is not practical for the rulesets considered above, there are cases, where we can use other ad-hoc methods to characterize the $\mathcal{P}$-positions on multiple heaps. However, we do not know any general method that would allow computing the outcome of Push-Subtraction games on multiple heaps in polynomial time.

Open Problem 18. Find a polynomial-time algorithm to compute the outcome of PUSH SUBTRACTION games on several heaps.

The theorem below is a simple example where we can give an explicit characterization of the $\mathcal{P}$-positions on multiple heaps. From the statement of the theorem, we can see that if we fix the size of the smallest heap, then deciding whether a position is a $\mathcal{P}$-position or not is easy: it only depends on the parity of certain quantities related to the position. Investigating whether this property holds for PUSH-SUBTRACTION games in general, or is only specific to this particular game could be a first step to solve general PUSH-SUBTRACTION game.

Theorem 98. *Consider the ruleset* $\mathcal{R} = \text{SUB}(\{1\}) \circledcirc \text{SUB}(\{1,2\})$. *A position* s *is a* $\mathcal{P}$*-position if and only if one of the following holds:*

- s *has an odd number of heaps and there is an even number of heaps with size* $(2 \mod 3)$ *and an odd number of heaps with size* $(1 \mod 3)$*;*
- s *has an even number of heaps and satisfies the criterion above after removing the smallest heap and subtracting it from all the other heaps.*

Proof. Note that the $\mathcal{SG}$-value for a heap of size i for $\text{SUB}(\{1,2\})$ is equal to $(i \mod 3)$. Denote by B be the set of positions which satisfy one of the two conditions above. In the following, give a position s and $i \in \{0,1,2\}$, we write $\Gamma_i(s)$ the set of heaps of s which have size:

- $(i \mod 3)$ if s has an odd number of heaps,
- $(\min(s) + i \mod 3)$ if s has an even number of heaps.

We also write $\gamma_i(s)$ the parity of $|\Gamma_i(s)|$. With this notation, we have $s \in B \iff (\gamma_1(s), \gamma_2(s)) = (\mathbf{odd}, \mathbf{even})$. To prove that B is the set of $\mathcal{P}$-positions for $\text{SUB}(\{1\}) \circledcirc \text{SUB}(\{1,2\})$, we will prove that it satisfies the two conditions from Lemma 80.

First, we check that for every position s not in B, there is either a winning move by pushing the button, or a move to $s' \in B$ by removing one token from a heap. The different possible cases are considered in the table below.

$\gamma_1(s)$	$\gamma_2(s)$	
even	**even**	Pushing the button is a winning move.
even	**odd**	Removing 1 in a heap of $\Gamma_2(s)$ does not change the number of heaps, and leads to s' with $(\gamma_1(s'), \gamma_2(s')) = (\mathbf{odd}, \mathbf{even})$.
odd	**odd**	If the number of heaps is odd, then $\gamma_0(s)$ is odd, and in particular $\Gamma_0(s)$ is not empty. Removing one token from a heap in $\Gamma_0(s)$ leads to a position s' with the same number of heaps, and $s' \in B$. If the number of heaps is even, removing one token in the smallest heap leads to s', with $(\gamma_1(s'), \gamma_2(s')) = (\mathbf{odd}, \mathbf{even})$ if the smallest heap had size at least 2. If the smallest heap had size 1, then this move leads to a position s' with an odd number of heaps, and $(\gamma_1(s'), \gamma_2(s')) = (\mathbf{odd}, \mathbf{even})$.

Then we check that for all $s \in B$, pushing the button is a losing move, and all options s' of s are not in B.

Let s be a position with $\gamma_1(s) = \mathbf{odd}$ and $\gamma_2(s) = \mathbf{even}$. We start by showing that pushing the button on s is a losing move. First, observe that the $\mathcal{SG}$-value of $(s)_{\mathrm{SUB}(\{1,2\})}$ is zero if and only if there is an even number of heaps with size $(1 \mod 3)$ and $(2 \mod 3)$. If there is an odd number of heaps in s, then the number of heaps with size $(1 \mod 3)$ is equal to $|\Gamma_1(s)|$ which is odd by assumption. Hence, the $\mathcal{SG}$-value after pushing the button is **odd**, and in particular non-zero. Consequently pushing the button is a losing move.

If the number of heaps is even, then we must have $\gamma_0(s) = \mathbf{odd}$. Let n_1 and n_2 denote the number of heaps of size $(1 \mod 3)$ and $(2 \mod 3)$ respectively. We cannot have both n_1 and n_2 even, since only one of $\gamma_0(s), \gamma_1(s), \gamma_2(s)$ is even. Again, the $\mathcal{SG}$-value after pushing the button is non-zero, and pushing the button is a losing move.

Hence, we only need to prove that for all the options of s' of s, we have $s' \notin B$. If s and s' have the same number of heaps, then the move to s' changes either $\gamma_1(s)$ or $\gamma_2(s)$, and consequently $s' \notin B$. Assume that the move from s to s' decreases the number of heaps by 1, and consequently the smallest heap of s has size 1. If the number of heaps in s was even, then $|\Gamma_1(s')| = |\Gamma_0(s)| - 1$ which is even, consequently $s' \notin B$. If the number of heaps in s is odd, then in s' the number of heaps of size i modulo 3 for $i \in \{0, 1, 2\}$ is even, as a consequence, $\gamma_1(s')$ and $\gamma_2(s')$ are both even, and $s' \notin B$. □

8.4.4 The $(0, 1)$-universe

As we have seen in the examples above, in general the push-canonical forms seem to be complicated, even for very simple games. There are however some cases where these values can be simplified further if we do not wish to compute push-sum with any games, but restrict the kind of games we can do sums with. This is similar to the 'misère quotient' approach taken in [Pla05, PS08, MR13, DRSS15] for the disjunctive sum under misère-play convention. The main idea is, instead of considering sums of arbitrary games, to only consider sums of games from a particular 'universe'. This universe is a proper subset of all the possible games. For example, if we wish only to study one particular ruleset $\mathcal{R}$, then the universe could be restricted only to games of the form $(p)_{\mathcal{R}}$ for some position p. By doing so, we can define an equivalence in this restricted universe, and this equivalence might contain only 'few' equivalence classes, which would simplify the study of the sums of these games. Of course, these values do not give any information if we want to sum these games with other games outside the universe. The main issue with this method it that the equivalence in the restricted universe might not satisfy anymore the simple properties of the general case. In particular, there is no guarantee that in the restricted universe there is a simple method to decide whether two games are equivalent.

In this subsection, we restrict the push-games we consider to only certain games which satisfy some properties. We show that in this restricted universe, the values become more simple, and canonical representative for these values can still be computed efficiently.

We consider the subset $\mathfrak{G}^{\circledcirc}_{0,1}$, which is the subset of push-games such that pushing the button can only produce games with $\mathcal{SG}$-values 0 or 1. In other words, this means that we can assume that the labels of the nodes in the game tree of $G \in \mathfrak{G}^{\circledcirc}_{0,1}$ are all either $*0$ or $*1$. We call $(0, 1)$-push-games these games. In this case, we modify the definition of equivalence we used at the beginning of this section, by quantifying only over all games in $\mathfrak{G}^{\circledcirc}_{0,1}$, instead of considering all the possible

push-games. We denote by $\overset{\circledcirc}{\equiv}_{0,1}$ this new equivalence, which is formally defined by:

$$G \overset{\circledcirc}{\equiv}_{0,1} H \Leftrightarrow \forall X \in \mathfrak{G}^{\circledcirc}_{0,1},\ o(G+X) = o(H+X)\ .$$

Note that the only difference with the push-equivalence is the set of games we quantify over. In particular, the push-equivalence $\overset{\circledcirc}{\equiv}$ is refinement of the $(0,1)$-equivalence $\overset{\circledcirc}{\equiv}_{0,1}$. This means that if $G \overset{\circledcirc}{\equiv} H$, then $G \overset{\circledcirc}{\equiv}_{0,1} H$. If we restrict our attention to $(0,1)$-push-games, the values (i.e., the equivalence classes) for the $(0,1)$-equivalence become more simple due to the following observation.

Observation 99. *Let G be a $(0,1)$-push-game, and G' be a normal option of G. If $\mathrm{push}(G) \not\equiv \mathrm{push}(G')$, then we can assume that the player with a winning strategy never plays the move from G to G', even if G is a component in a push-sum.*

Proof. Let G be the $(0,1)$-push-game with a normal option G' as in the observation, and X be an arbitrary $(0,1)$-push-game. Assume that in the game $G \oplus X$, playing in G to G' is a winning move. Hence $o(G' \oplus X) = \mathcal{P}$, and in particular, $\mathrm{push}(G' \oplus X)$ has a $\mathcal{SG}$-value different from 0. This implies that $\mathrm{push}(G')$ and $\mathrm{push}(X)$ have different $\mathcal{SG}$-values. Since there are only two possible cases for the $\mathcal{SG}$-value after pushing the button, and we assumed that $\mathrm{push}(G) \not\equiv \mathrm{push}(G')$, then this means that $\mathrm{push}(G) \equiv \mathrm{push}(X)$. In particular the player also has a winning move on $G \oplus X$ by pushing the button, and can win by playing this instead of playing in G. □

The observation above means that we can simplify the game tree of a $(0,1)$-push-game, by just removing all the nodes which are labelled with games which are not equivalent to the label of the root. After this simplification, we can apply Theorem 94, and we immediately obtain the following result.

Theorem 100. *Any game $G \in \mathfrak{G}^{\circledcirc}_{0,1}$ is $(0,1)$-equivalent to $\circledast i \oplus \circled{*j}$, for some integers $i \in \mathbb{N}$ and $j \in \{0,1\}$.*

Note that the value of i and j in the statement above can be easily computed. Indeed, the value for j is 0 if pushing the button is a winning move, and 1 otherwise. The value of i can be obtained from the $\mathcal{SG}$-values of an auxiliary game G' which is obtained from G by removing all the nodes in the game tree which have a label different from the root, and removing the push-move (i.e., pushing the button is no longer permitted). This implies in particular the following result for some very simple Push-Subtraction games. Note that in particular for the ruleset $\mathrm{Sub}(\{1,2\}) \circledcirc \mathrm{Sub}(\{1\})$, we showed in Theorem 96 that the sequence of push-canonical forms contained infinitely many different values. If we consider only the $(0,1)$-push-values of this game, then this sequence is ultimately periodic, and in particular takes only a finite number of different values. This result is a special case of the following theorem.

Theorem 101. *Let S_1 and S_2 be two finite set of positive integers, such that $\mathcal{SG}_{\mathrm{Sub}(S_2)}(n) \in \{0,1\}$ for every position $n \geq 0$. Then the sequence of $(0,1)$-push-canonical forms for $\mathrm{Sub}(S_1) \circledcirc \mathrm{Sub}(S_2)$ are periodic.*

Proof. We can first remark that there is only a finite number of different push-values . This can be seen easily from the way we compute the canonical forms as described just above. The periodicity follows immediately using a similar argument as in the proofs of Theorems 79 and 95, and the fact that the $(0,1)$-canonical form of a game can be computed from the $(0,1)$-canonical forms of its normal options. □

8.5 Conclusion

We have seen in this chapter a construction which allows us to combine several rulesets, and an almost disjunctive sum for these new games. We proved how to compute canonical forms for these games, and how these canonical forms shared some similarities with $\mathcal{SG}$-values, but also had significant differences. In particular, the number of push-canonical games seem to grow very quickly, and there are cases where computing the push-canonical forms does not seem to help. We also saw that these push-canonical forms became very simple if we restrict greatly the universe of games we consider. It could be interesting to see if it is possible to find an intermediate universe, for which the values retain a simple structure, and which captures a larger class of games. This could lead to a method to find a polynomial time algorithm to compute the outcome of Push-Subtraction games on multiple heaps.

Finally, note that all our analysis was done for impartial games only. Another direction of research could be to investigate if (and how) these results can be generalised to partisan games.

Chapter 9

Rules Decomposition: Partizan Subtraction Games

In this chapter, we consider Partizan Subtraction *games, a generalisation of standard* Subtraction *games. We investigate the asymptotic behaviours of the outcome sequences for this family of games. In addition to the notion of dominance introduced in [FK87], we define three other behaviours and investigate the problem of computing this behaviour for different instances. The results presented here were published in [DHNP19].*

This chapter is organized as follows. In Section 9.1 we formally describe Partizan Subtraction games, and define the properties of these games that will be studied in the other sections. In Section 9.2 we consider the problem from a complexity point of view. We show that two problems related to computing the outcome of a given position are **NP**-hard. In Sections 9.3 and 9.4, we investigate properties of the outcome sequence when one of the subtraction set is respectively fixed or contains only one element. Finally, in Section 9.5 we consider the case of subtraction sets of size 2, and characterize some cases for which a dominance property holds.

9.1 Introduction

Partizan Subtraction games

Partizan Subtraction *games* were introduced by Fraenkel and Kotzig in 1987 [FK87]. They are a generalisation of (impartial) Subtraction games. Recall from Section 7.3 that Subtraction games are parametrized by a finite set S of positive integers. At his turn, a player can remove x tokens from a single heap, provided $x \in S$. In the partizan version, each player is assigned a finite set of integers, respectively denoted $S_{\mathcal{L}}$ (for the $\mathcal{L}$eft player), and $S_{\mathcal{R}}$ (for the $\mathcal{R}$ight player). A move consists in removing a number m of tokens from the heap, provided m belongs to the set of the player. We consider the game with the normal play convention: the first player unable to move loses. When $S_{\mathcal{L}} = S_{\mathcal{R}}$, the game is impartial and is known as the standard Subtraction game.

Partizan Subtraction games are a special case of a more general construction which consists in building a new partizan ruleset from two impartial rulesets $\mathcal{R}_0$ and $\mathcal{R}_1$. Given these two rulesets, we can build a partizan ruleset where $\mathcal{L}$eft plays according to the rules of $\mathcal{R}_0$, and $\mathcal{R}$ight plays according to $\mathcal{R}_1$. For this construction to work, the two rulesets must be compatible in the sense defined in Chapter 8 (i.e., they must use the same set of positions). Partizan Subtraction games corresponds to the case where $\mathcal{R}_0$ and $\mathcal{R}_1$ are both Subtraction rulesets. In Chapter 8 we

studied a construction which builds an impartial ruleset from a partizan one. On the contrary, the operation we consider here is the reverse: it builds partizan rulesets from impartial ones.

In the rest of the section, it will always be clear from the context which ruleset we are considering. Hence we will omit to write it in the notations and just write $o(n)$ for the outcome of the Partizan Subtraction ruleset currently under consideration on a single heap of size n. We also use simplified notations and just write $(S_\mathcal{L}, S_\mathcal{R})$ the Partizan Subtraction ruleset with subtraction sets $S_\mathcal{L}$ and $S_\mathcal{R}$. Although Partizan Subtraction games can be played on multiple heaps, we will concentrate here on the single heap version of the game. As such, a game position will be simply denoted by an integer n corresponding to the size of the heap.

The *outcome sequence* of a Partizan Subtraction ruleset $(S_\mathcal{L}, S_\mathcal{R})$ is the sequence of the outcomes for $n = 0, 1, 2, 3, \ldots$, i.e., $o(0), o(1), o(2), \ldots$. A well-known result ensures that the outcome sequence of any impartial subtraction game is ultimately periodic (see Theorem 79). Note that in that case, the outcomes can only take the values $\mathcal{P}$ or $\mathcal{N}$ since the game is impartial. In [FK87], this result is extended to partizan subtraction games.

Theorem 102 (Fraenkel and Kotzig [FK87])**.** *The outcome sequence of any partizan subtraction game is ultimately periodic.*

Example 1. Consider the partizan subtraction ruleset $(\{1, 2\}, \{1, 3\})$. Its outcome sequence is

$$\mathcal{P}\,\mathcal{N}\,\mathcal{L}\,\mathcal{N}\,\mathcal{L}\,\mathcal{L}\,\mathcal{L}\,\mathcal{L}\,\ldots$$

In this particular case, the periodicity of the sequence can be easily proved by showing by induction that the outcome is $\mathcal{L}$ for $n \geq 4$.

A behaviour as in Example 1 where the outcome sequence has period 1 seem rather frequent for partizan subtraction games. In this case, the period is either only $\mathcal{L}$ or only $\mathcal{R}$. In their paper, Fraenkel and Kotzig called this property *dominance*. More precisely, we say that $S_\mathcal{L} \succ S_\mathcal{R}$ – or that $S_\mathcal{L}$ dominates $S_\mathcal{R}$ – if there exists an integer n_0 such that the outcome of the game $(S_\mathcal{L}, S_\mathcal{R})$ is always $\mathcal{L}$ for all $n \geq n_0$. By symmetry, a game satisfying $S_\mathcal{L} \prec S_\mathcal{R}$ is always $\mathcal{R}$ for all sufficiently large heap sizes. When a game satisfies neither $S_\mathcal{L} \succ S_\mathcal{R}$ nor $S_\mathcal{L} \prec S_\mathcal{R}$, the sets $S_\mathcal{L}$ and $S_\mathcal{R}$ are said *incomparable*, which is denoted by $S_\mathcal{L} \| S_\mathcal{R}$. In [FK87], several instances have been proved to satisfy the dominance property such as the rulesets $(\{1, 2m\}, \{1, 2n + 1\})$ and $(\{1, 2m\}, \{1, 2n\})$. Others were shown to be incomparable such as $\{a\}$ and $\{b\}$. It is also shown that the dominance relation is not transitive. Note that unlike their impartial variants, the canonical forms of partizan subtraction games can be quite complicated, and are not necessarily periodic. In [Pla95], the canonical forms have been computed for partizan subtraction games with $S_\mathcal{L} = \{1, 2\}$, and $S_\mathcal{R} = \{1, k\}$.

