université de BORDEAUX

THÈSE

présentée à

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

par **Thomas Bellitto**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Walks, Transitions and Geometric Distances in Graphs.

Soutenue publiquement le 27 août 2018

après avis des rapporteurs :
Jørgen Bang-Jensen, Professeur, University of Southern Denmark
Sylvain Gravier, Directeur de recherche, Université Joseph Fourier

devant la commission d'examen composée de :
Christine Bachoc, professeure, Université de Bordeaux, directrice de thèse
Jørgen Bang-Jensen, professeur, University of Southern Denmark, rapporteur
Sylvain Gravier, directeur de recherche, Université Joseph Fourier, rapporteur
Mickael Montassier, professeur, Université de Montpellier, examinateur
Arnaud Pêcher, professeur, Université de Bordeaux, directeur de thèse
Éric Sopena, professeur, Université de Bordeaux, président du jury

– 2018 –

Walks, Transitions and Geometric Distances in Graphs.

Abstract This thesis studies combinatorial, algorithmic and complexity aspects of graph theory problems, and especially of problems related to the notions of walks, transitions and distances in graphs.

We first study the problem of traffic monitoring, in which we have to place as few censors as possible on the arcs of a graph to be able to retrace walks of objects. The characterization of instances of practical interests brings us to the notion of forbidden transitions, which strengthens the model of graphs. Our work on forbidden-transition graphs also includes the study of connecting transition sets, which can be seen as a translation to forbidden-transition graphs of the notion of spanning trees.

A large part of this thesis focuses on geometric graphs, which are graphs whose vertices are points of the real space and whose edges are determined by geometric distance between the vertices. This graphs are at the core of the famous Hadwiger-Nelson problem and are of great help in our study of the density of sets avoiding distance 1 in various normed spaces. We develop new tools to study these problems and use them to prove the Bachoc-Robins conjecture on several parallelohedra. We also investigate the case of the Euclidean plane and improve the bounds on the density of sets avoiding distance 1 and on its fractional chromatic number.

Finally, we study the complexity of graph homomorphism problems and establish dichotomy theorems for the complexity of locally-injective homomorphisms to reflexive tournaments.

Keywords Graphs, walks, forbidden transitions, graph homomorphisms, independence number, geometric distances, sets avoiding distance 1, NP-completeness

Affiliation University of Bordeaux, CNRS, LaBRI, UMR 5800, F-33400 Talence, France

Marches, Transitions et Distances Géométriques dans les Graphes.

Résumé Cette thèse étudie les aspects combinatoires, algorithmiques et la complexité de problèmes de théorie des graphes, et tout spécialement de problèmes liés aux notions de marches, de transitions et de distance dans les graphes.

Nous nous intéressons d'abord au problème de *traffic monitoring*, qui consiste à placer aussi peu de capteurs que possible sur les arcs d'un graphe de façon à pouvoir reconstituer des marches d'objets. La caractérisation d'instances intéressantes dans la pratique nous amène à la notion de transitions interdites, qui renforce le modèle de graphe. Notre travail sur les graphes à transitions interdites comprend aussi l'étude de la notion d'ensemble de transitions connectant, que l'on peut voir comme l'analogue en terme de transitions de la notion d'arbre couvrant.

Une partie importante de cette thèse porte sur les graphes géométriques, qui sont des graphes dont les sommets sont des points de l'espace réel et dont les arêtes sont déterminées par les distances géométriques entre les sommets. Ces graphes sont au cœur du célèbre problème de Hadwiger-Nelson et nous sont d'une grande aide dans notre étude de la densité des ensembles qui évitent la distance 1 dans plusieurs types d'espaces normés. Nous développons des outils pour étudier ces problèmes et les utilisons pour prouver la conjecture de Bachoc-Robins sur plusieurs parallélotopes. Nous nous penchons aussi sur le cas du plan euclidien et améliorons les bornes sur la densité des ensembles évitant la distance 1 et sur son nombre chromatique fractionnaire.

Enfin, nous étudions la complexité de problèmes d'homomorphismes de graphes et établissons des théorèmes de dichotomie sur la complexité des homomorphismes localement injectifs vers les tournois réflexifs.

Mots-clés Graphes, marches, transitions interdites, homomorphismes de graphes, nombre de stabilité, distances géometriques, ensemble évitant la distance 1, NP-complétude

Affiliation Université de Bordeaux, CNRS, LaBRI, UMR 5800, F-33400 Talence, France

Acknowledgements

Je commence bien sûr cette section par un très grand merci à Arnaud, qui m'encadre depuis mon stage de master et qui m'a énormément aidé à trouver ce que je voulais faire et à arriver là où j'en suis. Merci pour ton soutien, ton aide et tes conseils dans tellement de domaines : la recherche, l'enseignement, la rédaction, la recherche de stages puis de postdocs, les candidatures... Merci aussi pour ta confiance et la liberté que tu m'as laissée pendant ma thèse pour trouver les problèmes sur lesquels je voulais travailler et les directions que je voulais explorer. Merci surtout pour tout le temps que tu as toujours réussi à trouver dans ton emploi du temps chargé pour écouter, relire et essayer de comprendre mes idées tordues. Un grand merci aussi à Christine, qui m'a beaucoup aidé à enrichir mon domaine de recherche, qui m'a fait découvrir des problèmes passionnants et qui m'a énormément aidé à les relier à ce que je savais faire et à partir dans la bonne direction.

I also thank Sylvain Gravier and Jørgen Bang-Jensen for accepting to read the manuscript and Éric Sopena and Mickael Montassier for being part of the jury.

I would also like to thank Jørgen Bang-Jensen again and Anders Yeo for choosing me for an open postdoc position in their team. I am really excited for this next important step of my journey.

Of course, I would like to thank my co-authors and all the people I have worked with and without whose ideas this thesis would probably never have existed. Arnaud et Christine, une fois encore. Philippe, mon frère de thèse, avec qui ça a toujours été un grand plaisir de travailler ou de discuter. Gary, of course; I had an amazing time in Victoria thanks to you and you have done a lot to help me since. Thank you to my other co-authors from Victoria too: Chris, Stefan and Feiran. Merci à Benjamin avec qui j'ai beaucoup aimé travailler et échanger. Merci aussi à Antoine avec qui ça a été un plaisir de travailler ces derniers mois.

Thank you to my previous advisors: Gunnar Klau, Tobias Marschall, Alexander Schönhuth, David Coudert, Nicolas Nisse and Endre Boros.

Merci évidemment à toute l'équipe de théorie des graphes du LaBRI pour de nombreux échanges passionnants et pour un cadre de travail et une ambiance très agréables. Merci aussi à mon équipe INRIA, Réalopt, et à mes co-bureaux, Henri et Théo.

Merci à Hervé et à toute l'équipe de Maths à Modeler pour cette expérience très enrichissante.

Un grand merci aussi à tous mes collègues d'équipes pédagogiques. Je pense

notamment à Carole, Olivier, Feri et Cyril avec qui ça a été un grand plaisir de travailler.

Merci aussi à tous ceux que j'ai oubliés et qui m'ont aidé à arriver là où je suis aujourd'hui.

Merci à tous mes amis, du labo et d'ailleurs. Un énorme merci tout particulièrement à Théo et Antonin pour leur soutien et pour tous les bons moments passés ensemble ces dernières années.

Merci enfin à ma famille et surtout à ma mère et mon frère pour leur soutien inconditionnel depuis toutes ces années.

Contents

Introduction (en français)

Cette thèse étudie des problème et des notions liées à la structure mathématique de graphe. Les graphes et graphes orientés sont un modèle puissant qui permet de décrire n'importe quelle relation binaire sur un ensemble. Les éléments de cet ensemble sont appelés des *sommets* et les paires d'éléments liés (ou adjacent) sont appelées *arêtes*. Grâce à leur expressivité, les graphes trouvent des applications dans d'innombrables domaines: systèmes d'information, télécommunications, bio-informatique, traitement d'images, réseaux de transports, réseaux sociaux, planification... et bien d'autres. Les graphes ont été introduits il y a presque trois siècles et l'intérêt qui leur est porté n'a cessé de grandir depuis, surtout avec l'émergence de l'informatique.

Plus précisément, cette thèse s'intéresse à la notion de marche et de distance dans les graphes, ainsi qu'aux aspects combinatoires, algorithmiques et à la complexité de problèmes liés. Une marche dans un graphe est une suite de sommets adjacents qui permet de relier deux sommets. Le nombre d'éléments dans la marche permet de définir sa longueur et la longueur des marches entre deux sommets définit leur distance. Nous étudions également de près les graphes géométriques, qui sont des graphes dont les sommets sont des points de l'espace réel. La relation entre la distance définie par l'adjacence dans le graphe et la distance géométrique nous intéressera tout particulièrement. Nos travaux font aussi beaucoup intervenir d'autres notions connues de théorie des graphes, dont celles d'ensemble stable, de coloration et d'homomorphisme.

Les problèmes et résultats présentés dans cette thèse peuvent se diviser en trois parties.

Transitions dans les graphes

La première partie se penche sur le modèle de graphes à transitions interdites. La puissance des modèles de graphe et de marche en fait les modèles de choix pour étudier des problèmes de routage dans de nombreux contextes. Par exemple, le réseau routier d'une ville peut être modélisé par un graphe dans lequel chaque endroit d'intérêt et chaque croisement sont représentés par des sommets et où l'existence d'une route directe entre deux sommets est traduite par une arête (ou un arc dans le cas d'une route à sens unique). Ce faisant, nous définissons implicitement un ensemble des marches entre chaque paire de sommets qui peut-être utilisé pour résoudre de nombreux problèmes. On peut par exemple résoudre des problèmes d'optimisation pour trouver le plus court chemin entre deux points, éviter des embouteillages, ou

chercher des attributs qui distinguent un ensemble de marches. Cependant, dans la pratique, beaucoup des marches définies par le graphe représentent des itinéraires qu'un conducteur n'a pas le droit de prendre. Pour modéliser une situation où un conducteur n'est pas autorisé à tourner à gauche ou à droite à une intersection, nous définissons les transitions dans le graphe comme des paires d'arêtes consécutives. L'étude des graphes à transitions interdites, où la définition du graphe inclut un ensemble de transitions autorisées ou interdites, est un domaine de la théorie des graphes qui émerge rapidement. Il existe aussi plusieurs autres modèles proches, parmi lesquels les marches proprement colorées sur des graphes arêtes-colorés ou les graphes à sous-chemins interdits.

Le premier des deux principaux problèmes liés à la notion de transitions interdites que nous étudions dans cette thèse est le problème de *traffic monitoring*. Dans le cadre de ce problème, on nous donne un graphe dans lequel des objets se déplacent et nous connaissons à l'avance un ensemble de marches possibles que ces objets peuvent prendre. Nous avons la possibilité de placer des capteurs sur les arcs du graphe qui nous indiquent quand un objet passe par un arc équipé. Ainsi, pour chaque objet, nous connaissons la suite ordonnée des arcs équipés qu'il a utilisé. Notre objectif est de trouver comment placer aussi peu de capteurs que possible sur le graphe de façon à ce que les informations qu'ils renvoient suffisent quand même à reconstituer exactement l'itinéraire des marcheurs. Prenons par exemple le graphe représenté en figure 1, où les arcs tu, vw et wx sont équipés de capteurs que nous appelons a, b et c respectivement. Un objet qui suit la marche (t, u, v, w, x, z) activera successivement les capteurs a, b et c. Un objet qui suit la marche (t, u, w, x, y, v, w, z) activera les même capteurs mais activera c avant b. Ainsi, l'ensemble de capteurs $\{a, b, c\}$ permet de distinguer ces deux marches.

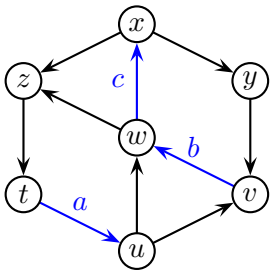


Figure 1: Un graphe dont trois arcs (représentés en bleu) sont équipés de capteur.

La complexité de ce problème a déjà été étudié dans [85], mais avant nos travaux, les seuls algorithmes conçus pour ce problème [88] se plaçaient dans le cas des graphes acycliques, dans lequel les marches possibles sont très limitées et ne peuvent notamment pas passer plusieurs fois par le même sommet ou le même arc.

Le problème de *traffic monitoring* requiert de trouver des moyens efficaces de décrire des ensembles des marches dans un graphe, ce que nous faisons grâce a des outils de théorie des langages. Nous faisons apparaître des liens entre le problème de *traffic monitoring* et le problème de code séparateur grâce à un nouveau modèle de séparation basé sur les langages, qui nous permet de prendre en compte le nombre de fois et l'ordre dans lequel les capteurs sont activés. Nous essayons en-

suite d'identifier les types d'instances les plus utiles en pratique et d'exploiter leur spécificités pour développer des algorithmes aussi efficace que possible. C'est ici que le modèle de transitions interdites prend toute son importance. Nous étudions plusieurs types d'instances, dont certains font apparaître des transitions interdites, et nous développons des algorithmes pour reformuler le problème de *traffic monitoring* sous la forme d'un programme linéaire en nombres entiers [9].

L'autre problème que nous étudions dans cette partie de la thèse est celui d'ensemble de transitions connectant minimum, qui peut être vu comme l'analogue en terme de transitions du problème d'arbre couvrant minimum. Étant donné un graphe connexe, un arbre couvrant minimum est un sous-ensemble d'arêtes de taille minimum qui assure la connectivité du graphe, *i.e.* tel qu'il existe une marche entre toute paire de sommets qui n'utilise que des arêtes de l'arbre couvrant minimum. De la même façon, un ensemble de transitions connectant minimum est un ensemble de transitions aussi petit que possible tel qu'il existe une marche entre chaque paire de sommets qui n'utilise que des transitions de l'ensemble connectant minimum. D'après la définition standard d'une transition dans un graphe non orienté, utiliser deux fois de suite la même arête n'est pas une transition et est donc toujours autorisé. Par exemple, la figure 2 représente un graphe où les deux transitions autorisées sont uvy et wvy. Il existe des marches autorisées entre w et n'importe quel autre sommet du graphe: les marches (w, v) et (w, x) n'utilisent aucune transition, la marche (w, v, y) utilise une transition autorisée et même si la marche (w, v, u) est interdite, il est toujours possible de relier w à u par la marche (w, v, y, v, u). Néanmoins, il est impossible d'aller de x à u ou y et l'ensemble de transitions $\{wvy, uvy\}$ n'est donc pas connectant. Il le devient si on lui ajoute par exemple la transition uvx.

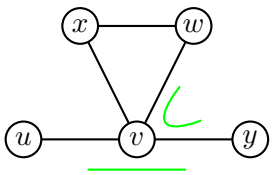


Figure 2: Un graphe avec deux transitions autorisées, représentées en vert.

Dans beaucoup de champs d'application où le modèle de transitions interdites est pertinent (c'est le cas par exemple des réseaux routiers, des réseaux de transports ou des réseaux optique de télécommunications), il peut arriver qu'à cause de travaux de maintenance ou d'un malfonctionnement, certaines transitions deviennent inutilisables. Le problème de robustesse à la suppression de transitions a déjà été étudié pour plusieurs propriétés des graphes [105] et la connexité est une propriété fondamentale qu'on attend des réseaux dans tous les champs d'application des transitions interdites. Toutefois, ce que nous étudions ici n'est pas le plus petit nombre de transitions à enlever pour déconnecter le graphe, mais à l'inverse, le plus petit nombre de transitions à assurer pour garantir la connexité du graphe. Sans donner d'information pertinente sur la robustesse aux pannes du réseau, cette grandeur permet toutefois de mettre en évidence quelle partie du réseau sont les plus importantes à son bon fonctionnement et peut donc s'avérer utile dans la conception

de réseau robuste, comme les arbres couvrants sont utiles dans les domaines où les pannes affectent les arêtes et non pas les transitions.

Cette thèse présente les résultats d'un travail commun avec Benjamin Bergougnoux et étudie différents aspects de ce problème. Nous analysons les aspects structurels des ensembles connectants minimaux, et en ressortons une reformulation du problème sous forme d'un problème de décomposition de graphe que nous appelons *hypergraphes connectants optimaux*. Ce nouveau problème s'avère plus facile à manipuler et se montre d'une grande aide dans les preuves des résultats suivants. Nous nous intéressons ensuite à l'aspect algorithmique du problème et à sa complexité et nos résultats principaux sont la preuve de NP-complétude du problème et la mise au point d'une $\frac{3}{2}$-approximation polynomiale [10].

Graphes géométriques et ensemble évitant la distance 1

La deuxième partie de cette thèse présente les résultats de projets menés avec Christine Bachoc, Philippe Moustrou, Arnaud Pêcher et Antoine Sedillot sur la densité maximale qui peut être atteinte par des ensembles de points A de l'espace réel qui ne contiennent pas deux points à distance exactement 1. Intuitivement, la densité est la portion d'espace qu'occupe A. La plus grande densité atteignable dépend aussi de la distance dont on munit $\mathbb{R}^n$. Dans cette thèse, nous ne considérons que des distances induites par des normes et nous notons cette quantité $m_1(\mathbb{R}^n, \|\cdot\|)$ où $\|\cdot\|$ est notre norme sur $\mathbb{R}^n$. La densité maximale des ensembles évitant la distance 1 a été étudiée depuis au moins le début des années 1960 [91], surtout dans le cas du plan euclidien, mais s'avère être un problème très difficile qui est encore très ouvert.

Prenons un empilement de disques de rayon 1 (*i.e.* un ensemble de disques de rayon 1 deux à deux disjoints). Un exemple d'ensemble qui ne contient pas deux points à distance 1 est l'union de disques ouverts de rayon $\frac{1}{2}$ et de même centre que les disques de notre empilement. La densité maximale d'un empilement de disque dans le plan euclidien est d'environ 0.9069 et on sait ainsi que $m_1(\mathbb{R}^2) \geqslant \frac{0.9069}{4} \geqslant 0.2267$. Un ensemble réalisant cette borne est illustré en figure 3. La meilleure borne inférieure est à peine meilleure ; elle a été obtenue par Croft en 1967 en affinant la construction précédente et est d'environ 0.2293. À l'inverse, les bornes supérieures ont été améliorées plusieurs fois au fil des années mais sont toujours loin de la borne inférieure. Avant nos travaux, la meilleure borne supérieure connue était de $m_1(\mathbb{R}^2) \leqslant 0.258795$ et a été établie par Keleti et al. en 2015 [70] à l'aide d'une approche basée sur l'analyse harmonique très différente de la notre. Dans cette thèse, nous développons une nouvelle approche et améliorons la borne en $m_1(\mathbb{R}^2) \leqslant 0.256828$.

Un objectif important des travaux portant sur ce sujet est de prouver une conjecture d'Erdős selon laquelle $m_1(\mathbb{R}^2) < \frac{1}{4}$. Remarquons que si la norme est telle que l'ensemble des points à distance 1 de l'origine est un polytope qui pave parfaitement l'espace (c'est à dire qu'il existe un empilement de polytopes unité de densité 1), la construction présentée dans le paragraphe précédent produit un ensemble évitant la distance 1 de densité exactement $\frac{1}{4}$. Ainsi, la conjecture d'Erdős dit que m_1 est plus petit dans le plan euclidien à cause des espaces vides entre les disques d'un empilement optimal.

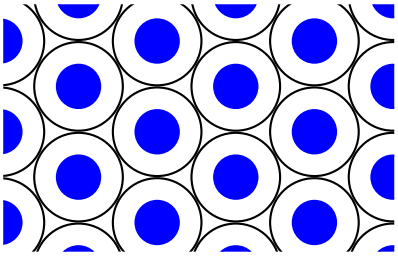


Figure 3: Un exemple d'ensemble évitant la distance euclidienne 1 (en bleu).

Pour progresser vers la conjecture d'Erdős, Bachoc et Robins ont étudié la densité des ensembles évitant la distance 1 pour des distantes induites par des normes pour lesquelles le polytope unité pave parfaitement l'espace (on appelle de telles normes des *normes paralléloèdres*). Ils ont conjecturé que dans ce cas, la construction précédente était optimale et donc, que si $\|\cdot\|_{\mathcal{P}}$ est une norme paralléloèdre, $m_1(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}}) = \frac{1}{2^n}$. Dans cette thèse, nous prouvons cette conjecture en dimension 2 ainsi que pour une famille de normes paralléloèdres en dimension quelconque et nous établissons des bornes sur $m_1(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ pour une autre famille de normes [5].

Ce problème est étroitement lié au célèbre problème de Hadwiger-Nelson, qui consiste à déterminer combien de couleurs sont nécessaires pour colorer tous les points du plan de sorte que deux points à distance exactement 1 n'aient jamais la même couleur. La figure 4 illustre une coloration d'une portion du plan. Par exemple, un triangle équilatéral de côté 1 dessiné dans un plan proprement coloré aura forcément ses trois sommets de couleurs différentes.

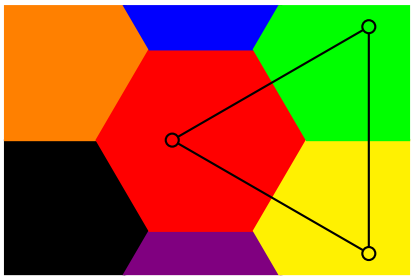


Figure 4: Une coloration d'une portion du plan euclidien.

De nombreuses variantes de ce problème ont également été étudiées, parmi lesquelles la coloration d'espaces munis de distances non-euclidiennes, la coloration mesurable (où on impose que les classes de couleur soient mesurables) ou encore la coloration fractionnaire, où l'on cherche le plus petit nombre $\frac{a}{b}$ tel que a couleurs différentes soient suffisantes pour donner à chaque point du plan b couleurs distinctes de façon à ce que deux points à distance 1 n'aient aucune couleur en commun.

Le problème de Hadwiger-Nelson a fait l'objet de nombreux travaux depuis au moins 1960 [52]. Les bornes inférieures et supérieures de 4 et 7 ont été rapidement établies, mais personne n'a su les améliorer jusqu'à ce que De Grey prouve en avril 2018 dans [34] qu'au moins 5 couleurs sont nécessaires pour colorer proprement le plan euclidien.

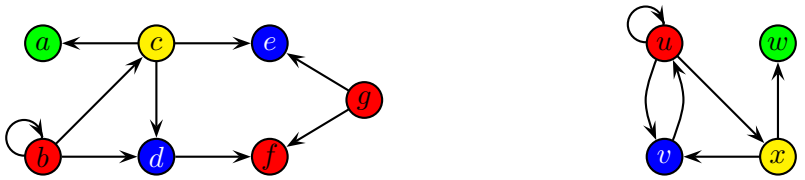
Tous ces problèmes peuvent être reformulés comme des problèmes dans des graphes géométriques. Le graphe géométrique dont les sommets sont tous les points de l'espace et dont les arêtes sont les paires de points à distance géométrique 1 est appelé le graphe distance-unité. La densité des ensembles évitant la distance 1, le problème de Hadwiger-Nelson et ses variantes reviennent tous à déterminer la valeur d'invariants de graphes bien connus, tels que le taux de stabilité, le nombre chromatique ou le nombre chromatique fractionnaire, sur le graphe distance-unité du plan euclidien.

Les graphes distance-unité ont une infinité non dénombrable de sommets et sortent du cadre des mathématiques discrètes dans lequel s'inscrit traditionnellement la théorie des graphes. Cependant, beaucoup d'informations sur le graphe distance-unité peuvent être déduites de ses sous-graphes. Par exemple, le triangle équilatéral représenté en figure 4 définit un sous-graphe complet de taille 3 pour lequel trois couleurs sont nécessaires. On prouve ainsi qu'au moins trois couleurs sont requises pour colorer le plan euclidien entier. Nous verrons que des propriétés similaires sont vérifiées par la densité des ensembles stables (qui sont des ensembles de sommets deux-à-deux non-adjacents et sont donc directement reliés à m_1).

Nous étudions algorithmiquement la notion de taux de stabilité pondéré optimal et l'utilisons pour étudier les problèmes décrits précédemment. Cette méthode est au cœur de plusieurs projets en cours et a déjà amené des résultats intéressants. Parmi ces résultats, on peut citer l'amélioration de la borne supérieure sur $m_1(\mathbb{R}^2)$, mais aussi l'amélioration de la borne inférieure sur le nombre fractionnaire chromatique du plan de 3.61904 [31] à 3.89366 et une borne supérieure sur $m_1(\mathbb{R}^3, \|\cdot\|, \|_{\mathcal{P}})$ quand $\mathcal{P}$ est un paralléloèdre régulier [4] [11].

Complexité des homomorphismes de graphes

La troisième partie de cette thèse porte sur la complexité de problèmes d'homomorphismes de graphes, et plus particulièrement sur les homomorphismes localement injectifs de graphes orientés. Un homomorphisme de graphe est une fonction des sommets d'un graphe vers les sommets d'un autre telle que l'image d'une arête est une arête, ou dans le cas orienté, telle que l'image d'un arc est un arc. Par exemple, dans la figure 5, la fonction qui associe b, f et g à u, d et e à v, a à w et c à x est un homomorphisme du graphe représenté en figure 5a vers le graphe de la figure 5b.



(a) (b)

Figure 5: On attribue une couleur à chaque sommet du graphe cible (à droite). La couleur des sommets du graphe de gauche indique leur image par l'homomorphisme.

Les homomorphismes sont une généralisation de la coloration de graphes. Dans le cas non-orienté, la coloration est le cas particulier d'homomorphisme où chaque

sommet du graphe cible est relié à chaque autre, mais pas à lui-même. Dans le cas orienté, les colorations sont définies comme des homomorphismes vers des graphes qui contiennent exactement un arc entre chaque paire de sommets distincts. De tels graphes s'appellent des *tournois* et sont très importants dans l'étude des homomorphismes orientés.

Les homomorphismes localement injectifs sont un cas particulier d'homomorphismes où on ajoute la contrainte que l'homomorphisme doit être injectif sur le voisinage des sommets du graphe en entrée. Les homomorphismes localement injectifs sont intéressants du fait qu'ils préservent la structure locale du graphe en entrée bien mieux que ne le font les homomorphismes standards. Ainsi, les homomorphismes localement-injectifs trouvent des applications dans une grande variété de domaines, parmi lesquels la détection de motifs en analyse d'image, la bio-informatique ou encore la théorie des codes. Cependant, en fonction de ce que l'on veut modéliser, nos contraintes ne sont pas exactement les mêmes et l'on trouve ainsi plusieurs définitions de l'injectivité locale dans la littérature. Par exemple, on peut vouloir que l'homomorphisme soit injectif sur le voisinage ouvert des sommets ou sur leur voisinage fermé (qui inclut le sommet lui-même). En fonction de la définition, le fait que sur la figure 5, le sommet g ait la même image que son voisin f peut être compatible ou non avec le critère d'injectivité locale. De la même façon on peut imposer que l'homomorphisme soit injectif sur les voisinages entrants et sortants des sommets séparément ou sur leur union. Ici encore, en fonction de la définition, le fait qu'un voisin entrant (le sommet b) et qu'un voisin sortant (f) de d aient la même image peut être autorisé ou non. On choisit en fonction des applications les propriétés du graphe initial qui doivent être préservées et on choisit la définition d'injectivité locale qui le permet. Par exemple, si l'homomorphisme est injectif sur l'union des voisinages entrants et sortants des sommets, on assure que l'image d'un chemin (une marche où les sommets sont deux à deux distincts) de longueur 2 sera également un chemin de longueur 2. Néanmoins, renforcer le modèle peut rendre algorithmiquement plus difficile le problème de déterminer l'existence d'un homomorphisme localement injectif entre deux graphes. C'est ce que nous étudions dans cette partie de la thèse.

L'objectif habituel des études sur la complexité des problèmes comme la k-coloration ou l'existence d'un homomorphisme vers un graphe G donné est la mise au point d'un théorème de dichotomie : on partitionne les problèmes en deux classes, on prouve que ceux de la première classe sont polynomiaux et que ceux de la deuxième sont NP-complets. Par exemple, dans le cas non-orienté, il est connu que la k-coloration est polynomiale pour $k \leqslant 2$ et NP-complète pour $k > 2$. Hell et Nešetřil ont prouvé dans [61] que le problème d'homomorphisme vers un graphe non-orienté G est polynomial si G possède une boucle ou est biparti et NP-complet dans le cas contraire. Ces résultats sont des théorèmes de dichotomie.

Notre point de départ dans l'étude de la complexité des homomorphismes localement injectifs est le cas où la cible est un tournoi. En effet, la complexité des variantes d'homomorphismes étudiées dans la littérature dépend souvent du nombre chromatique de la cible. Par exemple, dans le théorème de Hell-Nešetřil, les graphes bipartis sont exactement les graphes 2-colorables. Parmi les quatre définitions non équivalentes de l'injectivité locale que nous avons recensées dans le cas orienté, la

complexité de trois est ouverte dans le cas des tournois. Ces trois définitions sont équivalentes dans le cas où la cible ne possède pas de boucles et l'une d'elles ne tient pas compte des boucles sur la cible. Sa complexité est donc impliquée par celle des deux autres et il nous reste donc à étudier deux définitions d'injectivité locale. Sous ces définitions, Campbell, Clarke et MacGillivray ont déjà établi dans [20] que le problème est polynomial vers le tournoi réflexif à deux sommets et NP-complet vers les deux tournois réflexifs à trois sommets. Cependant, aucun théorème de dichotomie n'avait émergé sur une classe de tournois infinie avant nos travaux.

Avec Stefan Bard, Christopher Duffy, Gary MacGillivray et Feiran Yang, nous avons établi un théorème de dichotomie sur la complexité des homomorphismes localement injectifs vers les tournois réflexifs pour nos deux définitions d'injectivité locale : dans les deux cas, le problème est polynomial si la cible a deux sommets ou moins et NP-complet si la cible en a trois ou plus [7]. Nos résultats impliquent également un théorème de dichotomie sur la complexité des colorations réflexives localement injectives.

Contenu de la thèse

Le chapitre 1 présente les notions de base que nous utilisons tout au long de la thèse. Ces notions viennent de domaines variés des mathématiques et de l'informatique dont la théorie des graphes, de la complexité, la géométrie, la théorie des langages et la programmation mathématique.

Le chapitre 2 introduit le problème de *traffic monitoring*, présente le modèle et les outils que nous développons pour l'étudier et montre comment les utiliser pour résoudre le problème sur plusieurs types d'instances intéressantes en pratique.

Le chapitre 3 présente le problème d'ensemble de transitions connectant minimum, l'étudie sur plusieurs classes de graphes et présente une reformulation, une approximation polynomiale et une preuve de NP-complétude dans le cas général.

Le chapitre 4 présente le problème de la densité des ensembles évitant la distance 1 ainsi que son contexte, et notamment le problème de Hadwiger-Nelson. Nous y présentons une méthode qui utilise des bornes sur le taux de stabilité de graphes géométriques infinis et l'utilisons pour prouver la conjecture de Bachoc-Robins sur plusieurs familles de paralléloèdres dont tous ceux de dimension 2, et nous établissons des bornes supérieures sur $m_1(\mathbb{R}^n, \|\cdot\|)$ pour d'autres.

Le chapitre 5 présente la notion de taux de stabilité pondéré optimal, l'étudie sur des graphes géométriques et l'utilise pour étendre les résultats du chapitre précédent. Ce chapitre contient des améliorations sur la borne du nombre fractionnaire chromatique du plan euclidien et sur la densité des ensembles évitant la distance 1 dans le plan euclidien et dans l'espace en trois dimensions équipé de normes paralléloèdres régulières.

Le chapitre 6 présente le problème d'homomorphismes localement injectifs et les deux définitions que nous considérons et prouve pour les deux un théorème de dichotomie sur les tournois réflexifs.

Le chapitre 7 conclut la thèse en résumant les contributions principales et en présentant des problèmes ouverts que nos travaux soulèvent et qui pourraient faire l'objet de futurs projets.

Introduction

This thesis studies problems and notions that revolve around the mathematical objects called graphs. Graphs and directed graphs are very powerful models that allow to describe any binary relation on a set. The elements of this set are called *vertices* and the pair of related (or adjacent) vertices are called *edges*. Because of their expressiveness, graphs find application in a countless variety of fields: information systems, telecommunication, bio-informatics, image processing, transportation systems, social networks, scheduling.... among many others. Graphs were introduced almost three centuries ago and have received ever-increasing attention since, especially with the emergence of computer science.

More precisely, this thesis pays close attentions to the notions of walks and distances in graphs and to the combinatorial, algorithmic and complexity aspects of some of the related problems. A walk in a graph is a sequence of adjacent vertices that allows to connect two vertices. The number of elements in the walk allows to define its length and the length of the walks between two vertices leads to the definition of their distance. We closely study geometric graphs, which are graphs whose vertices are point of the real space. The connection between the distance defined by the adjacency in the graph and the geometric distance will be of great interest to us. Our works also involve extensively other well-known notions of graph theory, such as independent sets, graph colourings and homomorphisms.

The problems and results presented in this thesis can be divided into three parts.

Transitions in graphs

The first part focuses on the model of forbidden-transition graphs. The strength of the models of graphs and walks makes them the model of choice to address routing problems in various contexts. For example, the road network in a city can be modelled by a graph where every potential destination or crossroad is denoted by a vertex and roads between two vertices, by edges (or arcs if the road is halfway). We thereby implicitly define a set of possible walks between each pair of vertices that can then be used for many purposes. Examples include several optimization problem, such as finding the shortest walk between two points, avoiding traffic jams, or monitoring and separating a set of possible walks. However, in practice, many of the walks that the graph defines denote routes that drivers are not allowed to take. To model a situation where a driver may not turn left or right at a given crossroad, we define a transition in a graph as a pair of consecutive edges. The study of forbidden-transition graphs, where graphs are defined together with a set of

permitted or forbidden transitions, is a fast emerging area of graph theory. Several other related models are also widely studied, such as properly coloured walks in edge-coloured graphs and graphs with forbidden subpaths.

The first of the two main problems related to forbidden transitions that we study in this thesis is called traffic monitoring. Here, we are given a graph in which objects can walk and we know which walks these objects can use. We have the possibility to place censors on the arcs of the graph that indicate when an object goes through an equipped arc. Hence, for each object, we know the ordered sequence of equipped arcs they have used. Our objective is to find how to equip as few arcs as possible with censors in such a way that the information that the censors return is still sufficient to determine exactly which route the object has taken. For example, consider the graph depicted in Figure 1 and let the arcs tu, vw and wx be equipped with censors that we call a, b and c respectively. An object that uses the walk (t, u, v, w, x, z) would activate successively the censors a, b and c. An object that uses the walk (t, u, w, x, y, v, w, z) activates the same censors but activates the censor c before b. Thus, the set of censors $\{a, b, c\}$ allows to distinguish those two walks.