In the literature, partizan taking and breaking games have not been so much considered. A more general version, where it is also allowed to split the heap into two heaps, was introduced by Fraenkel and Kotzig in [FK87], and is known as partizan octal games. A particular case of such games, called *partizan splittles*, was considered in [MI05], where in addition, $S_\mathcal{L}$ are $S_\mathcal{R}$ are allowed to be infinite sets. Another variation with infinite sets is when $S_\mathcal{L}$ and $S_\mathcal{R}$ make a partition of $\mathbb{N}$ [LMNS18]. In such cases, the ultimate periodicity of the outcome sequence is not necessarily preserved.

In the current chapter, we propose a refinement of the structure of the outcome sequence for partizan subtraction games. More precisely, when the sets $S_\mathcal{L}$ and $S_\mathcal{R}$ are incomparable, different kinds of periodicity can occur. The following definitions presents a classification for them.

Definition 9. We characterize the ruleset $(S_\mathcal{L}, S_\mathcal{R})$ depending on the outcome values which appear in the periodic part of outcome sequences. More precisely, we say that $(S_\mathcal{L}, S_\mathcal{R})$ is:

strongly dominating, $\mathcal{SD}$
- $\mathcal{SD}$ (*Strongly Dominating*) for $\mathcal{L}$eft (resp. $\mathcal{R}$ight), and we write $S_\mathcal{L} \succ S_\mathcal{R}$ (resp. $S_\mathcal{L} \prec S_\mathcal{R}$) if any position n large enough has outcome $\mathcal{L}$ (resp. $\mathcal{R}$). In other words, the period is reduced to $\mathcal{L}$ (resp. $\mathcal{R}$).

weakly dominating, $\mathcal{WD}$
- $\mathcal{WD}$ (*Weakly Dominating*) for $\mathcal{L}$eft (resp. $\mathcal{R}$ight), and we write $S_\mathcal{L} \succ_w S_\mathcal{R}$ if the period contains at least one $\mathcal{L}$ and no $\mathcal{R}$ (or resp. one $\mathcal{R}$ and no $\mathcal{L}$).

fair, $\mathcal{F}$
- $\mathcal{F}$ (*Fair*) if the period contains both $\mathcal{L}$ and $\mathcal{R}$.

ultimately impartial, $\mathcal{UI}$
- $\mathcal{UI}$ (*Ultimately Impartial*) if the period contains no $\mathcal{L}$ and no $\mathcal{R}$.

Remark 3. Note that inside a period, not all the combinations of $\mathcal{P}$, $\mathcal{N}$, $\mathcal{L}$ and $\mathcal{R}$ are possible. For example, a period that includes $\mathcal{P}$ must also include $\mathcal{N}$. Indeed, assume on the contrary that there is $(S_\mathcal{L}, S_\mathcal{R})$ such that the period of the outcome sequence contains $\mathcal{P}$ but not $\mathcal{N}$. Let n be a position of outcome $\mathcal{P}$ in the period, and p the length of the period. Let $a \in S_\mathcal{L}$. Now the position $n+a$ is in the period, and $o(n+a) = \mathcal{L}$ since $\mathcal{L}$eft can win by playing a as a first player, and using the assumption that $o(n) \neq \mathcal{N}$. For the same reason, we also have $o(n+2a) = \mathcal{L}$. By repeating this argument, $o(n+ka) = \mathcal{L}$ for all k. In particular, $n+pa$ is a $\mathcal{L}$-position. However, since n is in the period, we must have $o(n+pa) = o(n) = \mathcal{P}$, a contradiction.

The literature detailed above mentions examples of $\mathcal{SD}$ and $\mathcal{UI}$ games (for example, impartial subtraction games are $\mathcal{UI}$). We will see later in this chapter examples of $\mathcal{WD}$ games (e.g., in Lemma 110) and fair games (e.g., in Example 2). In the following section, we will say that $\mathcal{L}$eft dominates in $(S_\mathcal{L}, S_\mathcal{R})$ to mean the ruleset $(S_\mathcal{L}, S_\mathcal{R})$ is strongly dominating (i.e., $\mathcal{SD}$) for $\mathcal{L}$eft. Before considering the dominance property of specific Partizan Subtraction games, we start by studying these games from a complexity point of view. In the following, if A is a set of integers, and x an integer, then $A+x$ denotes the set obtained by shifting the elements of A by x, in other words, $A+x = \{y+x,\ y \in A\}$. Additionally, if B is a set of integers, then we also will also write $A+B$ as the set $A+B = \{x+y,\ x \in A, y \in B\}$.

9.2 Complexity

Computing the outcome of a game position is a natural question when studying combinatorial games. For partizan subtraction games, we know that the outcome sequence is eventually periodic. This implies that, if $S_\mathcal{L}$ and $S_\mathcal{R}$ are fixed then computing the outcome of a given position n can be done in polynomial time[1]. However, if the subtraction sets are part of the input, then the algorithmic complexity of the problem is not so clear. This problem can be expressed as follows:

PSG Outcome
Input: two sets of integers $S_\mathcal{L}$ and $S_\mathcal{R}$, a game position n
Output: the outcome of n for the game $(S_\mathcal{L}, S_\mathcal{R})$

In the next result, we show that this problem is in fact **NP**-hard. Note that it is not clear whether the problem is in **NP** or not, and the complexity of the problem could be higher. Recall

[1]Note that this only works if we consider the game on a single heap. On multiple heaps, the complexity is still open, even if the subtraction sets are fixed.

from Section 7.3 that in the case of impartial subtraction games (i.e. if $S_{\mathcal{L}} = S_{\mathcal{R}}$), there is no known result about the complexity of this problem. This is surprising as these games have been thoroughly investigated in the literature.

Theorem 103. PSG Outcome *is* ***NP****-hard, even in the case where the set of one of the players is reduced to one element.*

Proof. We use a reduction from Unbounded Knapsack Problem defined below. This problem was shown to be **NP**-complete in [Lue75].

Unbounded Knapsack Problem
Input: a set S and an integer n
Output: can n be written as a sum of non-negative multiples of S?

Let S, n be an instance of unbounded knapsack problem, where S is a finite set of integers, and n is a positive integer. Without loss of generality, we can assume that $1 \notin S$ since otherwise the problem is trivial. We consider the partizan subtraction game where $\mathcal{L}$eft can only play 1, and $\mathcal{R}$ight can play any number x such that $x + 1 \in S$. In other words, we have $S_{\mathcal{L}} = \{1\}$ and $S_{\mathcal{R}} = S - 1$. We claim that for this game, $\mathcal{R}$ight has a winning strategy playing second if and only if n can be written as a sum of non-negative multiples of elements of S.

Observe that during one round (i.e. one move of $\mathcal{L}$eft followed by one move of $\mathcal{R}$ight), if x is the number of tokens that were removed, then $x \in S$. Suppose that $\mathcal{R}$ight has a winning strategy, and consider any play where $\mathcal{R}$ight plays according to this strategy. Then $\mathcal{R}$ight makes the last move, and after this move no token remains. Indeed, if there was at least one token remaining, then $\mathcal{L}$eft could still remove this token and continue the game. At each round an element of S was removed, and at the end, no tokens remains. This implies that n is a sum of non-negative multiples of S.

In the other direction, if n is a sum of non-negative multiples of S, we can write $n = \sum_{x \in S} n_x x$. A winning strategy for $\mathcal{R}$ight is simply to play n_x times the number $(x - 1)$ for each $x \in S$. □

The second question that emerged from partizan subtraction games is the behaviour of the outcome sequence, according to Definition 9. It can also be formulated as a decision problem.

PSG Sequence
Input: two sets of integers $S_{\mathcal{L}}$ and $S_{\mathcal{R}}$
Output: is the game $(S_{\mathcal{L}}, S_{\mathcal{R}})$ $\mathcal{SD}$, $\mathcal{WD}$ (and not $\mathcal{SD}$), $\mathcal{F}$ or $\mathcal{UI}$?

Unlike PSG Outcome, the algorithmic complexity is open for PSG Sequence. The next sections will consider this problem for some particular cases. In particular, we will investigate some conditions on the subtraction sets which ensure one type of outcome sequence. In addition, one can wonder whether the knowledge of the behaviour of the outcome sequence could help to compute the outcome of a game position. The answer is no, even if the game is $\mathcal{SD}$:

Proposition 104. *Let* $S_{\mathcal{L}} = \{a_1, \ldots, a_n\}$ *be such that* $\gcd(a_1+1, \ldots, a_n+1) = 1$*, and let* $S_{\mathcal{R}} = \{1\}$*. The game* $(S_{\mathcal{L}}, S_{\mathcal{R}})$ *is* $\mathcal{SD}$ *for* $\mathcal{L}$*eft but computing the length of the preperiod is NP-hard.*

The proof will be based on the well-known Coin Problem (also called Frobenius problem).

Coin Problem
Input: a set of n positive integers $a_1, \ldots, a_n$ such that $\gcd(a_1, \ldots, a_n) = 1$
Output: the largest integer that cannot be expressed as a positive linear combination of $a_1, \ldots, a_n$.

This value is called the *Frobenius number*. For $n = 2$, the Frobenius number equals $a_1a_2 - a_1 - a_2$ [S+84][2]. No explicit formula is known for larger values of n. Moreover, computing the Frobenius number was proved to be NP-hard in the general case [RA96].

Proof. Under the assumptions of the proposition, we will show that the length of the preperiod is exactly the Frobenius number of $\{a_1 + 1, \ldots, a_n + 1\}$. Indeed, let N be the Frobenius number of $\{a_1 + 1, \ldots, a_n + 1\}$. Then $N + 1, N + 2 \ldots$ can be written as a linear combinations of $\{a_1 + 1, \ldots, a_n + 1\}$. Note that in the game $(S_\mathcal{L}, S_\mathcal{R})$, any round (sequence of two moves) can be seen as a linear combination of $\{a_1 + 1, \ldots, a_n + 1\}$, as $\mathcal{L}$eft plays an a_i and $\mathcal{R}$ight plays 1. Hence if $\mathcal{R}$ight starts from $N + 1$, $\mathcal{L}$eft follows the linear combination for $N + 1$ to choose her moves, so as to play an even number of moves until the heap is empty. For the same reasons, if $\mathcal{R}$ight starts from $N + 2$, $\mathcal{L}$eft has a winning strategy as a second player. Since $\mathcal{R}$ight's first move is necessarily 1, it means that $\mathcal{L}$eft has a winning strategy as a first player from $N + 1$. Thus the position $N + 1$ has outcome $\mathcal{L}$. Using the same arguments, this remains true for all positions greater than $N + 1$. In other words, it proves that the game is $\mathcal{SD}$ for $\mathcal{L}$eft. Now, we consider the position N and show that $o(N) \neq \mathcal{L}$. Indeed, assume that $\mathcal{R}$ight starts and $\mathcal{L}$eft has a winning strategy. It means that an even number of moves will be played. According to the previous remark, the sequence of moves that is winning for $\mathcal{L}$eft is necessarily a linear combination of $\{a_1 + 1, \ldots, a_n + 1\}$. This contradicts the Frobenius property of N. □

This correlation between partizan subtraction games and the coin problem will be reused later in the chapter. We now continue with the study of Partizan Subtraction games for specific subtraction sets.

9.3 When $S_\mathcal{L}$ is fixed

In this section, we consider the case where $S_\mathcal{L}$ is fixed and study the behaviour of the sequence when $S_\mathcal{R}$ varies. In particular, we look for sets $S_\mathcal{R}$ that make the game $(S_\mathcal{L}, S_\mathcal{R})$ favourable for $\mathcal{R}$ight. This can be seen as a prelude to the game where players would choose their sets before playing: if $\mathcal{L}$eft has chosen her set $S_\mathcal{L}$, can $\mathcal{R}$ight force the game to be asymptotically more favourable for him?

9.3.1 The case $|S_\mathcal{R}| > |S_\mathcal{L}|$

If $S_\mathcal{R}$ can be larger than $S_\mathcal{L}$, then it is always possible to obtain a game favourable for $\mathcal{R}$ight, as it is proved in the following theorem.

Theorem 105. *Let $S_\mathcal{L}$ be any finite set of integers. Let p be the period of the impartial subtraction game played with $S_\mathcal{L}$ and let $S_\mathcal{R} = S_\mathcal{L} \cup \{p\}$. Then $\mathcal{R}$ight strongly dominates the game $(S_\mathcal{L}, S_\mathcal{R})$, i.e., the game $(S_\mathcal{L}, S_\mathcal{R})$ is ultimately $\mathcal{R}$.*

Proof. Let n_0 be the preperiod of the impartial subtraction game played on $S_\mathcal{L}$ and m be the maximal value of $S_\mathcal{L}$. We prove that $\mathcal{R}$ight wins if he starts on any heap of size $n > n_0 + p$, which implies that the outcome on $(S_\mathcal{L}, S_\mathcal{R})$ is $\mathcal{R}$ for any heap of size $n > n_0 + p + m$.

[2] Although not germane to this chapter, Sylvester's solution is central to the strategy stealing argument that proves that naming a prime 5 or greater is a winning move in SYLVER COINAGE[BCG04, pages 610-631].

If n is a $\mathcal{N}$-position for the impartial subtraction game on $S_{\mathcal{L}}$, then $\mathcal{R}$ight follows the strategy for the first player and never use the value p. Then the game is impartial and $\mathcal{R}$ight wins.

If n is a $\mathcal{P}$-position, $\mathcal{R}$ight takes p tokens and which leaves $\mathcal{L}$eft with a heap of size $n-p > n_0$ which is, using periodicity, also a $\mathcal{P}$-position in the impartial game. After $\mathcal{L}$eft's move, we are in the case of the previous paragraph and $\mathcal{R}$ight wins. □

Note that in the previous theorem, $S_{\mathcal{R}}$ contains the set $S_{\mathcal{L}}$, and thus has a large common intersection. We prove in the next theorem that if $S_{\mathcal{R}}$ cannot contain any value in $S_{\mathcal{L}}$, then it is still possible to have a game that is at least fair for $\mathcal{R}$ight (i.e., it contains an infinite number of $\mathcal{R}$-positions). Note that we do not know if for any set $S_{\mathcal{L}}$, there is always a set $S_{\mathcal{R}}$ with $|S_{\mathcal{R}}| = |S_{\mathcal{L}}|+1$ and $S_{\mathcal{R}} \cap S_{\mathcal{L}} = \emptyset$ that is (weakly or strongly) dominating for $\mathcal{R}$ight.

Theorem 106. *For any set $S_{\mathcal{L}}$, there exists a set $S_{\mathcal{R}}$ with $S_{\mathcal{L}} \cap S_{\mathcal{R}} = \emptyset$ and $|S_{\mathcal{R}}| = |S_{\mathcal{L}}| + 1$ such that the resulting game contains an infinite number of $\mathcal{R}$-position.*

Proof. Let n be any integer such that the set $A = \{n-m, m \in S_{\mathcal{L}}\}$ is a set of positive integers that is disjoint from $S_{\mathcal{L}}$. Putting $S_{\mathcal{R}} = A \cup \{n\}$ gives a set which satisfies the condition of the theorem and in the game $(S_{\mathcal{L}}, S_{\mathcal{R}})$ all the multiples of n are $\mathcal{R}$-positions.

Indeed, if $\mathcal{L}$eft starts on a position kn with $k \in \mathbb{N}^*$ by removing m tokens, then $\mathcal{R}$ight can answer by taking $n-m$ tokens and leaves $(k-1)n$ tokens, and by induction, $\mathcal{R}$ight wins. If $\mathcal{R}$ight starts, he takes n tokens and again, $\mathcal{L}$eft has a multiple of n and loses. □

Consequently, if $\mathcal{R}$ight has a small advantage on the size of the set, he can ensure that the sequence of outcome contains an infinite number of $\mathcal{R}$-positions. So having a larger subtraction set seems to be an important advantage. However, having a larger set is not always enough to guarantee dominance. Indeed, we have the following result:

Theorem 107. *Let $S_{\mathcal{L}}$ and $S_{\mathcal{R}}$ be two sets of positive integers. Assume that $|S_{\mathcal{L}}| \geq 2$ and that $\mathcal{L}$eft dominates in $(S_{\mathcal{L}}, S_{\mathcal{R}})$, with a preperiod at most p. Let $x_1, x_2 \in S_{\mathcal{L}}$, with $x_1 < x_2$, and let d be an integer with $d > p + \max(S_{\mathcal{R}} \cup \{x_2 - x_1\})$, then $\mathcal{L}$eft also dominates in $(S_{\mathcal{L}}, S_{\mathcal{R}} \cup \{d\})$ and the preperiod is at most $(d+x_2)\lceil \frac{d+x_2}{x_2-x_1} \rceil$.*

Proof. Let $S_{\mathcal{L}}$, $S_{\mathcal{R}}$, d, x_1 and x_2 be as in the statement of the theorem. We start by proving the following claim:

Claim 108. Consider the ruleset $(S_{\mathcal{L}}, S_{\mathcal{R}} \cup \{d\})$. If $\mathcal{L}$eft has a winning strategy on $n \in \mathbb{N}$ as first (resp. second) player, then $\mathcal{L}$eft also has a winning strategy on $n + (d+x)$ as first (resp. second player), for any $x \in S_{\mathcal{L}}$.

Proof. We will show this result by induction on $n \geq 0$. First, assume that $\mathcal{L}$eft has a winning strategy on n as second player. We will show that there is a strategy for $\mathcal{L}$eft playing second on $n+d+x$. Starting from the position $n+d+x$, there are three possible cases:

- $\mathcal{R}$ight plays $y \in S_{\mathcal{R}}$, with $y \leq n$. By the assumption on n, $\mathcal{L}$eft wins as first player on $n-y$, and using the induction hypothesis, he also wins as first player on $n-y+d+x$. Therefore, $\mathcal{L}$eft wins as second player on $n+d+x$.

- $\mathcal{R}$ight plays $y \in S_{\mathcal{R}}$, with $y > n$. Now $\mathcal{L}$eft answers by playing x. This leads to the position $(n-y)+d$, with is such that $(n-y)+d > p$ by assumption on d. Additionally, $n-y+d < d$ by assumption on y. Since $n-y+d < d$, $\mathcal{R}$ight can no longer play his move d, and the position $n-y+d$ has the same outcome in $(S_{\mathcal{L}}, S_{\mathcal{R}} \cup \{d\})$ and $(S_{\mathcal{L}}, S_{\mathcal{R}})$. Since $n-y+d > p$ $\mathcal{L}$eft wins playing second on this position in both rulesets.

- $\mathcal{R}$ight plays d, then $\mathcal{L}$eft answers by playing x, leading to the position n on which $\mathcal{L}$eft wins as second player by assumption.

Suppose now that $\mathcal{L}$eft wins playing first on n, and let $y \in S_{\mathcal{L}}$ be a winning move for $\mathcal{L}$eft. Then $\mathcal{L}$eft wins playing second on $n-y$, and using the induction hypothesis, she wins playing second on $n-y+d+x$. Consequently, y is a winning move for $\mathcal{L}$eft on $n+d+x$. □

For $i \geq 0$, denote by X_i the set of integers $k < d+x_2$ such that the position $i(d+x_2)+k$ is an $\mathcal{L}$-position for ruleset $(S_{\mathcal{L}}, S_{\mathcal{R}} \cup \{d\})$. To prove the theorem, it is enough to show that if i is large enough, then $X_i = [0, x_2+d[$. From the claim above, we know that $X_i \subseteq X_{i+1}$.

Additionally, using the hypothesis on d, we have that $[p+1, d-1] \subseteq X_0$. Finally, we have the following property: for any $x \geq 0$, if $x \in X_i$ then $x-(x_2-x_1) \mod (d+x_2) \in X_{i+1}$. Indeed, if $x \in X_i$, then $i(d+x_2)+x$ is an $\mathcal{L}$-position, and using the claim above, so is $i(d+x_2)+x+d+x_1 = (i+1)(d+x_2)+x-(x_2-x_1)$.

Let $0 \leq x < d+x_2$, and write $(d-x) \mod (d+x_2) = \alpha(x_2-x_1)+\beta$ the euclidian division of $(d-x) \mod (d+x_2)$ by (x_2-x_1). We have $0 < \beta \leq x_2-x_1$, and $\alpha \leq \lceil \frac{d+x_2}{x_2-x_1} \rceil$. This can be rewritten as:

$$x = (d-\beta) - \alpha(x_2-x_1) \mod (d+x_2)$$

Since we know that $d-\beta \geq p$ by assumption on d, we have that $(d-\beta) \in X_0$, and using the observation above, this implies that $x \in X_\alpha \subseteq X_{\lceil \frac{d+x_2}{x_2-x_1} \rceil}$.

Consequently, $\mathcal{L}$eft dominates in $(S_{\mathcal{L}}, S_{\mathcal{R}} \cup \{d\})$ with a preperiod at most $(d+x_2)\lceil \frac{d+x_2}{x_2-x_1} \rceil$. □

By applying iteratively Theorem 107 with a game that is initially $\mathcal{SD}$ for $\mathcal{L}$eft (like the one from Example 1), we obtain the following corollary.

Corollary 109. *There are sets $S_{\mathcal{L}}$ and $S_{\mathcal{R}}$ with $|S_{\mathcal{L}}| = 2$ and $|S_{\mathcal{R}}|$ arbitrarily large such that $(S_{\mathcal{L}}, S_{\mathcal{R}})$ is $\mathcal{SD}$ for $\mathcal{L}$eft.*

The condition that $d > p + \max(S_{\mathcal{R}} \cup \{x_2-x_1\})$ in Theorem 107 is optimal. Indeed, if we take $S_{\mathcal{L}} = \{c, c+1\}$ and $S_{\mathcal{R}} = \{1\}$, as seen in the proof of Proposition 104 the game $(S_{\mathcal{L}}, S_{\mathcal{R}})$ is $\mathcal{SD}$ for $\mathcal{L}$eft, with preperiod the Froebenius number of $\{c+1, c+2\}$, which is $p = (c+1)(c+2)-(c+1)-(c+2) = c(c+1)-1$.

Thus, by Theorem 107, the game $(\{c, c+1\}, \{1, d\})$ with $d > c(c+1)$ is also $\mathcal{SD}$ for $\mathcal{L}$eft. But, the example below shows that this is not true for $d = c(c+1)$ since this game is $\mathcal{F}$.

Example 2. Let $S_{\mathcal{L}} = \{c, c+1\}$ and $S_{\mathcal{R}} = \{1, d\}$ with $d = c(c+1)$ and $c > 1$. Then the game $(S_{\mathcal{L}}, S_{\mathcal{R}})$ is $\mathcal{F}$.

Proof. Let us start by showing by induction on n that $o(n) = o(n+d+c)$ for all $n \geq 0$. If $n = 0$, then we only need to prove that $o(d+c) = \mathcal{P}$. On this position, $\mathcal{R}$ight playing first can either

remove d tokens, in which case $\mathcal{L}$eft can remove the c remaining ones and win, or remove one token in which case $\mathcal{L}$eft can answer $c+1$. In this second case, after the two moves the position becomes $d-2=(c-1)(c+2)$. From this position, $\mathcal{R}$ight can only play 1, and $\mathcal{L}$eft can always remove $c+1$ tokens removing $c+2$ tokens in two rounds. Since $(c+2)$ is a divisor of $d-2=(c-1)(c+2)$, the game ends after an even number of rounds with no tokens remaining, and $\mathcal{L}$eft wins the game.

If $\mathcal{L}$eft plays first, then she can play either c, in which case $\mathcal{R}$ight answers with d and win, or she can play $c+1$ to the position $c(c+1)-1$. Since $c(c+1)-1$ is the Frobenius number of $\{c+1, c+2\}$, using the same argument as in the proof of Proposition 104 it follows that $\mathcal{L}$eft playing second on $c(c+1)-1$ loses.

Let us now assume that $n>0$, suppose that the property holds for every integer smaller than n. If $\mathcal{L}$eft (resp. $\mathcal{R}$ight) wins playing first on n by removing x tokens, then she can win as the first player on the position $n+d+c$ by playing x to reach the position $(n-x)+d+c$. Since $\mathcal{L}$eft (resp. $\mathcal{R}$ight) has a winning strategy playing second on $(n-x)$, using the induction hypothesis she also has a winning strategy playing second on $(n-x)+d+c$.

If $\mathcal{L}$eft wins playing second on n, then starting from the position $n+d+c$, either $\mathcal{R}$ight plays d, and $\mathcal{L}$eft can answer with c and win by assumption on n. Or, $\mathcal{R}$ight plays 1 to the position $n-1+d+c$. Since $\mathcal{L}$eft wins playing second on n, she must win playing first on $n-1$. Using the induction hypothesis, she also wins playing first on $n-1+c+d$, and consequently $\mathcal{L}$eft wins playing second on $n+c+d$.

Finally, assume that $\mathcal{R}$ight has a winning strategy on n playing second. On position $n+c+d$, $\mathcal{L}$eft start by playing c, then $\mathcal{R}$ight can answer with d and win. If $\mathcal{L}$eft plays $c+1$, we distinguish two sub-cases:

- $n \geq c+1$, in which case $\mathcal{R}$ight wins playing first on $n-c-1$ by assumption on n, and using the induction hypothesis she also wins playing first on $n-+d-1$.
- $n<c+1$, in which case $\mathcal{R}$ight can answer by playing d to the position $n-1$. On this position $\mathcal{L}$eft has no move available since $n-1<c$.

In both cases, $\mathcal{R}$ight wins as the second player on $n+c+d$. This ends the induction step, and shows that for every $n \geq 0$, we have $o(n)=o(n+c+d)$. The result of the lemma immediately follows from the observation that $o(1)=\mathcal{R}$ since $c>1$, and $o(c+2)=\mathcal{L}$. □

9.3.2 The case $|S_{\mathcal{R}}| \leq |S_{\mathcal{L}}|$

We first consider the case $S_{\mathcal{L}}=\{1,\ldots,k\}$ and prove that the game is always favourable to $\mathcal{L}$eft who strongly dominates in all but a few cases.

Lemma 110. *Let $S_{\mathcal{L}}=\{1,\ldots,k\}$, and $|S_{\mathcal{R}}|=k$, then:*

1. *If $S_{\mathcal{R}}=\{c+1, c+2, \ldots c+k\}$ for some integer c, then $\mathcal{L}$eft weakly dominates if $c>0$ and the game is impartial if $c=0$,*
2. *otherwise, $\mathcal{L}$eft strongly dominates.*

Proof. 1. In this case, the game is purely periodic, with period $\mathcal{P}\mathcal{L}^c\mathcal{N}^k$. This can be proved by induction on the size of the heap n. If $0<n\leq c$, only $\mathcal{L}$eft can play and the game is trivially $\mathcal{L}$. Otherwise, let $x=n \mod (c+k+1)$. If $x=0$, then if the first player removes i tokens, the second player answers by removing $c+k+1-i$ tokens, leading to the position $n-c-k-1$

which is $\mathcal{P}$ by induction, and so is n. If $0 < x < c+1$, when $\mathcal{L}$eft starts she takes one token, leading to a $\mathcal{L}$ or a $\mathcal{P}$-position, and wins. If she is second, she plays as before to $n-c-k-1$ which is a $\mathcal{L}$-position. Finally, if $x \geq c+1$, both players win playing first by playing $x-c$ for $\mathcal{L}$eft and x for $\mathcal{R}$ight.

2. We show that if we assume that there is a position $n > 0$ such that $\mathcal{R}$ight wins playing second on n, then $S_{\mathcal{R}}$ contains k consecutive integers. Let n_0 be the smallest positive integer for which $\mathcal{R}$ight wins playing second on n_0. We know that $n_0 > k$ since otherwise $\mathcal{L}$eft can win playing first by playing to zero. Since $\mathcal{R}$ight has a winning strategy playing second then $\mathcal{R}$ight has a winning first move on all the position $n-i$ for $1 \leq i \leq k$. This means that for each of these positions, $\mathcal{R}$ight has a winning move to some position m_i such that $\mathcal{R}$ight wins playing second on m_i. By minimality of n_0, this implies that $m_i = 0$, and consequently $n - i \in S_{\mathcal{R}}$ for all $1 \leq i \leq k$. Consequently, if $S_{\mathcal{R}}$ does not contain k consecutive integers, there is no position $n > 0$ such that $\mathcal{R}$ight wins playing second. In particular, there is no $\mathcal{R}$ nor $\mathcal{P}$-positions in the period. By Remark 3, this implies that the period only contains $\mathcal{L}$-positions, meaning that the game is strongly dominating for $\mathcal{L}$eft.

□

The set $S_{\mathcal{L}} = \{1, \ldots, k\}$ is somehow optimal for $\mathcal{L}$eft, since the exceptions of strongly domination for $\mathcal{L}$eft in the previous lemma appear for any set of k elements:

Lemma 111. *For any set $S_{\mathcal{L}}$, there is a set $S_{\mathcal{R}}$ with $|S_{\mathcal{R}}| = |S_{\mathcal{L}}|$ and $S_{\mathcal{R}} \cap S_{\mathcal{L}} = \emptyset$ such that $\mathcal{L}$eft does not strongly dominate.*

Proof. Let $S_{\mathcal{R}} = n_0 - S_{\mathcal{L}}$ for an integer n_0 larger than all the values of $S_{\mathcal{L}}$ and such that $S_{\mathcal{R}} \cap S_{\mathcal{L}} = \emptyset$. Then $\mathcal{R}$ight wins playing second in all the multiples of n_0. □

9.4 When one set has size 1

We now consider the case where one of the set, say $S_{\mathcal{R}}$ has size 1. As seen in Section 9.2, the study of the game is closely related to Unbounded Knapsack Problem and to the coin problem. Indeed, $\mathcal{R}$ight does not have any choice and thus the result is only depending on the possibility or not for n to be decomposed as a combination of the values in $S_{\mathcal{L}} + S_{\mathcal{R}}$. Our aim in this section is to exhibit the precise periods.

9.4.1 Case $|S_{\mathcal{L}}| = |S_{\mathcal{R}}| = 1$

In this really particular case, the game is always $\mathcal{WD}$ for the player that have the smallest integer.

Lemma 112. *Let $S_{\mathcal{L}} = \{a\}$ and $S_{\mathcal{R}} = \{b\}$ with $a < b$. The outcome sequence of $(S_{\mathcal{L}}, S_{\mathcal{R}})$ is purely periodic, the period length is $a+b$ and the period is $\mathcal{P}^a\mathcal{L}^{b-a}\mathcal{N}^a$. In particular, the game is weakly dominating for $\mathcal{L}$eft.*

Proof. We prove that for all $n \geq 0$, if one of the player has a winning move playing first (resp. second) on n, then he also has one playing first (resp. second) on $n+a+b$. Indeed, suppose for example that $\mathcal{L}$eft has a winning move on position n playing first (the other cases are treated in the same way). If $\mathcal{L}$eft plays first on position $n+a+b$, then after two moves, it's again $\mathcal{L}$eft's turn to play, and the position is now n, and $\mathcal{L}$eft wins the game.

Heap sizes	$\mathcal{L}$eft move range	$\mathcal{R}$ight move range	Outcome
$[0, a-1]$	no moves	no moves	$\mathcal{P}$
$[a, b-1]$	$[0, a-1]$	no moves	$\mathcal{L}$
$[b, b+a-1]$	$[b-a, b-1]$	$[0, a-1]$	$\mathcal{N}$
$[b+a, b+2a-1]$	$[b, b+a-1]$	$[a, 2a-1]$	$\mathcal{P}$
$[b+2a, 2b+2a-1]$	$[b+a, 2b+a-1]$	$[2a, b+2a-1]$	$\mathcal{L}$

Table 9.1: Outcomes with $S_{\mathcal{L}} = \{a\}$ and $S_{\mathcal{R}} = \{b\}$ for first values

The result then follows from computing the outcome of the positions $n \leq a+b$. These outcomes are tabulated in Table 9.1.

□

9.4.2 Case $|S_{\mathcal{L}}| = 2$ and $|S_{\mathcal{R}}| = 1$

In these cases, we are able to give the complete periods.

Theorem 113. *Let a, b and c be three positive integers, and let $g = \gcd(a+c, b+c)$. The game $(\{a, b\}, \{c\})$ is:*

- *strongly dominated by $\mathcal{L}$eft if $g \leq c$,*
- *weakly dominated by $\mathcal{L}$eft with period $(\mathcal{P}^{g-c}\mathcal{L}^{2c-g}\mathcal{N}^{g-c})$ if $c < g < 2c$,*
- *ultimately impartial with period $(\mathcal{P}^c\mathcal{N}^c)$ if $g = 2c$,*
- *weakly dominated by $\mathcal{R}$ight with period $(\mathcal{P}^c\mathcal{R}^{g-2c}\mathcal{N}^c)$ if $g > 2c$.*

Proof. Throughout this proof we write $n = qg + r$, with $0 \leq r < c$.

We start by proving the following claim which holds in all four cases.

Claim 114. If $(n \bmod g) < c$ and n is large enough then $\mathcal{L}$eft has a winning move on n playing second.

Proof. After both players play once, the number of tokens decreased by either $a + c$ or $b + c$ depending on which move $\mathcal{L}$eft played. By the results on the coin problem, we know that if q is large enough, then qg can be written as $\alpha(a+c)+\beta(b+c)$, with α and β two non-negative integers. If $\mathcal{L}$eft is playing second, a strategy can be to play a α times, and b β times. After these moves, it is $\mathcal{R}$ight's turn to play, and the position is $r < c$. Consequently $\mathcal{R}$ight has no move left and loses the game. □

We will now use this claim to prove the result in the four different cases.

For the first case, we have $g \leq c$. For any integer n, we have $(n \mod g) < g \leq c$. Consequently, by the claim above, there is an integer n_0 such that for any $n \geq n_0$, $\mathcal{L}$eft wins playing second on n. This also implies that for any $n \geq n_0 + a$, $\mathcal{L}$eft also wins as first player by playing a. Indeed, $\mathcal{L}$eft plays to the position $n - a$ on which $\mathcal{L}$eft has a winning strategy as second player by the claim above. Thus the outcome is $\mathcal{L}$ for any position n large enough.