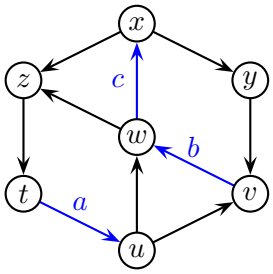


Figure 1: A graph with three monitored arcs depicted in blue.

The complexity of this problem had already been studied in [85], but prior to our work, the only algorithms that had been designed [88] focused on the special case of acyclic graph, in which walks are very limited and cannot use twice the same vertex or arc.

The traffic monitoring problem requires to find efficient ways to describe sets of walks in a graph, which we do by using tools stemming from language theory. We draw parallels between the problem of traffic monitoring and the well-known problem of separating code by developing a new model of separation based on languages, which allows us to take into account the number of times and the order in which the censors are activated. We then try to identify which kind of instances can be relevant for the practical applications of traffic monitoring and use their specificity to develop solutions as efficient as possible. This is where the model of forbidden transitions comes useful. We study several kinds of instances, some of which involve forbidden transitions and we develop algorithms to reformulate traffic monitoring as an integer linear program [9].

The other problem that we study in this part of the thesis is called minimum connecting transition set and can be seen as an adaptation with transitions of the well-known minimum spanning trees. Given a connected graph, a minimum spanning tree is a subset of edges of minimum size that keeps the graph connected *i.e.* such

that there exists a walk between every pair of vertices that only uses edges of the minimum spanning tree. Similarly, a minimum connecting transition set is a set of transition as small as possible such that there exists a walk between every pair of vertices that only uses transitions of the minimum connecting transition set. Note that according to the standard definition of transitions in undirected graphs, using twice the same edge in a row is not a transition and is always permitted. For example, consider the graph depicted in Figure 2 where the two permitted transitions are uvy and wvy. There are permitted walks leading from the vertex w to any other vertex of the graph: the walks (w, v) and (w, x) do not even require a transition, the walk (w, v, y) uses a permitted transition and while the walk (w, v, u) is forbidden, one can still go from w to u by using the walk (w, v, y, v, u). However, there is no way to go from x to u or y and the transition set $\{wvy, uvy\}$ is therefore not connecting. This can be fixed for example by adding the transition uvx.

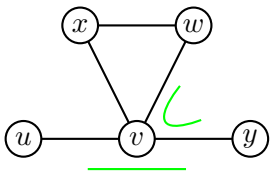


Figure 2: A graph with two permitted transitions depicted in green.

In many of the application fields where the model of transitions is relevant (such as road networks, transportation systems or optical telecommunication networks), it is possible that because of maintenance work or malfunction, some transitions become unusable. The problem of robustness to the removal of transitions has already been studied with several graph properties [105] and connectivity is an important requirement of the networks in every application fields of graphs with forbidden transitions. Here, what we study is not the smallest number of transitions that has to break down to disconnect the graph but the smallest number of transitions we have to secure for the graph to stay connected. While this does not provide an interesting measure of the robustness of the network, it highlights which parts of a network are the most useful for its proper functioning and can thus help designing robust network, like minimum spanning trees help in the cases where breakdowns impact edges and not transitions.

This thesis presents the results of joint work with Benjamin Bergougnoux and studies several aspects of the problem. We investigate structural aspects of minimum connecting sets, which leads to a reformulation of the problem as a problem of graph decomposition that we call *optimal connecting hypergraph*. This new problem turns out to be easier to manipulate and of great help in the subsequent proofs. We then study the problem under its algorithmic and complexity aspects and our main results are the establishment of its NP-completeness and the design of a polynomial $\frac{3}{2}$-approximation [10].

Geometric graphs and sets avoiding distance 1

The second part of this thesis presents the results of joint projects with Christine Bachoc, Philippe Moustrou, Arnaud Pêcher and Antoine Sedillot on the maximum density that can be achieved by a set of points A of the real space that does not contain two points at geometric distance exactly 1. Intuitively, the density denotes the portion of space that A fills. This maximum density also depends on the distance we use on $\mathbb{R}^n$. Throughout this thesis, we only consider distances induced by norms and we denote this number by $m_1(\mathbb{R}^n, \|\cdot\|)$ where $\|\cdot\|$ is the norm $\mathbb{R}^n$ is equipped with. The maximum density of sets avoiding distance 1 has been studied since at least the early 1960's [91] especially in the Euclidean plane, but turns out to be a very difficult problem and is still wide open.

Consider a packing of discs of radius 1 (*i.e.* a set of discs of radius 1 that do not overlap). An example of set that does not contain two points at distance 1 is the union of open discs of radius $\frac{1}{2}$ and of same center as the discs of our packing. The optimal density of a packing of discs in the Euclidean plane is about 0.9069 and we thus know that $m_1(\mathbb{R}^2) \geqslant \frac{0.9069}{4} \geqslant 0.2267$. A set achieving this density is illustrated in Figure 3. The best known lower bound is barely better; it was obtained by Croft in 1967 by refining the previous construction and is of about 0.2293. On the other hand, upper bounds have been improved more often through the years but are still far from the lower bounds. Prior to our works, the best upper bound was $m_1(\mathbb{R}^2) \leqslant 0.258795$ and was established by Keleti et al. in 2015 [70] through an approach based on harmonic analysis very different from ours. In this thesis, we develop a new approach and improve the bound to $m_1(\mathbb{R}^2) \leqslant 0.256828$.

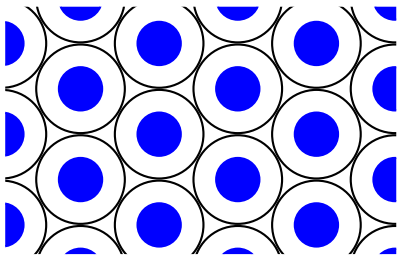


Figure 3: An example of set avoiding Euclidean distance 1 (in blue).

An important objective of the works that have been carried out on this topic is to prove a conjecture by Erdős that $m_1(\mathbb{R}^2) < \frac{1}{4}$. Note that if the norm is such that the set of points at distance 1 from the origin is a polytope that tiles the space perfectly (*i.e.* there exists a packing of unit polytopes of density 1), the construction described in the previous paragraph leads to a set avoiding distance 1 of density exactly $\frac{1}{4}$. Thus, the conjecture of Erdős says that m_1 is lower with the Euclidean norm because of the empty space between the unit discs of an optimal packing,

As a step toward Erdős' conjecture, Bachoc and Robins started studying the density of sets avoiding distance 1 for distance induced by norms whose unit polytopes perfectly tile the space (such norms are called *parallelohedron norms*). They conjectured that in this case, the construction described previously of sets avoiding distance 1 was optimal, and thus, that if $\|\cdot\|_{\mathcal{P}}$ is a parallelohedron norm,

$m_1(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}}) = \frac{1}{2^n}$. In this thesis, we prove this conjecture in dimension 2 as well as for a family of n-dimensional parallelohedron norms, and establish bounds on $m_1(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ for an other [5].

This problem is closely related to the famous Hadwiger-Nelson problem that aims at determining how many colours are needed to colour all the points of the Euclidean plane in such a way that no two points at distance exactly one receive the same colour. Figure 4 depicts a colouring of a portion of the plane. For example, every equilateral triangle of diameter 1 drawn in a properly-coloured plane has its three vertices of different colours.

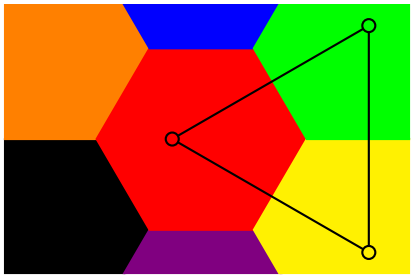


Figure 4: A colouring of a portion of the Euclidean plane.

Many variants of this problem have also been studied, including the colouring of other spaces or of space equipped with non-Euclidean distances, the measurable colouring (where we require the colour classes to be measurable), and the fractional colouring where we look for the smallest number $\frac{a}{b}$ such that we can give b different colours to all the points of a space chosen among a set of possible colours of size a, in such a way that no two points at distance exactly one share a common colour.

The problem of Hadwiger-Nelson has received a lot of attention since at least 1960 [52]. Lower and upper bounds of 4 and 7 were quickly established, but no one has been able to improve them until De Grey proved in April 2018 in [34] that at least 5 colours are needed to properly colour the Euclidean plane.

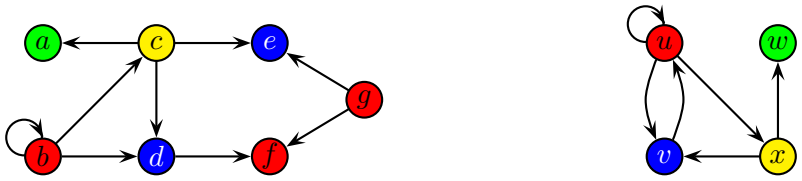
All these problems can be reformulated as problems in geometric graphs. The geometric graph whose vertices are all the points of the space and whose edges are the pairs of points at geometric distance 1 is called the unit-distance graph. The density of sets avoiding distance 1, the Hadwiger-Nelson problem and its variants all come down to determining the values of well-known graph invariants such as the independence ratio, the chromatic number or the fractional chromatic number of the unit-distance graph of the Euclidean plane.

Unit-distance graphs have uncountably many vertices and bring us out of the range of discrete mathematics in which graph theory is traditionally considered to belong. However, many information on the unit-distance graph can be inferred from its subgraphs. For example, the equilateral triangle depicted in Figure 4 provides a complete subgraph of size 3, for which three colours are required. This also proves that at least three colours are needed to colour the entire Euclidean plane. We will see that similar properties holds for the density of independent sets (sets of pairwise non-adjacent vertices, which are thus directly related to m_1).

We study algorithmically the notion of optimal weighted independence ratio of a graph and use it to study the aforementioned problems. This method is at the core of several still ongoing projects and has already provided interesting results. Those results include the improvement of the upper bound on $m_1(\mathbb{R}^2)$ but also the improvement of the best lower bound on the fractional chromatic number of the plane from 3.61904 [31] to 3.89366 and an upper bound on $m_1(\mathbb{R}^3, \|\cdot\|, \|\cdot\|_{\mathcal{P}})$ when $\mathcal{P}$ is a regular parallelohedron [4] [11].

Complexity of graph homomorphisms

The third part of this thesis studies the complexity of graph homomorphism problems, and especially of locally-injective directed graph homomorphisms. A graph homomorphism is a function from the vertex set of a graph to the vertex of a second graph such that the image of an edge is an edge, or in the directed case, such that the image of an arc is an arc. For example, in Figure 5, the function that maps b, f and g to u, d and e to v, a to w and c to x is a homomorphism from the graph depicted in Figure 5a to the graph depicted in Figure 5b.



(a) (b)

Figure 5: We attribute a colour to each vertex of the target graph (on the right). The colour of the vertices of the graph on the left indicates their image by the homomorphism.

Homomorphisms generalize graph colouring. In the undirected case, colouring is a specific case of graph homomorphism where every vertex of the target graph is connected to every other but not to itself. In the directed case, colourings are defined as homomorphisms to graphs where there is exactly one arc between each pair of distinct vertices. Such graphs are called *tournaments* and are very important in the study of directed homomorphisms.

Locally-injective homomorphisms are a specific case of homomorphisms where we add the constraint that the homomorphism must be injective on the neighbourhood of the vertices of the input graph. Locally-injective homomorphisms are interesting because they preserve the local structure of the input graph much better than standard homomorphisms. Hence, locally-injective homomorphisms find application in a wide range of areas including pattern detection in image processing, bio-informatics or coding theory. However, depending on what we want to model, our constraints are not exactly the same and we can thus find several definitions of local injectivity in the literature. For example, we can ask the homomorphism to be injective either on the open neighbourhood of the vertices or on their close neighbourhood (which includes the vertex itself). Hence, depending on the definition, the fact that in Figure 5, the vertex g has the same image as its neighbour f may or may not violate the

injectivity requirement. Similarly, we may ask the homomorphism to be injective on the in- and out-neighbourhoods of the vertices separately or on their union. Here again, depending on the definition, the fact that an in-neighbour (the vertex b) and an out-neighbour (f) of d have the same image may or may not be allowed. We choose depending on the applications which properties of the initial graph have to be preserved by the locally-injective homomorphism and we choose our definition of local injectivity accordingly. For example, if the homomorphism is injective on the union of the in- and out-neighbourhoods of vertices, we can ensure that the image of a path (a walk whose vertices are pairwise distinct) of length 2 is also a path of length 2. However, strengthening the model might also make the algorithmic problem of determining the existence of a locally-injective homomorphism between two graphs more difficult. This is what we study in this part of the thesis.

When studying the complexity of problems such as k-colouring or homomorphism to a given graph G, the objective is to establish what we call a dichotomy theorem: we partition the problems into two classes, prove that the problems of the first one are polynomial and that the problem of the other ones are NP-complete. For example, in the undirected case, it is well-known that k-colouring is polynomial for $k \leqslant 2$ and NP-complete for $k > 2$. Hell and Nešetřil have proved in [61] that undirected homomorphism to a graph G is polynomial if G has a loop or is bipartite and NP-complete otherwise. These results are dichotomy theorems.

Our starting point in the study of the complexity of locally-injective homomorphisms is the case where targets are tournaments. Indeed, the complexity of the variants of homomorphisms studied in the literature often depends on the chromatic number of the target. For example, in the Hell-Nešetřil theorem, bipartite graphs are exactly the 2-colourable graphs. Among the four non-equivalent definitions of local injectivity that we know of in the directed case, the complexity of three is open in the case of tournaments. Those three definitions are equivalent in the case of a loopless target and one of them does not take into account the loops on the target. Its complexity is therefore implied by the complexity of the other two and we are left with two definitions of local injectivity to study. Under those definitions, Campbell, Clarke and MacGillivray have already established in [20] that the problem is polynomial on the reflexive tournament on two vertices (reflexive tournaments are the ones that have loops on every vertex) and NP-complete on the two reflexive tournaments on three vertices. However, no dichotomy theorem on an infinite class of tournaments had emerged prior to our work.

Jointly with Stefan Bard, Christopher Duffy, Gary MacGillivray and Feiran Yang, we successfully established a dichotomy theorem on the complexity of locally-injective homomorphisms to reflexive tournaments for our two definitions of local injectivity: in both case, the problem is polynomial if the target has two vertices or fewer and NP-complete if the target has three vertices or more [7]. Our results also imply a dichotomy theorem on the complexity of locally-injective reflexive colouring.

Outline of the thesis

Chapter 1 introduces basic notions that we use throughout this thesis. Those notions come from various areas of mathematics and computer science, including

graph theory, complexity theory, geometry, language theory and mathematical programming.

Chapter 2 introduces the problem of traffic monitoring, presents the model and the tools that we have developed to address it and shows how to use them to solve the problem on several kind of instances of practical interest.

Chapter 3 introduces the problem of minimum connecting transition set, studies it on several classes of graphs, presents a reformulation, a polynomial approximation and a proof of NP-completeness in the general case.

Chapter 4 presents the problem of the density of sets avoiding distance 1 and its context, including the related problem of Hadwiger-Nelson. We present a method that requires to bound the independence ratio of infinite geometric graphs and we use it to prove the Bachoc-Robins conjecture on several families of parallelohedra including all those in dimension 2 and to establish upper bounds on $m_1(\mathbb{R}^n, \|\cdot\|)$ for some others.

Chapter 5 presents the notion of optimal weighted independence ratio, studies it on geometric graphs and uses it to extends the results of the previous chapter. This chapter contains improvement of the bounds on the fractional chromatic number of the Euclidean plane and on the density of sets avoiding distance 1 in the Euclidean plane and in the 3-dimensional space equipped with regular parallelohedron norm.

Chapter 6 presents the problem of locally-injective homomorphisms and the two definitions that we consider and proves for both of them a dichotomy theorem on reflexive tournaments.

Chapter 7 concludes this thesis by summing up the main contributions and presenting some open problems that our work raises and that may be at the core of future works.

Chapter 1

Preliminaries

This chapter introduces notions and results from different areas of computer science and mathematics that we need throughout this thesis. Its purpose is also to fix notations and definitions for which different variants can be found in the literature.

Section 1.1 presents generic notions of graph theory that are at the core of all the work we present in this thesis. Section 1.2 presents basic notions of complexity that we use throughout this thesis and introduces the notion of NP-completeness that is fundamental in Chapters 3 and 6. Section 1.3 extends Section 1.1 by presenting basic notions of graph theory but also presents the emerging field of forbidden-transition graphs. Those notions are crucial in Chapters 2 and 3. Section 1.4 introduces key notions of geometry and number theory and especially studies parallelohedra. The notions developed in this section play an important role in Chapters 4 and 5. Section 1.5 presents fundamentals of language theory and notably introduces tools that are extremely useful in Chapter 2 to describe sets of walks in a graph. Finally, Section 1.6 introduces a very powerful optimization technique called linear programming. Due to their efficiency and expressive power, linear programs are used in practice in a wide range of areas and are very important in Chapters 2 and 5 of this thesis.

Contents

Common mathematical notations

The cardinality of a set S is noted $|S|$ and its powerset is noted $\mathscr{P}(S)$. We use the notation $[a, b]$, $]a, b[$ and $[a, b[$ or $]a, b]$ for closed, open and semi-open intervals respectively. Integer intervals are noted $[\![a, b]\!]$.

We say that a set S that has a given property is maximal (respectively minimal) if none of its supersets (resp. subsets) possess this property. We say it is maximum

(resp. minimum) if no set of strictly bigger (resp. smaller) cardinality has this property.

When working in $\mathbb{R}^n$, we may denote by $\mathbf{0}$ the vector $(0, 0, \ldots, 0)$.

We denote by $\mathbb{P}(A)$ the probability of an event A and by $\mathbb{E}[X]$ the expected value of a random variable X.

The limit superior of a sequence U is noted $\limsup_{n\to\infty} U_n$ and is the supremum of the limits achieved by subsequence of U.

1.1 Fundamentals of graph theory

For a standard introduction to graph theory, we refer the reader to [36].

1.1.1 Core definitions

Undirected graphs. A *graph* G (sometimes referred to as a *simple graph* to avoid any ambiguity) is an ordered pair (V, E) where V is a non-empty finite set whose elements are called *vertices* and E is a set of unordered pairs of vertices whose elements are called *edges*. We can simply write uv to denote the edge $\{u, v\}$. The vertices u and v are the *endpoints* of the edge uv. The edge uv is *incident* to the vertices u and v. To avoid ambiguities, we may denote respectively by $V(G)$ and $E(G)$ the sets of vertices and edges of a graph G.

A vertex u is a *neighbour* of a vertex v if and only if $\{u, v\} \in E$. The *open neighbourhood* of a vertex v, noted $N(v)$, is the set $\{u : uv \in E\}$ and the *closed neighbourhood* of a vertex v, noted $N[v]$, is the set $N(v) \cup \{v\}$. If $\{u, v\} \in E$, u and v are *adjacent* and two edges are *adjacent* if they have a common endpoint.

The *degree* of a vertex v in a simple graph is noted $d(v)$ and is its number of neighbours. The *maximum degree* and *minimum degree* of a graph G, denoted respectively by $\Delta(G)$ and $\delta(G)$, are the maximum and minimum degree of a vertex of the graph.

A *multigraph* is a graph that may have loops and parallel edges. A *loop* is an edge whose two endpoints are the same vertex and *parallel edges*, also called *multiple edges*, are edges that have the same endpoints. Formally, a multigraph is also defined as an ordered pair (V, E) but here, edges are multisets of vertices of cardinality 2 and E is a multiset of edges.

Example 1.1.

Let us consider the graph G depicted in Figure 1.1. Here, $V(G) = \{u, v, w, x, y\}$ and $E(G) = \{uv, uw, vx, wx, wy\}$. The open neighbourhood of v is $N(v) = \{u, x\}$ and v therefore has degree 2. The maximum and minimum degrees of the graph are respectively 3 and 1 and are achieved by w and y respectively.

The work presented in this thesis also involves directed graphs. We now translate the basic notions of undirected graphs into the case of directed graphs.

Directed graphs. A *directed graph* G (or simple directed graph) is an ordered pair (V, A) where V is a non-empty finite set whose elements are called *vertices* and

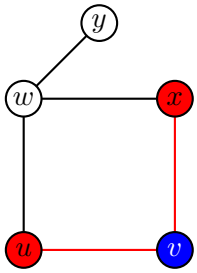


Figure 1.1: A simple graph G. The neighbours of v are depicted in red.

A is a set of ordered pairs of vertices whose elements are called *arcs*. As previously, we may simply write uv to denote the arc (u, v) and we denote respectively by $V(G)$ and $A(G)$ the sets of vertices and arcs of a graph G. The vertex u is the *origin* of the arc uv and v is its *target*. Two arcs a and b are *consecutive* if and only if the target of a is the origin of b. The arcs (u, v) and (v, u) are *opposite*.

A vertex u is an *in-neighbour* of a vertex v if and only if $(u, v) \in A$ and u is an *out-neighbour* of a vertex v if and only if $(v, u) \in A$. The vertex u is a *neighbour* of v if and only if u is an in- or an out-neighbour of v. The *open neighbourhood*, *open in-neighbourhood* and *open out-neighbourhood* of a vertex v are denoted respectively by $N(v)$, $N^-(v)$ and $N^+(v)$. The *closed neighbourhood* $N[v]$, the *closed in-neighbourhood* $N^-[v]$ and the *closed out-neighbourhood* $N^+[v]$ are defined as previously.

The number of in- and out-neighbours of a vertex v are noted $d^-(v)$ and $d^+(v)$ respectively and are called *in-degree* and *out-degree* of v.

An arc from a vertex to itself is called a *loop*. Observe that even simple directed graphs have loops since (u, u) is an ordered pair of vertices. The existence of an arc from a vertex u to a vertex v defines a relation on the vertex set of G. Thus, a graph is *irreflexive* if no vertex has a loop and *reflexive* if every vertex has a loop. A directed graph is *symmetric* if $\forall (u, v) \in A$, $(v, u) \in A$ too. We define *directed multigraphs* as an ordered pair (V, A) where A is a multi-set of arcs (and arcs are still ordered pair of vertices). This definition thus allows *parallels arcs*.

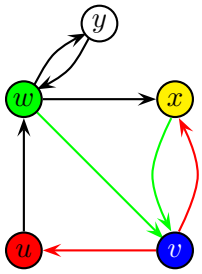
The following subclass of directed graphs plays an important role in the study of graph homomorphisms.

Definition 1.2. Oriented graphs:

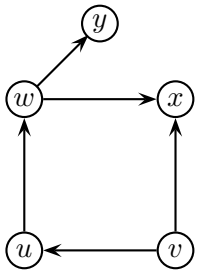
A directed graph is an *oriented graph* if and only if it has no pair of opposite arcs *i.e.* if there are no vertices $u \neq v$ such that uv and $vu \in A$. An oriented graph $\overrightarrow{G}$ is an *orientation* of an undirected graph G if and only if $V(\overrightarrow{G}) = V(G)$ and $E(G) = \{\{u, v\} : (u, v) \in A(\overrightarrow{G}) \text{ or } (v, u) \in A(\overrightarrow{G})\}$.

Example 1.3.

In the graph G_1 depicted in Figure 1.2a, w is an in-neighbour of v, u is an out-neighbour of v and x is both. There is a pair of opposite arcs between v and x and another between w and y. Hence, the graph G_1 is not oriented. If we remove an arc



(a) A directed graph G_1.



(b) An oriented graph G_2.

Figure 1.2

in each pair of opposite arcs, G_1 becomes oriented but is still not an orientation of the graph G depicted in Figure 1.1 because G has no edge between w and v.

The graph G_2 depicted in Figure 1.2b is an orientation of the graph G depicted in Figure 1.1.

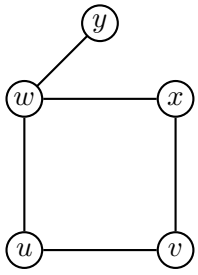
We now present the notions of subgraphs and complementary graphs that apply to both directed and undirected graphs.

Subgraphs and induced subgraphs. Let G be a graph. The graph H is a *subgraph of* G if and only if $V(H) \subset V(G)$ and $E(H) \subset E(G)$.

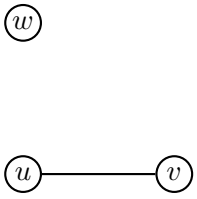
Let $S \subset V$ be a subset of vertex of G. The *subgraph of* G *induced by* S, noted $G[S]$, is the graph whose vertex set is S and whose edge set is $\{uv \in E(G) : (u, v) \in S^2\}$. The graph H is an *induced subgraph* of G if and only if $H = G[V(H)]$.

Let $v \in V$. We denote respectively by $G - S$ and $G - v$ the graphs $G[V \setminus S]$ and $G[V \setminus \{v\}]$.

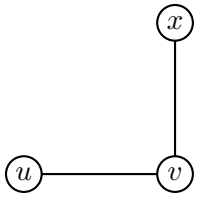
Example 1.4.



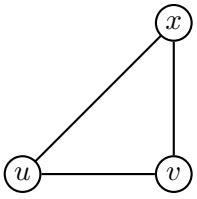
(a) A graph G.



(b) A graph H_1.



(c) A graph H_2.



(d) A graph H_3.

Figure 1.3

The graph H_1 depicted in Figure 1.3b is a subgraph of the graph G in Figure 1.3a but not an induced subgraph. Indeed, the vertex set $\{u, v, w\}$ induces the edge uw. The graph H_2 depicted in Figure 1.3c is the subgraph of G induced by $\{u, v, x\}$. The graph H_3 in Figure 1.3d contains the edge ux which is not in G ; H_3 is therefore not a subgraph of G.

Complement of a graph. Let G be a graph. The *complementary graph* of G, noted $\overline{G}$, is the graph defined by $V(\overline{G}) = V(G)$ and $E(\overline{G}) = \{uv : uv \notin E(G)\}$.

Example 1.5. Figure 1.4 depicts a graph and its complement.

(a) A graph G.

(b) Its complementary graph $\overline{G}$.

Figure 1.4

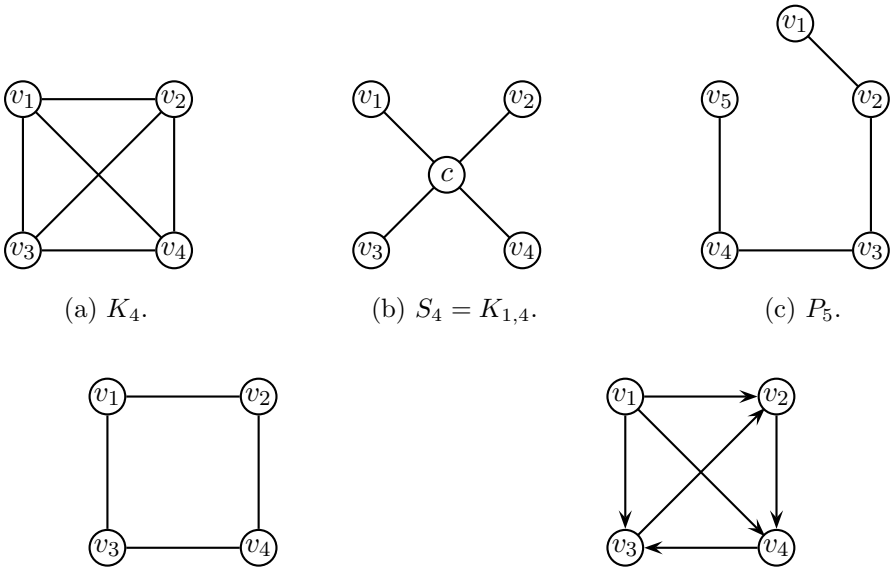
We now introduce some key families of graphs that are useful throughout this thesis.

Some basic graph families.

- A *path of n vertices*, noted P_n, is the graph defined by $V(P_n) = \{v_0, \dots, v_{n-1}\}$ and $E(P_n) = \{v_i v_{i+1} : 0 \leqslant i \leqslant n-2\}$.
- A *cycle of n vertices*, noted C_n, is the graph defined by $V(C_n) = \{v_0, \dots, v_{n-1}\}$ and $E(C_n) = \{v_i v_{(i+1) \mod n} : 0 \leqslant i \leqslant n-1\}$.
- A *complete graph of n vertices*, noted K_n, is the graph defined by $V(K_n) = \{v_0, \dots, v_{n-1}\}$ and $E(K_n) = \{v_i v_j : i \neq j\}$.
- A graph $G = (V, E)$ is *bipartite* if there exists a partition of V into two sets V_1 and V_2 such that every edge of G has one endpoint in V_1 and one in V_2. If in addition, $\forall u \in V_1, \forall v \in V_2, uv \in E$, the graph is *complete bipartite*. We denote by $K_{i,j}$ the complete bipartite graph with $|V_1| = i$ and $|V_2| = j$.
- A *star of $n+1$ vertices* or *star with n branches*, noted S_n, is the graph defined by $V(S_n) = \{c, v_1, v_2, \dots, v_n\}$ and $E(S_n) = \{cv_i : i \in [\![1, n]\!]\}$. The vertex c is the center of the star. Note that $S_n = K_{1,n}$.
- A *tournament on n vertices* is an orientation of the complete graph of n vertices K_n. If $n \geqslant 3$, there are several tournaments on n vertices. Note that tournaments are by definition irreflexive but we can also define *reflexive tournaments* as tournaments where we add a loop on every vertex.

Example 1.6. Figure 1.5 illustrates the aforementioned graph classes.

We conclude this subsection by introducing the notion of planarity.



(a) K_4. (b) $S_4 = K_{1,4}$. (c) P_5.

(d) $C_4 = K_{2,2}$. (e) A tournament on 4 vertices.

Figure 1.5

Definition 1.7. Planar graphs:

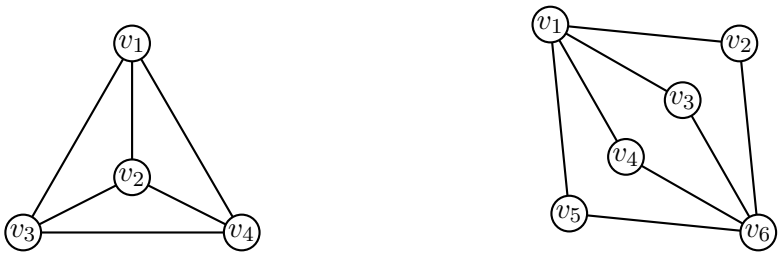
A *planar embedding* of a graph G is a drawing of the graph in the plane such that no two edges of G intersect each other. A *planar graph* is a graph that can be embedded in the plane. A *co-planar graph* is a graph whose complement is planar.

Example 1.8.

The drawing of K_4 depicted in Figure 1.5a is not a planar embedding since the edge v_1v_4 crosses the edge v_2v_3. However, the graph K_4 is still planar, as illustrated by Figure 1.6a.

Figure 1.6b depicts a planar embedding of $K_{2,4}$ and we can prove more generally that $K_{2,n}$ is planar for all n.

The graphs K_5 and $K_{3,3}$ are famous examples of non-planar graphs.



(a) A planar embedding of K_4. (b) A planar embedding of $K_{2,4}$.

Figure 1.6

1.1.2 Homomorphisms and colouring

This subsection presents the notion of graph homomorphism and its relation to graph colourings, which is central to Chapter 6.

Definition 1.9. Undirected graph homomorphisms:

A *homomorphism* from an undirected graph $G_1 = (V_1, E_1)$ to $G_2 = (V_2, E_2)$, also called *G_2-colouring of G_1*, is a function $f : V_1 \to V_2$ such that $\forall\{u, v\} \in E_1$, $\{f(u), f(v)\} \in E_2$. The graph G_2 is called the *target* of the homomorphism.

The famous problem of graph colouring is a specific case of homomorphism.

Definition 1.10. Undirected n-colouring:

A *n-colouring* of an undirected graph $G = (V, E)$, sometimes called *proper* n-colouring to avoid any ambiguity, is a function $c : V \to \{c_1, \ldots, c_n\}$ such that $\forall\{u, v\} \in E, c(u) \neq c(v)$. The elements $c_1, \ldots, c_n$ are called *colours*.

Finding a n-colouring of a graph G comes down to finding a homomorphism from G to the irreflexive complete graph on n vertices K_n.

If there exists a n-colouring of a graph G, G is *n-colourable*. The smallest number n such that G is n-colourable is called the *chromatic number* of G and is noted $\chi(G)$. If G is n-colourable but not $(n-1)$-colourable, G is *n-chromatic*.

The sets $C_i = \{v \in V : c(v) = c_i\}$ of vertices of the same colour are called *colour classes*.

Example 1.11.

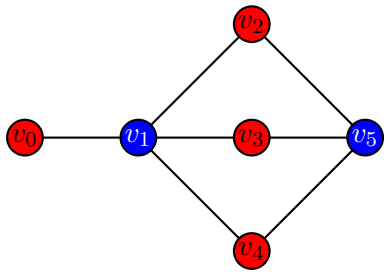


Figure 1.7: A 2-colouring of an undirected graph G.

Figure 1.7 presents a 2-colouring of a graph G. Since G is not 1-colourable, it is 2-chromatic.

While the definition of homomorphism is straightforward to generalize to the oriented case (the image of an arc must be an arc), the generalization of colouring is a little more subtle. Indeed, a n-colouring of an undirected graph G is a homomorphism from G to K_n and a n-colouring of an oriented graph G is a homomorphism from G to an orientation of K_n (*i.e.* a tournament on n vertices). However, while there is only one complete graph of n vertices, there are exponentially many tournaments. Colouring of an oriented graph consists of finding not only the colouring function but also the target tournament. For example, once we have coloured the two endpoints of an arc (u, v), we know that there must be an arc from $f(u)$ to $f(v)$

in the target tournament. Hence, there cannot be an arc from $f(v)$ to $f(u)$. We end up with the following equivalent definition:

Definition 1.12. Oriented graph homomorphisms and colouring:

An *homomorphism* from an oriented graph $G_1 = (V_1, A_1)$ to $G_2 = (V_2, A_2)$, also called *G_2-colouring of G_1*, is a function $f : V_1 \to V_2$ such that $\forall (u, v) \in A_1$, $(f(u), f(v)) \in A_2$.