For the three remaining cases, we will show that the following four properties hold when n is large enough. The result of the theorem immediately follows from these four properties.

1. if $r < c$, then $\mathcal{L}$eft wins playing second,
2. if $r \geq g - c$, then $\mathcal{L}$eft wins playing first,
3. if $r \geq c$, then $\mathcal{R}$ight wins playing first,
4. if $r < g - c$, then $\mathcal{R}$ight wins playing second.

We now prove these four points:

1. This point is exactly the claim above.
2. If $r \geq g - c$, and n is large enough, then $\mathcal{L}$eft can play a. The position after the move is such that $n - a = r - a = r + c \mod g$. Moreover, since $g - c \leq r < g$, we know that $g \leq r + c < g + c$. From the first item, we know that $\mathcal{L}$eft wins playing second on this position if $n - a$ is large enough, so $\mathcal{L}$eft has a winning strategy as a first player if $r \geq g - c$.
3. If $r \geq c$, and $\mathcal{R}$ight plays first, then whatever $\mathcal{L}$eft plays, after an even number of moves, $\mathcal{R}$ight still has a move available. Indeed, let n' be the position reached after an even number of moves. The number of tokens removed, $n - n'$ is a multiple of g. Consequently, we have $n' = (n \mod g)$. Since $(n \mod g) \geq c$, this implies that $n' \geq c$, and $\mathcal{R}$ight can play c. This proves that $\mathcal{R}$ight will never be blocked, and $\mathcal{L}$eft will eventually lose the game.
4. Finally, if $r < g - c$, then $\mathcal{L}$eft playing first can move to a position n' equal to either $n - a$ or $n - b$. Since $a = b = -c \mod g$, in both cases, we have $n' = r + c \mod g$. Since $c \leq r + c < g$, by the argument above, we know that $\mathcal{R}$ight playing first on n' wins. Consequently, $\mathcal{L}$eft playing first on n looses.

□

When $c > b$ and $b \geq 2a$, which is included in the first case, we know the whole outcome sequence. This will be useful in next Section.

Theorem 115. *The outcome sequence of the game $(\{a, b\}, \{c\})$, with $c > b$ and $b \geq 2a$ is the following:*

$$\mathcal{P}^a \mathcal{L}^{c-a} \mathcal{N}^a \mathcal{L}^\infty$$

Proof. We show the result by induction on n, the position of the game.

- If $n < a$, then none of the player has a move, and thus $o(n) = \mathcal{P}$.
- If $a \leq n < c$, then only $\mathcal{L}$eft has a valid move, and thus $o(n) = \mathcal{L}$.
- If $c \leq n < a + c$, then $\mathcal{R}$ight has a winning move to a position $n - c < a$ which has outcome $\mathcal{P}$, and $\mathcal{L}$eft has a winning move to a position with outcome either $\mathcal{P}$ or $\mathcal{L}$. Consequently, we have $o(n) = \mathcal{L}$.
- Finally, if $n \geq a + c$, then as the first player $\mathcal{R}$ight has no winning move, and $\mathcal{L}$eft has at least one winning move. Indeed, since $k \geq a$, we can't have at the same time $n - a$ and $n - a - k$ in the interval $[c, a + c[$. So at least one of $n - a$ and $n - a - k$ is not in this interval, and is either a $\mathcal{P}$-position or a $\mathcal{L}$-position by induction.

□

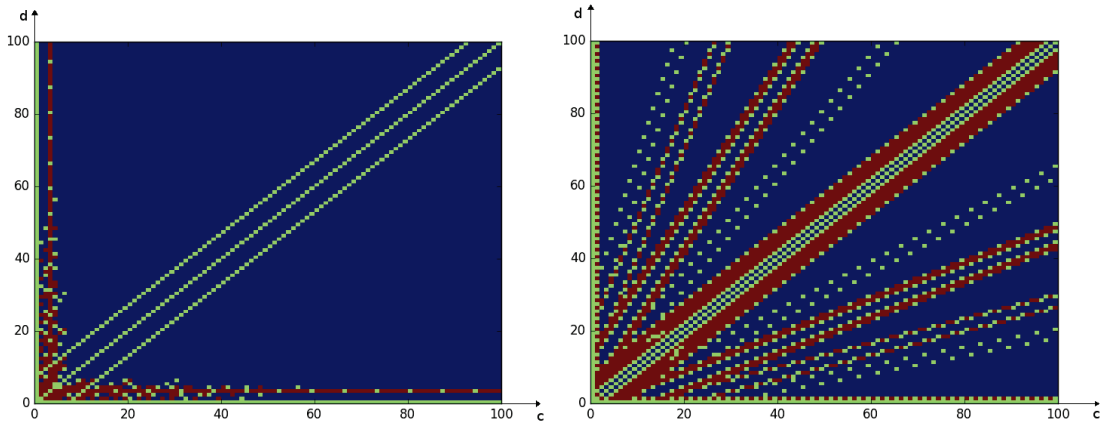


Figure 9.2: Properties of the outcome sequences for $(\{a, b\}, \{c, d\})$. The parameters a and b are fixed, and the pictures are obtained by varying the parameters c and d. The point at coordinate (c, d) is blue if $\mathcal{L}$eft dominates, red if $\mathcal{R}$ight dominates, and green if there is a mixed period.

9.5 When both sets have size 2

The goal of this section is to investigate the sequence of outcomes for $(S_\mathcal{L}, S_\mathcal{R})$ with $S_\mathcal{L} = \{a, b\}$ and $S_\mathcal{R} = \{c, d\}$. In particular, if we suppose that a and b are fixed, we would like to characterize for which choice of c and d we can ensure that $\mathcal{L}$eft dominates. The pictures on Figure 9.2 give an insight of what is happening. On the figure on the left, we have an example with $b \geq 2a$. In this case, $\mathcal{L}$eft almost always dominates, except when the point (c, d) is close to the diagonal or to one of the axis. When (c, d) is close to the diagonal (i.e., when $|d - c|$ is close to zero), the behaviour seems more complicated, and we don't have a characterization for this case.

When $b < 2a$, the behaviour is more complex, but shares some similarities with the previous case. From the picture on the right in Figure 9.2 we can see that there are some lines such that if the point (c, d) is far enough from these lines, then the game is eventually $\mathcal{L}$. Again, when the point is close to these lines, again the behaviour is more complicated, and we won't try to characterize it here. In all cases, we can see that if a and b are fixed, for almost all of the choices of c and d, $\mathcal{L}$eft dominates.

In the rest of this section, we will assume that we have $d > c > b$. We start by the case $b \geq 2a$ which is easier to analyse.

9.5.1 Case $b \geq 2a$

We start by the case where $b \geq 2a$, and show that in this case $\mathcal{L}$eft dominates if (c, d) is far enough from the diagonal and from the coordinates axes.

Theorem 116. *Assume $b \geq 2a$, and $d > c + b$, then $\mathcal{L}$eft dominates. More precisely, the outcome sequence is:*

$$\mathcal{P}^a \mathcal{L}^{c-a} \mathcal{N}^a \mathcal{L}^{d-c-a} \mathcal{N}^a \mathcal{L}^\infty.$$

Proof. We will show the result of the lemma by induction on n, the starting position of the game. If $n < d$, then the outcome of n is the same for rules $(\{a, a + k\}, \{c\})$ or $(\{a, a + k\}, \{c\})$ since d

cannot be played. Consequently we can just apply Theorem 115, and get the desired result. If $n \geq d$ there are two possible cases:

- If $d \leq n < d + a$, then $\mathcal{R}$ight has a winning move to the position $n - d < a$, and $\mathcal{L}$eft has a winning move by playing his strategy for n with rules $(\{a, b\}, \{c\})$. Indeed, this leads to a position $n - x < d$ for some $x \in \{a, b\}$. This position has outcome $\mathcal{P}$ or $\mathcal{L}$ for the rules $(\{a, b\}, \{c\})$, and consequently also for the rules $(\{a, b\}, \{c, d\})$, since d cannot be played anymore at this point.

- If $n \geq d+a$, denote by I_1 and I_2 the two intervals containing the $\mathcal{N}$-position, i.e., $I_1 = [c, c+a[$, and $I_2 = [d, d + a[$. Since $b \geq 2a$, we can't have that $n - a$ and $n - b$ are both in I_1, or both in I_2. Additionally, since $d > c + b$, we can have both $n - b \in I_1$ and $n - a \in I_2$ at the same time. Consequently, one of $n - a$ and $n - a - k$ has outcome either $\mathcal{L}$ or $\mathcal{P}$, and $\mathcal{L}$eft has a winning move on n.

□

9.5.2 General case

In the general case, we will again prove that if we fix a and b, for most choices of c and d the outcome is ultimately $\mathcal{L}$. The exceptional cases are slightly more complicated to characterize. The characterization is related to the following definition:

Definition 10. Given an integer a, and a real number $\alpha \geq 1$, we denote by $T_{a,\alpha}$ the set of points defined by:

$$T_{a,\alpha} = \left\{ (c, d), \quad \gcd(a + c, a + d) \geq \frac{\max(c, d)}{\alpha} \right\} .$$

Note that by definition, $T_{a,\alpha}$ can be obtained from $T_{0,\alpha}$ by a translation of $(-a, -a)$. We can also remark that, for any α and β, with $\beta \geq \alpha$, we have $T_{0,\alpha} \subseteq T_{0,\beta}$. We now prove some properties of the sets $T_{a,\alpha}$ which will be usefull for the proofs later on.

Lemma 117. *Assume that there are some positive integers x, y, u and v such that $xu - yv = 0$ with $(u, v) \neq (0, 0)$, then $(x, y) \in T_{0,\max(u,v)}$.*

Proof. Up to dividing u and v by $\gcd(u, v)$, we can assume that u and v are coprimes. The equation can be written $xu = yv$. Consequently, u is a divisor of yv, and since u and v are coprimes, this means that u is a divisor of v. We can write $y = gu$, and consequently we have $xu = yv = vgu$. This means that $x = vg$, and $g = \gcd(x, y)$. Consequently, $\frac{\max(x,y)}{\gcd(x,y)} = \max(u, v)$, and $(x, y) \in T_{0,\max(u,v)}$. □

Given two points $p = (x, y)$ and $p' = (x', y')$, we denote by $\mathbf{d}(p, p')$ the distance between these two points according to the 1-norm: $\mathbf{d}(p, p') = |x - x'| + |y - y'|$. If $\mathcal{D}$ is a subset of $\mathbb{N}^2$, we denote by $\mathbf{d}(p, \mathcal{D}) = \min\{\mathbf{d}(p, p''), p'' \in \mathcal{D}\}$ the distance of the point p to the set $\mathcal{D}$.

Lemma 118. *Assume that there are some positive integers x, y, u, v and a such that $|xu - yv| \leq a$, then $\mathbf{d}((x, y), T_{0,\max(u,v)}) \leq a(u + v)$.*

Proof. Let $r = xu - yv$, with $|r| \leq a$, and $g = \gcd(u, v)$. By definition, r is a multiple of g, and we can write $r = qg$ for some integer q. Additionally, by Bézout's identity, we know that there exists two integers u' and v' such that $uu' + vv' = g$, and $|u'| \leq u$ and $|v'| \leq v$. Consider the point (x', y'), with $x' = x - qu'$, and $y' = y + qv'$. We have the following:

$$x'u - y'v = xu + yv - q(uu' + vv') = r - qg = 0$$

By Lemma 117, we know that $(x', y') \in T_{0,\max(u,v)}$. Additionally, $\mathbf{d}((x, y), (x', y')) = |qu'| + |qv'| \leq |r|(u + v) \leq a(u + v)$. This proves the Lemma. □

The two lemmas above relate the set $T_{a,\alpha}$ to lines of equation $xu - yv = 0$ for some fixed u and v. Using these results we show in the lemma below that $T_{a,\alpha}$ can be written as a union of such lines.

Lemma 119. *For any a and α, the set $T_{a,\alpha}$ is the union of a finite set of lines.*

Proof. Since $T_{a,\alpha}$ can be obtained from $T_{0,\alpha}$ by a translation, we only need to prove the result in the case $a = 0$. Let $\mathcal{D}$ be the union of the lines with equation $xu - yv = 0$, for all $u, v \leq \alpha$. The set $\mathcal{D}$ is the union of a finite number of lines. By Lemma 117, we know that $\mathcal{D} \subseteq T_{0,\alpha}$. Reciprocally, let (x, y) be a point in $T_{0,\alpha}$, and let $g = \gcd(x, y)$. Since $(x, y) \in T_{0,\alpha}$, it follows that $y \geq \frac{g}{\alpha}$. We can also write $x = x'g$, and $y = y'g$ for some integers x' and y'. We have the following:

$$xy' - yx' = x'y'g - y'gx = 0$$

Additionally, we have $x' = \frac{x}{g} \leq x\frac{\alpha}{\max(x,y)} \leq \alpha$, and similarly for y'. Consequently, by Lemma 117 it follows that $(x, y) \in \mathcal{D}$, and $T_{0,\alpha} = \mathcal{D}$. □

The goal in the remaining of this section is to prove the following theorem:

Theorem 120. *Let a, b, c and d be positive integers, let $A = \lceil \frac{a}{b-a} \rceil + 1$. Assume that $\mathbf{d}((c, d), T_{a,A}) \geq 2A(a + 2b)$, then $\mathcal{L}$eft dominates for the partizan subtraction game $(\{a, b\}, \{c, d\})$.*

The proof of this theorem will be done by giving a characterization of the sets of $\mathcal{P}$-, $\mathcal{N}$-, and $\mathcal{L}$-positions for this ruleset. This characterisation is done using the following definition. Given two integers i and j, we define the following intervals:

- $I^{\mathcal{P}}_{i,j} = [\alpha_{i,j}, \alpha_{i,j} + a - (i + j)(b - a)[$
- $I^{\mathcal{N}}_{i,j} = [\beta_{i,j}, \beta_{i,j} + a - (i + j - 1)(b - a)[$

where

- $\alpha_{i,j} = i(d + b) + j(c + b)$,
- and $\beta_{i,j} = \alpha_{i,j} - b$.

Denote by $I^{\mathcal{P}}$ the set $\cup_{i,j} I^{\mathcal{P}}_{i,j}$, and similarly, $I^{\mathcal{N}} = \cup_{i,j} I^{\mathcal{N}}_{i,j}$. Note that $I^{\mathcal{P}}_{i,j}$ is empty if $i + j \geq \lceil \frac{a}{k} \rceil$, and $I^{\mathcal{N}}_{i,j}$ is empty if $i + j \geq \lceil \frac{a}{k} \rceil + 1$. Our goal is to show that, under the conditions in the statement of the theorem, the set $I^{\mathcal{N}}$ is the set of $\mathcal{N}$-positions, $I^{\mathcal{P}}$ the set of $\mathcal{P}$-positions, and all the other positions have outcome $\mathcal{L}$. In particular, since both $I^{\mathcal{P}}$ and $I^{\mathcal{N}}$ are finite, this will imply that the outcome sequence is eventually $\mathcal{L}$. Before proving this result, we show that under the conditions of the theorem the intervals $I^{\mathcal{P}}_{i,j}$ and $I^{\mathcal{N}}_{i,j}$ satisfy the following properties.

Lemma 121. *Fix the parameters a and b, and let $A = \lceil \frac{a}{b-a} \rceil + 1$. Assume that c and d are such that $\mathbf{d}((c,d), T_{b,A}) \geq 2A(a+2b)$, then the intervals $I^{\mathcal{N}}_{i,j}$ and $I^{\mathcal{P}}_{i,j}$ satisfy the following properties:*

(i) they are pairwise disjoint,

(ii) there is no interval $I^{\mathcal{P}}_{i',j'}$ or $I^{\mathcal{N}}_{i',j'}$ intersecting any of the b positions preceding $I^{\mathcal{N}}_{i,j}$,

(iii) $I^{\mathcal{P}}_{i,j} + c = I^{\mathcal{N}}_{i,j+1}$,

(iv) $I^{\mathcal{P}}_{i,j} + d = I^{\mathcal{N}}_{i+1,j}$,

(v) $(I^{\mathcal{N}}_{i,j} + a) \cap (I^{\mathcal{N}}_{i,j} + b) = I^{\mathcal{P}}_{i,j}$.

Proof. The points (iii), (iv) and (v) are just consequences of the definitions of $I^{\mathcal{P}}_{i,j}$ and $I^{\mathcal{N}}_{i,j}$. Consequently, we only need to prove the two other points.

We know that $I^{\mathcal{N}}_{i,j}$ and $I^{\mathcal{P}}_{i,j}$ are empty when $i+j \geq \lceil \frac{a}{b-a} \rceil + 1 = A$, consequently, we will assume in all the following that the indices i, j, i' and j' are all upper bounded by A. We first show the following claim. The rest of the proof will simply consists in applying this claim several times.

Claim 122. Assume that there is an integers B, and indices $i, j, i', j' \leq A$, such that one of the following holds:

- $|\alpha_{i,j} - \alpha_{i',j'}| \leq B$
- $|\beta_{i,j} - \beta_{i',j'}| \leq B$
- $|\alpha_{i,j} - \beta_{i',j'}| \leq B$

Then in all three cases we have $\mathbf{d}((c,d), T_{b,A}) \leq 2A(B+b)$.

Proof. The first two cases are equivalent to the inequality $|(i-i')(d+b) + (j-j')(c+b)| \leq B$, and the result follows by applying Lemma 118. The third case is equivalent to $|(i-i')(d+b) + (j-j')(c+b) + b| \leq B$. Using the triangle inequality, this implies $|(i-i')(d+b) + (j-j')(c+b)| \leq B+b$, and the result follows from Lemma 118. □

We will prove the points (i) and (ii) by proving their contrapositives. In other words, assuming that one of these two conditions does not hold, we want to show that $\mathbf{d}((c,d), T_{b,\alpha}) \leq 2\alpha(a+b)$.

We first consider the point (i). First, assume that there are two intervals $I^{\mathcal{P}}_{i,j}$ and $I^{\mathcal{P}}_{i',j'}$ such that the two intervals intersect. Then, the left endpoint of one of these two intervals is contained in the other interval. Without loss of generality, we can assume that $\alpha_{i,j} \in I^{\mathcal{P}}_{i',j'}$. This implies:

$$\alpha_{i',j'} \leq \alpha_{i,j} \leq \alpha_{i',j'} + a - (b-a)(i'+j')$$
$$0 \leq \alpha_{i,j} - \alpha_{i',j'} \leq a - (b-a)(i'+j') \leq a$$

By Claim 122, this implies $\mathbf{d}((c,d), T_{b,A}) \leq 2A(a+b)$.

Similarly, if we assume that $I^{\mathcal{N}}_{i,j}$ and $I^{\mathcal{N}}_{i',j'}$ intersect, then this implies without loss of generality that $\beta_{i,j} \in I^{\mathcal{N}}_{i',j'}$, and consequently, $0 \leq \beta_{i,j} - \beta_{i',j'} \leq a - (i+j-1)k \leq a$. Again, using Claim 122, this implies $\mathbf{d}((c,d), T_{b,A}) \leq 2(a+b)A$.

Finally, if $I^{\mathcal{N}}_{i',j'}$ and $I^{\mathcal{P}}_{i,j}$ intersect, then either $0 \leq \alpha_{i,j} - \beta_{i',j'} \leq a$ if $\alpha_{i,j} \in I^{\mathcal{N}}_{i',j'}$ or $0 \leq \beta_{i',j'} - \alpha_{i,j} \leq a$ if $\beta_{i',j'} \in I^{\mathcal{P}}_{i,j}$. In both cases, the Claim 122 gives the desired result.

The proof for the point (ii) is essentially the same as above. If $I^{\mathcal{N}}_{i',j'}$ intersects one of the b positions preceding $I^{\mathcal{N}}_{i,j}$, then we have the two inequalities:

$$\beta_{i',j'} + a - (i' + j' - 1)k \geq \beta_{i,j} - b \qquad\qquad \beta_{i',j'} \leq \beta_{i,j}$$

From these inequalities we can immediately deduce $-a - b \leq \beta_{i',j'} - \beta_{i,j} \leq 0$. The inequality $\mathbf{d}((c,d), T_{a+k,A}) \leq 2A(3a + 2k)$ follows immediately from Claim 122. Similarly, if the interval $I^{\mathcal{P}}_{i',j'}$ intersects one of the b positions preceding $I^{\mathcal{N}}_{i,j}$, then we have the two inequalities:

$$\alpha_{i',j'} + a - (i' + j')(b - a) \geq \beta_{i,j} - b \qquad\qquad \alpha_{i',j'} \leq \beta_{i,j}$$

This implies $-(a + b) \leq \alpha_{i',j'} - \beta_{i,j} \leq 0$, and again the result holds by Claim 122. □

We now have all the tools needed to prove the theorem.

Proof of Theorem 120. Let a, b, c, d be integers, and let $A = \lceil \frac{a}{b-a} \rceil$. Let us assume that we have $\mathbf{d}((c,d), T_{b,A}) \geq 2A(a + 2b)$. We know that the four properties of Lemma 121 hold. We will show by induction on n that for any position $n \geq 0$, if $n \in I^{\mathcal{P}}$, then n is a $\mathcal{P}$-position, if $n \in I^{\mathcal{N}}$, then it is a $\mathcal{N}$-position, and otherwise it is an $\mathcal{L}$-position. The inductive case is treated in the same way as the base case.

First, assume that $n \in I^{\mathcal{N}}_{i,j}$ for some indices i and j such that $i + j \geq 1$. $\mathcal{L}$eft has a winning move by playing a. Indeed, the interval $I^{\mathcal{N}}_{i,j}$ has length at most a, and using the condition (ii) from Lemma 121 and the induction hypothesis, $n - a$ is a $\mathcal{L}$-position. If $i > 0$, then $\mathcal{R}$ight playing c leads to the position $n - c \in I^{\mathcal{P}}_{i-1,j}$ by condition (iii). This position is a $\mathcal{P}$-position using the induction hypothesis. If $j > 0$, then similarly, $\mathcal{R}$ight can play d, and put the game in the position $n - d \in I^{\mathcal{P}}_{i,j-1}$ by condition (iv). This position is a $\mathcal{P}$-position using the induction hypothesis.

Suppose now that $n \in I^{\mathcal{P}}_{i,j}$. If i and j are both zero, then none of the players have any move, and n is a $\mathcal{P}$-position. Otherwise, if $\mathcal{L}$eft plays either a or b, this leads to a position $n' \in I^{\mathcal{N}}_{i,j}$ by condition (v). Using the induction hypothesis, n' is an $\mathcal{N}$-position, and $\mathcal{L}$eft has no winning move. Right's only possible winning move would be to a $\mathcal{P}$-position n'. Using the induction hypothesis this means $n' \in I^{\mathcal{P}}$. However, this would mean by conditions (iii) and (iv) that $n \in I^{\mathcal{N}}$, which is a contradiction of the property (i) that $I^{\mathcal{N}}$ and $I^{\mathcal{P}}$ are disjoint. Consequently, $\mathcal{R}$ight has no winning move.

Finally, suppose that $n \notin I^{\mathcal{P}} \cup I^{\mathcal{N}}$. We will show that $\mathcal{L}$eft has a winning move on n, and $\mathcal{R}$ight does not. Since $I^{\mathcal{P}}_{0,0} = [0, a[$, we can assume $n \geq a$, and $\mathcal{L}$eft can play a. Suppose that $\mathcal{L}$eft's move to $n - a$ is not a winning move, and let us show that $\mathcal{L}$eft has a winning move to $n - a - k$. Since $\mathcal{L}$eft's move to $n - a$ is not a winning move, this means that $n - a \in I^{\mathcal{N}}_{i,j}$ for some integer i, j with $i + j \geq 1$. Consequently we have $n \geq b$, and playing b is a valid move for $\mathcal{L}$eft. By condition (v), we can't have $n - b \in I^{\mathcal{N}}_{i,j}$ since otherwise we would have $n \in I^{\mathcal{P}}_{i,j}$. Moreover, we can't have either $n - b \in I^{\mathcal{N}}_{i',j'}$ for some $(i', j') \neq (i, j)$ since it would contradict condition (ii). Consequently, $n - b \in I^{\mathcal{L}}$, and using the induction hypothesis, this is a winning move for $\mathcal{L}$eft. The only possible winning move for $\mathcal{R}$ight would be to play to a position n' which is a $\mathcal{P}$-position. Using the induction hypothesis, this means that $n' \in I^{\mathcal{P}}$. However using the conditions (iii) and (iv) this would also imply $n \in I^{\mathcal{N}}$, a contradiction. □

Corollary 123. *Under the conditions of the theorem, the game G is ultimately $\mathcal{L}$.*

Proof. Since $I^{\mathcal{N}}_{i,j}$ and $I^{\mathcal{P}}_{i,j}$ are both empty if $i + j > a$, the two sets $I^{\mathcal{L}}$ and $I^{\mathcal{N}}$ are finite, and the result follows from the theorem. □

9.6 Conclusion

Our study of Partizan Subtraction games in this chapter is an example of a more general construction which consists in giving different rulesets to each of the players. It is interesting to investigate this construction for other games as well. The work of [CSD+12] on Chessfight, an other game obtained by this construction which is related to Wythoff is another example of such game.

In addition, if we were able to characterize the asymptotic behaviour of some Partizan Subtraction games, there are still many open cases. For example, we have seen a large class of games where one of the player dominates, but it seems that there are only few choices for the subtraction sets leading to ultimately impartial or fair games. Hence, it might be interesting to try to characterize some of the subtraction sets which leads to such games. Note that in [Wal02], Subtraction games (and more generally Octal games) were expressed as a form of string rewriting games, and some properties of these games were connected to the rationality of some language associated to these games. An interesting approach would be to extend these results to Partizan Subtraction games.

Finally, the question of the complexity of Partizan Subtraction games is still very open. If we showed that computing the outcome of a given position (on one heap) is **NP**-hard, it is not clear whether this problem is in **NP** or not, and it seems that it could be much more difficult than that. It might be easier to prove the hardness of the problem on multiple heaps. To this end, it is interesting to continue the investigation of [Pla95] on the canonical forms for these games.

Conclusion

We have seen in this thesis several problems related to one-player and two-player games, or in other terms, reconfiguration problems and combinatorial games. On the reconfiguration side, we studied problems related to the reconfiguration of two particular objects: graph colourings and perfect matchings. These two problems were studied both from a complexity point of view, and from a structural point of view. Concerning complexity, we proved both hardness results, as well as polynomial algorithms when the input graph is restricted to certain classes. From a structural point of view we improved some upper bounds on the length of the transformation for the reconfiguration of colourings.

After looking at reconfiguration problems, we considered other questions related to reconfiguration, but which are not directly reconfiguration problems. We studied the problem of sampling random colourings, for which we considered the performance of a certain type of algorithm called Glauber Dynamics; and a variant of online colouring algorithms, for which we constructed and analysed several algorithms for different classes of graphs.

On the combinatorial games side, we considered two construction operators and investigated how games built using these operators were modified. We proved some structural properties of these constructions, and used these properties for concrete examples. In particular, we analysed several games obtained from these constructions by combining other well known games. We were able to characterise the outcome of some of these games, and proved some hardness results for others.

There are still many open problems on the various topics that we considered, and we recall some of the major ones here. On the reconfiguration part, there are still many classes of graphs for which the complexity of reconfiguration of colouring and perfect matching is open. Figuring out for which classes of graphs these problems are difficult could help to understand the structure of the underlying reconfiguration graph. Additionally, in relation to our results about online colouring from Chapter 6, it could be interesting to investigate these problems when there are additional constraints restricting the type of operations permitted.

One other important open problem is Cereceda's conjecture which asserts that the reconfiguration graph of colourings has quadratic diameter if the number of colours is at least $\mathrm{col}(G) + 2$. Although our work in Chapter 3 made some important progress towards proving this conjecture, it still remains widely open, and it is not clear how our techniques could be improved further. One way to tackle the conjecture would be to consider specific classes of graphs.

In relation to this conjecture is the problem of sampling random colourings and the corresponding conjecture that Glauber dynamics has a polynomial mixing time if the number of colours is at least $\Delta + 2$. As we have seen, this conjecture is largely open, and there are still a large number of interesting classes of graphs for which the conjecture can be tested and that have not yet been

considered in the literature. Currently, the best bound on the number of colours for which the conjecture holds is $k \geq (\frac{11}{6} - \varepsilon)\Delta$ for some $\varepsilon > 0$. It could be interesting to investigate the limits of the current techniques and see if there are ways to decrease significantly the number of colours from the $\frac{11}{6}\Delta$ bound and still keep a polynomial mixing time. Another interesting question would be to investigate the mixing time for graphs of small degeneracy, and explore whether the mixing time remains polynomial for these graphs, even for a number of colours below $\Delta + 2$.

On the combinatorial game part, there are still a very large number of games whose complexity is unknown (see for example a list on the website [Bur]). We mentioned the complexity of subtraction games, but there are many other games, and in particular games played on graphs, which appear to be difficult but for which there is no formal proof of their hardness. Finding a placement game with simple local rules (for example played on bounded degree graph) which is provably difficult might be a lead to proving the hardness of these other games.

Concerning subtraction games, there is very little known about these games and their structure, despite being some of the most simple possible games that we can consider. Getting a better understanding of these games and their periods might be a first step towards analysing octal games, a generalisation of subtraction games, which are conjectured to be periodic as well.

Finally, we have seen in Chapter 8 a particular construction which leads to an almost disjunctive sum of games. If we were able to prove a nice theory for these games, it is not clear whether this theory, and in particular the values we derive for these games, can be put to a practical use except in very restricted settings. It could be interesting to investigate whether it is possible to find a more general setting for which the structure of the values remain simple.

Bibliography

[AB95] Ingo Althöfer and Jörg Bültermann. Superlinear period lengths in some subtraction games. *Theoretical computer science*, 148(1):111–119, 1995.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ACM10a] Lowell Abrams and Dena S Cowen-Morton. Algebraic structure in a family of nim-like arrays. *Journal of Pure and Applied Algebra*, 214(2):165–176, 2010.

[ACM10b] Lowell Abrams and Dena S. Cowen-Morton. Periodicity and other structure in a colorful family of nim-like arrays. *The Electronic Journal of Combinatorics [electronic only]*, 17(1):Research Paper R103, 21 p., electronic only–Research Paper R103, 21 p., electronic only, 2010.

[ACM14] Lowell Abrams and Dena S Cowen-Morton. A family of nim-like arrays: The locator theorem. *Theoretical Computer Science*, 535:31–37, 2014.

[Ada04] Colin Conrad Adams. *The knot book: an elementary introduction to the mathematical theory of knots*. American Mathematical Soc., 2004.

[AEH+18] John Asplund, Kossi Edoh, Ruth Haas, Yulia Hristova, Beth Novick, and Brett Werner. Reconfiguration graphs of shortest paths. *Discrete Mathematics*, 341(10):2938–2948, 2018.

[AEOP02] Ravindra K Ahuja, Özlem Ergun, James B Orlin, and Abraham P Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002.

[AH89] Kenneth I Appel and Wolfgang Haken. *Every planar map is four colorable*, volume 98. American Mathematical Soc., 1989.

[Ald82] David J Aldous. Some inequalities for reversible markov chains. *Journal of the London Mathematical Society*, 2(3):564–576, 1982.

[AMMVB04] Dimitris Achlioptas, Mike Molloy, Cristopher Moore, and Frank Van Bussel. Sampling grid colorings with fewer colors. In *Latin American Symposium on Theoretical Informatics*, pages 80–89. Springer, 2004.

[AMMVB05] Dimitris Achlioptas, Mike Molloy, Cristopher Moore, and Frank Van Bussel. Rapid mixing for lattice colourings with fewer colours. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(10):P10012, 2005.

[AMP15] Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is **NP**-complete. *Discrete & computational geometry*, 54(2):368–389, 2015.

[ANW07] Michael Albert, Richard Nowakowski, and David Wolfe. *Lessons in play: an introduction to combinatorial game theory*. AK Peters/CRC Press, 2007.

[AS17] Susanne Albers and Sebastian Schraink. Tight Bounds for Online Coloring of Basic Graph Classes. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BB73] Umberto Bertele and Francesco Brioschi. On non-serial dynamic programming. *Journal of Combinatorial Theory, Series A*, 14(2):137–148, 1973.

[BB00] J. (Rob) van den Berg and R. Brouwer. Random sampling for the monomer–dimer model on a lattice. *Journal of Mathematical Physics*, 41(3):1585–1597, 2000.

[BB14a] Marthe Bonamy and Nicolas Bousquet. Recoloring graphs via tree decompositions. *ArXiv preprint arXiv:1403.6386*, 2014.

[BB14b] Marthe Bonamy and Nicolas Bousquet. Reconfiguring independent sets in cographs. *ArXiv preprint arXiv:1406.1433*, 2014.

[BB18] Marthe Bonamy and Nicolas Bousquet. Recoloring graphs via tree decompositions. *European Journal of Combinatorics*, 69:200–213, 2018.

[BBFJ19] Marthe Bonamy, Nicolas Bousquet, Carl Feghali, and Matthew Johnson. On a conjecture of mohar concerning kempe equivalence of regular graphs. *Journal of Combinatorial Theory, Series B*, 135:179 – 199, 2019.

[BBH+19] Marthe Bonamy, Nicolas Bousquet, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Arnaud Mary, Moritz Muehlenthaler, and Kunihiro Wasa. The perfect matching reconfiguration problem. *ArXiv preprint arXiv:1904.06184*, 2019.

[BBP18] Marthe Bonamy, Nicolas Bousquet, and Guillem Perarnau. Frozen colourings of bounded degree graphs. *Electronic Notes in Discrete Mathematics*, 68:167–172, 2018.

[BC09] Paul Bonsma and Luis Cereceda. Finding paths between graph colourings: Pspace-completeness and superpolynomial distances. *Theoretical Computer Science*, 410(50):5215–5226, 2009.

[BCD+18] Laurent Beaudou, Pierre Coupechoux, Antoine Dailly, Sylvain Gravier, Julien Moncel, Aline Parreau, and Eric Sopena. Octal games on graphs: The game 0.33 on subdivided stars and bistars. *Theoretical Computer Science*, 746:19–35, 2018.