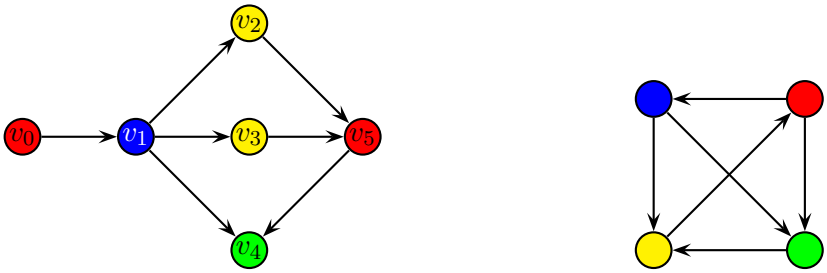
A *n-colouring* or *proper* n-colouring of an oriented graph $G = (V, E)$ is a function $c : V \to \{c_1, \ldots, c_n\}$ such that

$$\begin{cases} \forall (u, v) \in A, c(u) \neq c(v) \\ \forall (u, v) \in A, \forall (x, y) \in A, c(u) \neq c(y) \text{ or } c(v) \neq c(x) \end{cases}$$

The notion of *n-colourability*, *n-chromaticity*, *chromatic number* and *colour classes* are defined similarly as in the undirected case.

Oriented colourings have been introduced by Courcelle to study monadic second order logic in graphs [30]. Its purpose was to create a labelling of the vertices of the graph that would determine the orientation of the arcs.

Example 1.13.



(a) A 4-colouring of an oriented graph G. (b) The target tournament T.

Figure 1.8

Let us try to colour the graph G depicted in Figure 1.8a. We can pick arbitrarily the colour of v_1, say blue. The vertex v_0 can then be any colour but blue, let us say it is red. At this point, we know that the arc between the blue and red vertex in the target tournament goes from the red vertex to the blue. Thus, the vertices v_2, v_3 and v_4 cannot be red. We pick the colour yellow for v_2. The vertex v_5 cannot be yellow (it has a yellow neighbour) nor blue (it has a yellow in-neighbour which is not compatible with the orientation of the blue-yellow arc in the target tournament), but it can be red. The vertex v_3 has the same in- an out-neighbour as v_2 and can therefore have the same colour. However, the vertex v_4 can neither be red nor blue because it already has blue and red neighbours and it cannot be yellow because it has a red in-neighbour. Hence, we need a fourth colour for v_4. We can pick arbitrarily the orientation of the arc between the yellow and the green vertex in the target tournament (depicted in Figure 1.8b). We have exhibited a 4-colouring of G and proved that G cannot be coloured with three colours or less and is therefore 4-chromatic.

Note that a directed graph that has opposite arcs between the vertices u and v cannot be coloured because it would require the target tournament to have an arc from $f(u)$ to $f(v)$ and one from $f(v)$ to $f(u)$, which is impossible. Finally, note that G_2-colouring is trivial if G_2 has a loop on a vertex v. Indeed, the function f that maps every vertex to v is a homomorphism. This is why directed colouring is defined as a homomorphism from an oriented graph to an irreflexive tournament.

We now present a variant of colouring called edge-colouring. This variant is especially useful to study the complexity of homomorphisms problems.

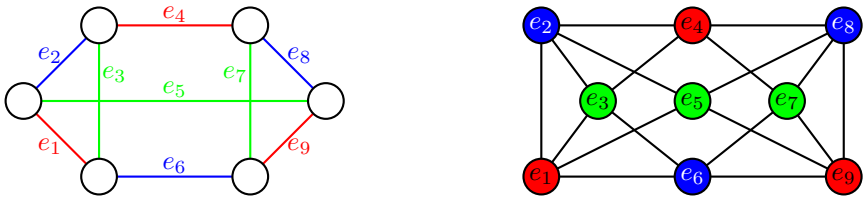
Definition 1.14. Edge-colouring:

A *k-edge-colouring* (sometimes called proper k-edge-colouring) of a graph $G = (V, E)$ is a function $c : E \to \{c_1, \ldots, c_n\}$ such that if two edges e_1 and e_2 are adjacent, then $c(e_1) \neq c(e_2)$. The *k-edge-colourability*, the *k-edge-chromaticity* and the *edge-chromatic number* are defined similarly.

Let G be a graph. The *line graph of G* is the graph H such that $V(H) = E(G)$ and two vertices of H are adjacent in H if and only if they denote adjacent edges in G. One can notice that edge-colouring G comes down to colouring its line graph.

Example 1.15.

Figure 1.9 illustrates the connection between colouring and edge-colouring.



(a) A graph G.

(b) The line graph of G.

Figure 1.9: An illustration of a proper 3-edge-colouring of a graph G and the associated 3-colouring of its line graph.

Finally, we present the notion of automorphism which greatly helps study the symmetry of a graph.

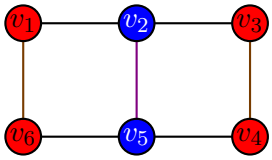
Definition 1.16. Isomorphisms, automorphisms and orbits:

An *isomorphism* from a graph $G_1 = (V_1, E_1)$ to $G_2 = (V_2, E_2)$ is a bijection $f : V_1 \to V_2$ such that $\forall (u, v) \in V_1^2, \{u, v\} \in E_1$ if and only if $\{f(u), f(v)\} \in E_2$. Note that the inverse function of an isomorphism from G_1 to G_2 is an isomorphism from G_2 to G_1. Two graphs are said to be *isomorphic* if and only if there exists an isomorphism from one to the other.

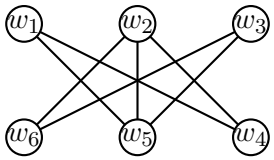
An *automorphism* is an isomorphism from a graph $G = (V, E)$ to itself. The automorphisms of a graph form a group under composition. Two vertices u and v are in the same *orbit* if and only if there exists an automorphism f of G such that $f(u) = v$. One can easily prove that this is an equivalence relation and since orbits are equivalence classes, they define a partition of the vertices of a graph. If all the vertices of the graph are in the same orbit, the graph is *vertex-transitive*.

We can define analogously orbit of edges by stating that two edges $\{u, v\}$ and $\{w, x\}$ are in the same orbit if there exists an automorphism such that $\{f(u), f(v)\} = \{w, x\}$. If all the edges of the graph are in the same orbit, the graph is *edge-transitive*.

Example 1.17.



(a) A graph G and a representation of its orbits.



(b) A graph H isomorphic to G.

Figure 1.10

The function f such that $f(v_1) = w_1$, $f(v_2) = w_5$, $f(v_3) = w_3$, $f(v_4) = w_6$, $f(v_5) = w_2$ and $f(v_6) = w_4$ is an isomorphism from the graph G depicted in Figure 1.10a to the graph H depicted in Figure 1.10b. The function $f_1 : V(G) \to V(G)$ that transposes v_1 with v_3 and v_4 with v_6 is an automorphism of G. So are the function f_2 that transposes v_1 with v_6, v_2 with v_5 and v_3 with v_4 and the function $f_3 = f_1 \circ f_2$. Those automorphisms prove that v_1, v_3, v_4 and v_6 are in the same orbit and that v_2 and v_5 are in the same orbit. Since $\deg(v_1) \neq \deg(v_2)$, no automorphism can map v_1 to v_2 and G has exactly two orbits. The orbit of edges are $\{v_1v_2, v_2v_3, v_4v_5, v_5v_6\}$, $\{v_1v_6, v_3v_4\}$ and $\{v_2v_5\}$.

The graph depicted in Figure 1.9 is vertex-transitive but not edge-transitive. Indeed, the edges e_4, e_5 and e_6 belong to no triangle while the six others do. The graph S_4 (see Figure 1.5b) is edge-transitive but not vertex-transitive. The graphs K_4 and C_4 (see Figures 1.5a and 1.5d) are both vertex- and edge-transitive.

1.1.3 Special vertex sets

This subsection presents the notions of independence, domination and separation in graphs.

Definition 1.18. Cliques, independent sets and dominating sets:

Let $G = (V, E)$ be a graph. A *clique* is a set of vertices $S \subset V$ such that $\forall (u, v) \in S^2, \{u, v\} \in E$. An *independent set* is a subset of vertices $S \subset V$ such that $\forall (u, v) \in S^2, \{u, v\} \notin E$. A *dominating set* is a set of vertices $S \subset V$ such that $\forall u \in V \setminus S, \exists v \in S, \{u, v\} \in E$.

The size of the biggest clique in G, noted $\omega(G)$, is called the *clique number* of G. The size of the biggest independent set in G, noted $\alpha(G)$, is called the *independence number* of G. The size of the smallest dominating set in G, noted $\gamma(G)$, is called the *domination number* of G.

A few fundamental properties follow directly from the definitions:

Proposition 1.19.

- *The empty vertex set is a clique and an independent set in every graph. More generally, any subset of a clique (resp. independent set) is a clique (resp. independent set) as well. The set $V(G)$ of vertices of a graph G is always a dominating set. Any superset of a dominating set is dominating too.*
- *Given a graph G and a proper colouring c on G, the colour classes are independent sets by definition. Hence, $\chi(G) \times \alpha(G) \geqslant V(G)$ since every vertex belongs to a colour class.*
- *In a colouring of a graph, all the vertices of a clique must have different colour. Hence, $\omega(G) \leqslant \chi(G)$. This bound is not always tight.*
- *A clique in a graph is an independent set in its complement. Hence, $\omega(G) = \alpha(\overline{G})$.*

Example 1.20.

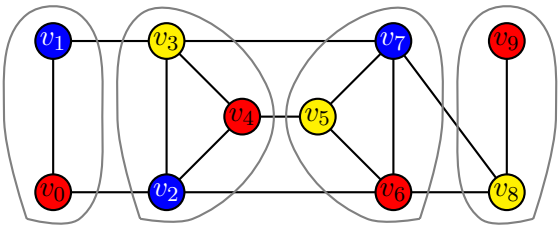


Figure 1.11: A graph G.

The set $\{v_2, v_3, v_4\}$ is a clique in the graph G depicted in Figure 1.11. Since G is 3-colourable (see the figure for an example of 3-colouring), we know that $\omega(G) = 3$.

The colour classes are independent set and notably $\{v_0, v_4, v_6, v_9\}$ is an independent set of size 4. The same argument applied to $\overline{G}$ indicates that since G can be partitioned into 4 cliques (circled in grey), $\alpha(G) = 4$ (an independent set has at most one vertex of each clique).

One can check that the set $\{v_0, v_4, v_8\}$ is a minimum dominating set on G and $\gamma(G) = 3$.

The cycle of 5 vertices C_5 cannot be coloured with less than 3 colours but contains no clique of size more than 2. This proves that the bound $\omega(G) \leqslant \chi(G)$ is not tight. Since $C_5 = \overline{C_5}$, we also observe that C_5 cannot be partitioned in less than 3 cliques but its independence number is 2.

Finally, we present the notions of separating and identifying codes, which are studied in Chapter 2.

Definition 1.21. Separating and identifying codes:

Let $G = (V, E)$ be a graph and $\mathcal{C}$ be a set of vertices of G. The *signature* of a vertex $v \in V$ is the set $\text{sign}(v) = N[x] \cap \mathcal{C}$ and $\mathcal{C}$ is said to be a *separating code* of G if and only if all the vertices of V have pairwise distinct signatures. A set $\mathcal{C}$ that is both separating and dominating is called an *identifying code*. In this case, the signature of all the vertices are pairwise distinct and non-empty.

Example 1.22.

(a) The code $\mathcal{C}_1$. (b) The code $\mathcal{C}_2$.

Figure 1.12: Two examples of codes in the graph C_6.

Consider the code $\mathcal{C}_1 = \{u, v, w\}$ depicted in red in Figure 1.12a. The signature of the vertices are respectively $\text{sign}(u) = \{u, v\}$, $\text{sign}(v) = \{u, v, w\}$, $\text{sign}(w) = \{v, w\}$, $\text{sign}(x) = \{u\}$, $\text{sign}(y) = \varnothing$ and $\text{sign}(z) = \{w\}$. Since they are pairwise distinct, $\mathcal{C}_1$ is a separating code but the vertex y is not dominated and $\mathcal{C}_1$ is not an identifying code. The code $\mathcal{C}_2 = \{u, w, y\}$ depicted in blue in Figure 1.12b is both separating and dominating and is therefore an identifying code.

Note that every superset of a separating set is separating too. Since this also holds for domination, it holds for identification. However, not every graph admits a separating code (look at the graph P_2 for example). Hence, a graph G admits a separating and an identifying code if and only if $V(G)$ is a separating code itself.

The conjunction of domination and separation is important in many practical applications. In fault detection and security, domination generally ensures that we know if a problem occurs and separation allows us to determine what the problem is. For example, imagine we model a building with a graph where the vertices denote the rooms and an edge between two vertices means that we can see a room from the other. If the set of rooms we equip with smoke detectors forms a dominating set, we are sure to know if a fire occurs. If it forms an identifying code, not only would we be able to detect a fire, but the set of activated detectors would indicate the exact room where the fire is starting. More concrete examples are presented in details in [108].

1.1.4 Hypergraphs

This subsection presents a generalization of graphs called hypergraphs where edges may connect more than two vertices.

Definition 1.23. Hypergraphs:

An *hypergraph* H is an ordered pair $(V, \mathscr{E})$ where V is a non-empty finite vertex set and $\mathscr{E} \subset \mathscr{P}(V)$ is a set whose elements are subsets of V and are called *hyperedges*. An hypergraph may simply be defined as a set $\{E_1, \ldots, E_n\}$ of hyperedges.

Separating codes are generalized to hypergraphs as follows:

Definition 1.24. Separating and identifying codes in hypergraphs:

Let $H = (V, \mathscr{E})$ be a hypergraph and let $\mathcal{C} \subset \mathscr{E}$ be a subset of hyperedges. The *signature* of a vertex v is the set of hyperedges it belongs to: $\text{sign}(v) = \{E \in$

$\mathscr{E} : v \in E\}$. The hyperedge set $\mathscr{C}$ is a *separating code* if all the vertices of V have pairwise distinct signatures and $\mathscr{C}$ is an *identifying code* if it is separating and $\forall v \in V, \exists E \in \mathcal{C}, v \in E$.

Example 1.25.

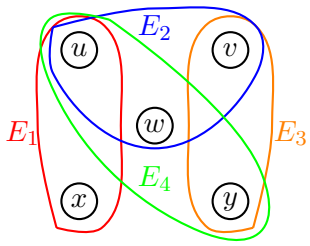


Figure 1.13: An hypergraph $H = (V, \{E_1, E_2, E_3, E_4\})$.

Let H be the hypergraph depicted in Figure 1.13. The hyperedge set $\{E_1, E_3, E_4\}$ is an identifying code in the hypergraph H. Indeed, the signatures of the vertices u, v, w, x and y are respectively $\{E_1, E_4\}$, $\{E_3\}$, $\{E_4\}$, $\{E_1\}$, $\{E_3, E_4\}$.

Note that if $\mathcal{C}$ is a code of size 2, there are only four possible signatures including the empty signature which is forbidden in an identifying code. Hence, a graph on five vertices (resp. four) or more cannot have a separating (resp. identifying) code of size 2. Thus, the code $\{E_1, E_3, E_4\}$ is a minimum separating and identifying code.

Equivalent formulations of this problem can also be found in the literature under the name of *test cover* or *discriminating code* (see [90] and [14] for important examples). The problem is generally defined by a set of individuals $\mathcal{I}$ and a set of attributes $\mathcal{A}$ where the attributes have no value and are simply properties that each individual may or may not have. Hence, the attributes can be defined by the set of individuals that possess them. The objective is to find a set of attributes $\mathcal{C}$ as small as possible such that each individual is characterized by the attributes of $\mathcal{C}$ it possesses. This is the same as finding a minimum separating code in the hypergraph whose vertex set is $\mathcal{I}$ and whose hyperedge set is $\mathcal{A}$.

The problem of separating and identifying codes in graphs were introduced in [69] and are a particular case of the same problems in hypergraphs. Indeed, solving the problem in a graph $G = (V, E)$ comes down to solving the problem in the hypergraph $H = (V, \mathscr{E})$ where $\mathscr{E} = \{N[v] : v \in V\}$ (where $N[v]$ denotes the closed neighbourhood of v in G). The problem on graphs is strictly less expressive since it only models problems in hypergraphs $(V, \mathscr{E})$ where $|\mathscr{E}| = |V|$. Separating codes have also been studied on directed irreflexive graphs in [101], [102] or [63]. The expressiveness of this model lies between the problem in graphs and the problem in hypergraphs.

This problem also appears in bipartite graphs in [22] and [23] for example. We are given a bipartition (V_1, V_2) of the vertices of a graph and a separating code is defined as a subset of V_2 such that all the vertices of V_1 have a different signature. This formulation is equivalent to the problem on a hypergraph where the vertex of the hypergraph are V_1 and the hyperedges are the $N[v]$ for $v \in V_2$. The expressiveness of different models of separation plays an important role in Chapter 2.

In many practical applications, some attributes are more expensive to test than others. Let a cost be associated to each attribute. Then, the weighted separation problem is the problem of finding a separating code of minimum cost. We return to these problems and present a method to solve them in Section 1.6.

1.2 Elements of complexity

For further information on the subject developed in this Section, we refer the reader to [94].

1.2.1 P, NP and polynomial reductions

Complexity theory aims at classifying computational problems according to their difficulty, which is measured by how much time is required to find a solution. This definition therefore depends on the model of computation we use. We first present two important complexity classes.

Definition 1.26. P, NP:

We denote by P the set of problems that can be solved on a deterministic Turing machine in polynomial-time in the size of the input. The set NP denotes the class of problems that can be solved in polynomial-time on a non-deterministic Turing machine.

However, defining formally the different variants of Turing machines is out of the scope of this thesis where we only consider algorithms on deterministic Turing machines. Alternatively, the class NP can be defined as the class of problems for which an answer can be verified in polynomial time on a deterministic Turing machine.

We now present the notion of reduction which allows to compare the complexity of two problems.

Definition 1.27. Polynomial reduction:

Given two problems $\mathscr{P}_1$ and $\mathscr{P}_2$, a *polynomial reduction* of $\mathscr{P}_1$ to $\mathscr{P}_2$ is a function that transforms an instance of $\mathscr{P}_1$ to an instance of $\mathscr{P}_2$ in polynomial time such that a solution of $\mathscr{P}_1$ can be deduced in polynomial time from the solution of $\mathscr{P}_2$. If there exists a polynomial reduction of $\mathscr{P}_1$ to $\mathscr{P}_2$, $\mathscr{P}_1$ is *reducible* (or *polynomially-reducible*) to $\mathscr{P}_2$. If $\mathscr{P}_2$ is also reducible to $\mathscr{P}_1$, $\mathscr{P}_1$ and $\mathscr{P}_2$ are *equivalent* (or *polynomially-equivalent*). If all the problems off a class C are polynomially reducible to a problem $\mathscr{P}$, $\mathscr{P}$ is *C-hard*, which intuitively means that solving $\mathscr{P}$ is at least as hard as solving any problem in C. If $\mathscr{P}$ is also in C, $\mathscr{P}$ is *C-complete*.

This notion is important because of the following consequence:

Proposition 1.28. *If $\mathscr{P}_1$ is reducible to $\mathscr{P}_2$ and $\mathscr{P}_2$ can be solved in polynomial time, then $\mathscr{P}_1$ can be solved in polynomial time too.*

Example 1.29.

Let k be an integer. Consider the two following problems:

$\mathscr{P}_1$: k-identifying code in graphs
Input: A graph G.
Output: Determines if there exists an identifying code in G of size k or less.

$\mathscr{P}_2$: k-identifying code in hypergraphs
Input: An hypergraph H.
Output: Determines if there exists an identifying code in H of size k or less.

Both problems are in NP. Indeed, given a code $\mathcal{C}$, one can compute and compare the signatures of the vertices of G or H and check whether $\mathcal{C}$ is dominating in polynomial time. Let now assume that we are given an instance $G = (V, E)$ of $\mathscr{P}_1$. We can build in polynomial time a hypergraph H whose vertex set is V and whose hyperedges are the $E_v = N[v]$ for $v \in V$. If there is no identifying code of size k or less in H, we know that there is no identifying code of size k or less in G either. If there exists an identifying code $\mathcal{C}_H$ of size k in H, we know that the code $\mathcal{C}_G = \{v : E_v \in \mathcal{C}_H\}$ is an identifying code in G of size $|\mathcal{C}_H| \leqslant k$. Hence, $\mathscr{P}_1$ is polynomially-reducible to $\mathscr{P}_2$ and if we find a polynomial-time algorithm for $\mathscr{P}_2$, we can deduce one for $\mathscr{P}_1$ easily.

The transformation of a graph into its line graph (see Definition 1.14) provides a polynomial reduction of edge-colouring to colouring (illustrated in Example 1.15).

Computing the clique number of a graph is polynomially equivalent to computing its independence number (see Definition 1.18) since one can compute the complement of a graph in polynomial time (see Proposition 1.19).

1.2.2 3-SAT and NP-completeness

Because of its expressiveness and complexity, the problem of boolean satisfiability is extremely important in complexity theory.

Definition 1.30. Basic definitions of boolean satisfiability:

A *boolean expression* is an expression built from *variables* that can be set either to True or False and from the operators $\wedge$ (the logical "*and*", also called *conjunction*), $\vee$ (the logical "*or*", also called *disjunction*) and $\neg$ (the logical "not", also called *negation*). A formula is said to be *satisfiable* if and only if there exists an assignment of each variable to True or False such that the formula is True.

We define the problem of *satisfiability*, noted *SAT* as follows:

SAT
Input: A boolean expression.
Output: True if the formula is satisfiable and False if it is not.

Occurrences of a variable x or of its negation $\neg x$ are called literals. A disjunction of literals is called a *clause*. A formula is written in *conjunctive normal form* if it is written as a conjunction of clauses. Every formula can be written in conjunctive normal form. For a given integer k, a formula is written in *k-conjunctive normal form* if it is written as a conjunction of clauses of size at most k. We define the problem of *k-satisfiability*, noted *k-SAT* as follows:

k-**SAT**
Input: A boolean expression given under k-conjunctive form. **Output:** True if the formula is satisfiable and False if it is not.

Example 1.31. Let a, b, c and d be variables and let us consider the formula $\mathscr{F} = a \vee (b \wedge \neg c) \vee \neg b \wedge (\neg a \vee b \vee c \vee \neg d)$. It is satisfied by the assignment $a =$ True, $b =$ True, $c =$ False and $d =$ True and is therefore satisfiable. It can be written in conjunctive normal form as follows:

$$\mathscr{F} = \underbrace{(a \vee b \vee \neg b)}_{=\text{True}} \wedge (a \vee \neg c \vee \neg b) \wedge (\neg a \vee b \vee c \vee \neg d) = (a \vee \neg c \vee \neg b) \wedge (\neg a \vee b \vee c \vee \neg d).$$

Thus, $\mathscr{F}$ is satisfiable if and only if the formula $(a \vee \neg c \vee \neg b) \wedge (\neg a \vee b \vee e) \wedge (c \vee \neg d \vee \neg e)$ is satisfiable. Indeed, by adding a new variable, we can replace a clause of size $2n$ by two clauses of size $n+1$ and by iterating this process, we find that every formula can be written under 3-conjunctive normal form. However, the polynomial equivalence between SAT and 3-SAT is not trivial. Note that our method to write a formula $\mathscr{F}$ under normal conjunctive form does not provide a formula whose size is polynomially bounded by the size of $\mathscr{F}$.

The problem of 3-SAT is useful in the seminal theorem of Cook and Levin.

Theorem 1.32. *Cook-Levin (1971) [29] [80]*
3-SAT is NP-complete.

In other words, every problem in NP is polynomially reducible to 3-SAT. Determining whether P=NP is unarguably one of the most important open problems in modern mathematics. The reason why NP-completeness is so important is because finding a polynomial-time algorithm for an NP-complete problem would answer the conjecture and prove that every problem in NP can be solved in polynomial time. Conversely, if P≠NP, proving that a problem is NP-complete proves that there exists no polynomial-time algorithm that can solve it. The Cook-Levin theorem considerably simplifies the proofs of NP-completeness since we now only have to reduce 3-SAT. This theorem was therefore the first of a long series of proof of NP-completeness. See [68] for one of the most important examples.

Theorem 1.33.
The following problems are NP-hard:

- *Determining the chromatic number of a graph G.*

  For a given integer $k \geqslant 3$, determining if a graph G is k-colourable (Karp, 1972 [68]).

- *Determining the chromatic number of a directed graph G.*

  For a given integer $k \geqslant 4$, determining if a directed graph G is k-colourable (Klostermeyer and MacGillivray, 2004 [74]).

- *Determining the edge-chromatic number of a graph G.*

  For a given integer $k \geqslant 3$, determining if a graph G is k-edge-colourable (Holyer, 1981 [62]).

- *Determining the clique number or the independence number of a graph G (Karp, 1972 [68]).*

- *Determining the domination number of a graph G (Garey and Johnson, 1979 [53]).*

- *Determining the size of the smallest separating / identifying code in a hypergraph (Gary and Johnson in updated editions of [53], 1990).*

- *Determining the size of the smallest separating / identifying code in a graph (Charon et al., 2003 [24]).*

1.2.3 Approximations

When no efficient algorithm is known for a problem that we have to solve on big instances, we often have to settle for *heuristic algorithms* that are faster but do not always provide optimal or exact answers. If the answer provided by our algorithm is within a constant multiplicative factor k of the optimal answer, our algorithm is a *k-approximation*. If such an algorithm exists and run in polynomial-time, the problem is said *k-approximable* (or *k-polynomially-approximable*).

For example, one can prove that finding maximal independent sets on a graph G of maximum degree 4 is NP-complete and we know no algorithm to solve it in polynomial time. However, it is proved in [59] that the greedy algorithm that consists of picking a vertex of minimum degree, adding it to the independent set and removing it and its neighbourhood from the graph until the graph is empty always returns an independent set at least half as big as the maximal independent set of the graph. Thus, finding a maximal independent set on a graph of maximum degree 4 is 2-approximable.

However, unless P=NP, there are problems that are not k-approximable for any k. One such problem is finding a maximal independent set in a general graph [8]. Hence, when faced with a problem we can prove NP-complete, a natural question is to determine whether the problem can be approximated in polynomial time or not.

1.3 Walks, connectivity and transitions

1.3.1 Walks and connectivity in usual graphs

This subsection presents basic notions of graph theory that are especially interesting in forbidden-transition graphs.

Paths, walks, distances. A *walk* W in an undirected graph or multigraph between a vertex v_0 and a vertex v_k is an alternating sequence of vertices and edges

$(v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$ where $e_i = \{v_{i-1}, v_i\}$. The vertices v_0 and v_k are the *extremities* of W. The integer k is called the *length* of the walk W.

A walk in a directed graph from a vertex v_0 to a vertex v_k is defined similarly as an alternating sequence of vertices and arcs $W = (v_0, a_1, v_1, a_2, v_2, \ldots, a_k, v_k)$ where $a_i = (v_{i-1}, v_i)$. Here, v_0 is the *starting point* of W and v_k is its *destination*. The *length* is defined similarly.

Note that a walk in a directed graph is entirely determined by the sequence of arcs it uses and that a walk in a simple graph (directed or not) is entirely determined by its sequence of vertices. Hence, in the appropriate structures, we can define a walk simply as a sequence of vertices or as a sequence of arcs.

A *path* or *elementary path* is a walk that does not use twice the same vertex or edge (except the extremities which may be the same). A walk whose extremities are the same is called a *cycle (walk)* and an *elementary cycle* is both a cycle and an elementary path.

The *distance* $\text{dist}(u, v)$ between a vertex u and a vertex v is the length of a shortest walk from u to v. If there is no walk between u and v, then $\text{dist}(u, v) = +\infty$.

The next proposition states an important property of elementary walk.

Proposition 1.34. *We can extract from every walk W on a graph G (directed or not) a path P that only uses vertices and edges that W uses and has the same starting point and destination.*

Proof. If a walk W uses twice a vertex v, we remove all the edges and vertices between the first and last occurrence of v in W. This process can be iterated until the walk is elementary. □

The next proposition follows directly from this proof.

Corollary 1.35. *The shortest walk between two vertices is always a path.*

Since the concatenation of a walk from u to v and a walk from v to w is a walk from u to w whose length is thus necessarily greater or equal than $\text{dist}(u, w)$, the distance in graphs satisfies the following property:

Proposition 1.36. *Triangle inequality:*
For all vertices u, v and w, $\text{dist}(u, v) + \text{dist}(v, w) \geqslant \text{dist}(u, w)$.

Example 1.37.

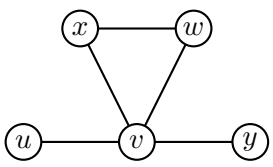


Figure 1.14: An example of graph G.

Consider the graph G depicted in Figure 1.14. The walk $W = (u, uv, v, vw, w, wx, x, xv, v, vy, y)$, that we can simply denote by the vertex sequence (u, v, w, x, v, y) is a

walk of length 5 between u and y. The walk W is not elementary since it uses twice the vertex v. We can however extract from W the path $P = (u, v, y)$ by removing everything between the first and last occurrence of the vertex v in W. Since P is the shortest path between u and y, $\text{dist}(u, y) = 2$.

There does not always exist a walk between two vertices but determining whether there exists a walk and therefore a path between two vertices in a graph $G = (V, E)$ can be done in time $O(|V| + |E|)$. This leads us to the notion of connectivity.

Definition 1.38. Connectivity, connected components:

Let $G = (V, E)$ be an undirected graph. Two vertices u and v are *connected* if and only if there exists a walk between u and v. A G is *connected* if every pair of vertices of G is connected. The *connected components* of G are the maximal vertex subsets of G that induce a connected graph. Since the connectivity between two vertices induces an equivalence relation, the connected components of G are its equivalence classes and are therefore pairwise disjoint.

Let $G = (V, A)$ be a directed graph. Two vertices u and v are *strongly connected* if and only if there exists a walk from u to v and a walk from v to u. Similarly, a graph G is *strongly connected* if every pair of vertices is strongly connected and the *strongly connected components* of G are the maximal vertex sets of G that induce a strongly connected graph. Since strong connectivity also induces an equivalence relation, the components are pairwise disjoint.

Example 1.39.

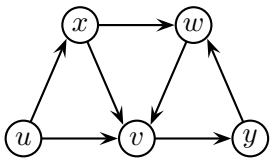


Figure 1.15: An example of graph G.

Let G be the directed graph depicted in Figure 1.15. While there is a walk leading from u to every vertex of the graph, there is no walk leading to u from any other vertex. Hence, the graph is not strongly connected and $\{u\}$ is a strongly-connected component of the graph. The two other strongly connected components of the graph are $\{x\}$ and $\{v, w, y\}$.

Connectivity and cycles allow us to introduce the important notion of tree.

Definition 1.40. Trees and spanning trees:

An undirected graph G is said *acyclic* if and only if there is no elementary cycle of non-zero length in G. A *tree* is a graph that is both connected and acyclic. Alternatively, a tree is a graph such that there exists exactly one walk between every pair of vertices. A tree of n vertices always has $n-1$ edges. A vertex of degree one in a tree is called a *leaf*.

A *spanning tree* T of a graph G is an acyclic and connected subgraph of G such that $V(T) = V(G)$. The set $E(T)$ is a minimum set of edges of G that keeps the graph connected. Every connected graph G admits a spanning tree.

Example 1.41.

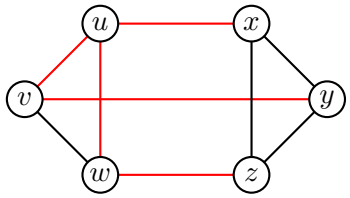


Figure 1.16: An illustration of a spanning tree on a graph.

The graph $G = (V, E)$ depicted in Figure 1.16 is not acyclic. For example, the walk (u, uv, v, vw, w, wu, u) is a cycle. The tree $T = (V, \{uv, uw, ux, vy, wz\})$ depicted in red is acyclic and connected and is therefore a spanning tree on G. The vertices x, y and z are leaves in T.

1.3.2 Forbidden-transition graphs

Walks in graphs are the model of choice to solve routing problems in all sorts of networks. So far, we have defined walks in undirected graphs and in directed graphs. However, in many practical applications, the set of possible routes an object can take is much more complicated than the set of possible walks in a graph. For example, it is completely possible in a road network that a driver coming from a given road may not turn left while the road on his left can be used by drivers coming from different roads. To model this kind of situations, we need to take into account not only the edges of the graphs but also the transitions.

Definition 1.42. Transitions and forbidden-transition graphs:

A *transition in an undirected graph* is an unordered pair of adjacent distinct edges. If a walk uses two edges uv and vw consecutively (with $u \neq w$), it uses the transition $\{uv, vw\}$. We can simply write uvw to denote the transition $\{uv, vw\}$. A *forbidden-transition graph* (or FTG) is a graph where certain pair of edges, although adjacent, cannot be taken consecutively. More formally, we denote by $T(G)$ and $F(G)$ respectively the set of permitted and forbidden transitions ($F(G)$ and $T(G)$ are complementary) of a FTG. Depending on the applications, we define a FTG either as a triplet (V, E, T) or (V, E, F). A walk $W = (v_0, e_0, v_1, \ldots, e_k, v_k)$ in a FTG (V, E, T) is compatible (or T-compatible) if and only if it only uses transitions of T *i.e.* for all $i \leqslant k - 2$, we have $v_i v_{i+1} v_{i+2} \in T$ or $v_i = v_{i+2}$ (*i.e.* $v_i v_{i+1}$ and $v_{i+1} v_{i+2}$ are the same edge). Observe that a walk of length one uses no transition and is therefore always T-compatible.

A *transition in a directed graph* is an ordered pair of adjacent arcs $((u, v), (v, w))$. The definitions of *directed forbidden-transition graphs* and *T-compatibility* are similar as above.

An undirected FTG is *connected* if there exists a walk between u and v for every pair of vertices u, v and that a directed FTG is *strongly-connected* if there exists a walk from u to v for every ordered pair of vertices (u, v).

In the parts of this thesis where we deal with forbidden-transition graphs, we may call the graphs with no forbidden-transitions *usual graphs* to avoid any ambiguity.

While the notions of distance and connected components seem easy to generalize to FTGs, their properties are drastically different. For example, the fundamental Propositions 1.34 and 1.36 do not hold anymore in FTGs.

Example 1.43.

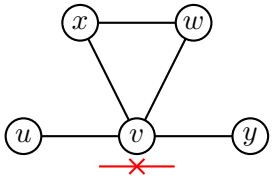


Figure 1.17: An example of undirected FTG. The red crossed line denotes a forbidden transition between the edges uv and vy.

In the FTG depicted in Figure 1.17, the walk $W = (u, v, w, x, v, y)$ is a T-compatible walk leading from u to y. We saw in Example 1.37 that we can extract from W the path $P = (u, v, y)$ which has same starting point and destination as W and only uses vertices and edges that W uses. However, P uses the transition uvy that W does not use and we notice that P is not T-compatible. While there is a T-compatible walk from u to y, there is no T-compatible path. Also note that the shortest compatible walk from u to y has length 4 (the walk (u, v, w, v, y) for example) while there are walks of length 1 from u to v and from v to y.