[BCG82] E. R. Berlkamp, J. H. Conway, and R. K. Guy. *Winning ways for your mathematical plays*. New York: Academic Press, 1982.

[BCG04] Elwyn R Berlekamp, John H Conway, and Richard K Guy. *Winning Ways for Your Mathematical Plays, Volumes 1-4*. AK Peters/CRC Press, 2001-2004.

[BCG19] Suman K Bera, Amit Chakrabarti, and Prantar Ghosh. Graph coloring via degeneracy in streaming and other space-conscious models. *Preprint*, 2019.

[BCK+17] Luis Barba, Jean Cardinal, Matias Korman, Stefan Langerman, André van Renssen, Marcel Roeloffzen, and Sander Verdonschot. Dynamic graph coloring. In *Workshop on Algorithms and Data Structures*, pages 97–108. Springer, 2017.

[BD97] Russ Bubley and Martin Dyer. Path coupling: A technique for proving rapid mixing in markov chains. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 223–231. IEEE, 1997.

[BDGJ99] Russ Bubley, Martin Dyer, Catherine Greenhill, and Mark Jerrum. On approximately counting colorings of small degree graphs. *SIAM Journal on Computing*, 29(2):387–400, 1999.

[BDJ98] Russ Bubley, Martin Dyer, and Mark Jerrum. An elementary analysis of a procedure for sampling points in a convex body. *Random Structures & Algorithms*, 12(3):213–235, 1998.

[BDK05] Magnus Bordewich, Martin Dyer, and Marek Karpinski. Path coupling using stopping times. In *International Symposium on Fundamentals of Computation Theory*, pages 19–31. Springer, 2005.

[BDKK17] Boštjan Brešar, Paul Dorbec, Sandi Klavzar, and Gašper Košmrlj. How long can one bluff in the domination game? *Discussiones Mathematicae Graph Theory*, 37(2), May 2017.

[Bea76] Dwight R Bean. Effective coloration. *The Journal of Symbolic Logic*, 41(2):469–480, 1976.

[BEMPS10] Laura Beaudin, Joanna Ellis-Monaghan, Greta Pangborn, and Robert Shrock. A little statistical mechanics for the graph theorist. *Discrete Mathematics*, 310(13-14):2037–2053, 2010.

[Ber88] Elwyn R Berlekamp. Blockbusting and domineering. *Journal of Combinatorial Theory, Series A*, 49(1):67–116, 1988.

[Ber90] Marc E Bertschi. Perfectly contractile graphs. *Journal of Combinatorial Theory, Series B*, 50(2):222–230, 1990.

[BF90] Uri Blass and Aviezri S Fraenkel. The sprague-grundy function for wythoff's game. *Theoretical Computer Science*, 75(3):311–333, 1990.

[BH14] Sarah-Marie Belcastro and Ruth Haas. Counting edge-kempe-equivalence classes for 3-edge-colored cubic graphs. *Discrete Mathematics*, 325:77–84, 2014.

[BH19] Nicolas Bousquet and Marc Heinrich. A polynomial version of cereceda's conjecture. *ArXiv preprint arXiv:1903.05619*, 2019.

[BHI+19a] Marthe Bonamy, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Moritz Mühlenthaler, Akira Suzuki, and Kunihiro Wasa. Diameter of colorings under kempe changes. 2019.

[BHI+19b] Marthe Bonamy, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Moritz Mühlenthaler, Akira Suzuki, and Kunihiro Wasa. Shortest reconfiguration of colorings under kempe recoloring. 2019.

[BI17] Sergey Bereg and Takehiro Ito. Transforming graphs with the same graphic sequence. *Journal of Information Processing*, 25:627–633, 2017.

[BJL+14] Marthe Bonamy, Matthew Johnson, Ioannis Lignos, Viresh Patel, and Daniël Paulusma. Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs. *Journal of Combinatorial Optimization*, 27(1):132–143, 2014.

[BKMP05] Noam Berger, Claire Kenyon, Elchanan Mossel, and Yuval Peres. Glauber dynamics on trees and hyperbolic graphs. *Probability Theory and Related Fields*, 131(3):311–340, 2005.

[BKW14] Paul Bonsma, Marcin Kamiński, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In *Scandinavian Workshop on Algorithm Theory*, pages 86–97. Springer, 2014.

[BLR06] John Billingham, Robert Leese, and Hannu Rajaniemi. Frequency reassignment in cellular phone networks. Technical report, Motorola, 2006.

[BM76] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*, volume 290. Citeseer, 1976.

[BMMN16] Richard C Brewster, Sean McGuinness, Benjamin Moore, and Jonathan A Noel. A dichotomy theorem for circular colouring reconfiguration. *Theoretical Computer Science*, 639:1–13, 2016.

[BMNR14] Paul Bonsma, Amer E Mouawad, Naomi Nishimura, and Venkatesh Raman. The complexity of bounded length graph recoloring and csp reconfiguration. In *International Symposium on Parameterized and Exact Computation*, pages 110–121. Springer, 2014.

[BMP17] Nicolas Bousquet, Arnaud Mary, and Aline Parreau. Token jumping in minor-closed classes. In *Fundamentals of Computation Theory, FCT 2017, Bordeaux, France*, pages 136–149, 2017.

[BN15] Richard C Brewster and Jonathan A Noel. Mixing homomorphisms, recolorings, and extending circular precolorings. *Journal of Graph Theory*, 80(3):173–198, 2015.

[Bon13] Paul Bonsma. The complexity of rerouting shortest paths. *Theoretical computer science*, 510:1–12, 2013.

[Bon16] Paul Bonsma. Independent set reconfiguration in cographs and their generalizations. *Journal of Graph Theory*, 83(2):164–195, 2016.

[BOR+18] Marthe Bonamy, Paul Ouvrard, Mikaël Rabie, Jukka Suomela, and Jara Uitto. Distributed recoloring. *ArXiv preprint arXiv:1802.06742*, 2018.

[Bou01] Charles L Bouton. Nim, a game with a complete mathematical theory. *Annals of Mathematics*, 3(1/4):35–39, 1901.

[BP16] Nicolas Bousquet and Guillem Perarnau. Fast recoloring of sparse graphs. *European Journal of Combinatorics*, 52:1–11, 2016.

[Bre77] Wayne M Brehaut. An efficient outerplanarity algorithm. In *Proceedings of the 8th South-Eastern Conference on Combinatorics, Graph Theory, and Computing*, pages 99–113, 1977.

[BUH00] Dennis M Breuker, Jos WHM Uiterwijk, and H Jaap van den Herik. Solving 8× 8 domineering. *Theoretical Computer Science*, 230(1-2):195–206, 2000.

[Bul02] Nathan Bullock. Domineering: Solving large combinatorial search spaces. *ICGA Journal*, 25(2):67–84, 2002.

[Bur] Kyle Burke. Combinatorial games rulesets. https://turing.plymouth.edu/~kgb1013/rulesetTable.php.

[CD69] Alfred J Cole and AJT Davie. A game based on the euclidean algorithm and a winning strategy for it. *The Mathematical Gazette*, 53(386):354–357, 1969.

[Cer07] Luis Cereceda. *Mixing graph colourings*. PhD thesis, The London School of Economics and Political Science (LSE), 2007.

[CFKP07] Ioannis Caragiannis, Aleksei V Fishkin, Christos Kaklamanis, and Evi Papaioannou. A tight bound for online colouring of disk graphs. *Theoretical Computer Science*, 384(2-3):152–160, 2007.

[CHJ08] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(5-6):913–919, 2008.

[CHJ09] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Mixing 3-colourings in bipartite graphs. *European Journal of Combinatorics*, 30(7):1593–1606, 2009.

[CHJ11] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Finding paths between 3-colorings. *Journal of graph theory*, 67(1):69–82, 2011.

[CHMW87] Vasek Chvátal, Chính T Hoàng, Nadimpalli VR Mahadev, and Dominique De Werra. Four classes of perfectly orderable graphs. *Journal of graph theory*, 11(4):481–495, 1987.

[CL08] David Collins and Tamás Lengyel. The game of 3-euclid. *Discrete Mathematics*, 308(7):1130 – 1136, 2008.

[CLLW18] Wai Hong Chan, Richard M Low, Stephen C Locke, and Oi Lin Wong. A map of the p-positions in 'nim with a pass' played on heap sizes of at most four. *Discrete Applied Mathematics*, 244:44–55, 2018.

[CLN17] Matthew Cook, Urban Larsson, and Turlough Neary. A cellular automaton for blocking queen games. *Natural Computing*, 16(3):397–410, 2017.

[CLSB81] Derek G. Corneil, H. Lerchs, and Lorna K. Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981.

[CM11] Kelly Choo and Gary MacGillivray. Gray code numbers for graphs. *Ars Mathematica Contemporanea*, 4(1):125–139, 2011.

[CM18] Sitan Chen and Ankur Moitra. Linear programming bounds for randomly sampling colorings. *arXiv preprint arXiv:1804.03156*, 2018.

[CNS18] Alda Carvalho, João Pedro Neto, and Carlos Santos. Ordinal sums of impartial games. *Discrete Applied Mathematics*, 243:39–45, 2018.

[Con59] Ian G Connell. A generalization of wythoff's game. *Canadian Mathematical Bulletin*, 2(3):181–190, 1959.

[Con76] John Horton Conway. *On numbers and games*. Academic Press, 1976.

[Con00] John H Conway. *On numbers and games*. AK Peters/CRC Press, 2000.

[CPS85] Derek G. Corneil, Yehoshua Perl, and Lorna K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[CSD+12] Alda Carvalho, Carlos P Santos, Cátia Lente Dias, Francisco Coelho, João Pedro Neto, and Sandra Vinagre. A recursive process related to a partizan variation of wythoff. *Integers*, 12(5):1029–1045, 2012.

[CUL99] J CULBARSON. Sokoban is pspace-complete. *Proceedings in Informatics, Waterloo, Canada, 1999*, 4:65–76, 1999.

[DDF+14] Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Polynomial-time algorithm for sliding tokens on trees. In *Algorithms and Computation - 25th International Symposium, ISAAC, Proceedings*, pages 389–400, 2014.

[DDHL16] Eric Duchêne, Matthieu Dufour, Silvia Heubach, and Urban Larsson. Building nim. *International Journal of Game Theory*, 45(4):859–873, 2016.

[DEGI09] Camil Demetrescu, David Eppstein, Zvi Galil, and Giuseppe F Italiano. Dynamic graph algorithms. In *Algorithms and Theory of Computation Handbook, Second Edition, Volume 1*, pages 228–255. Chapman and Hall/CRC, 2009.

[DF03] Martin Dyer and Alan Frieze. Randomly coloring graphs with lower bounds on girth and maximum degree. *Random Structures & Algorithms*, 23(2):167–179, 2003.

[DF10] Martin Dyer and Alan Frieze. Randomly coloring random graphs. *Random Structures & Algorithms*, 36(3):251–272, 2010.

[DFFV06] M. Dyer, A. D. Flaxman, A. M. Frieze, and E. Vigoda. Randomly coloring sparse random graphs with fewer colors than the maximum degree. *Random Structures & Algorithms*, 29(4):450–465, 2006.

[DFHV04] Martin Dyer, Alan Frieze, Thomas P Hayes, and Eric Vigoda. Randomly coloring constant degree graphs. In *null*, pages 582–589. IEEE, 2004.

[DFI04] Camil Demetrescu, Irene Finocchi, and Giuseppe F. Italiano. Dynamic graphs. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications.* Chapman and Hall/CRC, 2004.

[DFK91] Martin Dyer, Alan Frieze, and Ravi Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM (JACM)*, 38(1):1–17, 1991.

[DG00] Martin Dyer and Catherine Greenhill. On markov chains for independent sets. *J. Algorithms*, 35(1):17–49, 2000.

[DGH01] Persi Diaconis, Ronald Graham, and Susan P Holmes. Statistical problems involving permutations with restricted positions. *State of the Art in Probability and Statistics: Festschrift for Willem R. Van Zwet*, 36:195, 2001.

[DGJ06] Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Systematic scan for sampling colorings. *The Annals of Applied Probability*, 16(1):185–230, 2006.

[DGM02] Martin Dyer, Catherine Greenhill, and Mike Molloy. Very rapid mixing of the glauber dynamics for proper colorings on bounded-degree graphs. *Random Structures & Algorithms*, 20(1):98–114, 2002.

[DHLP18] Eric Duchene, Marc Heinrich, Urban Larsson, and Aline Parreau. The switch operators and push-the-button games: a sequential compound over rulesets. *Theoretical Computer Science*, 715:71–85, 2018.

[DHNP19] Eric Duchêne, Marc Heinrich, Richard J. Nowakowski, and Aline Parreau. Partizan subtraction games. 2019.

[DHP18] Michelle Delcourt, Marc Heinrich, and Guillem Perarnau. The glauber dynamics for edges colourings of trees. *arXiv preprint arXiv:1812.05577*, 2018.

[Dir61] Gabriel Andrew Dirac. On rigid circuit graphs. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 25, pages 71–76. Springer, 1961.

[DJM17] Martin Dyer, Mark Jerrum, and Haiko Müller. On the switch markov chain for perfect matchings. *Journal of the ACM (JACM)*, 64(2):12, 2017.

[DL17] Maya Dotan and Nati Linial. Efficient generation of random one-factorizations for complete graphs. *arXiv preprint arXiv:1707.00477*, 2017.

[DM17] Martin Dyer and Haiko Müller. Counting perfect matchings and the switch chain. *ArXiv preprint arXiv:1705.05790*, 2017.

[DPP18] Michelle Delcourt, Guillem Perarnau, and Luke Postle. Rapid mixing of glauber dynamics for colorings below vigoda's 11/6 threshold. *arXiv preprint arXiv:1804.04025*, 2018.

[dR$^+$] H. N. de Ridder et al. Information system on graph classes and their inclusions (ISGCI). http://www.graphclasses.org/.

[DR08] Eric Duchêne and Michel Rigo. A morphic approach to combinatorial games: the tribonacci case. *RAIRO-Theoretical Informatics and Applications*, 42(2):375–393, 2008.

[DR10] Eric Duchêne and Michel Rigo. Invariant games. *Theoretical Computer Science*, 411:3169 – 3180, 2010.

[DRSS15] Paul Dorbec, Gabriel Renault, Aaron N Siegel, and Éric Sopena. Dicots, and a taxonomic ranking for misere games. *Journal of Combinatorial Theory, Series A*, 130:42–63, 2015.

[DSC93] P. Diaconis and L. Saloff-Coste. Comparison theorems for reversible Markov chains. *The Annals of Applied Probability*, pages 696–730, 1993.

[DSVW04] Martin Dyer, Alistair Sinclair, Eric Vigoda, and Dror Weitz. Mixing in time and space for lattice spin systems: A combinatorial view. *Random Structures & Algorithms*, 24(4):461–479, 2004.

[EF02] Thomas Erlebach and Jiri Fiala. On-line coloring of geometric intersection graphs. *Computational Geometry*, 23(2):243–255, 2002.

[Fár48] István Fáry. On straight-line representation of planar graphs. *Acta Sci. Math.*, 11:229–233, 1948.

[FB73] Aviezri S Fraenkel and I Borosh. A generalization of wythoff's game. *Journal of Combinatorial Theory, Series A*, 15(2):175–191, 1973.

[Feg19a] Carl Feghali. Paths between colourings of graphs with bounded tree-width. *Information Processing Letters*, 144:37–38, 2019.

[Feg19b] Carl Feghali. Paths between colourings of sparse graphs. *European Journal of Combinatorics*, 75:169–171, 2019.

[Feg19c] Carl Feghali. Reconfiguring 10-colourings of planar graphs. *ArXiv preprint arXiv:1902.02278*, 2019.

[FH18] Graham Farr and Nhan Bao Ho. The sprague–grundy function for some nearly disjunctive sums of nim and silver dollar games. *Theoretical Computer Science*, 732:46–59, 2018.

[Fis77] Steve Fisk. Geometric coloring theory. *Advances in Mathematics*, 24(3):298–340, 1977.

[FJP14] Carl Feghali, Matthew Johnson, and Daniël Paulusma. A reconfigurations analogue of brooks' theorem. In *International Symposium on Mathematical Foundations of Computer Science*, pages 287–298. Springer, 2014.

[FJP15] Carl Feghali, Matthew Johnson, and Daniël Paulusma. Kempe equivalence of colourings of cubic graphs. *Electronic notes in discrete mathematics*, 49:243–249, 2015.

[FK87] Aviezri S Fraenkel and Anton Kotzig. Partizan octal games: partizan subtraction games. *International Journal of Game Theory*, 16(2):145–154, 1987.

[FL81] Aviezri S Fraenkel and David Lichtenstein. Computing a perfect strategy for n× n chess requires time exponential in n. In *International Colloquium on Automata, Languages, and Programming*, pages 278–293. Springer, 1981.

[Fla82] Jim Flanigan. One-pile time and size dependent take-away games", Fibonacci Quart. *Fibonacci Quarterly*, pages 51–59, 1982.

[Fla00] Achim Flammenkamp. http://wwwhomes.uni-bielefeld.de/achim/octal.html, 2000.

[FMFPH+12] Ruy Fabila-Monroy, David Flores-Peñaloza, Clemens Huemer, Ferran Hurtado, Jorge Urrutia, and David R Wood. Token graphs. *Graphs and Combinatorics*, 28(3):365–380, 2012.

[FNPS05] Dimitris A Fotakis, Sotiris E Nikoletseas, Vicky G Papadopoulou, and Paul G Spirakis. Radiocoloring in planar graphs: complexity and approximations. *Theoretical Computer Science*, 340(3):514–538, 2005.

[Fra97] Aviezri S Fraenkel. Combinatorial game theory foundations applied to digraph kernels. *Electron. J. Combin*, 4(2), 1997.

[Fra04] Aviezri S Fraenkel. Complexity, appeal and challenges of combinatorial games. *Theoretical Computer Science*, 313(3):393–415, 2004.

[Fra07] Aviezri S. Fraenkel. The Raleigh game. *INTEGERS: Electronic Journal of Combinatorial Number Theory*, 7(2):A13, 2007.

[FS99] Sabino Jose Ferreira and Alan D Sokal. Antiferromagnetic potts models on the square lattice: A high-precision monte carlo study. *Journal of statistical physics*, 96(3-4):461–530, 1999.

[FT04] Rudolf Fleischer and Gerhard Trippen. Kayles on the way to the stars. In *International Conference on Computers and Games*, pages 232–245. Springer, 2004.

[FV06] Alan Frieze and Juan Vera. On randomly colouring locally sparse graphs. *Discrete Mathematics & Theoretical Computer Science*, 8(1), 2006.

[FV07] Alan Frieze and Eric Vigoda. A survey on the use of markov chains to randomly sample colourings. *Oxford Lecture Series in Mathematics and its Applications*, 34:53, 2007.

[Gal74] David Gale. A curious nim-type game. *The American Mathematical Monthly*, 81(8):876–879, 1974.

[GH18] Sylvain Gravier and Marc Heinrich. Online graph coloring with bichromatic exchanges. *Submitted*, 2018.

[GJK10] Leslie Ann Goldberg, Mark Jerrum, and Marek Karpinski. The mixing time of glauber dynamics for coloring regular trees. *Random Structures & Algorithms*, 36(4):464–476, 2010.

[GJMP06] Leslie Ann Goldberg, Markus Jalsenius, Russell Martin, and Mike Paterson. Improved mixing bounds for the anti-ferromagnetic potts model on $\mathbb{Z}^2$. *LMS Journal of Computation and Mathematics*, 9:1–20, 2006.

[GJT76] M. Garey, D. Johnson, and R. Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

[GK07] David Gamarnik and Dmitriy Katz. Correlation decay and deterministic fptas for counting list-colorings of a graph. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1245–1254. Society for Industrial and Applied Mathematics, 2007.

[GKMP09] Parikshit Gopalan, Phokion G Kolaitis, Elitza Maneva, and Christos H Papadimitriou. The connectivity of boolean satisfiability: computational and structural dichotomies. *SIAM Journal on Computing*, 38(6):2330–2355, 2009.

[GKRS15] David Galvin, Jeff Kahn, Dana Randall, and Gregory B Sorkin. Phase coexistence and torpid mixing in the 3-coloring model on \mathbbzˆd. *SIAM Journal on Discrete Mathematics*, 29(3):1223–1244, 2015.

[GL88] András Gyárfás and Jenö Lehel. On-line and first fit colorings of graphs. *Journal of Graph theory*, 12(2):217–227, 1988.

[GMP04] Leslie Ann Goldberg, Russell Martin, and Mike Paterson. Random sampling of 3-colorings in $\mathbb{Z}^2$. *Random Structures & Algorithms*, 24(3):279–302, 2004.

[GMP05] Leslie Ann Goldberg, Russell Martin, and Mike Paterson. Strong spatial mixing with fewer colors for lattice graphs. *SIAM Journal on Computing*, 35(2):486–517, 2005.

[GN96] Richard K Guy and Richard J Nowakowski. Unsolved problems in combinatorial games. *Games of No Chance*, 3, 1996.

[GR07] David Galvin and Dana Randall. Torpid mixing of local markov chains on 3-colorings of the discrete torus. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 376–384. Society for Industrial and Applied Mathematics, 2007.

[Gru39] Patrick M Grundy. Mathematics and games. *Eureka*, 2:6–9, 1939.

[GS09] Adrien Guignard and Éric Sopena. Compound node–kayles on paths. *Theoretical Computer Science*, 410(21-23):2033–2044, 2009.

[Gur07] Vladimir A Gurvich. On the misere version of game euclid and miserable games. *Discrete mathematics*, 307(9-10):1199–1204, 2007.

[Gur16] Venkatesan Guruswami. Rapidly mixing markov chains: a comparison of techniques (a survey). *arXiv preprint arXiv:1603.01512*, 2016.

[Hak63] S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph II. Uniqueness. *Journal of the Society for Industrial and Applied Mathematics*, 11(1):135–147, 1963.

[Hal97] Magnús M Halldórsson. Parallel and on-line graph coloring. *Journal of Algorithms*, 23(2):265–280, 1997.

[Hal00] Magnús M Halldórsson. Online coloring known graphs. *the electronic journal of combinatorics*, 7(1):7, 2000.

[Hay03] Thomas P Hayes. Randomly coloring graphs of girth at least five. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 269–278. ACM, 2003.

[Hay06] Thomas P Hayes. A simple condition implying rapid mixing of single-site dynamics on spin systems. In *null*, pages 39–46. IEEE, 2006.

[HD05] Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.

[Heu13] Jan van den Heuvel. The complexity of change. *Surveys in combinatorics*, 409(2013):127–160, 2013.

[HG96] Hacène Ait Haddadene and Sylvain Gravier. On weakly diamond-free berge graphs. *Discrete Mathematics*, 159(1-3):237–240, 1996.

[HGM98] Hacène Aït Haddadène, Sylvain Gravier, and Frédéric Maffray. An algorithm for coloring some perfect graphs. *Discrete mathematics*, 183(1-3):1–16, 1998.

[HHNRC05] Michael E. Houle, Ferran Hurtado, Marc Noy, and Eduardo Rivera-Campo. Graphs of triangulations and perfect matchings. *Graphs and Combinatorics*, 21(3):325–331, 2005.

[HIZ15] Tatsuhiko Hatanaka, Takehiro Ito, and Xiao Zhou. The list coloring reconfiguration problem for bounded pathwidth graphs. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 98(6):1168–1178, 2015.

[HIZ17] Tatsuhiko Hatanaka, Takehiro Ito, and Xiao Zhou. The coloring reconfiguration problem on specific graph classes. In *International Conference on Combinatorial Optimization and Applications*, pages 152–162. Springer, 2017.

[HLZ16] Lingxiao Huang, Pinyan Lu, and Chihao Zhang. Canonical paths for mcmc: from art to science. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 514–527. Society for Industrial and Applied Mathematics, 2016.

[HM97] Hacène Aït Haddadène and Frédéric Maffray. Coloring perfect degenerate graphs. *Discrete Mathematics*, 163(1-3):211–215, 1997.

[HN03] D G Horrocks and Richard J Nowakowski. Regularity in the g- sequences of octal games with a pass. *Integers: Electronic Journal of Combinatorial Number Theory*, 3(G01):2, 2003.

[HN19] Melissa Huggan and Richard J. Nowakowski. Conjoined games: Go-cut and sno-go. *Games of No Chance 5*, 2019. To appear.

[HNU99] Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

[Ho12a] Nhan Bao Ho. Subtraction games with three element subtraction sets. *ArXiv preprint arXiv:1202.2986*, 2012.

[Ho12b] Nhan Bao Ho. Variations of the game 3-Euclid. *International Journal of Combinatorics*, 2012.

[HRR03] Arthur Holshouser, Harold Reiter, and James Rudzinski. Dynamic one-pile Nim. *Fibonacci Quarterly*, 41(3):253–262, 2003.

[HRR04] Arthur Holshouser, Harold Reiter, and James Rudzinski. Pilesize Dynamic One-Pile Nim and Beatty's Theorem. *INTEGERS: Electronic Journal of Combinatorial Number Theory*, 4(G03):G03, 2004.

[HS92] Magnús M Halldórsson and Mario Szegedy. Lower bounds for on-line graph coloring. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 211–216. Society for Industrial and Applied Mathematics, 1992.

[HS05] Thomas P Hayes and Alistair Sinclair. A general lower bound for mixing of single-site dynamics on graphs. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 511–520. IEEE, 2005.

[Hub98] Mark Huber. Exact sampling and approximate counting techniques. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 31–40. ACM, 1998.

[HV03] Thomas P Hayes and Eric Vigoda. A non-markovian coupling for randomly sampling colorings. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 618–627. IEEE, 2003.

[HV07] Thomas P Hayes and Eric Vigoda. Variable length path coupling. *Random Structures & Algorithms*, 31(3):251–272, 2007.

[HVV15] Thomas P Hayes, Juan C Vera, and Eric Vigoda. Randomly coloring planar graphs with fewer colors than the maximum degree. *Random Structures & Algorithms*, 47(4):731–759, 2015.

[IDH+11] Takehiro Ito, Erik D Demaine, Nicholas JA Harvey, Christos H Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011.

[IKO14] Takehiro Ito, Marcin Kamiński, and Hirotaka Ono. *Fixed-Parameter Tractability of Token Jumping on Planar Graphs*, pages 208–219. Springer International Publishing, 2014.

[Ira94] Sandy Irani. Coloring inductive graphs on-line. *Algorithmica*, 11(1):53–72, 1994.

[Jal09] Markus Jalsenius. Strong spatial mixing and rapid mixing with five colours for the kagome lattice. *LMS Journal of Computation and Mathematics*, 12:195–227, 2009.

[Jal12] Markus Jalsenius. Sampling colourings of the triangular lattice. *Random Structures & Algorithms*, 40(4):501–533, 2012.

[Jer95] Mark Jerrum. A very simple algorithm for estimating the number of k-colorings of a low-degree graph. *Random Structures & Algorithms*, 7(2):157–165, 1995.

[JKK+16] Matthew Johnson, Dieter Kratsch, Stefan Kratsch, Viresh Patel, and Daniël Paulusma. Finding shortest paths between graph colourings. *Algorithmica*, 75(2):295–321, 2016.

[JM96] Mark T Jacobson and Peter Matthews. Generating uniformly distributed random latin squares. *Journal of Combinatorial Designs*, 4(6):405–437, 1996.

[JS79] Wm Woolsey Johnson and William Edward Story. Notes on the "15" puzzle. *American Journal of Mathematics*, 2(4):397–404, 1879.

[JS89] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM journal on computing*, 18(6):1149–1178, 1989.

[JS96] Mark Jerrum and Alistair Sinclair. The markov chain monte carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1996.

[JVV86] Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[KMM12] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical computer science*, 439:9–15, 2012.

[KQ95] Hal A Kierstead and Jun Qin. Coloring interval graphs with first-fit. *Discrete Mathematics*, 144(1-3):47–57, 1995.

[KT81] HA Kierstead and WT Trotter. An extremal problem in recursive combinatorics,. *Congressus Numerantium*, 33:143–153, 1981.

[Lar11] Urban Larsson. Blocking wythoff nim. *the electronic journal of combinatorics*, 18(1):120, 2011.

[Lar13] Urban Larsson. Impartial games emulating one-dimensional cellular automata and undecidability. *Journal of Combinatorial Theory, Series A*, 120(5):1116–1130, 2013.

[Law77] Charles L. Lawson. Software for c_1 surface interpolation. In *Mathematical software*, pages 161–194. Elsevier, 1977.

[Len03] Tamás Lengyel. A nim-type game and continued fractions. *Fibonacci Quarterly*, 41(4):310–320, 2003.

[LM06] Lap Chi Lau and Michael Molloy. Randomly colouring graphs with girth five and large maximum degree. In *Latin American Symposium on Theoretical Informatics*, pages 665–676. Springer, 2006.

[LMNS18] Urban Larsson, Neil A McKay, Richard J Nowakowski, and Angela A Siegel. Wythoff partizan subtraction. *International Journal of Game Theory*, 47(2):613–652, 2018.

[LMP09] Brendan Lucier, Michael Molloy, and Yuval Peres. The glauber dynamics for colourings of bounded degree trees. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 631–645. Springer, 2009.

[LMR02] Michael Lachmann, Cristopher Moore, and Ivan Rapaport. Who wins domineering on rectangular boards. *More Games of No Chance*, 42:307–315, 2002.

[LP15] Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is **NP**-complete. *Computational Geometry*, 49:17–23, 2015.

[LP17] D. A. Levin and Y. Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[LR17] Urban Larsson and Israel Rocha. Eternal picaria. *Recreational Mathematics Magazine*, 4(7):119–133, 2017.

[LST89] László Lovász, Michael Saks, and William T Trotter. An on-line graph coloring algorithm with sublinear performance ratio. *Discrete Mathematics*, 75(1-3):319–325, 1989.

[Lue75] George S Lueker. Two np-complete problems in nonnegative integer programming. Technical report, Princeton University. Department of Electrical Engineering, 1975.

[LV98] Stefano Leonardi and Andrea Vitaletti. Randomized lower bounds for online path coloring. In *RANDOM*, volume 98, pages 232–247. Springer, 1998.

[ŁV05] Tomasz Łuczak and Eric Vigoda. Torpid mixing of the wang–swendsen–koteckỳ algorithm for sampling colorings. *Journal of Discrete Algorithms*, 3(1):92–100, 2005.

[LV12] Julien Lemoine and Simon Viennot. Nimbers are inevitable. *Theoretical Computer Science*, 462:70–79, 2012.

[LVM81] Michel Las Vergnas and Henri Meyniel. Kempe classes and the hadwiger conjecture. *Journal of Combinatorial Theory, Series B*, 31(1):95–104, 1981.

[Maa05] Thomas Maarup. Everything you always wanted to know about hex but were afraid to ask. *University of Southern Denmark*, 2005.

[Mar99] F. Martinelli. Lectures on Glauber dynamics for discrete spin models. In *Lectures on probability theory and statistics*, pages 93–191. Springer, 1999.

[Mey78] Henry Meyniel. Les 5-colorations d'un graphe planaire forment une classe de commutation unique. *Journal of Combinatorial Theory, Series B*, 24(3):251–257, 1978.

[MFL11] Rebecca E Morrison, Eric J Friedman, and Adam S Landsberg. Combinatorial games with a pass: A dynamical systems approach. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 21(4):043108, 2011.

[MI05] GA Mesdal III. Partizan splittles. In *Games of No Chance III, Proc. BIRS Workshop on Combinatorial Games*, pages 447–461. Citeseer, 2005.

[MIZ17] Haruka Mizuta, Takehiro Ito, and Xiao Zhou. Reconfiguration of steiner trees in an unweighted graph. *IEICE Transactions*, 100-A(7):1532–1540, 2017.

[MMS12] Jessica McDonald, Bojan Mohar, and Diego Scheide. Kempe equivalence of edge-colorings in subcubic and subquartic graphs. *Journal of Graph theory*, 70(2):226–239, 2012.

[MN13] Neil A McKay and RJ Nowakowksi. Outcomes of partizan euclid. *Combinatorial Number Theory*, pages 123–137, 2013.

[MO07] GA Mesdal and Paul Ottaway. Simplification of partizan games in misere play. *Integers*, 7:G06, 2007.

[Moh06] Bojan Mohar. Kempe equivalence of colorings. In *Graph Theory in Paris*, pages 287–297. Springer, 2006.

[Mol04] Michael Molloy. The glauber dynamics on colorings of a graph with high girth and maximum degree. *SIAM Journal on Computing*, 33(3):721–737, 2004.

[Mou15] Amer Mouawad. *On Reconfiguration Problems: Structure and Tractability*. PhD thesis, University of Waterloo, 2015.

[MP99] Frédéric Maffray and Myriam Preissmann. Sequential colorings and perfect graphs. *Discrete Applied Mathematics*, 94(1):287 – 296, 1999. Proceedings of the Third International Conference on Graphs and Optimization GO-III.

[MR13] Rebecca Milley and Gabriel Renault. Dead ends in misere play: the misere monoid of canonical numbers. *Discrete Mathematics*, 313(20):2223–2231, 2013.

[MS10a] Bojan Mohar and Jesús Salas. On the non-ergodicity of the swendsen–wang–koteckỳ algorithm on the kagomé lattice. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(05):P05016, 2010.

[MS10b] Elchanan Mossel and Allan Sly. Gibbs rapidly samples colorings of $G(n, d/n)$. *Probability theory and related fields*, 148(1-2):37–69, 2010.

[Mul16] Todd Mullen. The self-referential games minnie and wynnie and some variants. Master's thesis, Dalhousie University, 2016.

[NB08] N. S. Narayanaswamy and R. Subhash Babu. A note on first-fit coloring of interval graphs. *Order*, 25(1):49–53, Feb 2008.

[Nis18] Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.

[Niv05] Gabriel Nivasch. More on the sprague-grundy function for wythoff's game. In *Games of No Chance III, Proc. BIRS Workshop on Combinatorial Games*, pages 377–410, 2005.

[OSIZ18] Hiroki Osawa, Akira Suzuki, Takehiro Ito, and Xiao Zhou. The complexity of (list) edge-coloring reconfiguration problem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 101(1):232–238, 2018.

[Ped08] Kasper Pedersen. *On systematic scan.* PhD thesis, The University of Liverpool, 2008.

[Pla95] Thane Plambeck. Notes on partizan subtraction games, 1995.