Since there is a walk between every pair of vertices, this graph is connected.

Example 1.44.

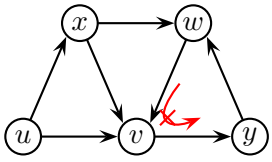


Figure 1.18: An example of directed FTG. The transition from the arc wv to vy is forbidden.

Consider the FTG depicted in Figure 1.18. Because of the forbidden transition, it is now impossible to go from w to y even though w and v are still strongly connected and so are v and y. The strong connectivity is no longer transitive and therefore no longer an equivalence relation. Hence, the maximal vertex sets that induce strongly connected components may now overlap. In this graph, they are $\{u\}$, $\{x\}$, $\{v, w\}$ and $\{w, y\}$.

Determining whether there exists a walk between two vertices u and v can still easily be done in polynomial time but the existence of a walk does not imply the existence of a path.

Theorem 1.45. *Szeider (2003) [106]*
Determining whether there exists a path between two vertices u and v of a forbidden-transition graph is NP-complete.

Graphs with forbidden transitions were introduced in 1968 by Kotzig [75] and have since been used in a wide range of applications. As illustrated by Theorem 1.45, many problems are harder in graphs with forbidden transitions. Finding elementary path between two vertices is a well-studied problem in the general case [66] and also on some subclasses of graphs such as grids [67]. Problems of graph-decomposition on FTGs have also received a lot of attention, see [39] and [50] for important examples.

Compatible walks in FTGs also generalize other well-studied models such as properly coloured paths. Given a graph with coloured edges (the colouring of the edges does not have to be proper), properly coloured paths are paths that do not use two edges of the same colour consecutively. This comes down to forbidding the transitions between edges of the same colour. This model was introduced by Daykin in 1976 [25] and has applications in various areas where the model of edge-colouring is relevant and in bio-informatics [37]. We refer the reader to [56] for a survey on this subject.

The model of forbidden-transition graph as well as a more general model where we can forbid any given set of subpaths (a transition is a path of length 2) are also used in optical telecommunication networks. Here, in order to travel through the fibers of a network, a ray of light needs specific properties that depend on the fiber. While every fiber can be used by certain rays, there are subpaths that no ray can follow. See [79] for more details on the optical constraints and [1] for an example of study of paths avoiding forbidden subpaths.

1.4 Polytopes and lattices

1.4.1 Norms and distances

The vector space used in this thesis is the space $\mathbb{R}^n$. All definitions are given in this specific case.

Definition 1.46. Norms, distances:
A *norm* is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}^+$ such that:
- $\forall x \in \mathbb{R}^n$, $\|x\| = 0$ if and only if $x = 0$;
- $\forall x \in \mathbb{R}^n$ and for all $a \in \mathbb{R}$, $\|ax\| = |a|\|x\|$;
- $\forall (x, y) \in (\mathbb{R}^n)^2$, $\|x + y\| \leqslant \|x\| + \|y\|$.

A *distance* is a function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ such that:
- $\forall (x, y) \in (\mathbb{R}^n)^2$, $\text{dist}(x, y) = 0$ if and only if $x = y$;
- $\forall (x, y) \in (\mathbb{R}^n)^2$, $\text{dist}(x, y) = \text{dist}(y, x)$;
- $\forall (x, y, z) \in (\mathbb{R}^n)^3$, $\text{dist}(x, y) + \text{dist}(y, z) \leqslant \text{dist}(x, z)$.

Given a norm $\|\cdot\|$, the *distance induced by the norm* is the function $d : (x, y) \mapsto \|x - y\|$.

Given a distance, we denote by $\mathcal{B}(x, r)$ the ball of center x and of radius r defined as $\{y \in \mathbb{R}^n : \text{dist}(x, y) \leqslant r\}$. The *unit ball* (or sometimes unit polytope when it

happens to be a polytope) is the ball of center 0 and of radius 1.

Example 1.47. Here are three important examples of norms:

- the *Euclidean norm*, noted $\|\cdot\|_2$, is defined by $\|(x_1,\dots,x_n)\|_2 = \sqrt{x_1^2+\cdots+x_n^2}$;
- the norm $\|\cdot\|_1$ is defined by $\|(x_1,\dots,x_n)\|_1 = |x_1|+\cdots+|x_n|$;
- the norm $\|\cdot\|_\infty$ is defined by $\|(x_1,\dots,x_n)\|_\infty = \max(x_1,\dots,x_n)$.

The unit balls associated to these norms are illustrated in Figure 1.19.

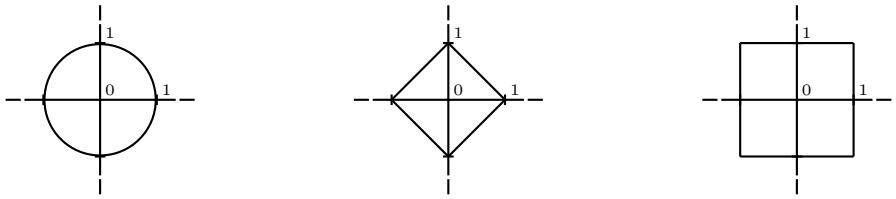


Figure 1.19: The unit balls associated respectively to the norms $\|\cdot\|_2$, $\|\cdot\|_1$ and $\|\cdot\|_\infty$.

Among the possible unit balls, polytopes are especially important in this thesis.

Definition 1.48. Polyhedra, polytopes:

A *n-dimensional polyhedron* is a set of points defined as an intersection of half-spaces (*i.e.* solutions of equations of the form $a_1x_1+a_2x_2+\cdots+a_nx_n \leqslant b$ for given a_i and $b \in \mathbb{R}$). A *polytope* is a bounded polyhedron.

We say that a polytope is *regular* if and only if all its edges have same length.

We would like to point out that several definitions of regularity can be found in the literature and that some of them are stronger than ours and require the regular polytopes to to be edge- and face-transitive. On the other hand, some definitions of regularity only requires the edges of the same orbit to have same length, which is weaker than our definition.

Note that the unit ball defined by any distance induced by a norm is convex, symmetric, centered at 0 and has a non-empty interior. The converse holds and allows to define polytope norms.

Definition 1.49. Polytope norms:

Let $\mathcal{P}$ be a convex, symmetric polytope centered at 0 and whose interior is non-empty. The *polytope norm* associated to $\mathcal{P}$, noted $\|\cdot\|_\mathcal{P}$, is the norm defined by the formula

$$\|x\|_\mathcal{P} = \inf\{\lambda \in \mathbb{R}^+ : x \in \lambda\mathcal{P}\}$$

The distance induced by $\|\cdot\|_\mathcal{P}$ is the *polytope distance* associated to $\mathcal{P}$: its unit ball is equal to $\mathcal{P}$.

Example 1.50.

Suppose we want to determine the distance between the points A and B on Figure 1.20 with the distance associated to the regular hexagon $\mathcal{P}$ (depicted in blue). From $\overrightarrow{AB} = \overrightarrow{OC} = 2\overrightarrow{OI}$ and $\mathrm{dist}(O,I) = 1$ (as I is on the border of the polytope), we find $\mathrm{dist}(A,B) = 2$.

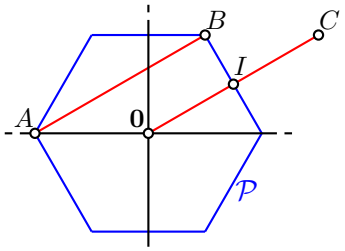


Figure 1.20: A polytope norm.

1.4.2 Lattices

Lattices are helpful in our study of polytopes. For further information, the reader may refer to [55].

Definition 1.51. Lattices:

A *lattice* of a n-dimensional vector space E is a subset $\Lambda \subset E$ such that there exists a basis $\mathcal{B} = (e_1, \dots, e_n)$ of E for which Λ is the set of points with integer coordinates: $\Lambda = e_1\mathbb{Z} \oplus \cdots \oplus e_n\mathbb{Z}$. In this case, $\mathcal{B}$ is a *basis* of Λ.

The *Voronoï cell* (or Voronoï region) of a lattice Λ, denoted by $\mathcal{V}_\Lambda$, is the set of points of the space that are closer to 0 than to any other point of the lattice:

$$\mathcal{V}_\Lambda = \{x \in \mathbb{R}^n : \forall y \in \Lambda, \|y - x\| \geqslant \|x\|\}.$$

A *coset* is the translate of a lattice by a vector of E. Let Λ be a lattice of E and let Λ' be a sublattice of Λ (*i.e.* a lattice included in Λ). By translating Λ' by vectors of $\Lambda \setminus \Lambda'$, we can create a partition of Λ into cosets. A coset of Λ is simply a coset included in Λ and is thus the translate of a sublattice of Λ.

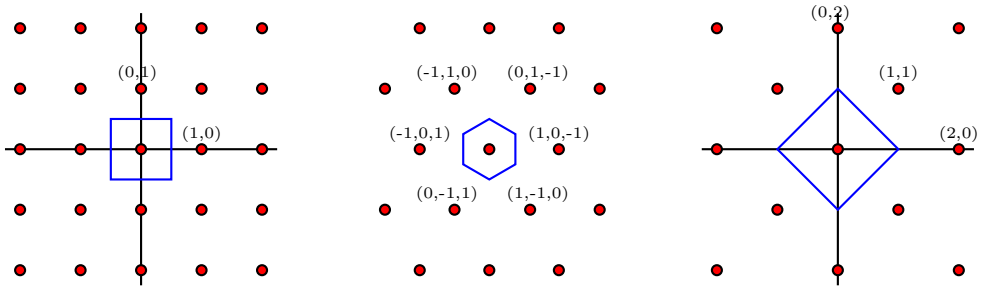
Example 1.52.

The following lattices are of particular interest in Chapter 4.

• The set $\mathbb{Z}^n$ is the *cubic lattice* and a basis is $\mathcal{B} = \{(1, 0, \dots, 0), (0, 1, \dots, 0), \dots, (0, 0, \dots, 1)\}$. Its Voronoï cell is the hypercube whose vertices are the points of the form $\left(\pm\frac{1}{2}, \pm\frac{1}{2}, \dots, \pm\frac{1}{2}\right)$. It is illustrated in Figure 1.21a. The set $(2\mathbb{Z})^2$ is a sublattice of $\mathbb{Z}^2$ and the cosets $(2\mathbb{Z})^2$, $(2\mathbb{Z}+1)^2$, $2\mathbb{Z} \times (2\mathbb{Z}+1)$ and $(2\mathbb{Z}+1) \times 2\mathbb{Z}$ partition $\mathbb{Z}^2$.

• The lattice A_n is defined by $A_n = \left\{(x_1, x_2, \dots, x_{n+1}) \in \mathbb{Z}^n : \sum_{i=0}^{n} x_i = 0\right\}$. Note that it is a n-dimensional lattice since all its vertices belong to the hyperplane of $\mathbb{R}^{n+1}$ defined by $x_1 + x_2 + \cdots + x_{n+1} = 0$. A basis of A_n in $\mathbb{R}^{n+1}$ is $\mathcal{B} = \{(1, -1, 0, \dots, 0), (1, 0, -1, \dots, 0), \dots, (1, 0, 0, \dots, -1)\}$. The Voronoï cell of A_2 is illustrated in Figure 1.21b and the Voronoï cells in higher dimension are more extensively studied in Subsection 4.4.1.

• The lattice D_n consists of the points whose coordinates are integers of even sum: $D_n = \left\{(x_1, \dots, x_n) \in \mathbb{Z}^n : \sum_{i=0}^{n} x_i \equiv 0 \mod 2\right\}$. A basis of D_n is $\{(1, 1, 0, \dots, 0), (1, -1, 0, \dots, 0), (1, 0, -1, \dots, 0), \dots, (1, 0, 0, \dots, -1)\}$. The Voronoï cell of D_2 is illustrated in Figure 1.21c and higher dimensions are studied in further details in Subsection 4.4.2.



(a) The lattice $\mathbb{Z}^2$. (b) The lattice A_2. (c) The lattice D_2.

Figure 1.21: Some lattices (points depicted in red) and their Voronoï cells (depicted in blue).

1.4.3 Polytopes

We say that a set $S \in \mathbb{R}^n$ is *discrete* if and only if every ball of finite radius in $\mathbb{R}^n$ only contains a finite number of elements of S.

Definition 1.53. Tilings and parallelohedra:

A polytope $\mathcal{P}$ *tiles* $\mathbb{R}^n$ *by translation* if there exists a discrete set of vectors S such that $\bigcup_{s \in S}(P+s) = \mathbb{R}^n$ and $\forall s, s' \in S^2$, $P+s$ and $P+s'$ have disjoint interiors. If in addition to the previous conditions, the intersection of $P+s$ and $P+s'$ for s and s' in S is always either empty or a common face of both of them, S is a *face-to-face tiling of* $\mathbb{R}^n$ *by translation*.

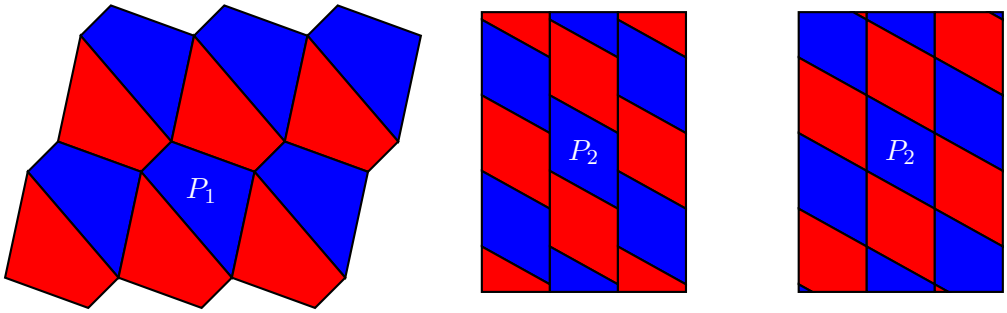
A *n-dimensional parallelohedron* is a convex polytope $\mathcal{P}$ that tiles $\mathbb{R}^n$ face-to-face by translation.

Note that a polytope of empty interior cannot tile the space because the set S that it would require is not discrete.

Several alternative definitions of parallelohedra exist in the literature. In this thesis, parallelohedra are by definition convex, which is not always the case. Furthermore, some authors only use the term "parallelohedra" in dimension 3 and talk about parallelotopes in general dimension and similarly use the terms polygons, polyhedra and polytopes in dimension 2, 3 and in the general case. However, some other authors use the term "parallelotopes" for a n-dimensional generalization of parallelepipeds. To avoid any confusion, we avoid the term "parallelotope" and use "parallelohedra" in the general case.

The only tilings we are interested in in this thesis are the tilings by translation. More general tilings where rotations are allowed have been widely studied too. Those tilings make the list of polytopes that tile $\mathbb{R}^n$ much larger. The polygons that tile the plane notably include all the triangles and quadrilaterals. The classification of polygons that tile the plane has recently been completed by Rao in [99] but is not our subject here.

Example 1.54.



(a) A tiling of $\mathbb{R}^2$ by a polygon P_1. (b) A tiling by translation of $\mathbb{R}^2$ by a polygon P_2. (c) A face-to-face tiling by translation of $\mathbb{R}^2$ by P_2.

Figure 1.22

The polygon P_1 depicted in Figure 1.22a tiles the plane but it requires rotation (the red polygons are not translated of the blue ones) and P_1 is not a parallelohedron. The tiling depicted in Figure 1.22b is not a face-to-face tiling but P_2 also tiles the plane face-to-face as depicted in Figure 1.22c and is therefore a parallelohedron.

We only study convex polytopes in this thesis because only them define a norm. Tilings by translation are especially important to us because if we apply a rotation to a unit polytope, it loses the property that it only contains vertices at distance smaller than one to a center. The classification of the convex polytopes that tile $\mathbb{R}^n$ and especially $\mathbb{R}^2$ and $\mathbb{R}^3$ by translation is fundamental for Chapters 4 and 5: the rest of this section is devoted to the main results addressing this classification.

Theorem 1.55. *Venkov (1954) [109]*
The convex polytopes that tile $\mathbb{R}^n$ by translation are exactly the parallelohedra, i.e. the convex polytopes that tile $\mathbb{R}^n$ by translation face-to-face.

The vector set S of a face-to-face tiling of $\mathbb{R}^n$ by a parallelohedron (see Definition 1.53) is a lattice. For further details on the characterization of parallelohedra, we refer the reader to the works of Minkowski [89] and McMullen [86].

Note that the Voronoï cell of any lattice $\Lambda \in \mathbb{R}^n$ tiles $\mathbb{R}^n$ face to face when translated by Λ. Voronoï's conjecture states that the converse also holds:

Conjecture 1.56. *Voronoï (1908) [110]*
If $\mathcal{P}$ is a parallelohedron in $\mathbb{R}^n$, then there is an affine map $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ such that $\varphi(\mathcal{P})$ is the Voronoï region of a lattice $\Lambda \subset \mathbb{R}^n$.

This conjecture has already been proved for several families of parallelohedra such as primitive parallelohedra (Voronoï, 1908 [110]) or zonotopal parallelohedra (Erdahl, 1999 [40]). Delaunay solved it for low dimensions:

Theorem 1.57. *Delaunay (1929) [35]*
Voronoï's conjecture holds in dimension up to 4.

Hence, classifying the parallelohedra in dimension 2 and 3 comes down to classifying the Voronoï's cells of 2- or 3-dimensional lattices.

1.4.4 Classification of the parallelohedra in dimension 2 and 3

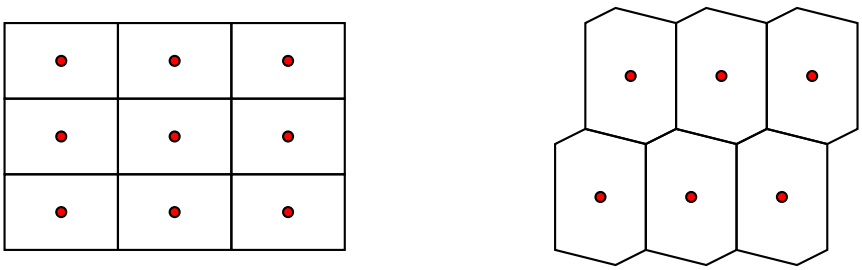
For more details on the classification we present in this subsection, we refer the reader to [27].

The only two possible Voronoï cells in dimension 2 are rectangles and a specific kind of hexagon called Voronoï hexagon (see Figure 1.23). The parallelohedra also contain the parallelograms since every parallelogram can be written as the image of a rectangle by an affine map.

Proposition 1.58. *There are two kinds of parallelohedra in dimension 2:*

- parallelograms*;*
- Voronoï hexagons*, which are hexagons whose opposite sides are parallels.*

The assumption that the vertices of our polytopes do not coincide is rarely useful in our proofs. Hence, we can see parallelograms as a specific degenerated case of Voronoï hexagons that have a pair of opposite sides of zero length. Voronoï hexagons can thus be seen as the most general 2-dimensional parallelohedra.



(a) Rectangular Voronoï cells. (b) Hexagonal Voronoï cells.

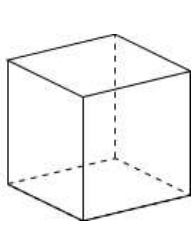
Figure 1.23: The two kinds of Voronoï cells in the plane.

The classification of parallelohedra in dimension 3 is slightly more complicated. The most general parallelohedron in dimension 3 is called the *truncated octahedron* (see Figure 1.24e). In the most regular case, it contains 36 edges of same length than can be grouped into six sets of six parallel edges. However, in the general case, we only need the edges within the same group to have the same length for the polytope to tile $\mathbb{R}^3$. Just like in the plane, one of those groups can consist of edges of length zero. In this case, we obtain an *elongated dodecahedron* (see Figure 1.24d). The elongated dodecahedron contains 28 edges that can be grouped in 4 groups of 6 and a group of 4 edges (the edges that separate the hexagonal faces) that must be parallel and have same length. If we shrink one of the groups of 6 edges, we obtain an *hexagonal prism* (see Figure 1.24c). If we shrink the edges of the group of size 4,

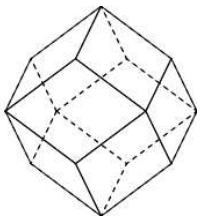
we obtain a *rhombic dodecahedron* (see Figure 1.24b). If we shrink both, we end up with a *cuboid* (see Figure 1.24a, it is the the Voronoï cell of $\mathbb{Z}^3$). The classification ends here as shrinking any other group of edges would lead to a polytope of empty interior.

Proposition 1.59. *There are five kinds of parallelohedra in dimension 3:*

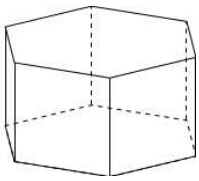
- *cuboids;*
- *rhombic dodecahedra;*
- *hexagonal prisms;*
- *elongated dodecahedra;*
- *truncated octahedra.*



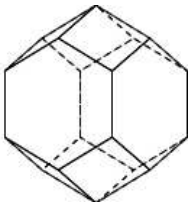
(a) A cube.



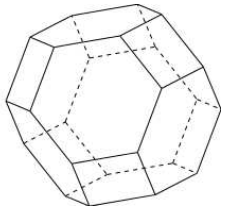
(b) A rhombic dodecahedron.



(c) An hexagonal prism.



(d) An elongated dodecahedron.



(e) A truncated octahedron.

Figure 1.24: The 5 kinds of 3-dimensional parallelohedra in their most regular form possible, where all the edges have same length.

The Voronoï cells of A_3 and D_3 are rhombic dodecahedra and of course, the Voronoï cell of Z_3 is a cube. Also note that the faces of 3-dimensional parallelohedra are always 2-dimensional parallelohedra.

The truncated octahedron and especially the case where all the edges have same length is at the core of Chapter 5. It is a specific case of permutohedron, which provides an interesting coordinate system to study it.

Definition 1.60. Permutohedron:

A *n-dimensional permutohedron* or *permutohedron of order $n+1$* is the convex hull of the set of points of $\mathbb{R}^{n+1}$ whose coordinates are a permutation of $\{1, 2, \ldots, n+$

1}. It is thus contained in the hyperplane of $\mathbb{R}^{n+1}$ defined by $\sum_{i=1}^{n+1} x_i = \frac{(n+1)(n+2)}{2}$ and is therefore a n-dimensional polytope. Two vertices are adjacent if and only if the coordinate of one can be obtained from the coordinate of the other by transposing two consecutive integers. Figure 1.25 illustrates the 2-dimensional permutohedron.

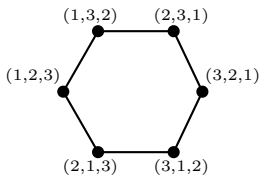


Figure 1.25: The permutohedron of order 3 is a regular hexagon.

A truncated octahedron whose edges all have same length is a permutohedron of order 4. We now describe it using the permutohedron coordinates.

A regular octahedron consists of 6 vertices of degree 4 and has 12 edges and 8 triangular faces. Since the vertices have degree 4, the truncation replaces them by a square. The truncated octahedron therefore has $6 \times 4 = 24$ vertices, 8 hexagonal faces and 12 edges between hexagonal faces (the faces and edges of the octahedron), 6 square faces and $6 \times 4 = 24$ edges that come from the truncation and belong to a square and an hexagonal face. This amounts to 36 edges and 14 faces.

If we see the truncated octahedron as a permutohedron in the hyperplane of $\mathbb{R}^4$ defined by $x_1 + x_2 + x_3 + x_4 = 10$, the coordinates of the 24 vertices are the permutation of $\{1, 2, 3, 4\}$. A vertex v has three neighbours whose coordinates can be obtained from the coordinates of v by transposing respectively 1 and 2, 2 and 3 and 3 and 4 (hence a total of 36 edges). By fixing one of the coordinate to 1 or 4, we define 8 planes that each contains a translation of a permutohedron of order 3 (which is a regular hexagon). Each vertex belongs to two of those 8 faces depending on which of their coordinates are equal to 1 and 4. From a given vertex, we can transpose 1 and 2 or 3 and 4 independently in any order, which defines a square. Each square is defined by which two of the four coordinates of its vertices are equal to 1 and 2, which amounts to 6 square faces and each vertex belongs to exactly one. Hence, the 12 edges that denote the transposition of 2 and 3 belong to two hexagonal faces and the 24 other edges belong to both a square and an hexagon. Figure 1.26 illustrates the neighbourhood of a vertex of the truncated octahedron.

Note that every permutohedron is vertex-transitive but a permutohedron of order $n \geqslant 4$ is not edge-transitive.

1.5 Rational languages and automata

This section only presents the few notions that we use in this thesis and does not intend to give a detailed overview of language theory. For further information on this subject, we refer the reader to [64].

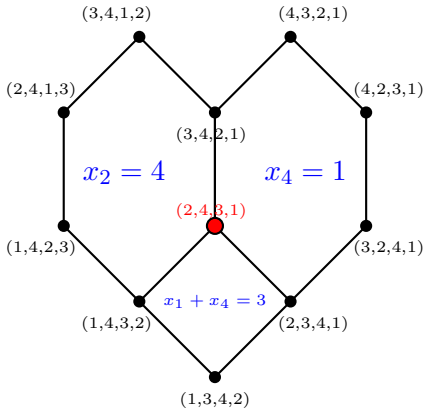


Figure 1.26: A representation of the vertex of coordinates (2,4,3,1) depicted in red and all the faces it belongs to. The equation of the faces are written in blue.

1.5.1 Rational languages

Definition 1.61. Alphabets, words, languages:

An *alphabet* is a non-empty finite set. The elements of an alphabet are called *letters*. A *word* of length k on the alphabet A is a sequence of k letters of A. The word of length 0, called the *empty word*, is denoted by ε. A *language* is a set of words on a given alphabet (unlike alphabets, language do not have to be finite).

Definition 1.62. Prefixes, suffixes, factors, subwords:

If a word u can be written as the concatenation vwx of three (possibly empty) words v, w and x, then v, w and x are respectively a *prefix*, a *factor* and a *suffix* of u. A word $w = w_0w_1 \dots w_n$ is a *subword* of a word v if and only if there exist $v_0, v_1, \dots, v_{n+1} \in A^*$ such that $u = v_0w_0v_1w_1 \dots v_nw_nv_{n+1}$.

Example 1.63.

The words ε and `exa` are prefixes of the word `example`. The word `xampl` is a factor of example. The word `xale` is a subword of `example`. All prefixes and suffixes of a word are factors and all factors are subwords.

A word of length n has $n+1$ distinct prefixes, $n+1$ distinct suffixes, at most $\frac{n(n+1)}{2} + 1$ distinct factors and at most 2^n distinct subwords.

The languages used in this thesis all belong to a specific subclass called rational languages that we introduce here.

Definition 1.64. Rational languages:

Rational languages on an alphabet A are defined inductively as follows:

- $\varnothing$ is rational;
- $\forall a \in A$, the language $\{a\}$ often simply denoted by a is rational;
- for all rational languages L_1 and L_2, their union denoted by $L_1 + L_2$ is rational;

- for all rational languages L_1 and L_2, their concatenation L_1L_2 is rational. The concatenation of L_1 and L_2 is the set of words than can be written uv with $u \in L_1$ and $v \in L_2$.
- For all rational languages L, $L^* = \sum_{k\in\mathbb{N}^*} L^k$ is rational. The operator * is called *Kleene star* or *Kleene closure* since it is the closure of L by the above operations.

We denote by $\mathrm{Rat}(A)$ the set of rational languages on an alphabet A. An expression that describes a language as a combination of letters or words by the above operations is called a *regular expression*.

Example 1.65.

The set of all words on an alphabet A is A^* and the set of all words of length k is A^k. These are regular expressions and those languages are therefore rational. A language that contains a single word is always regular too since it is the concatenation of languages of one letter.

The language of the words of even length on an alphabet A can be written $L_1 = (A^2)^*$ and is rational.

The language of the words on the alphabet $\{a, b, c\}$ that contain the letter a at most twice is rational too and can be written $L_2 = (b+c)^*(a+\varepsilon)(b+c)^*(a+\varepsilon)(b+c)^*$.

The language of the words on the alphabet $\{a, b\}$ that contain exactly one occurrence of one of their letter is $L_3 = a^*ba^* + b^*ab^*$ and is rational.

However, the language of words on $\{a, b\}$ that contains exactly as many occurrences of each letter or even the language $\sum_{n\in\mathbb{N}} a^nb^n$ are not rational (the only possible infinite sums in a regular expression are the Kleene stars).

1.5.2 Automata and recognition

This subsection presents the notion of automata, which creates interesting bridges between language and graph theories.

Definition 1.66. Automata:

An *automaton* is a 5-tuple (A, Q, T, I, F) where:

- A is the alphabet the automaton is defined on.
- Q is the set of states of the automaton.
- $T \subset Q \times A \times Q$ is the set of transitions of the automaton. A transition t of an automaton is a triplet (q_1, a, q_2) where q_1 is called the *origin* of t, a is called its *label* and q_2 its *target*. Transitions in automata can be seen as labelled arcs between states and should not be confused with transitions in graphs (Definition 1.42).
- $I \subset Q$ is the set of initial states of the automaton.
- $F \subset Q$ is the set of final states of the automaton.

The *transition function* of an automaton determines in which states the automaton ends up if it is in a given set of states $S \in \mathscr{P}(Q)$ and reads a given word $u \in A^*$. Intuitively, the automaton is at each step of the run in a subset of its states. Reading the empty word does not change the states the automaton is in. After reading a letter a, the automaton is in all the states that are the targets of a transition labelled by a whose origin is a state the automaton was in before reading a. We iterate this process for each letter of the word that the automaton reads. The automaton starts in the initial states I and the question is whether the states of the automaton after reading a given input word contains one of the final states $f \in F$.

Definition 1.67. Recognizable languages:
Let (A, Q, T, I, F) be an automaton. Its *transition function* $\delta : \mathcal{P}(Q)A^* \mapsto \mathcal{P}(Q)$ is defined inductively as follows:

- $\forall S \in \mathscr{P}(Q), \delta(S, \varepsilon) = S$
- $\forall S \in \mathscr{P}(Q), \forall a \in A, \delta(S, a) = \{q \in Q | \exists s \in S, \exists t \in T, t = (s, a, q)\}$
- $\forall S \in \mathscr{P}(Q), \forall u \in A^* \setminus \{\varepsilon\}, \delta(S, u) = \delta(\delta(S, a), u')$ where $u = au'$ with $a \in A, u' \in A^*$

A word u is *accepted* by an automaton (A, Q, T, I, F) if and only if $\delta(I, u) \cap F \neq \varnothing$. The *language recognized by an automaton* is the set of words that this automaton accepts. A language is *recognizable* if and only if there exists an automaton that recognizes exactly this language. The set of recognizable languages on an alphabet A is noted $\mathrm{Rec}(A)$.

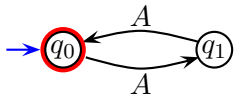
An automaton (A, Q, T, I, F) can be depicted by a directed multigraph whose vertex set is Q and that contains an arc (u, v) labelled by a for each transition (u, a, v). Two transitions (u, a, v) and (u, b, v) with same origin and target may be depicted by a single arc (u, v) labelled by both a and b. We depict initial states by incoming blue arrows and final states are circled in red.

Example 1.68.
The automata depicted in Figure 1.27 recognize the languages $L_1 = (A^2)^*$, $L_2 = (b + c)^*(a + \varepsilon)(b + c)^*(a + \varepsilon)(b + c)^*$ and $L_3 = a^*ba^* + b^*ab^*$ described in Example 1.65.

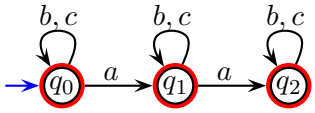
If the automaton depicted in Figure 1.27b reads the word *bacb*, its states are successively $\{q_0\}$ (before reading anything), $\{q_0\}$ (after reading b), $\{q_1\}$ (after reading ba), $\{q_1\}$ (after reading *bac*) and $\{q_1\}$ (after reading *bacb*). The automaton ends in the state q_1 which is final and thus, $bacb \in L_2$. If the same automaton reads the word *abaac*, its successive states before reading the word and after each letter are respectively $\{q_0\}$, $\{q_1\}$, $\{q_1\}$, $\{q_2\}$, $\varnothing$ and $\varnothing$. The set of states the automaton ends in does not contain a final state. Hence, $abaac \notin L_2$.

If the automaton depicted in Figure 1.27c reads the word *aba*, its successive states are $\{q_0, q_2\}$, $\{q_0, q_3\}$, $\{q_1, q_3\}$ and $\{q_1\}$. Since q_1 is a final state, $aba \in L_3$.

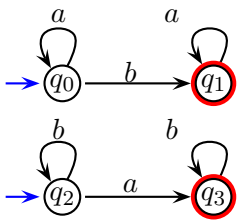
We are now ready to introduce a fundamental characterization of recognizable languages:



(a) An automaton that recognizes L_1.



(b) An automaton that recognizes L_2.



(c) An automaton that recognizes L_3.

Figure 1.27

Theorem 1.69. *Kleene (1956) [73]*
For all alphabets A, $\mathrm{Rat}(A) = \mathrm{Rec}(A)$.

In other words, a language L is rational if and only if there exists an automaton that recognizes exactly the words of L. The main consequence is that every recognizable language can be described by a regular expression. The rest of this section presents how one can deduce a regular expression of a language from an automaton that recognizes it.

Lemma 1.70. *Arden (1961) [2]*
Let K *and* L *be two languages such that* $\varepsilon \notin K$. *The only solution of the equation* $X = KX + L$ *is* $X = K^*L$.

Suppose we want to find a regular expression of the language L recognized by the automaton (A, Q, T, I, F). For each state $q \in Q$, we denote by L_q the language recognized by the automaton $(A, Q, T, \{q\}, F)$. We end up with the following system:

$$\begin{cases} L_q = \begin{cases} \displaystyle\sum_{(q,a,r)\in T} aL_r + \varepsilon \text{ if } q \in F \\ \displaystyle\sum_{(q,a,r)\in T} aL_r \text{ else} \end{cases} \\ L = \displaystyle\sum_{q\in I} L_q \end{cases}$$

This system can be solved by Gaussian elimination with Arden's lemma. At each step of the resolution, the languages are denoted by regular expressions.

Example 1.71.
Let us determine a regular expression of the language L recognized by the automaton depicted in Figure 1.28

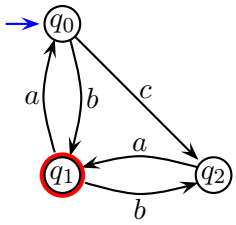


Figure 1.28: An example of automaton.

Our system is $\begin{cases} L_0 = bL_1 + cL_2 \\ L_1 = aL_0 + bL_2 + \varepsilon \\ L_2 = aL_1 \\ L = L_0 \end{cases}$, by elimination of L_2 this leads to

$\begin{cases} L_0 = (b+ca)L_1 \\ L_1 = aL_0 + baL_1 + \varepsilon = (ba)^*(aL_0 + \varepsilon) \\ L = L_0 \end{cases}$ and by elimination of L_1 we find:

$L = L_0 = (b+ca)(ba)^*aL_0 + (b+ca)(ba)^* = ((b+ca)(ba)^*a)^*(b+ca)(ba)^*$.

1.6 Linear programming

For further information on the subject developed in this section, we refer the reader to [12].

1.6.1 Definitions

We begin by defining the family of linear programs:

Definition 1.72. Linear Programs:
A *linear program* is a problem defined by

- A set of real *variables* $\{x_1, x_2, \ldots, x_n\}$ where n is the *number of variables* of the problem.
- An *objective function* described by a real vector $C = (c_1, c_2, \ldots, c_n)$ of size n too.
- A set of *constraints* defined by a real matrix $(A_{i,j})$ with $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$ and a real vector $B = (b_1, b_2, \ldots, b_m)$ where m is the *number of constraints* of the problem.

Solving the problem consists of assigning real values to the variables $x_1, \ldots, x_n$ such that $\forall j \leqslant m$, $\sum_{i=1}^{n} A_{i,j}x_i \leqslant b_j$ and that maximize $\sum_{i=1}^{n} c_i x_i$.

An assignment of the variables is called a *solution*. A solution that satisfies all the constraints is called a *feasible solution*. A feasible solution that maximizes the objective function is called an *optimal solution*. If there is no feasible solution, the problem in *unfeasible* and if there are feasible solutions but no optimal solution, the problem is *unbounded*.

Such problems are called linear because neither the objective function nor the constraints involve the product of two variables.

Minimizing the objective function described by a vector C comes down to maximizing the objective function described by $-C$. Similarly, satisfying the constraint $\forall j \leqslant m, \sum_{i=1}^{n} A_{i,j}x_i \geqslant b_j$ comes down to satisfying $\forall j \leqslant m, \sum_{i=1}^{n} -A_{i,j}x_i \leqslant -b_j$ and a constraint of the form $\forall j \leqslant m, \sum_{i=1}^{n} A_{i,j}x_i = b_j$ can be written as the conjunction of two inequalities. We can use whichever of these variants is the most convenient for the problem we study.

A solution defines a point of $\mathbb{R}^n$, a constraint defines a *hyperplane* and a solution satisfies a constraint or not depending on which side of the hyperplane it is on. Hence, the set of feasible solutions is defined by the intersection of a finite number of half-spaces and is therefore a convex polytope. One can prove that if optimal solutions exist, at least one of them is a vertex of the polytope and that a vertex of the polytope that reaches a strictly higher objective value than all its neighbour (also called a *local optimal*) is a global optimal *i.e.* an optimal solution. This is the key observation that led to Dantzig's seminal Simplex Algorithm [33] that solves linear programs very efficiently in practice.

Example 1.73.

The problem P_1 that consists of maximizing $x_1 - x_2$ under the constraints $x_1 \leqslant x_2 - 1$ and $x_2 \leqslant x_1 - 1$ is infeasible.

The problem P_2 that consists of maximizing $2x_1 - x_2$ under the constraints $x_1 - x_2 \leqslant 0$ is feasible. For example, $(3, 4)$ is a feasible solution. However, (k, k) is a feasible solution for all $k \in \mathbb{R}$ and reaches the objective value of k. Thus, this problem is unbounded.

The problem P_3 that consists of maximizing $x_1 - x_2$ under the constraints $-1 \leqslant x_1 \leqslant 5$, $x_2 \leqslant 2$ and $x_1 - 2x_2 \leqslant 0$ is bounded and feasible. An optimal solution is $(4, 2)$ and its value is 2. The problem is represented on Figure 1.29 where the feasible solutions are depicted in yellow, the objective function in blue, the constraints in red and the optimal solution in green.

The worst case for the simplex algorithm is when adjacent vertices of the polytope of feasible solutions achieve the same objective value, as illustrated by the Klee-Minty cube in [72]. Many variants exist to better deal with some pathological cases but all the improved versions also have exponential worst-case complexity. The complexity of linear programming remains open and improving the resolution of linear programs is a very active field. However, the simplex algorithm yields very good results on practical instances. For a probabilistic analysis of the simplex algorithm on random instances that explains its very good results in practice, we refer the reader to [15].

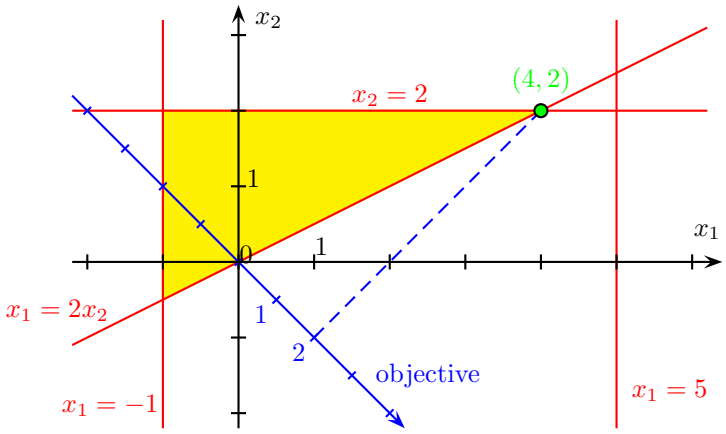


Figure 1.29: A graphical representation of the problem P_3.

1.6.2 Integer linear programming

In many decision or optimization problems including all the problems listed in Theorem 1.33, the set of possible solutions is discrete. Hence, those problems cannot be modelled by a linear program. In this subsection, we introduce a stronger model more suitable for this kind of problems.

Definition 1.74. Integer Linear Programs:
An *Integer Linear Program* (or ILP) is a linear program whose variables can only take integer values. A linear program where some of the variables have to take integer values and some others do not is a *Mixed Integer Linear Program* (or MILP).

Since the vertices of the polytope of feasible solutions do not necessarily have integer coordinates, the simplex algorithm does not work anymore. Given an integer linear program, it is possible to add constraints that do not discard integer solutions and ensure that the vertices of the polytope have integer coordinates but the number of constraints to add is not necessarily polynomially bounded in the number of constraints of the ILP. As a result, ILPs are unfortunately much harder to solve than regular linear programs.

Theorem 1.75. *Karp (1972) [68]*
ILP is NP-complete.

Since MILP is a generalization of ILP, it is NP-complete too. However, reduction to ILP are often natural and there exist several efficient solvers. Hence, when confronted to optimization problems on graphs, it is often helpful to reduce them to linear programming or ILP. The rest of this subsection presents two examples of reduction of important graph problems to ILP that are useful later in this thesis (in Chapters 5 and 2 respectively).

Example 1.76. Maximum Independent Set (see Definition 1.18)
Consider the following key problem:

Maximum Independent Set.
Input: A graph G.
Output: An independent set of maximum cardinality.

Let $V = (v_1, v_2, \ldots, v_n)$ be the set of vertices of G. We define a binary variable for each $v_i \in V$. This variable is equal either to 0 or 1 depending on whether v_i belongs to our maximum independent set.

The objective is to maximize $\sum_{i=1}^{n} x_i$ (the independent set has to be as big as possible).

The constraints are $\forall i, 0 \leqslant x_i \leqslant 1$ (binary variables) and $\forall \{v_i, v_j\} \in E(G), x_i + x_j \leqslant 1$ (two adjacent vertices cannot both belong to an independent set). The value of the objective function indicates the maximum size of an independent set and the variables set to 1 in an optimal solution give an example of a maximum independent set.

Before moving to the second example of reduction to ILP, we need to define separating sets, which are of great help in Chapter 2.

Definition 1.77. Separating set of two individuals:

Let $\mathcal{I}$ be a set of individuals and $\mathcal{A}$ be a set of attributes. Let $(i, i') \in \mathcal{I}^2$ be such that $i \neq i'$. The separating set of i and i', denoted by $\mathrm{Sep}(i, i')$, is the symmetric difference of their attributes *i.e.* the set of attributes that exactly one of them possesses.

Example 1.78. Minimum Test Cover / Minimum Separating Code in a hypergraph (see Definition 1.24):

Minimum Test Cover.

Input: A set of individuals $\mathcal{I}$ and a set of attributes $\mathcal{A} \subset \mathscr{P}(I)$
Output: A minimum set of attributes $\mathcal{C}$ such that all the individuals of $\mathcal{I}$ are characterized uniquely by the attributes of $\mathcal{C}$ they possess.

For each attribute $a \in \mathcal{A}$, let x_a be a binary variable that indicates whether $a \in \mathcal{C}$. Note that two individuals i and i' have distinct signatures according to a set of attributes $\mathcal{C}$ if and only if $\mathcal{C}$ possesses at least one attribute of their separating set. We obtain the following system:

$$\begin{cases} \text{minimize} \sum_{a \in \mathcal{A}} x_a \\ \forall (i, i') \in \mathcal{I}^2 \text{ with } i \neq i', \sum_{a \in \mathrm{Sep}(i,i')} x_a \geqslant 1 \end{cases}$$

This ILP has a solution if and only if all the separating sets are non-empty *i.e.* if and only if no two individuals possess exactly the same attributes. This program also solves the problem of separating code in a graph (Definition 1.21), which is a special case of this problem (see Example 1.29). Thus, there exists a separating code in a graph $G = (V, E)$ if and only if $\forall u, v \in V^2, N[u] \neq N[v]$.

Such ILP reformulation of identifying codes problems have been studied extensively by Argiroffo et al. in [3].

If one wants to solve identifying code instead, we need the solution to also be dominating, *i.e.* $\forall i \in \mathcal{I}, \exists a \in \mathcal{C}, i \in a$. While separation and domination seem

unrelated at first sight, one can actually easily reduce identification to separation. Indeed, notice that separation consists of making the signatures of the individuals pairwise distinct while identification requires that the signatures are both pairwise distinct and non-empty. Therefore, all we have to do is add to our set of individuals an artificial individual that has no attribute and whose signature is thus necessarily empty. We thereby force all the other individuals to have non-empty signatures. Hence, the identification problem is resolvable if and only if no two individuals possess exactly the same attributes and each individual possesses at least one attribute.

Finally, we mentioned in Subsection 1.1.4 that in some applications, some attributes are more expensive to test than others and we are more interested in a test cover of minimum cost than in a test cover of minimum size. The reduction we present above can easily be generalized to the weighted case: if each attribute a has a cost c_a, we replace our objective function by $\sum_{a \in \mathcal{A}} c_a x_a$.

Chapter 2

Separating codes and traffic monitoring

This chapter mainly presents the work published in [9].

Contents

2.1 Introduction

Characterizing objects by testing as few properties as possible is an important task in diagnosis or identification problems and has been broadly studied in varying forms including separating, identifying codes and test covers (see Definitions 1.21 and 1.24). Separating codes have many applications in a wide range of domains. In each case, we have to deliver a diagnosis with limited or expensive access to information. Notable examples include visualization and pattern detection [14] [93], routing [76] or fault detection [87] in telecommunication networks, as well as many areas of bio-informatics, such as analysis of molecular structures [60] or in medical diagnosis, where test covers are the core of diagnostic tables [111] and are therefore important for blood sampling or bacterial identification (see [112] for a survey on different methods). Separating codes have also been studied under the name of *sieves* in the context of logic characterizations of graphs; the size of a minimum separating code determines the complexity of the first-order logic formula required to describe a graph [71].

In this chapter, the studied problem is traffic monitoring. Assume that traffic is going through a network modelled as a directed graph (e.g. cars in a town, packets

in a telecommunication network, skiers in a ski resort...) and that we are given the opportunity to install sensors on the arcs of the graph. Each time an object walks in the graph and goes through an equipped arc, it activates a sensor, and we know how many times and in which order each sensor was activated by that object. We are given the set of possible walks the object can take. Our goal is to find where to place the sensors so that we are able to determine exactly which route the object took from the information given by the sensors. This problem has been proven NP-complete by Maheshwari in [85], even in the case of acyclic directed graphs. Aside from the complexity aspect, few results have been obtained on the problem. We have to take into account information such as the multiplicity and the order of the signals sent by the sensors, which place this problem beyond the expressive power of existing models for separating codes and their resolution methods.

For the special case of monitoring skiers, Meurdesoif et al. developed a solution for acyclic directed graphs [88]. Their algorithm is based on double-path detection and their approach is very different from ours. In this chapter, we present a new, more flexible approach based on separating codes that allows us to handle more general problems.

Section 2.2 gives a formal definition of the traffic monitoring problem and outlines the limitations of the separating code model that make it unsuitable for handling traffic monitoring. We present in Section 2.3 a new model of separation based on language theory that overcomes these limitations and even generalizes the traffic monitoring problem. The next three sections focus on particular cases of traffic monitoring. Section 2.4 studies the case where the set of walks to separate is finite and present a reduction to an integer linear program. Section 2.5 studies the case where we want to separate every walk starting from a given set of starting points and leading to a given set of destinations. Such sets of walks can be infinite and would therefore yield infinitely many constraints. We study the underlying language and exhibit some properties that enable us to reformulate the problem as a standard integer linear program with finitely many constraints. Section 2.6 solves the same problem in the more general case of graphs with forbidden transitions. This model is much more relevant for physical networks such as road networks.

2.2 The traffic monitoring problem

2.2.1 Definition

Here, we model a network with a directed graph and have the option of installing sensors on the arcs of the graph. All the graphs we consider in this chapter are directed and can be multigraphs unless explicitly stated otherwise. The aim is to determine where to place the sensors to be able to reconstruct the route of objects walking in the graph.

Definition 2.1. Traffic Monitoring:

Let $G = (V, A)$ be a directed graph and let $\mathcal{C} \subset A$ be the set of arcs equipped with sensors. We say that the arcs of $\mathcal{C}$ are *monitored*.

When an object walks in G, it activates a sensor each time it uses a monitored arc. By moving in the graph, the object thus activates the sensors a certain number

of times in a certain order. We call the ordered sequence of activated sensors the *signature* of the walk of the object.

In this problem, we are given the set $\mathcal{W}$ (which does not have to be finite) of potential walks (also called *routes*) that the object in the graph can take. A set of arcs $\mathcal{C}$ *separates* $\mathcal{W}$ if and only if all the walks in $\mathcal{W}$ have different signatures. If this is the case, the information given by the sensors is sufficient to determine exactly which route the object picked. Note that we make no assumption on the speed of the object; the time between the activation of two sensors cannot be used to determine what the object did in the meantime. However, we know the order in which the sensors were activated.

The *Traffic Monitoring problem* is defined as follows:

Traffic Monitoring

Input: A directed graph $G = (V, A)$ and a set $\mathcal{W}$ of walks in G.
Output: A minimum set of arcs $\mathcal{C}$ such that the signatures of the walks of $\mathcal{W}$ are pairwise distinct.

Example 2.2.

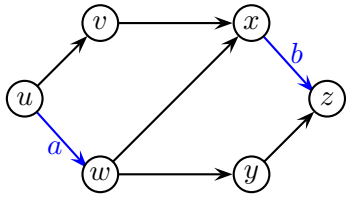


Figure 2.1: An acyclic graph equipped with two sensors.

Consider the graph depicted in Figure 2.1. Assume that we want to separate the set $\mathcal{W} = \{W_1 = (u, v, x, z), W_2 = (u, w, x, z), W_3 = (u, w, y, z)\}$ of walks from u to z. We monitor the set of arcs $\mathcal{C} = \{uw, xz\}$ depicted in blue and we note a the sensor we put on the arc uw and b the sensor on xz. Thus, the signatures of the walks of $\mathcal{W}$ are respectively $\text{sign}(W_1) = b$, $\text{sign}(W_2) = ab$ and $\text{sign}(W_3) = a$ which means that $\mathcal{C}$ separates $\mathcal{W}$. One can check that no set of one sensor separates $\mathcal{W}$ and $\mathcal{C}$ is therefore an optimal solution.

2.2.2 Limitations of the existing separation models

The problem of traffic monitoring consists of distinguishing a set of individuals (walks in a directed graph) by testing as few attributes (the arcs that the walks use) as possible. However, while this problem looks close to the the separation problems we have introduced so far (see Definition 1.24 for the most general model), it presents three major difficulties that we have not encountered yet which place it beyond the expressive power of the models described previously.

The set of activated sensors is not sufficient to identify a walk. Here, the attributes that we can test are the arcs that a walk uses. However, we expect

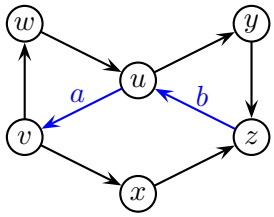


Figure 2.2: A graph G equipped with two sensors.

distinguishable individuals to have different attributes. In the graph G depicted in Figure 2.2, the walks (u, v, w, u) and (u, v, w, u, v, w, u) are different but the sets of arcs they use are the same. Since they do not use these arcs the same number of times, we still can distinguish them: their signatures according to the sensor set $\{a, b\}$ are respectively a and aa. Unfortunately, this forces us to take into account the multiplicity of the attributes of our individuals and leaves us with infinitely many possible value for each attribute. While this problem cannot be described by sets of individuals and attributes, we can still try to adapt the method presented in Example 1.78 by defining the separating sets of two walks (see Definition 1.77) as the set of edges that the two walks do not use the same number of times. We will see in the following that this does not work either.

The number of times each sensor is activated is not sufficient to identify a walk. In Figure 2.2, the walks (u, v, w, u, y, z, u) and (u, y, z, u, v, w, u) not only use the same arcs but they also use them the same number of times. They can still be separated; their signatures are indeed respectively ab and ba, but this requires considering the order of the attributes. This illustrates the limitation not only of the model but also of the resolution method we showed in Example 1.78. Indeed, we know that a code separates two individuals if and only if it contains an element that does so, but one can see here that the sensor set $\{a, b\}$ separates the walks (u, v, w, u, y, z, u) and (u, y, z, u, v, w, u) while neither $\{a\}$ nor $\{b\}$ does.

The set $\mathcal{W}$ of potential walks can be infinite. If the graph contains a cycle, the number of walks in it is infinite and $\mathcal{W}$ can be any subset of it. Therefore, even checking in finite time whether a given set of sensors separates $\mathcal{W}$ is non-trivial since it requires ensuring that all the walks of $\mathcal{W}$ have different signatures. A wrong intuition is that the problem can be reduced to separation on elementary paths since non-elementary walks are concatenations of elementary paths (which would be helpful since there are only a finite number of them), but this does not work. For example, assume that the set $\mathcal{W}$ we want to separate is the set of cycles starting from and leading to the vertex u. Our set of sensors $\{a, b\}$ does not separate $\mathcal{W}$ since the cycles (u, v, w, u, y, z, u) and (u, v, x, z, u) both have the signature ab. However, all the elementary cycles $((u, v, w, u)$, (u, y, z, u) and $(u, v, w, z, u))$ have different signatures (respectively, a, b and ab).

2.3 A new model of separation: separation on a language

2.3.1 Presentation of the problem

As highlighted in Subsection 2.2.2, the existing models for separation do not allow to address traffic monitoring. This section introduces a new model of separation based on language theory that overcomes the limitations we pointed out.

Definition 2.3. Separation on a language:

Let A be an alphabet, let $u \in A^*$ and let $\mathcal{C}$ be a subalphabet of A. The *projection of u on $\mathcal{C}$*, noted $p_{\mathcal{C}}(u)$, is the longest subword of u which uses only letters of $\mathcal{C}$.

We define the problem of *separation on a language* as follows:

Separation on a language

Input: A language L on an alphabet A.
Output: A minimum subalphabet $\mathcal{C} \subseteq A$ such that all of the words of L have different projections on $\mathcal{C}$.

Similarly, we define *identification on a language* by adding the constraint that no word of L can have an empty projection on $\mathcal{C}$.

Example 2.4.

The projection on $\{a, b\}$ of the word *abacacb* is *abaab*. Intuitively, all we have to do is read the word from left to right and select only the letters that belong to the subalphabet $\mathcal{C}$ we project on.

Let us now solve separation on the language $L = \{aabcc, acabc, baacb, cbaac\}$. One can immediately notice that *aabcc* and *acabc* use the same letters the same number of times. Hence, we cannot separate them with an alphabet of one letter. Furthermore, the projection of both these words on the subalphabet $\{a, b\}$ is *aab* so this subalphabet also does not separate them. The projections of *acabc* and *cbaac* on $\{b, c\}$ are also the same (*cbc*). However, the projections of the four words of L on the subalphabet $\{a, c\}$ are respectively $aacc, acac, aac$ and $caac$ and are pairwise distinct. Hence, $\{a, c\}$ is a solution of the problem and is the only optimal solution.

2.3.2 Expressiveness of the model

It is easy to reduce the problem of test cover to the problem of separation on a language. Indeed, let $\mathcal{I}$ be a set of individuals, let $\mathcal{A}$ be a set of attributes and let $\leqslant$ be a total order on $\mathcal{A}$. Let us associate with each individual i a word on $\mathcal{A}$ composed of all the attributes that i possesses, exactly once, in increasing order according to $\leqslant$. The subalphabet of $\mathcal{A}$ that separates the language of the words associated to the individuals of $\mathcal{I}$ are exactly the set of attributes that separates $\mathcal{I}$. Since test cover is NP-complete in the size of $\mathcal{I}$ and $\mathcal{A}$, it follows that:

Theorem 2.5. *Separation on a language is NP-complete in both the size of the language and the alphabet it is defined on.*

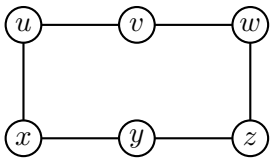


Figure 2.3: The graph C_6.

Example 2.6.

Consider the graph C_6 depicted in Figure 2.3 and let us express separating codes in it as a separating alphabet on a language. As explained in Example 1.29, our attributes are the $N[v]$ for $v \in V(C_6)$. We build a word associated to each vertex by looking at the closed neighbourhood they belong to. Finding a separating / identifying code in C_6 comes down to solving separation / identification on $L = \{uvx, uvw, vwz, uxy, xyz, wyz\}$. We saw in Example 1.12 that the subalphabet $\{u, v, w\}$ separates L and that $\{u, w, y\}$ identifies it.

However, even if we do not need to use it in the reduction of separating codes in a hypergraph, there can be multiple occurrences of a letter in a word, words are ordered sequences of letters and languages can be infinite. Hence, our new model overcomes the three limitations of previous models highlighted in Subsection 2.2.2.

Back to our problem of traffic monitoring, let $G = (V, A)$ be a directed graph and let $\mathcal{W}$ be a set of walks in G. Since the set A of arcs of G is finite and non-empty, it is an alphabet. If we see a walk in G as a sequence of arcs, it is a word of A^* and the set $\mathcal{W}$ of possible walks is a language on A. Given a set of sensors, the signature of a walk is its projection on the subalphabet composed of the monitored arcs. Hence, traffic monitoring is a particular case of separating code in a language.

Unless we allow G to be a multigraph, there are languages that cannot be written as a set of walks on $A(G)$ and the problems are not equivalent. For example, the words abd and acd both denote possible walks in G only if b and c have same origin and target. If G may be a multigraph, traffic monitoring and separation on a language are equivalent. Indeed, if G has only one vertex and contains a loop for each letter of A, then every word of A^* denotes a possible walk in G.

The reduction we presented in this subsection allows us to use tools arising from separating codes and language theory to address the traffic monitoring problem. However, our new separation model is still NP-complete in the size of the language that we want to separate and we saw that in some instances, the language we have to separate can be infinite. Hence, we do not intend to solve the problem in general, but we intend to address some classes of sets of walks which are of practical interest.

2.4 Separation of a finite set of walks

The easiest place to start is the case where the set of walks we want to separate is finite. This happens in particular when we want to solve traffic monitoring on an acyclic graph or when we can bound the length of the walks in the network. This problem already covers a wide range of applications.

Finite Traffic Monitoring

Input: A directed graph $G = (V, A)$ and a finite set $\mathcal{W}$ of walks in G.
Output: A minimum set of arcs $\mathcal{C}$ such that the signatures of the walks of $\mathcal{W}$ are pairwise distinct.

This problem comes down to solving separation on a finite language. The ILP we designed in Example 1.78 was based on the central notion of separating sets (Definition 1.77). The fundamental property of the separating set of two individuals i and i' is that a set of attributes separates i and i' if and only if it contains an attribute of their separating set. However, we saw that this property no longer holds for traffic monitoring and separation on a language. What still holds however is that if a set separates two words w and w', so do its supersets. Hence, we can still look for the minimal sets of letters that separate two words w and w' and we know that a subalphabet separates w and w' if and only if it contains one of those minimal sets.

Definition 2.7. Separating set of two words:
Given two words w and w' on an alphabet A, a subalphabet $\mathcal{C}$ of A belongs to the *separating set* $\mathrm{Sep}(w, w')$ of w and w' if and only if $\mathcal{C}$ separates w and w' and none of its strict subsets do. The separating set of two words is therefore a set of sets of letters.

It follows from this definition that a subalphabet $\mathcal{C} \subseteq A$ separates two words w and w' if and only if there exists $\mathcal{C}' \subseteq \mathcal{C}$ such that $\mathcal{C}' \in \mathrm{Sep}(w, w')$.

Example 2.8.
Let $A = \{a, b, c, d, e, f\}$, $u = bfcfabce$ and $v = bcfafbcb$. The subalphabet $\{c, d, f\}$ separates u and v but does not belong to $\mathrm{Sep}(u, v)$ since $\{c, f\}$ separates u and v too. However, neither $\{c\}$ nor $\{f\}$ separates u and v and $\{c, f\}$ therefore belongs to $\mathrm{Sep}(u, v)$. We can find that $\mathrm{Sep}(u, v) = \{\{b\}, \{e\}, \{a, f\}, \{c, f\}\}$. Since $\{b, c, d\}$ contains the subalphabet $\{b\}$ which belongs to $\mathrm{Sep}(u, v)$, it separates u and v.

We now exhibit some properties of the structure of separating sets that are useful in computing them efficiently (Theorem 2.11).

Lemma 2.9.
A word $u \in A^*$ is characterized by its projections on the subalphabets of A of cardinality 2.

Proof. The letter a is the first letter of a word u if and only if it is the first letter of all projections of u on the subalphabets of cardinality 2 containing a. One can then remove the a in the first position of the projection of u on the subalphabets containing a and find the second letter of u. This can be iterated until u is entirely determined. □

Example 2.10.
Let $A = \{a, b, c\}$ and $u \in A^*$ such that $p_{\{a,b\}}(u) = abba$, $p_{\{a,c\}}(u) = aca$, $p_{\{b,c\}}(u) = bbc$. Since a is the first letter of the projections of u on $\{a, b\}$ and $\{a, c\}$,

we know that a is the first letter of u. Thus, there exists v such that $u = av$. From the projection of u, we deduce that $p_{\{a,b\}}(v) = bba$, $p_{\{a,c\}}(v) = ca$, $p_{\{b,c\}}(v) = bbc$ and the first letter of v is therefore b. Thus, there exists w such that $u = abw$ and by iterating the process, we determine that $u = abbca$.

Theorem 2.11.
The separating set of two words contains only sets of cardinality at most 2.

Proof. Let u and v be two words on an alphabet A and let $\mathcal{C}$ be a subalphabet of A of cardinality at least 3 such that no strict subalphabet of $\mathcal{C}$ separates u and v. Hence, for every subalphabet $\mathcal{C}'$ of $\mathcal{C}$ of cardinality 2, $p_{\mathcal{C}'}(u) = p_{\mathcal{C}'}(v)$. Since $\mathcal{C}'$ is a subalphabet of $\mathcal{C}$, we know that $p_{\mathcal{C}'}(u) = p_{\mathcal{C}'}(p_{\mathcal{C}}(u))$ and $p_{\mathcal{C}'}(v) = p_{\mathcal{C}'}(p_{\mathcal{C}}(v))$. Hence, $p_{\mathcal{C}}(u)$ and $p_{\mathcal{C}}(v)$ are two words on $\mathcal{C}$ whose projections on every subalphabet $\mathcal{C}'$ of $\mathcal{C}$ are identical and therefore, by Lemma 2.9, $p_{\mathcal{C}}(u) = p_{\mathcal{C}}(v)$ which means that $\mathcal{C}$ does not separate u and v. □

Let u and v be two words of A^*. We can build their separating set as follows:

- If a letter a does not appear the same number of times in u and in v, then a alone suffices to separate them. We thus add $\{a\}$ to $\mathrm{Sep}(u, v)$ and while the pairs containing a all separate u and v, they do not belong to the separating set.

- If a letter a appears neither in u nor in v, containing it would be of no help to separate u and v. Thus, the separating set contains no pair containing a. What is left to investigate are pairs $\{a, b\}$ composed of two letters appearing the same number of times in u and v. Let k be the number of occurrences of a in u and v ($k \neq 0$). Thus, there exist $u_0, \cdots, u_k, v_0, \cdots, v_k$ such that $u = u_0 a u_1 a \cdots a u_k$ and $v = v_0 a v_1 a \cdots a v_k$. The pair $\{a, b\}$ belongs to the separating set if and only if there exists $i \in [\![0, k]\!]$ such that u_i and v_i do not contain the same number of occurrences of the letter b. These decompositions of u and v can then be reused when investigating other pairs containing a.

We now present how to take advantage of Theorem 2.11 to separate a finite language. Let $L \subseteq A^*$ be the language we want to separate. For each letter $a \in A$, let $x_{\{a\}}$ be a binary variable that indicates whether $a \in \mathcal{C}$ where $\mathcal{C}$ is the minimum separating subalphabet we want to build. For each pair of letters $\{a, b\}$, let $x_{\{a,b\}}$ be a binary variable that indicates whether both a and b belong to $\mathcal{C}$ (thus, $x_{\{a,b\}} = \min(x_{\{a\}}, x_{\{b\}})$). We obtain the following ILP:

$$\begin{cases} \text{minimize} \displaystyle\sum_{a \in A} x_{\{a\}} \\ \forall a \neq b \in A, 2x_{\{a,b\}} \leqslant x_{\{a\}} + x_{\{b\}} \quad \text{(ensures that } x_{\{a,b\}} \leqslant \min(x_{\{a\}}, x_{\{b\}})) \\ \forall a \neq b \in A, x_{\{a,b\}} + 1 \geqslant x_{\{a\}} + x_{\{b\}} \quad \text{(ensures that } x_{\{a,b\}} \geqslant \min(x_{\{a\}}, x_{\{b\}})) \\ \forall v \neq v' \in L, \displaystyle\sum_{S \in \mathrm{Sep}(v,v')} x_S \geqslant 1 \quad \text{(where } S \text{ can denote a singleton or a pair)} \end{cases}$$

The solution is thus $\mathcal{C} = \{a \in \mathcal{A} : x_{\{a\}} = 1\}$. The inequalities of the third line are not necessary to ensure the validity and the optimality of the solution but they ensure that the values of the pair-variables are determined exactly by the values of the singleton-variables. Hence the number of degrees of freedom of the problem is linear in the cardinality of A and not quadratic.

Solving identification instead of separation can be relevant too since domination let us know that an object is walking in the network and separation indicates which walk it uses. Like in Example 1.78, it can be solved simply by adding the empty word ε to the language we separate. Similarly, models where certain arcs of the network are more expensive to monitor than others can be useful in practice and can also be expressed by our model. We can solve them by replacing the objective function by $\sum_{a \in A} c_a x_{\{a\}}$ where c_a is the cost of monitoring the arc a.

Another interesting variant of *traffic monitoring* is the one where several objects are walking in the graph and all have different sets of possible walks $\mathcal{W}_1, \ldots, \mathcal{W}_k$. Our aim is still to determine which walks each object uses but this variant cannot easily be reduced to the standard model: indeed, one cannot solve the problem on each set $\mathcal{W}_i$ separately because we need to separate all the $\mathcal{W}_i$ with the same censor set but separating the union of the $\mathcal{W}_i$ is superfluous (the case where two walks have same signature but cannot be used by the same object cannot lead to a situation where we do not know which walk an object has used and does not have to be avoided) and leads to sub-optimal solution. We would like to point out that our method can easily be adapted solve this variant too.

2.5 Separation of walks with given sets of starting points and destinations

This subsection studies the problem that we call *complete traffic monitoring*. Here, we are given a set of potential starting points V_I and a set of potential destinations V_F. The set $\mathcal{W}$ of walks we want to separate is the set of all the walks leading from a vertex of V_I to a vertex of V_F. If the graph contains a cycle, there can be infinitely many such walks.

Complete Traffic Monitoring

Input: A directed graph $G = (V, A)$ and two subsets of vertices V_I and V_F.
Output: A minimum set of arcs $\mathcal{C}$ such that no two walks leading from a vertex of V_I to a vertex of V_F have the same signature.

Of course, since the number of walks to separate can be infinite, it is not feasible to compute the separating set of each pair of walks. However, notice that the separating set of two walks is included in the powerset of the set A of arcs of the graph and can therefore only take a finite number of values. Thus, while the linear program presented in the previous section would have infinitely many constraints, only a finite number of them would be distinct. If we can determine which values of $\mathrm{Sep}(v, v')$ are actually reached, we would therefore be able to describe the same

polytope with only a finite number of constraints. This is the main result of this section (Theorem 2.22).

2.5.1 Study of the reachable languages

Since we do not want to deal with NP-complete problems on general infinite instances, we know that our solution must take advantage of the specificities of the instances of complete traffic monitoring. To use our reduction to separation on languages, we need to understand how the constraints we have on walks translate on languages. We define reachable languages so that complete traffic monitoring is equivalent to separation on reachable languages.

Definition 2.12. Reachable languages:
A language L on an alphabet A is *reachable* if and only if there exists a directed graph $G = (V, A)$ (A is both the alphabet of L and the set of arcs of the graph), a set of vertices $V_I \subseteq V$ and a set of vertices $V_F \subseteq V$ such that L depicts the set of walks in G leading from a vertex of V_I to a vertex of V_F. We denote by $\text{Reach}(A)$ the set of reachable languages on an alphabet A.

We now investigate interesting properties of reachable languages.

Lemma 2.13. *If $L \subseteq A^*$ is reachable, then:*

$$\forall u, u', v, v' \in A^*, \forall a \in A, \begin{cases} uav \in L \\ u'av' \in L \end{cases} \Rightarrow \begin{cases} uav' \in L \\ u'av \in L \end{cases}$$

Proof. The idea of the proof is that in complete traffic monitoring, the choices that the walker has at a given time only depend on the current vertex he is on and cannot be restricted depending on where he comes from.