[Pla05] Thane E Plambeck. Taming the wild in impartial combinatorial games. *INTEGERS: Electronic Journal of Combinatorial Number Theory*, 5(1):1–36, 2005.

[Pla06] Thane E Plambeck. Advances in losing. *ArXiv preprint math/0603027*, 2006.

[Poo16] Chun Yeung Poon. Glauber dynamics for sampling an edge colouring of regular trees. Master's thesis, The Chinese University of Hong Kong, 2016.

[PS08] Thane E Plambeck and Aaron N Siegel. Misere quotients for impartial games. *Journal of Combinatorial Theory, Series A*, 115(4):593–622, 2008.

[RA96] Jorge Luis Ramírez-Alfonsín. Complexity of the frobenius problem. *Combinatorica*, 16(1):143–147, 1996.

[Ran06] Dana Randall. Rapidly mixing markov chains with applications in computer science and physics. *Computing in Science & Engineering*, 8(2):30–41, 2006.

[RT00] D. Randall and P. Tetali. Analyzing Glauber dynamics by comparison of Markov chains. *Journal of Mathematical Physics*, 41(3):1598–1615, 2000.

[S$^+$84] James J Sylvester et al. Mathematical questions with their solutions. *Educational times*, 41(21):6, 1884.

[San10] Carlos Manuel Ferreira Pereira dos Santos. *Some notes on impartial games and NIM dimension*. PhD thesis, University of Lisbon, 2010.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[Sen51] James K. Senior. Partitions and their representative graphs. *American Journal of Mathematics*, 73(3):663–689, 1951.

[Sie13] Aaron N Siegel. *Combinatorial game theory*, volume 146. American Mathematical Soc., 2013.

[Sie15] Aaron N Siegel. The structure and classification of misere quotients. *Games of No Chance 4*, 63:241, 2015.

[Sil76] Rober Silber. A fibonacci property of wythoff pairs. *Fibonacci Quart*, 14(4):380–384, 1976.

[Sin92] A. Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, probability and Computing*, 1(4):351–370, 1992.

[SJ89] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989.

[Smi10] David A. Smith. *The First-fit Algorithm Uses Many Colors on Some Interval Graphs*. PhD thesis, Arizon State University, Tempe, AZ, USA, 2010. AAI3428197.

[Spr35] Richard Sprague. Über mathematische kampfspiele. *Tohoku Mathematical Journal, First Series*, 41:438–444, 1935.

[SS97] Jesús Salas and Alan D Sokal. Absence of phase transition for antiferromagnetic potts models via the dobrushin uniqueness theorem. *Journal of Statistical Physics*, 86(3-4):551–579, 1997.

[STT12] Hadas Shachnai, Gal Tamir, and Tami Tamir. A theory and algorithms for combinatorial reoptimization. In *Latin American Symposium on Theoretical Informatics*, pages 618–630. Springer, 2012.

[SU93] Walter Stromquist and Daniel Ullman. Sequential compounds of combinatorial games. *Theoretical Computer Science*, 119(2):311–321, 1993.

[SVW18] Daniel Stefankovic, Eric Vigoda, and John Wilmes. On counting perfect matchings in general graphs. In *LATIN 2018- 13th Latin American Symposium, Proceedings*, pages 873–885, 2018.

[SW18] Shay Solomon and Nicole Wein. Improved dynamic graph coloring. In *26th Annual European Symposium on Algorithms (ESA 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[Tak18] Asahi Takaoka. Complexity of hamiltonian cycle reconfiguration. *Algorithms*, 11(9):140, 2018.

[Tuc87] Alan Tucker. Coloring perfect (k4- e)-free graphs. *Journal of Combinatorial Theory, Series B*, 42(3):313–318, 1987.

[Uit16] Jos WHM Uiterwijk. Polymerization and crystallization of snowflake molecules in domineering. *Theoretical Computer Science*, 644:143–158, 2016.

[Var17] Shai Vardi. Randomly coloring graphs of bounded treewidth. *arXiv preprint arXiv:1708.02677*, 2017.

[Vig00] Eric Vigoda. Improved bounds for sampling colorings. *Journal of Mathematical Physics*, 41(3):1555–1569, 2000.

[Viz64] Vadim G Vizing. On an estimate of the chromatic class of a p-graph. *Diskret analiz*, 3:25–30, 1964.

[Wal02] Johannes Waldmann. Rewrite games. In *International Conference on Rewriting Techniques and Applications*, pages 144–158. Springer, 2002.

[War16] Mark Daniel Ward. A conjecture about periods in subtraction games. *ArXiv preprint arXiv:1606.04029*, 2016.

[Wei05] Dror Weitz. Combinatorial criteria for uniqueness of gibbs measures. *Random Structures & Algorithms*, 27(4):445–475, 2005.

[Wel93] Dominic Welsh. *Complexity: Knots, Colourings and Countings*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1993.

[Wil74] Richard M Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86–96, 1974.

[Wil89] Herbert S Wilf. *Combinatorial algorithms: an update*, volume 55. SIAM, 1989.

[Wil99] Todd G. Will. Switching distance between graphs with the same degrees. *SIAM Journal on Discrete Mathematics*, 12(3):298–306, 1999.

[Wol93] David Wolfe. Snakes in domineering games. *Theoretical computer science*, 119(2):323–329, 1993.

[Wro14a] Marcin Wrochna. Homomorphism reconfiguration via homotopy. *ArXiv preprint arXiv:1408.2812*, 2014.

[Wro14b] Marcin Wrochna. Reconfiguration in bounded bandwidth and treedepth. *ArXiv preprint arXiv:1405.0847*, 2014.

[WS04] Dror Weitz and Alistair Sinclair. *Mixing in time and space for discrete spin systems*. University of California, Berkeley, 2004.

[Wyt07] Willem A Wythoff. A modification of the game of nim. *Nieuw Arch. Wisk*, 7(2):199–202, 1907.

[Zan15] Tom C. van der Zanden. Parameterized complexity of graph constraint logic. In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation*, volume 43 of *LIPIcs*, pages 282–293, 2015.

[ZZ07] Hamid Zarrabi-Zadeh. Online coloring co-interval graphs. In *Proc. 12th International CSI Computer Conference*, pages 1328–1332. Citeseer, 2007.

[ZZC09] Hamid Zarrabi-Zadeh and Timothy M Chan. An improved algorithm for online unit clustering. *Algorithmica*, 54(4):490–500, 2009.

Index

List of Figures and Tables

Appendix A

Missing proofs from Chapter 5

For any function $f : \Omega \to \mathbb{R}$, the *variance* and the *Dirichlet form* of $\mathcal{L}$ are defined respectively as

$$\mathrm{Var}_{\mathcal{L}}(f) = \frac{1}{2} \sum_{\alpha,\beta\in\Omega} \pi(\alpha)\pi(\beta)(f(\alpha) - f(\beta))^2 ,$$
$$\xi_{\mathcal{L}}(f,f) = \frac{1}{2} \sum_{\alpha,\beta\in\Omega} \pi(\alpha)\mathcal{L}[\alpha \to \beta](f(\alpha) - f(\beta))^2 .$$

Let $\mathcal{F}_{\mathcal{L}} = \{f : \mathrm{Var}_{\mathcal{L}}(f) > 0\}$ and note that $\mathcal{F}_{\mathcal{L}} = \mathcal{F}_{\mathcal{L}'}$ are the set of non-constant functions, as π and π' are positive on Ω. It is well-known (see e.g. Remark 13.13 in [LP17]) that the spectral gap of $\mathcal{L}$ satisfies

$$\mathrm{Gap}(\mathcal{L}) = \min_{f\in\mathcal{F}_{\mathcal{L}}} \frac{\xi_{\mathcal{L}}(f,f)}{\mathrm{Var}_{\mathcal{L}}(f)} . \tag{A.1}$$

A.1 Proof of Proposition 35

We first recall the statement of the proposition. Let $b := \max_{\alpha\in\Omega} \frac{\pi(\alpha)}{\pi'(\alpha)}$. For every $(\sigma,\eta) \in \mathcal{L}$ its congestion is defined as

$$\rho_{\sigma,\eta} := \frac{1}{\pi(\sigma)\mathcal{L}[\sigma \to \eta]\omega(\sigma,\eta)} \sum_{(\alpha,\beta)\in\mathcal{L}'} \sum_{\substack{\gamma\in\Gamma_{\alpha,\beta} \\ \gamma\ni(\sigma,\eta)}} g(\gamma)\pi'(\alpha)\mathcal{L}'[\alpha \to \beta] \cdot |\gamma|_\omega .$$

Let $\rho_{\max} = \max\{\rho_{\sigma,\eta} : (\sigma,\eta) \in \mathcal{L}\}$ be the maximum congestion over all transitions of $\mathcal{L}$. We want to show the following:

Proposition 35 (Weighted multi-commodity flows method)**.** *We have* $\tau(\mathcal{L}) \le b^2\rho_{\max} \cdot \tau(\mathcal{L}')$.

This result is proved by comparing the Dirichlet form and the variance of $\mathcal{L}$ and $\mathcal{L}'$. We have

$$\begin{aligned}
\xi_{\mathcal{L}'}(f,f) &= \frac{1}{2}\sum_{\alpha,\beta\in\Omega}\pi'(\alpha)\mathcal{L}'[\alpha\to\beta](f(\alpha)-f(\beta))^2 \\
&= \frac{1}{2}\sum_{\alpha,\beta\in\Omega}\sum_{\gamma\in\Gamma_{\alpha,\beta}} g(\gamma)\pi'(\alpha)\mathcal{L}'[\alpha\to\beta](f(\alpha)-f(\beta))^2 \\
&= \frac{1}{2}\sum_{\alpha,\beta\in\Omega}\sum_{\gamma\in\Gamma_{\alpha,\beta}} g(\gamma)\pi'(\alpha)\mathcal{L}'[\alpha\to\beta]\left(\sum_{(\sigma,\eta)\in\gamma}\sqrt{\omega(\sigma,\eta)}\frac{f(\sigma)-f(\eta)}{\sqrt{\omega(\sigma,\eta)}}\right)^2 \\
&\le \frac{1}{2}\sum_{\alpha,\beta\in\Omega}\sum_{\gamma\in\Gamma_{\alpha,\beta}} g(\gamma)\pi'(\alpha)\mathcal{L}'[\alpha\to\beta]\cdot|\gamma|_\omega\sum_{(\sigma,\eta)\in\gamma}\frac{(f(\sigma)-f(\eta))^2}{\omega(\sigma,\eta)} \\
&= \frac{1}{2}\sum_{(\sigma,\eta)\in\mathcal{L}}(f(\sigma)-f(\eta))^2\cdot\frac{1}{\omega(\sigma,\eta)}\sum_{(\alpha,\beta)\in\mathcal{L}'}\sum_{\substack{\gamma\in\Gamma_{\alpha,\beta}\\ \gamma\ni(\sigma,\eta)}} g(\gamma)|\gamma|_\omega\pi'(\alpha)\mathcal{L}'[\alpha\to\beta] \\
&\le \rho_{\max}\,\xi_{\mathcal{L}}(f,f)\ ,
\end{aligned}$$

where we used the Cauchy-Schwartz inequality in the first inequality. Additionally, we have

$$\begin{aligned}
\mathrm{Var}_{\mathcal{L}'}(f) &= \frac{1}{2}\sum_{\alpha,\beta\in\Omega}\pi'(\alpha)\pi'(\beta)(f(\alpha)-f(\beta))^2 \\
&\ge \frac{1}{2b^2}\sum_{\alpha,\beta\in\Omega}\pi(\alpha)\pi(\beta)(f(\alpha)-f(\beta))^2 \\
&= \frac{1}{b^2}\mathrm{Var}_{\mathcal{L}}(f)\ .
\end{aligned}$$

Combining the previous two inequalities and using (A.1), the desired result follows,

$$\mathrm{Gap}(\mathcal{L}') = \min_{f\in\mathcal{F}_{\mathcal{L}'}}\frac{\xi_{\mathcal{L}'}(f,f)}{\mathrm{Var}_{\mathcal{L}'}(f)} \le b^2\rho_{\max}\min_{f\in\mathcal{F}_{\mathcal{L}}}\frac{\xi_{\mathcal{L}}(f,f)}{\mathrm{Var}_{\mathcal{L}}(f)} = b^2\rho_{\max}\,\mathrm{Gap}(\mathcal{L})\ .$$

A.2 Proof of Proposition 42

We first recall the statement of the proposition:

Proposition 42. *Let $G=(V,E)$ be a graph on n vertices, and k be a positive integer. Let $v\in V$ such that $N(v)$ induces a clique of size at most $k-2$. For any choice of parameters $(p_1,\dots,p_n)$, the Glauber dynamics $\mathcal{L}_V$ and $\mathcal{L}_{V\setminus\{v\}}$ for k-colourings of V and $V\setminus\{v\}$ respectively and defined with the same parameters satisfy,*

$$\tau\left(\mathcal{L}_{V\setminus\{v\}}\right)\le\tau(\mathcal{L}_V)\ .$$

Let $U=\{u_1,\dots,u_m\}=V\setminus\{v\}$. Let $p_1,\dots,p_m,p_v$ be the parameters for $u_1,\dots,u_m,v$, respectively. Let Ω_V and Ω_U be the set of L-colourings of G and $G[U]$ respectively. Let $d=|N(v)|$ the degree of v. Since $N(v)$ is a clique, we have:

$$|\Omega_V| = |\Omega_U|(k-d)\ .$$

Indeed, for every colouring α_U of U, there are exactly $(k-d)$ possibilities to extend it into a colouring of V. Let π_V and π_U be the stationary distributions of $\mathcal{L}_V$ and $\mathcal{L}_U$, which are uniform as the transitions are symmetric. Recall that $\mathcal{F}_{\mathcal{L}_V}$ is the set of non-constant functions from Ω_V to $\mathbb{R}$. Let $\mathcal{F}^v_{\mathcal{L}_V}$ be the subset of these functions which are independent of v, i.e. which satisfy $f(\alpha) = f(\beta)$ whenever α and β agree on U.

Using (A.1), we have

$$\mathrm{Gap}(\mathcal{L}_V) = \min_{f \in \mathcal{F}_{\mathcal{L}_V}} \frac{\mathcal{E}_{\mathcal{L}_V}(f)}{\mathrm{Var}_{\mathcal{L}_V}(f)} \leq \min_{f \in \mathcal{F}^v_{\mathcal{L}_V}} \frac{\mathcal{E}_{\mathcal{L}_V}(f)}{\mathrm{Var}_{\mathcal{L}_V}(f)} \,. \tag{A.2}$$

Let $f \in \mathcal{F}_V$, then we have the following

$$\begin{aligned}
\mathrm{Var}_{\mathcal{L}_V}(f) &= \frac{1}{2} \sum_{\alpha,\beta \in \Omega_V} \pi_V(\alpha)\pi_V(\beta)(f(\alpha) - f(\beta))^2 \\
&= \frac{1}{2} \sum_{\alpha,\beta \in \Omega_V} \frac{1}{|\Omega_V|^2}(f(\alpha) - f(\beta))^2 \\
&= \frac{1}{2} \sum_{\alpha_U,\beta_U \in \Omega_U} \sum_{\substack{c \in [k] \\ c \notin \alpha(N(v))}} \sum_{\substack{c' \in [k] \\ c' \notin \beta(N(v))}} \frac{1}{|\Omega_V|^2}(f(\alpha_U) - f(\beta_U))^2 \\
&= \frac{1}{2} \sum_{\alpha_U,\beta_U \in \Omega_U} \frac{(k-d)^2}{|\Omega_V|^2}(f(\alpha_U) - f(\beta_U))^2 \\
&= \mathrm{Var}_{\mathcal{L}_U}(f)
\end{aligned} \tag{A.3}$$

Additionally, if $f \in \mathcal{F}^v_V$, then we have

$$\begin{aligned}
\mathcal{E}_{\mathcal{L}_V}(f) &= \frac{1}{2} \sum_{u_i \in U} \sum_{\substack{\alpha,\beta \in \Omega_V \\ \text{differ at } u_i}} \frac{1}{|\Omega_V|} p_i (f(\alpha) - f(\beta))^2 + \sum_{\substack{\alpha,\beta \in \Omega_V \\ \text{differ at } v}} \frac{1}{|\Omega_V|} p_v (f(\alpha) - f(\beta))^2 \\
&= \frac{1}{2} \sum_{u_i \in U} \sum_{\substack{\alpha_U,\beta_U \in \Omega_U \\ \text{differ at } u_i}} \sum_{\substack{c \in [k] \\ c \notin \alpha(N(v)) \\ c \notin \beta(N(v))}} \frac{1}{|\Omega_V|} p_i (f(\alpha_U) - f(\beta_U))^2 \\
&\leq \frac{1}{2} \sum_{u_i \in U} \sum_{\substack{\alpha_U,\beta_U \in \Omega_U \\ \text{differ at } u_i}} \frac{k-d}{|\Omega_V|} p_i (f(\alpha_U) - f(\beta_U))^2 \\
&= \mathcal{E}_{\mathcal{L}_U}(f) \,.
\end{aligned} \tag{A.4}$$

where we used that $N(v)$ is a clique in the inequality. Putting together (A.2)–(A.4), gives as required

$$\mathrm{Gap}(\mathcal{L}_V) \leq \mathrm{Gap}(\mathcal{L}_U) \,.$$