More formally, let $G = (V, A)$ be a directed graph and let V_I and $V_F \subseteq V$ be such that L is the set of walks leading from V_I to V_F. Let o_a and t_a be the origin and target of the arc a. If uav and $u'av'$ are in a reachable language L, this means that:

- u describes a walk that leads from a vertex $i \in V_I$ to o_a;
- u' describes a walk that leads from a vertex $i' \in V_I$ to o_a;
- v describes a walk that leads from t_a to a vertex $f \in V_f$;
- v' describes a walk that leads from t_a to a vertex $f' \in V_f$.

Hence, uav' and $u'av$ also describe valid walks leading from a vertex of V_I to a vertex of V_F and therefore belong to L too. □

Proposition 2.14. $\text{Reach}(A) \subsetneq \text{Rat}(A)$

Proof. The proof is in two steps:
• $\text{Reach}(A) \subseteq \text{Rat}(A)$. Let $G = (V, A)$, V_I and V_F be an instance of complete traffic monitoring and let L be the associated reachable language. The automaton

whose alphabet is A, set of states is V, set of transitions is $T = \{(u, uv, v) : (u, v) \in A\}$, set of initial states is V_I and set of final states is V_F recognizes exactly L. By Theorem 1.69 (Kleene), this proves that reachable languages are rational.

• $\mathrm{Reach}(A) \neq \mathrm{Rat}(A)$. Note that the previous construction provides very specific automata where each letter of the alphabet labels exactly one transition. There are rational languages that cannot be recognized by such automata. For example, let L be a reachable language such that $ababa \in L$. Then, $(ab)a(ba) \in L$ and $\varepsilon a(baba) \in L$. Hence, according to Lemma 2.13, $(ab)a(baba)$ and $a(ba)$ both belong to L too. Hence, the language $\{ababa\}$ although rational, is not reachable. If A contains only one letter a, we find similarly that only $\varnothing$, $\{a\}$ and A^* are reachable while $\{aa\}$ is rational. □

2.5.2 Reduction theorem and resolution

We first define the notion of restriction of a rational language which is useful for the reduction theorems (Theorem 2.22 and Theorem 2.31).

Definition 2.15. Restriction of a rational language:

Given a regular expression of a rational language L (see Definition 1.64), we define a *restriction* of L and we denote by $\overline{L}$ the language built inductively as follows:

- $\overline{\varnothing} = \varnothing$
- $\forall a \in A^*, \overline{\{a\}} = \{a\}$
- $\overline{L_1 + L_2} = \overline{L_1} + \overline{L_2}$
- $\overline{L_1 L_2} = \overline{L_1}\ \overline{L_2}$
- $\overline{L^*} = \varepsilon + \overline{L} + \overline{L}^2$.

Notice that the restriction of a language L is not unique. Indeed, two regular expressions can denote the same language but their associated restrictions can differ. For example, $L^{**} = L^*$ but unless $L = \varnothing$, $\overline{L^{**}} = \sum_{i=0}^{4} \overline{L}^i \neq \sum_{i=0}^{2} \overline{L}^i = \overline{L^*}$.

Proposition 2.16. *Every restriction $\overline{L}$ of a rational language L is finite.*

Proof. The proof by induction is immediate. Indeed, restricted languages are empty, singletons or built from other restricted languages using only finite unions or concatenations, which are operations that preserve the finiteness of the language. □

We now introduce the notion of acyclicity of walks and words which helps us prove the reduction theorems.

Definition 2.17. Acyclic walks and acyclic words:

A walk is *acyclic* if and only if it does not use twice the same vertex. Hence, acyclic walks are exactly the elementary walks that are not cycles (see Subsection 1.3.1). Given a graph $G = (V, A)$, a word $u \in A^*$ is *acyclic* if and only if it denotes an acyclic walk in G.

Proposition 2.18.
We can extract from every walk W on a graph G an acyclic walk that only uses vertices and edges that W uses and has the same starting point and destination.

Proof. By Proposition 1.34, we can extract from W a path P with same starting point and destination as W. Hence, P satisfies the proposition unless it is a cycle. If P is a cycle on a vertex v, the walk (v) of length 0 satisfies the proposition. □

Corollary 2.19. *Given an instance of traffic monitoring and the associated reachable language L, one can extract from every word $u \in L$ an acyclic subword of u that denotes a path with same starting point and destination as u. We denote this word by* $\mathrm{Acycl}(u)$. *Notice that* $\mathrm{Acycl}(u) \in L$ *and if u denotes a cycle, then* $\mathrm{Acycl}(u) = \varepsilon$.

Finally, we need the following lemma which helps us prove that a word v in a reachable language L belongs to every restriction $\overline{L}$ of L.

Lemma 2.20.
For all rational languages L, for all restrictions $\overline{L}$ of L and for all words $u \in L \setminus \overline{L}$, there exist words v, w_1, w_2, w_3 and x such that $u = vw_1w_2w_3x$, w_1, w_2 and w_3 are all non-empty and each of the 8 strings resulting by deleting zero or more of the substrings w_1, w_2 and w_3 still belongs to L.

Proof. Let $\overline{L}$ be a reduction of L. We prove the lemma by induction on the regular expression of L from which $\overline{L}$ is built:

- if $L = \varnothing$, then $L \setminus \overline{L} = \varnothing$ and the lemma holds for L;
- if $L = \{u\}$ with $u \in A^*$, then $L \setminus \overline{L} = \varnothing$ and the lemma holds;
- if $L = L_1 + L_2$ and the lemma holds for L_1 and L_2:
  $L \setminus \overline{L} = (L_1 + L_2) \setminus (\overline{L_1} + \overline{L_2}) \subseteq (L_1 \setminus \overline{L_1}) + (L_2 \setminus \overline{L_2})$ and the lemma holds for L;
- if $L = L_1L_2$ and the lemma holds for L_1 and L_2: let $u \in L_1$ and $v \in L_2$ and let us observe that for uv not to belong to $\overline{L_1L_2}$, it is necessary that $u \notin \overline{L_1}$ or $v \notin \overline{L_2}$. Hence, $L \setminus \overline{L} \subseteq (L_1 \setminus \overline{L_1})L2 + L1(L_2 \setminus \overline{L_2})$ and the lemma still holds for L;
- if $L = L'^*$ and the lemma holds for L': we set $M = L' \setminus \{\varepsilon\}$. Note that $L'^* = M^*$. We also set $\overline{M} = \overline{L'} - \varepsilon$.

$$
\begin{aligned}
L \setminus \overline{L} = \left(\sum_{i\in\mathbb{N}} L'^i\right) \setminus \left(\sum_{i=0}^{2} \overline{L'}^i\right) &= \left(\sum_{i\in\mathbb{N}} M^i\right) \setminus \left(\sum_{i=0}^{2} \overline{M}^i\right) \\
&\subseteq \sum_{i=0}^{2} (M^i \setminus \overline{M}^i) + \sum_{i\geqslant 3} M^i \\
&\subseteq \varnothing + (M \setminus \overline{M}) + \underbrace{M^2 \setminus \overline{M}^2}_{=M(M\setminus\overline{M})+(M\setminus\overline{M})M} + M^3M^*
\end{aligned}
$$

Since the lemma holds for all words of $M \setminus \overline{M} = L' \setminus \overline{L'}$, it holds for all words of $M \setminus \overline{M} + M^2 \setminus \overline{M}^2$. Let us now prove it for $u \in M^3M^*$. By definition, u is the concatenation of $v = \varepsilon$, w_1, w_2 and $w_3 \in M$ (which are non-empty by construction of M) and $x \in M^*$. Hence, even if we remove some of the w, u is still a concatenation of words of M and therefore still belongs to $M^* = L$.

This proves the lemma. □

In this section, we only need the following weaker version of Lemma 2.20:

Corollary 2.21.
Let G, V_I, V_F be an instance of complete traffic monitoring, let L be the associated reachable language, let $\overline{L}$ be a restriction of L, and $u \in L \setminus \overline{L}$. Then there exists a vertex $q \in V(G)$ such that u denotes a walk using at least four times the vertex q.

Proof. We decompose u with Lemma 2.20. The fact that u still denotes a walk in G if we remove some of the w_i proves that it uses the same vertex between v and w_1, between w_1 and w_2, between w_2 and w_3 and between w_3 and x. Since the w_i are non-empty, these are four different occurrences of the same vertex in the walk. □

We are now ready to present the main theorem of this section (Theorem 2.22). We prove this theorem in a more general case (Theorem 2.31) in Section 2.6 its proof is more technical. This proof of Theorem 2.22 gives a better intuition of why the theorem holds.

Theorem 2.22. *Reduction theorem (reachable language version)*
For all reachable languages L on an alphabet A, for all restrictions $\overline{L}$ of L, if $A' \subseteq A$ separates $\overline{L}$, then it separates L.

Proof. Let $L \subseteq A^*$ be a reachable language, let $\mathcal{C} \subseteq A$ and $u, v \in L$ be such that $u \neq v$ but $p_{\mathcal{C}}(u) = p_{\mathcal{C}}(v) = a_0 \cdots a_n$. Thus, $u = u_0a_0u_1a_1 \cdots a_nu_{n+1}$ and $v = v_0a_0v_1a_1 \cdots a_nv_{n+1}$ where for all i, u_i and v_i belong to $(A \setminus \mathcal{C})^*$. We want to prove that there exist two different words that belong to every restriction $\overline{L}$ of L and have the same signature.

Since $u \neq v$, we know that there exists i such that $u_i \neq v_i$. Moreover, since L is reachable, by using Lemma 2.13 twice, we find that $u_0a_0 \cdots u_{i-1}a_{i-1}\boxed{v_i}\underbrace{a_iu_{i+1} \cdots u_{n+1}}_{=y}$ still belongs to L. To keep the notation simple, we set $y = a_iu_{i+1} \cdots u_{n+1}$.

Let x be the longest common prefix of u_i and v_i. Hence, $u_i = xbu'_i$ and $v_i = xcv'_i$ where b and $c \in (A \setminus \mathcal{C}) \cup \{\varepsilon\}$ are the first letters of the suffix of u_i and v_i that start after x (or ε if this suffix is empty). Thus, $b \neq c$ and b or c is empty if and only if $x = u_i$ or $x = v_i$ respectively (hence, if $b = \varepsilon$ then $u'_i = \varepsilon$ too and the same holds for c and v'_i). We also set $z = u_0a_0 \cdots u_{i-1}a_{i-1}x$. Thus, $u = zbu'_iy$ and we know that $zcv'_iy \in L$ too.

We set $\overline{u} = \mathrm{Acycl}(z)b\,\mathrm{Acycl}(u'_i)\,\mathrm{Acycl}(y)$ and $\overline{v} = \mathrm{Acycl}(z)c\,\mathrm{Acycl}(v'_i)\,\mathrm{Acycl}(y)$. By definition of the function Acycl, a walk t has the same starting point and destination as $\mathrm{Acycl}(t)$. Hence, we still have $\overline{u}, \overline{v} \in L$.

We know that $b\,\mathrm{Acycl}(u_i')$ and $c\,\mathrm{Acycl}(v_i')$ cannot both be empty. If one of them is, they are thus necessarily distinct and if none of them are, we know that b and c denote different letters. In all cases, $b\,\mathrm{Acycl}(u_i') \neq c\,\mathrm{Acycl}(v_i')$ and thus, $\overline{u} \neq \overline{v}$.

Since letters denote arcs, the sequence of vertices visited by the walk associated to $\mathrm{Acycl}(z)b\,\mathrm{Acycl}(u_i')$ is the concatenation of the sequences of vertices visited by the walks associated to $\mathrm{Acycl}(z)$ and $\mathrm{Acycl}(u_i')$. By definition of acyclicity, it does not contain more than twice the same vertex. Thus, by construction of $\overline{u}$ and $\overline{v}$, the walks they denote cannot contain more than three times the same vertex. By contraposition of Corollary 2.21, this means that for all restrictions $\overline{L}$ of L, $\overline{u}$ and $\overline{v} \in \overline{L}$.

We know that for all words t, $\mathrm{Acycl}(t)$ only uses letters that t itself uses. Since u_i and v_i are words of $(A \setminus \mathcal{C})^*$, we know that $b\,\mathrm{Acycl}(u_i')$ and $c\,\mathrm{Acycl}(v_i') \in (A \setminus \mathcal{C})^*$ too. Hence,

$$\begin{aligned} p_{\mathcal{C}}(\overline{u}) &= p_{\mathcal{C}}(\mathrm{Acycl}(z))\,\underbrace{p_{\mathcal{C}}(b\,\mathrm{Acycl}(u_i'))}_{=\varepsilon}\,p_{\mathcal{C}}(\mathrm{Acycl}(y)) \\ &= p_{\mathcal{C}}(\mathrm{Acycl}(z))\,\underbrace{p_{\mathcal{C}}(c\,\mathrm{Acycl}(v_i'))}_{=\varepsilon}\,p_{\mathcal{C}}(\mathrm{Acycl}(y)) \\ &= p_{\mathcal{C}}(\overline{v}) \end{aligned}$$

We proved that for all reachable languages L on an alphabet A, for all subalphabets $\mathcal{C} \subseteq A$, if there exist $u \neq v$ in L such that $p_{\mathcal{C}}(u) = p_{\mathcal{C}}(v)$, then there also exist two words in every restriction $\overline{L}$ of L that are different but still have the same projection on $\mathcal{C}$. The contrapositive of this result is our theorem. □

Note that the converse is obviously true since $\overline{L} \subseteq L$. Thus, the subalphabets that separate $\overline{L}$ are exactly those that separate L.

Hence, given a directed graph, a set V_I of potential starting points and a set V_F of potential destinations, we proceed as follows to solve the complete traffic monitoring problem:

- We know that the language of possible walks leading from a vertex of V_I to a vertex of V_F is rational and therefore admits a regular expression. The graph directly provides an automaton which recognizes it and we can use Lemma 1.70 (Arden) to find an expression of the associated reachable language that we want to separate.
- We use the regular expression of the language to determine a restriction. We know by Proposition 2.16 that the restriction is finite.
- Due to Theorem 2.22, the solutions on the restricted language are exactly the solutions on the initial language. All that is left to do is to use the method described in Section 2.4 to solve the problem on the restricted language which is finite.

Example 2.23.

Let us separate all the cycles starting at u in the graph $G = (V, A)$ depicted in Figure 2.4.

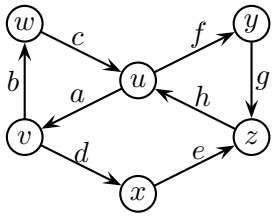


Figure 2.4: A graph $G = (V, A)$ with labelled arcs.

The language of walks we have to separate is $L = (abc + adeh + fgh)^*$ and is infinite. A restriction is $\overline{L} = \varepsilon + abc + adeh + fgh + abcabc + abcadeh + abcfgh + adehabc + adehadeh + adehfgh + fghabc + fghadeh + fghfgh$. We find that a minimum separating subalphabet on $\overline{L}$ (and therefore L) is for example $\mathcal{C} = \{b, d, f\}$.

2.6 Separation of walks with forbidden transitions

2.6.1 Motivation of the problem

Let us consider the very simple road depicted in Figure 2.5.

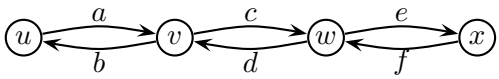


Figure 2.5: A bi-directed road between the vertices u and x.

Here, a driver who wants to go from vertex u to vertex x will simply use the path ace. Still, the model that we presented in the previous section requires us to distinguish all the walks of $(a(c(ef)^*d)^*b)^*a(c(ef)^*d)^*c(ef)^*e$. Taking into account such paradoxical behaviours is not only superfluous but it also leads to prohibitive computation times and tremendously increases the cost of the solutions. Here, at least three sensors are required to distinguish all the roads leading from u to x while only one of those roads makes sense in practice.

One could be tempted to get around this problem by contracting the arcs a, c and e and the arcs b, d and f in a pre-processing step, which would result in a bi-directed path of length 1 and bring the language to separate down to $(ab)^*a$. However, this would not be possible in a more complex road network. Indeed, a road network can for example feature crossroads on v and w and transversal roads could lead to those vertices or leave from them. While this does not change the fact that a driver who wants to go from u to x will always pick the route ace, it can make the contraction of the vertices v and w impossible and forces us to consider many absurd walks.

The approach we choose here is to forbid transitions in the graph. In this chapter, we choose to denote forbidden-transition graphs by a triplet (V, A, F) where F is the set of forbidden transitions which is usually smaller than the set of permitted transitions in a road network. In the rest of this section, we refer to graphs with no forbidden transition as *usual graphs*.

For example, in the road network of Figure 2.5, we can assume that drivers do not turn back in the middle of the road and forbid the transitions ab, cd and ef. We thereby reduce the set of roads leading from u to x down to $\{ace\}$. This also enables us to model situations where certain turns are prohibited, which is very common on road networks. By choosing wisely the forbidden transitions, we only discard routes that would be prohibited or absurd in practice and we can significantly reduce the computation time and the cost of the optimal solutions on large instances.

We call *restricted traffic monitoring* the problem that we study in this section:

Restricted Traffic Monitoring

Input: A directed FTG $G = (V, A, F)$ and two subsets of vertices V_I and V_F.
Output: A minimum set of arcs $\mathcal{C}$ such that no two compatible walks leading from a vertex of V_I to a vertex of V_F have the same signature.

2.6.2 Study of the FTG-reachable languages

As in Subsection 2.5.1, we define FTG-reachable languages so that restricted traffic monitoring is equivalent to separation on FTG-reachable languages.

Definition 2.24. FTG-reachable languages:
A language L on an alphabet A is *FTG-reachable* if and only if there exists a directed FTG $G = (V, A, F)$, a set of vertices $V_I \subseteq V$ and a set of vertices $V_F \subseteq V$ such that L is the set of compatible walks in G leading from a vertex of V_I to a vertex of V_F. We denote by $\mathrm{FTGR}(A)$ the set of reachable languages on an alphabet A.

We now investigate the properties of this new class of languages and compare it to reachable and rational languages.

Proposition 2.25. $\mathrm{Reach}(A) \subsetneq \mathrm{FTGR}(A)$.

Proof. The proof proceeds in two steps:
• $\mathrm{Reach}(A) \subseteq \mathrm{FTGR}(A)$. Since usual graphs are particular cases of FTGs (with $F = \varnothing$), the languages that are reachable by graphs are clearly reachable by FTGs.
• $\mathrm{Reach}(A) \neq \mathrm{FTGR}(A)$. Let L be a reachable language on $A = \{a, b, c, d\}$ such that $ac, ad, bc \in L$. Since both ac and ad are in L, we know that c and d denote arcs starting from the same vertex v and leading to a vertex of V_F. Furthermore, since both ac and bc belong to L, we know that a and b both denote arcs leading to v and starting from a vertex of V_I. Hence, $bd \in L$ too for every reachable language that contains ac, ad and bc.

Let us now consider the instance of restricted traffic monitoring given by the FTG depicted in Figure 2.6 where the transition (b, d) is forbidden, with $V_I = \{u, w\}$ and $V_F = \{w, x\}$. One can see that the associated FTG-reachable language contains ac, ad and bc since they describe compatible walks leading from a vertex of V_I to a vertex of V_F. However, it does not contain bd, which uses a forbidden transition. Hence, the language $\{ac, ad, bc\}$ is FTG-reachable but not reachable. □

It is important to note that Lemma 2.13 also holds for FTG-reachable languages:

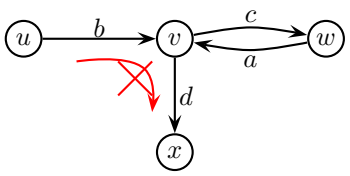


Figure 2.6: An example of FTG.

Lemma 2.26. *For all FTG-reachable languages $L \subseteq A^*$:*

$$\forall u, u', v, v' \in A^*, \forall a \in A, \begin{cases} uav \in L \\ u'av' \in L \end{cases} \Rightarrow \begin{cases} uav' \in L \\ u'av \in L \end{cases}$$

Proof. Let $G = (V, A, F), V_I, V_F$ be an instance of restricted traffic monitoring such that the set of walks to separate is L. Let us call L' the set of words that denote all the walks leading from a vertex of V_I to a vertex of V_F on the underlying usual graph $G = (V, A)$ with no forbidden transition. Hence, L' contains all the words of L plus eventually some words that contain transitions of F. Since L' is reachable by construction and $\{uav, u'av'\} \in L \subseteq L'$, we know by Lemma 2.13 that uav' and $u'av$ belong to L'. Therefore, the only way for uav' or $u'av$ not to belong to L is to contain two consecutive letters that denote a forbidden transition. However, every sequence of two letters in uav' and $u'av$ also appears in uav or $u'av'$ which both belong to L. Thus, uav' and $u'av$ are in L, which concludes the proof of the lemma. □

We now show that $\mathrm{FTGR}(A)$ is a proper subset of $\mathrm{Rat}(A)$:

Proposition 2.27. $\mathrm{FTGR}(A) \subsetneq \mathrm{Rat}(A)$.

Proof. The proof proceeds in two steps:

• $\mathrm{FTGR}(A) \subseteq \mathrm{Rat}(A)$. Let L be a FTG-reachable language and $G = (V, A, F), V_I, V_F$ be an instance of restricted traffic monitoring such that $\mathcal{W} = L$. To prove that L is rational, we create from G an automaton that recognizes L. To do so we create copies of each vertex of the graph for each possible incidence. For example, if an arc (u, v) leads to a vertex v, instead of just having a state v in our automaton, we create a state uv that is final if and only if v is final and from which we can reach any out-neighbour w of v unless $((u, v), (v, w)) \in F$. More formally, here is a construction of an automaton that recognizes L:

- its alphabet is the set A of arcs of the graph;
- its set of states is $V_I \cup \{uv : (u, v) \in A\}$;
- for all $v \in V_I$, for all out-neighbours w of v, we create a transition from v to vw labelled by the arc (v, w). For all states uv, for all out-neighbours w of v, we create a transition from uv to vw labelled by the arc (v, w) if and only if $((u, v), (v, w)) \notin F$;
- the set of initial states is still V_I;

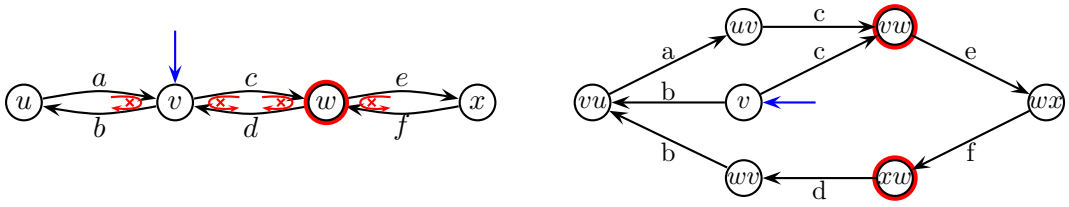
- the final states are those whose name ends with a vertex $v \in V_F$.

This construction is illustrated in Example 2.28.

• $\mathrm{FTGR}(A) \neq \mathrm{Rat}(A)$. The counter-example to the other inclusion is the same as in the proof of Proposition 2.14: the language *ababa* is rational but not FTG-reachable since it does not satisfy Lemma 2.26. □

Example 2.28.

For example, with the graph in Figure 2.7a, $F = \{(a,b),(c,d),(d,c),(f,e)\}, V_I = \{v\}$ and $V_F = \{w\}$, the associated FTG-reachable language is recognized by the automaton presented in Figure 2.7b whose initial and final states are respectively $\{v\}$ and $\{vw, xw\}$. Like in Section 1.5, initial and final states are denoted respectively by incoming blue arrows and red circles.



(a) The FTG of an instance of restricted traffic monitoring.

(b) The automaton recognizing the associated FTG-reachable language.

Figure 2.7

Note in particular that since FTG-reachable languages are rational, they admit a regular expression and therefore, a restriction.

2.6.3 Reduction theorem and resolution

We prove in this subsection that the reduction theorem also holds on FTG-reachable languages, that generalize reachable languages. However, the proof of the previous section does not work. Indeed, as illustrated in Example 1.43, Proposition 1.34 about extracting paths from walks does not hold anymore in FTGs. Similarly, Proposition 2.18 and Corollary 2.19 that play an important role in the proof of Theorem 2.22 do not hold either. Indeed, our method to extract a path from a walk involves transitions that are not in the initial walk. As long as the only forbidden transitions are transitions between two opposite arcs (as is the case in Example 2.28), an acyclic path extracted from a walk cannot involve a forbidden transition and the proof of Theorem 2.22 applies. In the general case, however, an acyclic path extracted from a permitted walk may involve forbidden transitions.

To overcome this problem, we look for an alternative characterization of acyclic walks in usual graphs. Note that acyclic walks are exactly those that cannot be decomposed as a concatenation of three walks $W_1W_2W_3$ such that W_2 is non-empty and W_1W_3 is also a walk in the graph. Indeed, such a decomposition is possible if and only if W_1 and W_2 end on the same vertex, which means that the walk is not acyclic. Therefore, by iterating the deletion of W_2 until the walk is acyclic (which

happens within a finite number of iterations since the length of the walk strictly decreases), one can extract from every walk W an acyclic walk W' with the same starting point and destination as W and that uses only vertices and edges that W uses. This reduction can be generalized to walks in FTGs.

Definition 2.29. Max-reduced form of a walk or a word:

A compatible walk W is *max-reduced* if and only if there does not exist three walks $W_1W_2W_3$ such that W_2 is non-empty and W_1W_3 is also a compatible walk in the graph. Similarly, a word u of a language L is *max-reduced* in L if and only if there does not exist $u_1u_2u_3$ with $u_2 \neq \varepsilon$ such that $u = u_1u_2u_3$ and $u_1u_3 \in L$. It follows that for all languages L and $u \in L$, there exists a subword v of u such that v is max-reduced and $v \in L$. The word v is a *max-reduced form of* u.

Note that unlike the acyclic form of a walk or a word, the max-reduced form is not unique. Indeed, the decomposition $u = u_1u_2u_3$ is not unique and depending on which decompositions we use for the reduction, we may end up on different max reduced words.

Example 2.30.

Let $G = (V, A, F)$ be the graph depicted in Figure 2.8 and let $G' = (V, A)$ be the usual graph associated to G.

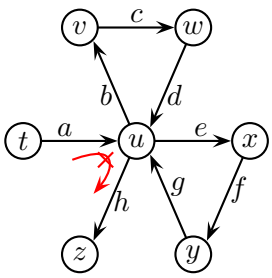


Figure 2.8: A graph G with two forbidden transitions.

Let $W = abcdefgh$. Since W uses three times the vertex u, it is not acyclic. In G', we can extract the acyclic walk ah from W. However, the transition ah is forbidden in G and one cannot extract from W an acyclic compatible walk. However, W is not a max-reduced compatible walk. Indeed, it can be written as the concatenation of $W_1 = abcd$, $W_2 = efg$ and $W_3 = h$ where W_2 is not empty and $W_1W_3 = abcdh$ is still a compatible walk in G. This reduction cannot be iterated again and $abcdh$ is a max-reduced compatible form of W. We could also have written W as the concatenation of $W_1 = a$, $W_2 = bcd$ and $W_3 = efgh$ and the resulting walk $aefgh$ is another max-reduced compatible form of W.

Note that the language L in Definition 2.29 does not have to be reachable or FTG-reachable. Thus, max-reduced words can be defined with respect to any constraints and not just compatibility with a set of forbidden transitions. Also note that Lemma 2.20 holds for all the rational languages L, which notably include the FTG-reachable languages.

Theorem 2.31. *Reduction theorem (FTG-reachable language version)*
For all FTG-reachable languages L on an alphabet A, for all restrictions $\overline{L}$ of L, if $A' \subseteq A$ separates $\overline{L}$, then it separates L.

Proof. Let $L \subseteq A^*$ be a FTG-reachable language, let $A' \subseteq A$ and $u, v \in L$ be such that $u \neq v$ but $p_{A'}(u) = p_{A'}(v)$. We want to prove that for all restriction $\overline{L}$ of L, there exist two different words in $\overline{L}$ that have the same signature.

Since Lemma 2.26 holds on FTG-reachable graphs, we can proceed like in the proof of Theorem 2.22 to build two words zbu'_iy and zcv'_iy in L where b and c belong to $A \setminus \{\mathcal{C}\} \cup \{\varepsilon\}$ and are different and u'_i and v'_i are words on the alphabet $A \setminus \{\mathcal{C}\}$ such that if b (resp. c) is empty, then u'_i (resp. v'_i) is empty too.

Let $\mathrm{red}(z)$ be a max-reduced form of z such that both $\mathrm{red}(z)b$ and $\mathrm{red}(z)c$ are compatible (we iterate the reduction as long as the condition holds). Similarly, let $\mathrm{red}(u'_i)$ and $\mathrm{red}(v'_i)$ be max-reduced forms of u'_i and v'_i such that $b\,\mathrm{red}(u'_i)$ and $c\,\mathrm{red}(v'_i)$ are compatible. Let $\mathrm{red}(y)$ be a max-reduced form of y such that $\overline{u} = \mathrm{red}(z)b\,\mathrm{red}(u'_i)\,\mathrm{red}(y)$ and $\overline{v} = \mathrm{red}(z)c\,\mathrm{red}(v'_i)\,\mathrm{red}(y)$ are both compatible. Since $\overline{u}$ and $\overline{v}$ are compatible and have same origin and destination as u, they still belong to L.

Case I: both $\overline{u}$ and $\overline{v}$ belong to every restriction $\overline{L}$ of L. We can prove that $\overline{u} \neq \overline{v}$ and $p_{\mathcal{C}}(u) = p_{\mathcal{C}}(v)$ like in the proof of Theorem 2.22.

Case II: at least one of $\overline{u}$ and $\overline{v}$, say $\overline{u}$ does not belong to a restriction $\overline{L}$ of L. Hence, by Lemma 2.20, we know that $\overline{u} = sw_1w_2w_3t$ where w_1, w_2 and w_3 are all non-empty and can be removed. If a word $\alpha\beta\gamma$ denotes a compatible walk, we say that β is a *removable factor* if and only if $\alpha\gamma$ is still compatible. For example, w_1, w_2 and w_3 are removable factors of $\overline{u}$. We recall that $\overline{u}$ can also be written $\mathrm{red}(z)b\,\mathrm{red}(u'_i)\,\mathrm{red}(y)$.

By definition, $\mathrm{red}(z)$ is a max-reduced form of z such that $\mathrm{red}(z)b$ and $\mathrm{red}(z)c$ are both compatible. This notably means that any further reduction of $\mathrm{red}(z)$ would change its last letter since it changes the letter we can write after it (the last letter of $\mathrm{red}(z)$ would become a letter l such that at least one of the transitions lb and lc is forbidden). Hence, a removable factor of $\mathrm{red}(z)$ is necessarily a suffix. Since we know by Lemma 2.20 that we can remove w_1 from $\overline{u}$ without making forbidden transitions appear, this proves in particular that w_1 cannot end strictly before $\mathrm{red}(z)$. Similarly, we can prove that a removable factor of $\mathrm{red}(y)$ is necessarily a prefix and therefore, that w_3 cannot begin strictly after $\mathrm{red}(y)$. Hence, we know that w_2 is a factor of $b\,\mathrm{red}(u'_i)$.

By definition of $\mathrm{red}(u'_i)$, we know that it does not contain any factor that we can remove without making a forbidden transition appear in $b\,\mathrm{red}(u'_i)$. Therefore, since w_2 is removable and a factor of $b\,\mathrm{red}(u'_i)$, it has to contain b. Putting it all together, we find out that w_2 is a prefix of $b\,\mathrm{red}(u'_i)$, which means that w_1 is a suffix of $\mathrm{red}(z)$. Since w_1 is non-empty, this proves that s is a strict prefix of $\mathrm{red}(z)$. Since w_3 is removable and $\mathrm{red}(u'_i)$ cannot contain a removable factor, we know that w_3 is the concatenation of a suffix of $\mathrm{red}(u'_i)$ and a non-empty prefix of $\mathrm{red}(y)$. Finally, we

know that t is a strict suffix of $\mathrm{red}(y)$. The relation between the two decompositions is illustrated in Figure 2.9.

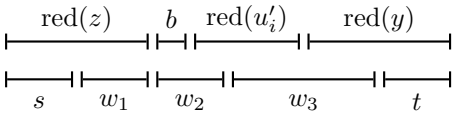


Figure 2.9: The two decompositions of $\overline{u}$.

Let $\mathrm{red}(w_2)$ be a max-reduced form of w_2 such that $\mathrm{red}(w_2)$ is non-empty (possible since w_2 is non-empty itself) and $s\,\mathrm{red}(w_2)t$ is still compatible. By Lemma 2.20 on $\overline{u} = sw_1w_2w_3t$, we know that st and sw_2t both belong to L and therefore, that $s\,\mathrm{red}(w_2)t \in L$ too.

- By definition, $\mathrm{red}(w_2)$ is non-empty so $st \neq s\,\mathrm{red}(w_2)t$.
- Since $\mathrm{red}(w_2)$ uses only letters that w_2 uses and w_2 is a factor of $b\,\mathrm{red}(u'_i) \in (A \setminus \mathcal{C})^*$, it has an empty signature. Therefore, $p_{\mathcal{C}}(st) = p_{\mathcal{C}}(s\,\mathrm{red}(w_2)t)$.
- By definition, $\mathrm{red}(w_2)$ cannot contain a removable factor of $s\,\mathrm{red}(w_2)t$ unless it is removable itself. We also know that a removable factor of $\mathrm{red}(z)$ is necessarily a suffix and that s is a strict prefix of $\mathrm{red}(z)$, which means that s does not contain a removable factor. Finally, we know that a removable factor of $\mathrm{red}(y)$ is necessarily a prefix and since t is a strict suffix of $\mathrm{red}(y)$, it cannot contain one. This means that $s\,\mathrm{red}(w_2)t$ can contain at most two disjoint removable factors (one starting in s and finishing in w_2 and one starting in w_2 and finishing in t). Similarly, st can contain at most one removable factor. By Lemma 2.20, this means that st and $s\,\mathrm{red}(w_2)t$ belong to every reduction $\overline{L}$ of L.

Like in the proof of Theorem 2.22, we proved that for all FTG-reachable languages L on an alphabet A, for all subalphabets $\mathcal{C} \subseteq A$, if $\mathcal{C}$ does not separate L, it does not separate any restriction $\overline{L}$ of L either. The contrapositive of this result is our theorem. □

Hence, the method we described at the end of Section 2.5 to solve complete traffic monitoring also applies on FTGs: given an instance of FTG-reachable separation, we use the construction described in the proof of Proposition 2.27 to build an automaton that recognises the associated language, we determine a regular expression of this language, restrict it and use the tools developed in Section 2.4 to solve the problem on the resulting language that is finite.

2.7 Conclusion

In this chapter, we studied the problem of traffic monitoring from the point of view of separating codes. To overcome the limitations of this approach (as described in Subsection 2.2.2), we introduced a new model of separation based on languages

and addressed the traffic monitoring with tools stemming from language theory. The problem of separation on a language being NP-complete in the size of the language, we outlined three subproblems relevant in practice, namely finite, complete and restricted traffic monitoring, and we outlined algorithms to solve each of these subproblems, even if the set of walks to separate is infinite. The strength and flexibility of our model enables us to address the case of non-acyclic graphs, infinite sets of roads to separate and even to impose additional constraints on the sets we separate such as avoiding certain transitions. The expressiveness of our new model of separation on a language and the limitations it overcomes also offer hope that it could be of help in a much wider range of applications than traffic monitoring alone.

Of course, this study also opens the door to many possibilities for improvement and raises open problems. We present some of them in Section 7.1.

Chapter 3

Minimum connecting transition sets in graphs

This chapter is based on [10] which is joint work with Benjamin Bergougnoux.

Contents

3.1 Introduction

As discussed in Subsection 2.6.1, forbidding transitions in graphs allow to express stronger constraints on the set of possible walks than what we can do with the standard graph definitions. This model is suitable to solve routing problems in many practical cases including optical networks, road networks or public transit systems among others. We also highlighted in Subsection 1.3.2 that the model of forbidden-transition graph generalizes other models such as edge-coloured graphs.

Another use of forbidden transitions is to measure the robustness of graph properties. In [105], Sudakov studied the Hamiltonicity of a graph with the idea that even an Hamiltonian graph can be more or less strongly Hamiltonian (an Hamiltonian graph is a graph in which there exists an elementary cycle that uses all the vertices). The number of transitions one needs to forbid for a graph to lose its Hamiltonicity gives a measure of its robustness: if the smallest set of forbidden transitions that makes a graph lose its Hamiltonicity has size 4, this means that this graph can hold the failure of three transitions, no matter where the failures happen.

In this chapter, the notion we are interested in is not Hamiltonicity but connectivity: the possibility to go from any vertex to any other, which is probably one of the most important properties we expect from any telecommunication or transport network. However, our work differs from others in that we are not looking for the minimum number of transitions to forbid to disconnect the graph but for the minimum number of transitions to allow to keep the graph connected. Our problem

can be seen as an equivalent of minimum spanning trees for transitions. Indeed, a minimum spanning trees (see Definition 1.40) is a minimum set of edges that keeps a graph connected.

In terms of robustness, we are looking for the maximum number of transitions that can fail without disconnecting the graph, provided we get to choose which transitions still work. This does not provide a valid measure of the robustness of the network but measuring the robustness is only one part (the definition of the objective function) of the problem of robust network design. In most practical situations, robustness is achievable but comes at a cost and the optimization problem consists of creating a network as robust as possible for the minimum cost. In this respect, it makes sense to be able to choose where the failure are less likely to happen. Our problem highlights which transitions are the most important for the proper functioning of the network and this is where special attention must be paid in its design or maintenance. As long as those transitions work, connectivity is assured.

We also would like to point out that in practice, unusable transitions are not always the result of a malfunction. Consider a train network and imagine that there is a train going from a town A to a town B and one going from the town B to a town C. In the associated graph, there is an edge from A to B and one from B to C but if the second train leaves before the first one arrives, the transition is not usable and this kind of situation is clearly unavoidable in practice even if no special problem happens. Highlighting the most important transitions in the network thus helps design the schedule, even before the question of robustness arises.

Unlike Hamiltonicity (proved NP-complete by Karp in [68]) or the existence of an elementary path between two vertices (polynomial in usual graphs but NP-complete in FTGs [106] as discussed in Subsection 1.3.2), testing the connectivity is an easy task to perform even on graphs with forbidden transitions (note that a walk connecting two vertices does not have to be elementary). However, we prove that the problem of determining the smallest set of transitions that maintains the connectivity of a given graph is NP-hard even on co-planar graphs, which is the main contribution of this chapter (Theorem 3.19 and 3.22). We also establish a $O(|V|^2)$-time $\frac{3}{2}$-approximation (Theorem 3.16) and a reformulation of the problem (Theorem 3.11) which is of great help in the proofs and could hopefully be useful again in subsequent works.

More formally, the problem we study in this chapter is defined as follows:

Definition 3.1. Connecting transition set:

Let $G = (V, E)$ be a graph and T be a set of transition of G. The graph G is *T-connected* and T is a *connecting transition set of G* if and only if for all vertices u and v of G, there exists a T-compatible walk between u and v.

Minimum Connecting Transition Set (MCTS)
Input: A connected undirected graph G.
Output: A minimum connecting transition set of G.

All the graphs we consider in this chapter are simple and undirected.

3.2 Polynomial algorithms and structural results

In this section, we only consider graphs with at least 2 vertices; our problem is trivial otherwise.

3.2.1 General bounds

We start by studying the specific case of trees, which are the smallest connected graphs.

Lemma 3.2.
If G is a tree then a minimum connecting transition set of G has size $|V(G)|-2$.

Proof. The proof is in two steps:

• We first prove that $|V(G)|-2$ transitions are enough to connect G.

For every vertex v of G, we pick a neighbour of v that we call $f(v)$. For every neighbour $u \neq f(v)$ of v, we allow the transition $uvf(v)$. We end up with the transition set $T = \{uvf(v) : v \in V(G), u \in N(v) \setminus \{f(v)\}\}$.

Let u and v be vertices of G. Since G is connected, there exists a walk $(u, u_1, u_2, \ldots, u_k, v)$ between u and v. The walk $(u, u_1, f(u_1), u_1, u_2, f(u_2), u_2, \ldots, u_k, f(u_k), u_k, v)$ is T-compatible and still leads from u to v. This proves that G is T-connected.

The size of T is $|T| = \sum_{v \in V(G)} (d(v)-1) = 2|E(G)| - |V(G)|$. Since G is a tree, $|E(G)| = |V(G)| - 1$ and thus, $|T| = |V(G)| - 2$.

• Let us now prove by induction on the number n of vertices of G that at least $n-2$ transitions are necessary to connect G.

This is obvious for $n = 2$. Let us assume that it holds for n and let G be a tree with $n+1$ vertices.

Let T be a minimum connecting transition set of G. Let uv be an internal edge of T if any (*i.e.* an edge such that u and v are not leaves). Let a and b be two vertices from different connected components of $G - \{u, v\}$. Every walk leading from a to b in G therefore uses the edge uv and thus, two transitions containing uv. This proves that every internal edge of T belongs to at least two transitions of T.

If every edge of G belongs to at least two transitions of T, T has size at least $|E(G)| = |V(G)| - 1$ which concludes the proof. Otherwise, let uv be an edge that belongs to at most one transition of T. This means that one of its vertices, say v, is a leaf. It is straightforward to check that uv must belong to one transition of T, otherwise G would not be T-connected.

Let t be the transition in T containing uv. The graph $G - v$ is $T \setminus \{t\}$-connected and is a tree. By the induction hypothesis, this means that $|T \setminus \{t\}| \geqslant n-3$ and $|T| \geqslant n-2$. This concludes the proof of the lemma. □

Let us also note that a linear-time algorithm to compute an optimal solution can be easily deduced from this proof.

Since every connected graph contains a spanning tree (Definition 1.40), we deduce the following upper bound on general graphs.

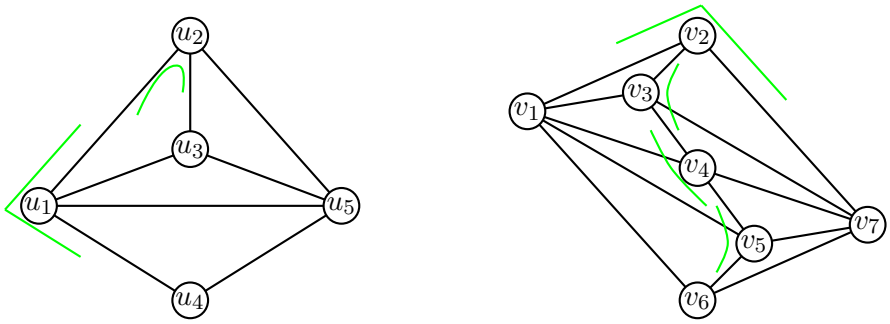
Proposition 3.3.
Every connected graph G has a connecting transition set of size $|V(G)| - 2$.

Note however that in the general case, this bound is far from tight. The most extreme case is the complete graph where every vertex can be connected to every other with a walk of one edge, that therefore uses no transition. Thus, the empty set is a connecting transition set of the complete graph. More generally, for every k$\in [\![0, n-2]\!]$, there exists a graph G of n vertices such that a minimum connecting transition sets on G has size k. Such a graph can be built from a tree of $k+2$ vertices by adding $n-k-2$ vertices connected to every other vertex of the graph (including the vertices of the tree).

While graphs with dominating vertices (*i.e.* vertices connected to every vertex of the graph) are the most obvious counter-examples, there are many other cases where a graph G has connecting transition sets smaller than $|V(G)| - 2$. It may suggest that the size of a minimum connecting transition set of a graph with k non-dominating vertices is $k-2$ but we show that this is neither a lower bound nor an upper bound.

Indeed, the graph G depicted in Figure 3.1a has 3 non-dominating vertices (u_2,u_3 and u_4) but we need 2 transitions to connect it (for example, $u_3u_2u_1$ and $u_2u_1u_4$, as depicted in green in the figure). Here, the non-dominating vertices do not induce a connected graph and one transition is not enough to connect them.

Conversely, the graph H depicted in Figure 3.1b has no dominating vertex and therefore 7 non-dominating vertices but can be connected with only 4 transitions (for example, $v_2v_3v_4$, $v_3v_4v_5$, $v_4v_5v_6$ and $v_1v_2v_7$). Indeed, the vertices v_1 and v_7, while not dominating, are connected to every vertex of the graph except each other. We need three transitions to connect the other vertices of the graph and only one more to connect both of them.



(a) A graph G.

(b) A graph H.

Figure 3.1

The key to understand what is going on here is to look at the connected components of the complementary graph.

Definition 3.4. Co-connected components of a graph
The *co-connected components* (or *co-cc*) of a graph G are the connected components (or *cc*) of its complementary graph $\overline{G}$.

The following result aims at tightening the upper bound on the size of a minimum connecting transition set of a graph.

Theorem 3.5.
Every connected graph G has a connecting transition set of size $\tau(G)$ where

$$\tau(G) = \sum_{\substack{C \text{ co-cc of } G \\ |C| \geqslant 2}} \begin{cases} |C| - 2 \text{ if } G[C] \text{ is connected} \\ |C| - 1 \text{ otherwise} \end{cases}$$

Proof. By definition, if u and v belong to different co-connected components of G, there is an edge $uv \in E(G)$ and there is therefore a walk between u and v that is compatible with any transition set. Thus, we only have to find a transition set that connects all the vertices that belong to the same co-connected component.

Let C be a co-connected component of G with at least 2 vertices. If $G[C]$ is connected, Proposition 3.3 provides a transition set of size $|C| - 2$ that connects C. Otherwise, since G is connected, we know that $V(G) \neq C$ and there exists a vertex $v \notin C$. Hence, v is adjacent to every vertex of C and $C \cup \{v\}$ induces a connected subgraph of G. Proposition 3.3 provides a set of size $|C \cup \{v\}| - 2 = |C| - 1$ that connects C. By iterating this on every C, we build a connecting transition set T of size $\tau(G)$. □

Example 3.6.
Looking back at Figure 3.1a, the co-connected components of the graph G are respectively $\{u_1\}$, $\{u_2, u_3, u_4\}$ and $\{u_5\}$. We need no transition to connect the dominating vertices to every other vertex but we need transitions to connect the vertices of $\{u_2, u_3, u_4\}$. However, they do not induce a connected graph and no transition can connect u_2 and u_4 using only those vertices. A solution is to add the vertex u_1 and to look for a connecting transition set on the subgraph induced by $\{u_1, u_2, u_3, u_4\}$. By Proposition 3.3, we know that there exists one of size 2, and $\{u_1u_2u_3, u_2u_1u_4\}$ is an example of such set.

The fact that the graph H depicted in Figure 3.1b has no dominating vertex only means that it has no co-connected component of size 1 but it can still have several co-connected components. Indeed, its co-connected components are $\{v_1, v_7\}$ and $\{v_2, v_3, v_4, v_5, v_6\}$. The second one induces a connected subgraph and Proposition 3.3 ensures that it can be connected with 3 transitions, for example $\{v_2v_3v_4, v_3v_4v_5, v_4v_5v_6\}$. The component $\{v_1, v_7\}$ does not induce a connected graph and we need a vertex from another component to connect them. For example, we can look for a connecting transition set on the graph induced by $\{v_1, v_2, v_7\}$ and we find $\{v_1v_2v_7\}$.

Notice that this bound can be computed in $O(|V(G)|^2)$. However, we show in the next section that the bound provided by Theorem 3.5 is not tight either.

3.2.2 Connecting hypergraphs

This subsection presents a reformulation of MCTS which is very useful in the subsequent proofs.

We first observe that the bound provided in Theorem 3.5 is not tight.

Example 3.7.

Let us consider the graph $\overline{P_7}$ whose vertex set is $\{v_1, \ldots, v_7\}$ and where every vertex v_i, $2 \leqslant i \leqslant 6$ is connected to every vertex of the graph except v_{i-1} and v_{i+1}. Since the graph is connected and co-connected, $\tau(\overline{P_7}) = 5$ but the set $T = \{v_3v_1v_4, v_2v_4v_1, v_6v_4v_7, v_5v_7v_4\}$ is a connecting transition set of size only 4. To better understand this solution, let us consider the spanning tree of $\overline{P_7}$ depicted in Figure 3.2:

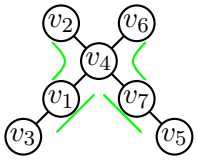


Figure 3.2: A spanning tree of $\overline{P_7}$ and a connecting transition set (depicted in green).

Note that the set T described above does not connect this spanning tree. Indeed, one cannot go from v_1, v_2 or v_3 to v_5, v_6 or v_7 using a T-compatible walk in the tree. However, these vertices are already connected to each other by edges that do not belong to the spanning tree. The optimal solution here does not consist of connecting a spanning tree of G but in connecting a spanning tree of $G[\{v_1, v_2, v_3, v_4\}]$ and one of $G[\{v_4, v_5, v_6, v_7\}]$ and the cost is $(4-2)+(4-2) = 4$ instead of $7-2 = 5$.

In fact, we prove that to each optimal connecting transition set T of a graph G corresponds an unique decomposition of G into subgraphs $G_1, G_2, \ldots, G_k$ such that T is the disjoint union of $T_1, T_2, \ldots, T_k$, where each T_i is the connecting transition set of some spanning tree of G_i. Observe that the size of T is uniquely determined by its associated decomposition, *i.e.* , $|T| = |V(G_1)| - 2 + \cdots + |V(G_k)| - 2$. Hence, finding an optimal connecting transition set is equivalent to finding its associated decomposition. In the following, we reformulate **MCTS** into this problem of graph decomposition which is easier to work with.

Definition 3.8. Connecting Hypergraph

Let G be a graph. A *connecting hypergraph* of G is a set H of subsets of $V(G)$ called *connecting hyperedges*, such that

- For all $E \in H$, $|E| \geqslant 2$.
- For all $E \in H$, $G[E]$ is connected.
- For all $uv \notin E(G)$, there exists $E \in H$ such that $u, v \in E$ (we say that the hyperedge E connects u and v).

We define the problem of *optimal connecting hypergraph* as follows:

Optimal Connecting HyperGraph (OCHG)

Input: A connected graph G.

Output: A connecting hypergraph H that minimizes $\text{cost}(H) = \sum_{E \in H} (|E| - 2)$.

We now prove that **OCHG** is a reformulation of **MCTS** (Theorem 3.11).

Proposition 3.9. *Let G be a graph and let H be a connecting hypergraph of G. There exists a connecting transition set T of size at most* $\mathrm{cost}(H)$.

Proof. By the definition of a connecting hypergraph, for all i, E_i induces a connected graph and by Proposition 3.3, there exists a subset of transitions T_i of size $|E_i| - 2$ such that $G[E_i]$ is T_i-connected. Let $T = \bigcup_{i \leqslant k} T_i$. By definition, for all $uv \notin E(G)$, there exists i such that $u, v \in E_i$. Since $G[E_i]$ is T_i-connected and $T_i \subseteq T$, there is a T-compatible walk between u and v in G which means that G is T-connected. Since $T = \bigcup_{i \leqslant k} T_i$, $|T| \leqslant \sum_{i \leqslant k} |T_i| = \sum_{i \leqslant k} (|E_i| - 2) = \mathrm{cost}(H)$ and we can use the construction given in the proof of Lemma 3.2 to build T in $O(|V(G)| + |E(G)|)$. □

Proposition 3.10. *Let T be a connecting transition set of G. There exists a connecting hypergraph $H = \{E_1, \ldots, E_k\}$ of cost at most $|T|$.*

Proof. Let $\sim$ be the relation on T such that $t \sim t'$ if and only if t and t' share at least one common edge. We denote by $\mathcal{R}$ the transitive closure of $\sim$. Let $T_1, \ldots, T_k$ be the equivalence classes of $\mathcal{R}$. For all $i \leqslant k$, we denote by E_i the set of vertices induced by T_i. We claim that the hypergraph $\{E_1, \ldots, E_k\}$ is a connecting hypergraph and that for all i, $|T_i| \geqslant |E_i| - 2$.

By construction, for all i, we have $|E_i| \geqslant 3$ since T_i contains at least one transition and thus, three vertices. Furthermore, since G is T-connected, there exists a T-compatible walk W between every pair $uv \notin E(G)$. All the transitions that W uses must be in T and are pairwise equivalent for $\mathcal{R}$. Thus, for all $uv \notin E(G)$, there exists i such that both u and v belong to E_i.

It remains to prove that for all i, $|E_i| - 2 \leqslant |T_i|$. We prove by induction on n that every set T of n pairwise equivalent transitions induces a vertex set of size at most $n + 2$. This property trivially holds for $n = 1$. Now, suppose that it is true for sets of size n and let T be a set of pairwise equivalent transitions of size $n + 1$. Let $P = t_1, \ldots, t_r$ be a maximal sequence of distinct transitions of T such that, for all $i \leqslant r - 1$, $t_i \sim t_{i+1}$. One can check that all the transitions of $T \setminus \{t_1\}$ are still pairwise equivalent (otherwise, P would not be maximal). By the induction hypothesis, $T \setminus \{t_1\}$ induces at most $n+2$ vertices. Since t_1 shares an edge (and thus at least 2 vertices) with t_2, it induces at most one vertex not induced by $T \setminus \{t_1\}$. Thus T induces at most $n + 3$ vertices. □

The next theorem follows directly from Propositions 3.9 and 3.10.

Theorem 3.11.
Let G be a graph.

- *The size of a minimum connecting transition set of G is equal to the cost of an optimal connecting hypergraph.*
- *A solution of one of these problems on G can be deduced in polynomial time from a solution of the other.*

Let us note that the bound provided in Theorem 3.5 suggests a $O(|V|^2)$-time heuristic for **OCHG** which consists of building the set H as follows:

$$H = \bigcup_{\substack{C \text{ co-cc of } G \\ |C| \geqslant 2}} \begin{cases} C \text{ if } G[C] \text{ is connected} \\ C \cup \{v\} \text{ with } v \notin C \text{ otherwise} \end{cases}$$

Example 3.12.

Let G be the graph depicted in Figure 3.1a (see Example 3.6 for a description of its co-connected components). The aforementioned heuristic provides the connecting hypergraph $\{\{u_2, u_3, u_4, u_1\}\}$ of cost 2. Since G achieves the bound provided by Theorem 3.5, this connecting hypergraph is optimal.

On the graph H depicted in Figure 3.1b, the heuristic provides the connecting hypergraph $\{\{v_1, v_7, v_2\}, \{v_2, v_3, v_4, v_5, v_6\}\}$ of cost 4, which is optimal by the same argument.

On the graph $\overline{P_7}$ depicted in Figure 3.2, the heuristic provides the connecting hypergraph $\{\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}\}$ of cost 5 while the connecting hypergraph $\{\{v_1, v_2, v_3, v_4\}, \{v_4, v_5, v_6, v_7\}\}$ has cost only 4.

Let G be a connected graph and $v \in V(G)$. If $G - v$ is not connected, we say that v is a *cut vertex* of G. Note that if G is a tree, every vertex of G that is not a leaf is a cut vertex.

We use the reformulation given by Theorem 3.11 to generalize Lemma 3.2.

Proposition 3.13. *If G has a cut vertex, then a minimum connecting transition set of G has size $|V(G)| - 2$.*

Proof. By Theorem 3.11, it is sufficient to prove that $H = \{V(G)\}$ is an optimal connecting hypergraph of G. Let p be a cut vertex of G and $C_1, \dots, C_r$ be the connected components of $G - p$. Let $H = \{E_1, \dots, E_k\}$ be an optimal connecting hypergraph of G.

Let $a \in C_1$. Suppose that there are two vertices $b, c \neq p$ that do not belong to C_1. Hence, $\{a, b\} \notin E(G)$ and there exists i such that $a, b \in E_i$. Since E_i must induce a connected subgraph of G, we know that $p \in E_i$. Similarly, we know that there exists E_j that contains a, c and p. Thus, $|E_i \cap E_j| \geqslant |\{a, p\}| \geqslant 2$ and $\text{cost}(\{E_i \cup E_j\}) = |E_i \cup E_j| - 2 \leqslant |E_i| - 2 + |E_j| - 2 = \text{cost}(\{E_i, E_j\})$.

Thus, $H \setminus \{E_i, E_j\} \cup \{E_i \cup E_j\}$ is also an optimal connecting hypergraph where the same hyperedge contains both b and c. By iterating this process, we prove that there is an optimal connecting hypergraph with one hyperedge E that contains a, p and $C_2, \dots, C_r$. This result trivially holds if there is only one vertex $b \neq p$ that does not belong to C_1.

By iterating the previous process on this hypergraph with a vertex in $E \cap C_2$ instead of a, we end up with the optimal connecting hypergraph $\{V(G)\}$ whose cost is $n - 2$. □

We now investigate in which case there exists an optimal connecting hypergraph whose hyperedges all induce co-connected graphs. Such hyperedges are easier to work with and this lemma helps us prove that **MCTS** is $\frac{3}{2}$-approximable and NP-hard.

Lemma 3.14.

Let G be a connected graph. If G is co-connected or G has a dominating vertex x and $G - x$ is connected and co-connected, then there exists an optimal connecting hypergraph $H = \{E_1, \ldots, E_k\}$ on G such that for all i, $G[E_i]$ is co-connected.

Proof. To facilitate the understanding, the construction we use in the proof is illustrated in Figure 3.3.

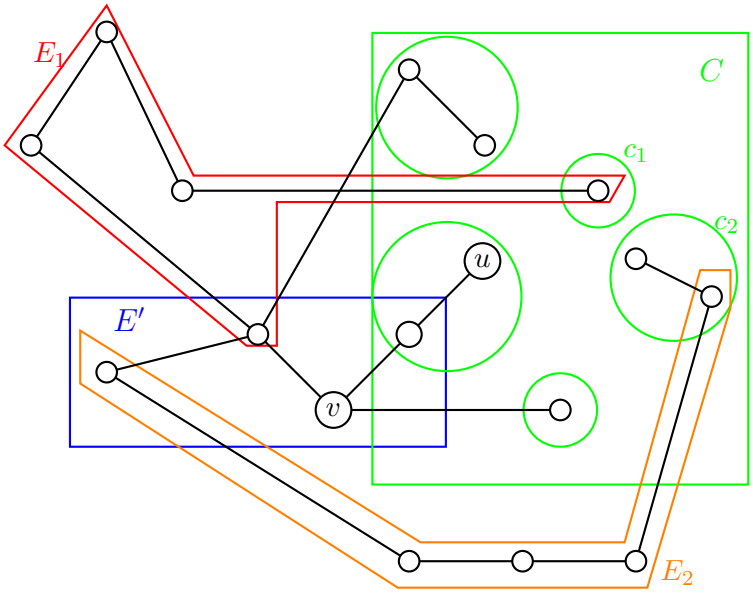


Figure 3.3: Here, $G[C]$ has five connected components (circled in green), two of which (c_1 and c_2) are not connected to E'. To ensure that $G[\mathscr{E}]$ is connected, we need the hyperedges E_1 and E_2.

Let H be an optimal connecting hypergraph on G and let E be a hyperedge of H such that $G[E]$ is not co-connected. If E is a clique, then E does not connect any pair of non-adjacent vertices and $H \setminus E$ is still a connecting hypergraph whose cost is less or equal than the cost of H. Otherwise, let a and b be two non-adjacent vertices of E. They therefore belong to the same co-connected component C of $G[E]$.

By way of contradiction, suppose that C is a co-connected component of G. Since $C \subsetneq E \subsetneq V(G)$, we know that G is not co-connected. By hypothesis, it therefore has a dominating vertex x and $C = V(G) \setminus \{x\}$. Thus, $E = V(G)$. Hence, $\text{cost}(H) \geqslant |V(G)| - 2$ which is absurd since $\{V(G) \setminus \{x\}\}$ is a connecting hypergraph of cost $|V(G)| - 3$.

Thus, C is not a co-connected component of G, which means that there exist $u \in C$ and $v \notin C$ such that u and v are not adjacent. Since C is a co-connected component of $G[E]$, we also know that $v \notin E$. Hence, there exists $E' \neq E$ in H that contains u and v. Let $c_1, \ldots, c_l$ be the connected components of $G[C]$ that are not connected to any vertex of E' (if any). By definition of connecting hypergraph, we know that for all $i \leqslant l$, there exists $E_i \in H$ that connects a vertex of E' and a vertex of c_i. We create H' from H by replacing E by $E \setminus C$ and by replacing $E_1, \ldots, E_l$ and E' by $\mathscr{E} = E' \cup C \cup E_1 \cup \cdots \cup E_l$.

We claim that $cost(H') \leqslant cost(H) - 1$. Indeed, replacing E by $E \setminus C$ decreases the cost by $|C|$ while replacing E' by $E' \cup C$ increases the cost of at most $|C| - 1$ because $E' \cap C$ contains at least the vertex u. Moreover, we can prove by induction on $i \leqslant l$ that the cost of $\{E' \cup C \cup E_1 \cup \cdots \cup E_{i-1}, E_i\}$ is greater or equal than the cost of $\{E' \cup C \cup E_1 \cup \cdots \cup E_{i-1} \cup E_i\}$. Indeed, $(E' \cup C \cup E_1 \cup \cdots \cup E_{i-1}) \cap E_i$ has size at least two (it contains at least one vertex in E' and one in $c_i \subset C$, by definition of E_i). It follows that $\text{cost}(\mathscr{E}) \leqslant \text{cost}(\{E' \cup C, E_1, \ldots, E_l\})$. Therefore, $\text{cost}(H') \leqslant \text{cost}(H) - 1$. Since H is an optimal connecting hypergraph and H' has a smaller cost, we know that H' is not a connecting hypergraph. But observe that H' satisfies the following properties:

- Every pair of non-adjacent vertices is still connected by a hyperedge of H'. Indeed, if two non-adjacent vertices are connected by E in H they are connected by $E \setminus C$ or $\mathscr{E}$ in H' depending on whether they belong to C or not; if they are connected by an E_i in H, they are connected by $\mathscr{E}$ in H' and otherwise, the hyperedge that connects them in H belongs to H' too.
- The graph $G[\mathscr{E}]$ is connected. Indeed, the sets $E', E_1, \ldots, E_l$ all induce connected subgraphs of G by definition and are connected to each other because for all i, $E_i \cap E' \neq \varnothing$. Furthermore, all the connected components of C are connected to a vertex of E', except the c_i which are by definition connected to the E_i.

Thus, either $|E \setminus C| < 2$ or $G[E \setminus C]$ is not connected. If $E \setminus C$ is a singleton, it does not connect any pair of non-adjacent vertices. Thus, $H' \setminus \{E \setminus C\}$ is a connecting hypergraph whose cost is strictly smaller than H, which is absurd. Hence, $G[E \setminus C]$ is not connected, which means it is co-connected. We can therefore apply to $E \setminus C$ the same method we used on C.

Just like before, we know that there exist two non-adjacent vertices $u' \in E \setminus C$ and $v' \notin E$. Let $F \in H'$ be the hyperedge that connects u' and v', let $d_1, \ldots, d_{l'}$ be the connected components of $E \setminus C$ that are not connected to F and let $F_1, \cdots, F_{l'}$ be hyperedges of H' such that F_i connects a vertex of F to a vertex of d_i. We create H'' from H' by removing $E \setminus C$ and by replacing $F_1, \ldots, F_{l'}$ and F by $\mathscr{F} = F \cup (E \setminus C) \cup F_1 \cup \cdots \cup F_{l'}$.

With the same arguments used for H', we can prove that $\mathscr{F}$ is connected and that H'' connects every pair of non-adjacent vertices, which means that H'' is a connecting hypergraph. Moreover, with these arguments, we can also prove that $\text{cost}(\mathscr{F}) \leqslant \text{cost}(\{F \cup (E \setminus C), F_1, \ldots, F_{l'}\})$. Furthermore, removing $E \setminus C$ from H' decreases the cost by $|E \setminus C| - 2$ and replacing F by $F \cup (E \setminus C)$ increases it by at most $|E \setminus C| - 1$ since $F \cap (E \setminus C)$ contains at least the vertex u'. Hence, $\text{cost}(H'') \leqslant \text{cost}(H') + 1 \leqslant \text{cost}(H)$. Thus, H'' is an optimal covering hypergraph. Observe that H'' has strictly fewer hyperedges E such that $G[E]$ is not co-connected than H. We prove the lemma by iterating this process. □

Even if G satisfies the condition of the lemma, there may still exist an optimal connecting hypergraph H of G and $E \in H$ such that $G[E]$ is not co-connected. In this case, the proof of the lemma implies that $G[E]$ contains two co-connected components C_1 and C_2 and that neither $G[C_1]$ nor $G[C_2]$ is connected.

3.2.3 Polynomial approximation

We prove in this subsection that the bound provided by Theorem 3.5 is actually a $\frac{3}{2}$-approximation of the size of a minimum connecting transition set and that this bound is tight. This bound, as well as a solution that achieves it, can be computed in $O(|V|^2)$.

We first study the case of graphs that are connected and co-connected.

Proposition 3.15. *Let G be a graph of n vertices. If G is connected and co-connected, then for every connecting hypergraph H of G, $\mathrm{cost}(H) \geqslant \frac{2(n-1)}{3}$.*

Proof. Let G be connected and co-connected. We know by Lemma 3.14 that there exists an optimal connecting hypergraph $H = \{E_1, \ldots, E_k\}$ of G such that for all i, $G[E_i]$ is co-connected.

First, observe that $\mathrm{cost}(H) \geqslant 2k$. Indeed, for every i, $G[E_i]$ is both connected and co-connected, which is only possible if $|E_i| \geqslant 4$. As $\mathrm{cost}(H) = \sum_{i \leqslant k} |E_i| - 2$, we deduce that $\mathrm{cost}(H) \geqslant 2k$.

Now, we prove that $\mathrm{cost}(H) \geqslant n - k - 1$. Observe that since $\overline{G}$ is connected, every vertex v belongs to at least one edge in $\overline{G}$. Hence, by definition of connecting hypergraph, there exists $E \in H$ such that $v \in E$ and $\bigcup_{i \leqslant k} E_i = V(G)$.

Also note that for all $i < k$, there exists a hyperedge E_j with $j > i$ that shares a vertex with a hyperedge of $E_1, \ldots, E_i$. Otherwise, $\bigcup_{j \leqslant i} E_j$ and $\bigcup_{j > i} E_j$ cover the vertices of G and since G is co-connected, this means that there exist $u \in \bigcup_{j \leqslant i} E_j$ and $v \in \bigcup_{j > i} E_j$ such that $(u, v) \notin E(G)$ but no set of H connects them, which is impossible. We can assume without loss of generality that this hyperedge that shares at least one vertex with $\bigcup_{j \leqslant i} E_j$ is E_{i+1}.

It is now immediate to prove by induction on $i \leqslant k$ that
$\sum_{j \leqslant i} |E_j| - 2 \geqslant |\bigcup_{j \leqslant i} E_j| - i - 1$. Thus, $\mathrm{cost}(H) \geqslant n - k - 1$.

By combining the two inequalities, we find that $\mathrm{cost}(H) \geqslant \frac{2(n-1)}{3}$. □

We now prove the theorem in the general case.

Theorem 3.16.
For every connected graph G and optimal connecting transition set T of G, the size of T is at least $\frac{2}{3}\tau(G)$, where $\tau(G)$ is the function defined in Theorem 3.5.

Proof. By Theorem 3.11, it is enough to prove that an optimal connecting hypergraph has cost at least $\frac{2}{3}\tau(G)$. We already know that the theorem holds if G is co-connected.

Let $H = \{E_1, \ldots, E_k\}$ be an optimal connecting hypergraph of G, let $C_1, \ldots, C_l$ be the co-connected components of G and for all $j \leqslant l$, let v_j be a vertex that does not belong to C_j.

For all $i \leqslant k$ and $j \leqslant l$ such that $|E_i \cap C_j| \geqslant 2$, we define

$$E_{i,j} = \begin{cases} E_i \cap C_j & \text{if } G[E_i \cap C_j] \text{ is connected} \\ E_i \cap C_j \cup \{v_j\} & \text{otherwise} \end{cases}$$

and we define F as the union of the $\{E_{i,j}\}$. Note that if G is co-connected, there is only one co-connected component $C_1 = V(G)$ and while there is no vertex $v_1 \notin C_1$, for all i, $G[E_i \cap C_1] = G[E_i]$ is connected by definition, so we do not need v_1 in the above construction.

Since v_j dominates C_j, it is easy to check that every hyperedge of F is connected and has size at least 2. Furthermore, any two non-adjacent vertices of G belong to the same C_j and are connected by an $E_i \in H$. Therefore, they belong to $E_i \cap C_j$ which has size at least two and thus belongs to F. Hence, F is a connecting hypergraph.

Let $E_i \in H$ and let S_i be the set of values of j such that $E_{i,j}$ exists. If there is only one such value j, then $\text{cost}(E_i) \geqslant \text{cost}(E_{i,j}) = \text{cost}(\bigcup_{j\in S_i}\{E_{i,j}\})$ follows immediately. Otherwise

$$\text{cost}(\bigcup_{j\in S_i}\{E_{i,j}\}) \leqslant \sum_{j\in S_i}(|E_i \cap C_j| + |\{v_j\}| - 2) \leqslant |E_i| - |S_i| \leqslant |E_i| - 2 = \text{cost}(E_i)$$

still holds. Since $F = \bigcup_{i\leqslant k}\bigcup_{j\in S_i}\{E_{i,j}\}$, it follows that $\text{cost}(F) \leqslant \text{cost}(H)$ which proves that F is optimal.

We know that two non-adjacent vertices necessarily belong to the same co-connected component of G and since a hyperedge $E_{i,j}$ only contains one vertex that does not belong to C_j it only connects non-adjacent vertices of one connected component. For all j, let F_j be the set of hyperedges of F that connect non-adjacent vertices of C_j. Since $E_{i,j} \subseteq C_j \cup \{v_j\}$, F_j is a connecting hypergraph of $G[C_j \cup \{v_j\}]$.

Let C_j be a co-connected component of G and observe that:

- if C_j is not connected, then v_j is a cut vertex of $C_j \cup \{v_j\}$. Hence, by Proposition 3.13, $\text{cost}\,(F_j) \leqslant |C_j| - 1$.
- if C_j is connected, since it is co-connected by definition, $C_j \cup \{v_j\}$ admits by Lemma 3.14 an optimal connecting hypergraph H_j such that for every hyperedge E of H_j, $G[E]$ is co-connected and thus, $v_j \notin E$. This proves that H_j is a connecting hypergraph of $G[C_j]$. As $G[C_j]$ is connected and co-connected, we know by the above claim that $\text{cost}(H_j) \geqslant \frac{2(|C_j|-1)}{3}$.

Thus,

$$\begin{aligned}
\text{cost}(F) = \sum_{j\leqslant l}\text{cost}(F_j) &\geqslant \sum_{j\leqslant l}\begin{cases} |C_j| - 1 & \text{if } C_j \text{ is not connected} \\ \dfrac{2(|C_j|-2)}{3} & \text{otherwise} \end{cases} \\
&\geqslant \frac{2}{3}\sum_{j\leqslant l}\begin{cases} |C_j| - 1 \text{ if } C_j \text{ is not connected} \\ |C_j| - 2 \text{ otherwise} \end{cases} \\
&\geqslant \frac{2}{3}\tau(G)
\end{aligned}$$

□

The bound given in Theorem 3.16 is tight. Indeed, let G be the complement of a star of n branches of 3 edges. The graph G has $3n+1$ vertices that we call c, $v_{i,1}$, $v_{i,2}$ and $v_{i,3}$ with $1 \leqslant i \leqslant n$ (in $\overline{G}$, c is the center of the star and $v_{i,1}$, $v_{i,2}$ and $v_{i,3}$ are

the three vertices of the branch i). Every vertex of G is connected to every other except c and $v_{i,1}$, $v_{i,1}$ and $v_{i,2}$ and $v_{i,2}$ and $v_{i,3}$ with $1 \leqslant i \leqslant n$. The complement of the case $i = 6$ is depicted in Figure 3.4.

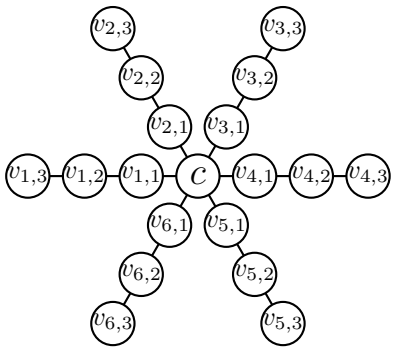


Figure 3.4: The star of 6 branches of 3 edges.

Since G is both connected and co-connected, our algorithm returns the connecting hypergraph $H_1 = \{V(G)\}$ whose cost is $3n - 1$ but the hypergraph H_2 whose hyperedges are the $\{c, v_{i,1}, v_{i,2}, v_{i,3}\}$ for $i \in [\![1, n]\!]$ is a connecting hypergrpah of cost $2n$. The case of co-P_7 that we studied in Example 3.7 to prove that the algorithm was not exact is the case $i = 2$.

3.3 NP-completeness

This section mainly proves the NP-completeness of **OCGH** and thus **MCTS**.

Subsection 3.3.1 discusses the generalization of the problem where the input graph is already a forbidden-transition graph (V, E, T) and we look for the minimum connecting transition set $T' \subset T$. We show that this problem is NP-complete by reducing the problem of finding elementary paths in FTGs. Section 3.3.2 is devoted to the proof of the NP-completeness of **MCTS** in usual graphs. We prove it by reducing from 3-SAT, and this subsection contains both the presentation of the gadget and the proof itself. Finally, Subsection 3.3.3 aims at explaining the intuition behind the previous proof. To do so, we highlight which properties of the gadgets were needed for the proof to work and present how the gadgets were designed.

3.3.1 MCTS in FTGs

We said in Section 3.1 that the problem of **MCTS** is helpful to highlight which transitions of a network are the most important to its connectivity and thus, which transitions we should strengthen to ensure the robustness of the network. However, we also pointed out that forbidden transitions are not always the consequence of a malfunction. In practical applications such as transit network scheduling, certain transitions cannot be made possible. Hence, we look for connecting transitions among a set of possible transitions which does not contain all the pair of adjacent edges of the graph. This subsection is dedicated to the following problem:

Minimum Connecting Transition Set in FTGs

Input: A connected undirected FTG $G = (V, E, T)$.
Output: A minimum transition set $T' \subset T$ such that of G is T' connected.

The vast majority of the results we proved in the previous section do not hold anymore. Indeed, most of them were based on Proposition 3.3, which states that a MCTS of a graph G has size at most $|V(G)| - 2$. We show that Proposition 3.3 does not hold anymore in FTGs.

Example 3.17.
Let G be the graph depicted in the Figure 3.5, where $T = \{v_{2i}uv_{2i+1} : i \in [\![0, k]\!]\} \cup \{uv_{2i+1}v_{2i+2} : i \in [\![0, k]\!]\} \cup \{v_{2i+1}v_{2i+2}u : i \in [\![0, k]\!]\}$. The only T-compatible walk leading from v_0 to v_{2k+1} is $(v_0, u, v_1, v_2, u, v_3, v_4, u, \dots u, v_{2k-1}, v_{2k}, u, v_{2k+1})$ which uses all the transitions of T. Therefore, no strict subset T' of T connects G and the only minimum connecting set in G has size $3k = \frac{3}{2}(|V(G| - 2)$.

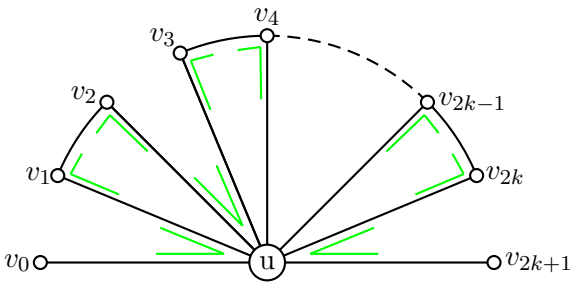


Figure 3.5: A counter-example to the generalization of Proposition 3.3 to FTGs.

Since FTGs are a generalization of usual graphs, the NP-completeness of MCTS in FTGs is an immediate consequence of the NP-completeness of MCTS in usual graphs, which we prove in the next section. However, while our proof of NP-completeness in usual graphs is quite complicated, we show in this subsection that the NP-completeness of MCTS in FTGs follows quickly from the NP-completeness of elementary paths in FTGs, which was established by Szeider in [106].

Theorem 3.18.
MCTS in FTGs is NP-complete.

Proof. The problem we reduce is the following:

Elementary path in FTGs

Input: A FTG $G = (V, E, T)$ and two vertices $s, t \in V$.
Output: A T-compatible elementary path in G leading from s to t.

Let $G = (V, E, T)$ be a FTG of n vertices and s and t be two vertices of G. We define the graph G_2 as follows:

- $V_2 = V \cup \{s'\} \cup \{w\}$;

- $E_2 = \{uv : u, v \in (V \cup \{w\})\} \cup \{s'w\} \cup \{s's\}$;
- $T_2 = T \cup \{s'wv : v \neq t\} \cup \{s'sv : sv \in E\}$.

Let us solve MCTS on G_2. We note that all the vertices are already connected to each other by an edge (a walk that does not use any transition) except s' which is only connected to s and w.

A T_2-compatible walk leaving from s' can go to w and then, to any vertex u of V except t, but after u, it cannot go anywhere, except back to w and then to s'. Hence, a walk W going from s' to t has to go from s' to s first. From s, it can then only go to a neighbour of s in G and must then go to t using only transitions of T_2 but the only transitions of T_2 it can use at this point belong to T too. Thus, a T_2 compatible walk from s' to t in G_2 is the concatenation of $s's$ and a T-compatible walk W in G from s to t. Hence, there is a T_2-compatible elementary walk from s' to t in G_2 if and only if there is a T-compatible walk from s to t in G.

Let $T' \subset T_2$ be a connecting transition set of G_2. In order to connect s' and t, T' must contain all the transitions of a walk W between s' and t. Let n_W be the number of distinct vertices that the walk W uses and m_W the length of the walk. We can assume that all the transitions that the walk uses are distinct. Indeed, if this is not the case, we can extract from W a compatible walk that only uses distinct transitions by removing everything between two occurrences of the same transition. The walk W therefore uses $m_W - 1$ transitions and T' has to contain them all. However, those transitions only connect s' to the vertices that are used by W and there are thus still $n + 1 - n_W$ vertices that are not connected to s'. We note that each transition can only connect s' to one new vertex and that one transition is always enough to connect s' to a new vertex u since we can always add $s'uw$. Hence, the size of our connecting transition set T' is $m_W - 1 + n + 1 - n_W$. The greatest value that n_W can achieve is $n_W = m_W + 1$ and requires W to be elementary.

Therefore, there exists a connecting transition set of size $n - 1$ in G_2 if and only if there exists a T-compatible elementary walk from s to t in G, which proves that MCTS in FTGs is NP-complete. □

An even stronger model to describe the practical situations where some transitions are harder to ensure than other is the weighted version of **MCTS** where every transition of the graph has a cost and we look for a connecting transition set of minimum weight.

Note that since the weighted version is stronger, Theorem 3.18 also implies the NP-completeness of this variant. Indeed, forbidden transitions can be modelled by giving them arbitrarily high weight and it is easy to check whether a weighted transition set uses a forbidden transition by looking at its cost.

3.3.2 MCTS in usual graphs

This subsection is devoted to the proof of the following theorem, which is stronger than Theorem 3.18:

Theorem 3.19.
OCGH *and* ***MCTS*** *are NP-complete.*

Those problems are in NP since the cost of a connecting hypergraph or the size of a transition set can be computed in linear time. It remains to prove that they are NP-hard.

The graph involved in this proof are very dense, which makes them hard to visualize. Hence, we prefer to work with their complementary graphs and we therefore prove the NP-hardness of the following problem that we call **co-OCHG**:

Definition 3.20. co-Connecting Hypergraph

Let G be a graph. A co-connecting hypergraph is a set of hyperedges $E_1, \dots, E_r \subseteq V(G)$ such that

- For all $i \leqslant r$, $|E_i| \geqslant 2$.
- For all $i \leqslant r$, $G[E_i]$ is co-connected.
- For all $uv \in E(G)$, there exists i such that $u, v \in E_i$ (we say that the hyperedge E_i covers the edge uv).

co-Optimal Connecting HyperGraph (co-OCHG)

Input: A co-connected graph G.

Output: A co-connecting hypergraph that minimizes $\text{cost}(H) = \sum_{E \in H} (|E| - 2)$.

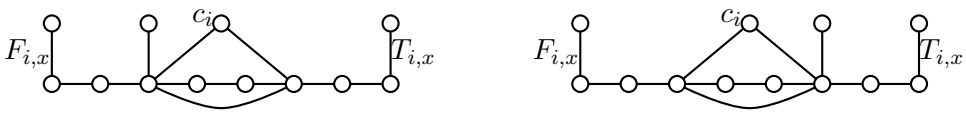
The problems of **OCHG** and **co-OCHG** are clearly polynomially equivalent since solving **OCHG** on a graph G is the same as solving **co-OCHG** on its complement $\overline{G}$.

We prove the NP-hardness of **co-OCHG** by reducing 3-SAT to it. Let $\mathscr{F}$ be a boolean formula in 3-SAT form, with n variables and m clauses. We may assume that each variable has positive and negative occurrences in $\mathscr{F}$. Indeed, if a variable x has no negative occurrence, we may set it to True and we create from $\mathscr{F}$ a formula $\mathscr{F}'$ by removing every clause that contains x. Determining if $\mathscr{F}$ is satisfiable is equivalent to determining if $\mathscr{F}'$ is.

We call $c_1, \dots, c_m$ the clauses of $\mathscr{F}$ and $x_1, \dots, x_n$ its variables. We are going to build from $\mathscr{F}$ a graph $G_{\mathscr{F}}$ such that $\mathscr{F}$ is satisfiable if and only if $G_{\mathscr{F}}$ admits a co-covering hypergraph of cost $25m$.

We start by describing how to construct $G_{\mathscr{F}}$. Additional information about how $G_{\mathscr{F}}$ is designed is given in Subsection 3.3.3. To simplify the construction and the proofs, we give labels to some vertices and some edges. The labels we use are c_i, $T_{i,x}$ and $F_{i,x}$ where $i \leqslant m$ and x is a variable of $\mathscr{F}$. For each clause c_i and each variable x occurring in c_i, we define a gadget $g(x, c_i)$. If x occurs positively in c_i, then $g(x, c_i)$ is the graph depicted in Figure 3.6a. If x occurs negatively in c_i, then $g(x, c_i)$ is the graph depicted in Figure 3.6b. Every gadget $g(x, c_i)$ contains a vertex labelled c_i, an edge labelled $F_{i,x}$ and an edge labelled $T_{i,x}$.

We then create a new vertex for each clause c_i that we connect to the three vertices labelled c_i and to an additional vertex of degree 1. We thus have for each clause a graph $g(c_i)$ like the one depicted in Figure 3.7.



(a) The gadget $g(x, c_i)$ if x appears in c_i. (b) The gadget $g(x, c_i)$ if $\neg x$ appears in c_i.

Figure 3.6

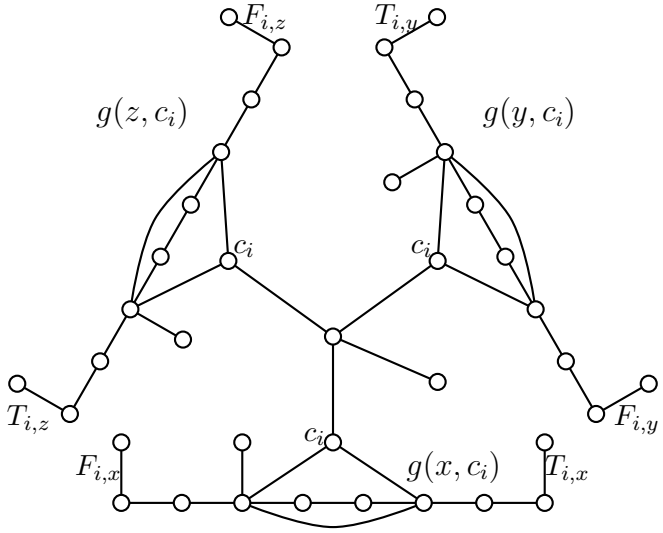


Figure 3.7: The clause-gadget $g(c_i)$ associated to the clause $c_i = (x \vee \neg y \vee \neg z)$.

Finally, for each variable x, let $c_{i_1}, \ldots, c_{i_\ell}$ be the clause in which x appears. Observe that $\ell \geqslant 2$ since every variable has a positive and a negative occurrence. For each $j \leqslant \ell$, we merge the edge labelled $T_{i_j,x}$ in $g(x, c_{i_j})$ with the edge labelled $F_{i_k,x}$ in $g(x, c_{i_k})$ (where $k = j + 1 \mod \ell$) such that the resulting edge has an extremity of degree one. We consider that this edge has both $T_{i_j,x}$ and $F_{i_k,x}$ as labels. For example, if a variable x appears positively in the clauses c_1 and c_4 and negatively in the clause c_3, the Figure 3.8 depicts what the graph looks like around the gadget associated to the variable x.

By connecting all the gadgets $g(x, c_i)$ as described above, we obtain the gadget graph $G_{\mathscr{F}}$. We may assume that $G_{\mathscr{F}}$ is connected. Otherwise, this means that $\mathscr{F}$ is the conjunction of two formulas that share no common variables and $\mathscr{F}$ is satisfiable if and only if those two formulas are. Observe that $G_{\mathscr{F}}$ is trivially co-connected. Moreover, the size of $G_{\mathscr{F}}$ is polynomial in n and m.

Let us prove that $\mathscr{F}$ is satisfiable if and only if $G_{\mathscr{F}}$ admits a co-covering hypergraph of cost $25m$. We start with the following lemma which proves the existence of an optimal co-covering hypergraph where every hyperedge is contained in the vertex set of some clause-gadget.

Lemma 3.21. *There exists an optimal co-connecting hypergraph H of $G_{\mathscr{F}}$ such that $H = H_1 \cup H_2 \cup \cdots \cup H_m$ and for all $i \leqslant m$, $V(H_i) \subseteq V(g(c_i))$ where $V(H_i)$ denotes the vertices used by the hyperedges of H_i.*

Proof. In the graph $G_{\mathscr{F}}$, the intersection between two clause-gadgets only contains

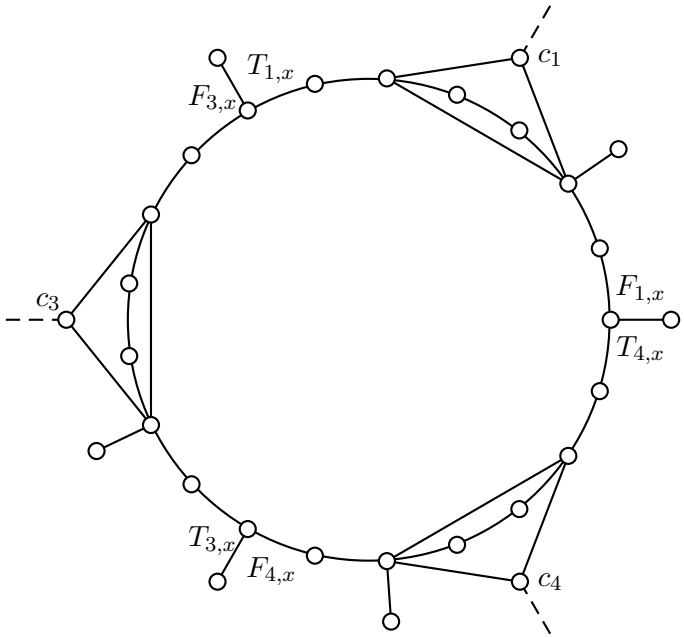


Figure 3.8: The gadgets associated to the variable x.

labelled edges. Thus, if a hyperedge E is not included in any clause-gadget, it means that E covers at least two non-labelled edges from two distinct clause-gadgets. The Figure 3.9 depicts what we call the junction between two clause-gadgets and outlines the vertices of interest in this proof. Here, the clause-gadget $g(c_1)$ contains the vertices v_0, v_1, v_2 and v_3 and $g(c_2)$ contains v_2, v_3, v_4 and v_5.

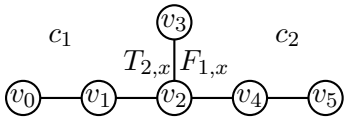


Figure 3.9: The junction between the clause-gadgets of c_1 and c_2.

Since $G_{\mathscr{F}}$ is connected and co-connected, we know by Lemma 3.14 that it admits an optimal co-connecting hypergraph $H = \{E_1, \ldots, E_k\}$ such that for all i, $G_{\mathscr{F}}[E_i]$ is connected. The only way for a hyperedge E_i, such that $G_{\mathscr{F}}[E_i]$ is connected, to cover non-labelled edges in several clause-gadgets is to contain the vertices labelled v_1, v_2 and v_4 at the junction between two clause-gadgets. In this case, we say that E_i covers the junction between these two gadgets. Let us assume that a hyperedge E_i covers the junction between two clause-gadgets $g(c_1)$ and $g(c_2)$. We make no assumption on whether x appears positively or negatively in c_1 and c_2.

By definition, E_i contains the vertices v_1, v_2 and v_4. If E_i does not contain v_3, there exists a hyperedge E_j that covers the edge $\{v_2, v_3\}$. Since $G_{\mathscr{F}}[E_j]$ is connected, E_j has to contain at least one of v_1 and v_4 and therefore shares at least two common vertices with E_i. This means that we can merge E_i and E_j without increasing

the cost of the solution and thus, we can assume that E_i contains v_3. However, $G_{\mathscr{F}}[\{v_1, v_2, v_3, v_4\}]$ is still not co-connected and E_i has to contain other vertices. By connectivity, E_i must contain at least one of v_0 and v_5.

- If E_i contains only one of $\{v_0, v_5\}$, say v_0, we remove v_4 from E_i and add v_2 to the hyperedge E_j that covers $\{v_4, v_5\}$. The hyperedges E_i and E_j still induce co-connected subgraphs of $G_{\mathscr{F}}$, cover the same edges as before and the cost of H does not increase.

- If $G_{\mathscr{F}}[E_i \setminus \{v_2, v_3\}]$ is connected, we replace E_i in the solution by the hyperedges $F_1 = \{v_0, v_1, v_2, v_3\}$ and $F_2 = E_i \setminus \{v_1, v_3\}$. The hyperedges F_1 and F_2 have the same cost as E_i and cover the same edges, $G_{\mathscr{F}}[F_1]$ is co-connected and so is $G_{\mathscr{F}}[F_2]$ (all the vertices are adjacent to v_2 in $\overline{G_{\mathscr{F}}[F_2]}$ except v_4 that can be connected to v_2 through v_0) . Let us also note that both $G_{\mathscr{F}}[F_1]$ and $G_{\mathscr{F}}[F_2]$ are connected.

- If E_i contains both v_0 and v_5 but $G_{\mathscr{F}}[E_i \setminus \{v_2, v_3\}]$ is not connected, we know it has two connected components C_1 and C_2 (since removing a vertex set of degree k cannot create more than k connected components). We replace E_i by $F_1 = \{v_2, v_3\} \cup C_1$ and $F_2 = \{v_2, v_3\} \cup C_2$. The hyperedges F_1 and F_2 have the same cost as E_i and cover the same edges. Furthermore, both $\overline{G_{\mathscr{F}}[F_1]}$ and $\overline{G_{\mathscr{F}}[F_2]}$ are connected since every vertex is adjacent to v_3 except v_2 that can be connected to v_3 through v_0 and v_5 in C_1 and C_2. We also notice that $G_{\mathscr{F}}[F_1]$ and $G_{\mathscr{F}}[F_2]$ are both connected.

In any case, we can build an optimal co-connecting hypergraph where $G_{\mathscr{F}}[E_i]$ is still connected for all i and the hyperedges cover strictly fewer junctions. We can iterate this process until H satisfies the lemma. □

Let $H = H_1 \cup \cdots \cup H_m$ be an optimal co-connecting hypergraph of G such that for all $i \leqslant m$, $V(H_i) \subseteq V(g(c_i))$.

Observe that the labelled edges are the only edges of $G_{\mathscr{F}}$ to belong to several clause-gadgets. Thus, for each $i \leqslant m$, the non-labelled edges of $g(c_i)$ must be covered by H_i. Consequently, the cost of H_i is fully determined by which labelled edges of $g(c_i)$ it covers. We want to prove that $G_{\mathscr{F}}$ is satisfiable if and only if the labelled edges can be covered in a way such that each H_j has cost 25.

Let c_i be a clause of $\mathscr{F}$ and let us study the cost of H_i with respect to the labelled edges it covers. Let x be a variable of c_i. The gadget $g(x, c_i)$ depends on whether x appears positively or negatively in c_i. In both cases, $g(x, c_i)$ contains an edge labelled $F_{i,x}$ and an edge labelled $T_{i,x}$ but those edges may of may not be covered by H_i. For each variable x that appears in c_i, H_i induces one of the 8 graphs depicted in Figure 3.10 on $g(x, c_i)$.

We observe that this graph can only take 4 values up to isomorphism:

- The two configurations of the first line are actually the same. We call this configuration N (for "none").

- The two configurations of the last line are the same. We call this configuration B (for "both").

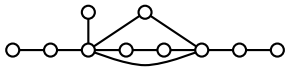
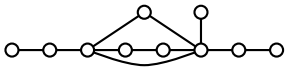
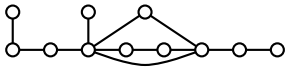
	x appears positively in c_i	x appears negatively in c_i
Neither $T_{i,x}$ nor $F_{i,x}$ are covered by a hyperedge of H_i.		
Only $F_{i,x}$ is covered by a hyperedge of H_i.		
Only $T_{i,x}$ is covered by a hyperedge of H_i.		
Both $T_{i,x}$ and $F_{i,x}$ are covered by a hyperedge of H_i.		

Figure 3.10: The eight possible subgraphs that $V(H_i)$ can induce on $g(x, c_i)$.

- The first configuration of the second line and the second configuration of the third line are the same. We call this configuration U (for "unsatisfied").
- The second configuration of the second line and the first configuration of the third line are the same. We call this configuration S (for "satisfied").

The edges that H_i covers are determined (up to isomorphism) by the configurations encountered for each of the three variables that appear in c_i. Since the clause-gadget is symmetric, the order does not matter: the configuration SUN is exactly the same as the configuration NSU. Thus, we find that H_i can cover 20 different sets of edges up to isomorphisms. We determined the optimal values of cost(H_i) for each case via a computer-assisted exhaustive search. The results are presented in Figure 3.11.

Configuration	Minimum cost	conf.	min.	conf.	min.	conf.	min.
BBB	28	BUS	26	UUU	26	UNN	25
BBU	27	BUN	26	UUS	25	SSS	25
BBS	27	BSS	26	UUN	25	SSN	25
BBN	27	BSN	26	USS	25	SNN	25
BUU	26	BNN	26	USN	25	NNN	25

Figure 3.11: The cost of an optimal connecting hypergraph on every possible configuration of H_i.

The first observation we make is that the optimal value of cost(H_i) is necessarily at least 25 and an optimal co-connecting hypergraph on $G_{\mathscr{F}}$ therefore always costs at least $25m$. We now investigate the case where the optimal cost is exactly $25m$. To this end, we suppose that H has a cost of $25m$.

We note that every configuration that contains a B costs at least 26. Thus, we

know that for each H_i and each x appearing in c_i, H_i covers at most one of the two labelled edges of $g(x, c_i)$.

Let us now look at the gadgets associated to a variable x that appears in ℓ clauses (cf. Figure 3.8). For all j such that x appears in c_j, the hypergraph H_j either covers the two labelled edges of $g(x, c_j)$ (B), one (S or U) or none (N). Since every edge must be covered at least once, this means that a solution where no configuration involves B also does not feature a configuration involving N. Hence, for every H_i, the only configurations that occur are S and U.

Let us suppose that H_i covers the edge $T_{i,x}$. Since the configuration B is impossible, we know that H_i does not cover the edge $F_{i,x}$. Let $T_{j,x}$ be the other label of the edge $F_{i,x}$. Since this edge has to be covered, this means that H_j must cover the edge $T_{j,x}$ and because the configuration B is impossible, it cannot cover the edge $F_{j,x}$. For each variable x, we can prove by induction that either, for all gadget $g(x, c_i)$, H_i covers the edge $T_{i,x}$ or for all gadget $g(x, c_i)$, H_i covers the edge $F_{i,x}$. In the first case, we say that the variable x is set to True, and in the second case, to False. If the variable x is set to True, this means all its positive occurrence leads to a S configuration in the clause where it appears and conversely.

Finally, we notice that the cost of an optimal co-connecting hypergraph on the configurations SSS, SSU and SUU is 25 while it is 26 on the configuration UUU. Therefore, there exists a solution of cost $25m$ if and only of there exists a way to affect all the variables to either True or False such that every clause is satisfied by at least one variable, which comes down to saying that the formula $\mathscr{F}$ is satisfiable.

This proves that **co-OCGH** and therefore **OCGH** and **MCTS** are all NP-hard. □

The *incidence graph* of a formula $\mathscr{F}$ is the bipartite graph representing the relation of belonging between the variables and the clauses of $\mathscr{F}$. Note that if the incidence graph of $\mathscr{F}$ is planar then the graph $G_{\mathscr{F}}$ we build in our proof is planar too. Moreover, Lichtenstein proved in [81] that 3-SAT remains NP-complete when restricted to formulas whose incidence graph is planar. We can therefore deduce from our proof the following stronger result:

Theorem 3.22.

*The restriction of **OCGH** and **MCTS** to co-planar graphs are NP-complete.*

3.3.3 Intuition of the proof

This subsection aims at explaining the intuition behind the conception of the gadget graph. If the gadget graph is connected, we know by Lemma 3.14 that there exists an optimal co-connecting hypergraph H such that for all hyperedge E_i, $G[E_i]$ is connected. We can also assume that $\forall i, j, |E_i \cap E_j| \leqslant 1$ because otherwise we can replace E_i and E_j by $E_i \cup E_j$ in H without increasing its cost. In the rest of this subsection, we only consider hypergraphs satisying these two properties.

Let us take a look back at the table in Figure 3.11. For the proof to work, we need all the following properties to hold:

1. Every configuration with a B must have a strictly higher cost than SSS. This includes for example the configuration BNN which has strictly fewer vertices and edges than SSS.

2. No configuration can have a strictly smaller cost than SSS. This notably implies that SSS may not cost more than NNN eventhough NNN has three fewer vertices and edges that SSS.

3. The configurations SSU and SUU must have the exact same cost as SSS and therefore be optimal. Otherwise, it would be possible to make up for an unsatisfied clause if a clause is satisfied by all its variables.

4. The configuration UUU must have a strictly higher cost than the configurations SSS, SSU and SUU.

To design the gadget graph, we define a new class of graphs as follows.

Definition 3.23. Gearwheels:
A *gearwheel with n teeth*, noted G_n, is the graph constituted of a cycle of length $2n$ where every vertex of the cycle of even rank has an additional neighbour that we call a tooth. More formally, $V(G_n) = \{v_1, \dots, v_{2n}, u_1, \dots, u_n\}$ and $E(G_n) = \{\{v_i, v_{i+1(\mathrm{mod}2n)}\} : i \leqslant 2n\} \cup \{\{u_i, v_{2i}\} : i \leqslant n\}$.

There are only two optimal co-connecting hypergraphs on a gearwheel G_n: the one whose hyperedges are the $E_i = \{u_i, v_{2i}, v_{2i+1}, v_{2i+2}\}$ and the one whose hyperedge are the $E_i = \{v_{2i}, v_{2i+1}, v_{2i+2}, u_{i+1}\}$. Those two optimal solutions and the gearwheel G_4 are illustrated in Figure 3.12.

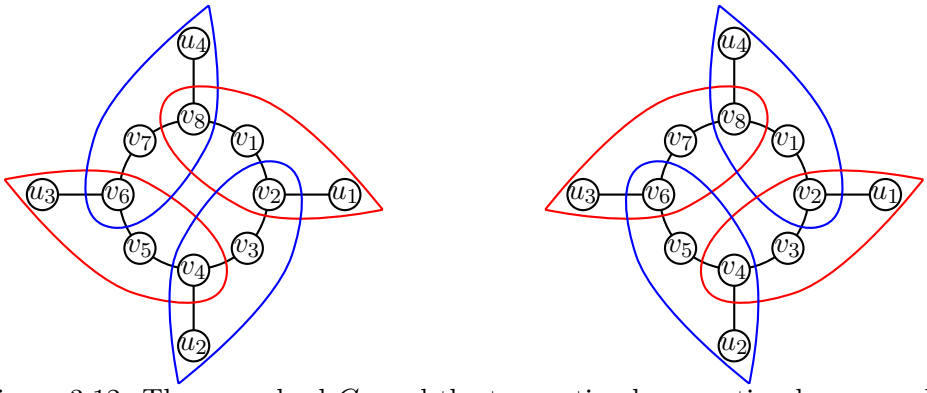
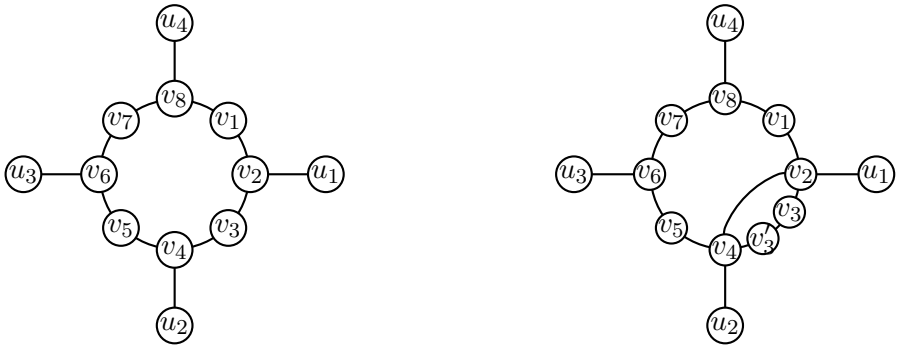


Figure 3.12: The gearwheel G_4 and the two optimal connecting hypergraphs.

It is useful to note that a similar result still holds if we replace the P_3 between two teeth by a C_4. More formally, our gearwheel contains the vertices $v_{2i}, v_{2i+1}, v_{2i+2}$ and the edges $\{v_{2i}, v_{2i+1}\}$ and $\{v_{2i+1}, v_{2i+2}\}$. We add the vertex v'_{2i+1}, the edge $\{v_{2i}, v_{2i+2}\}$ and we replace the edge $\{v_{2i+1}, v_{2i+2}\}$ by $\{v_{2i+1}, v'_{2i+1}\}$ and $\{v'_{2i+1}, v_{2i+2}\}$. This construction is illustrated in Figure 3.13 where we apply this transformation to the gearwheel depicted in Figure 3.13a between the teeth u_1 and u_2 and we obtain the gearwheel depicted in Figure 3.13b. The transformation does not change the fact that there are exactly two optimal solutions that we can obtain from the previous solutions by adding the vertex v'_{2i+1} to the hyperedge $\{u_i, v_{2i}, v_{2i+1}, v_{2i+2}\}$ or $\{v_{2i}, v_{2i+1}, v_{2i+2}, u_{i+1}\}$.

Our strategy is the following: given a formula, we create a graph as a union of gearwheels. Each gearwheel represents a variable and its length depends on



(a) The gearwheel G_4.

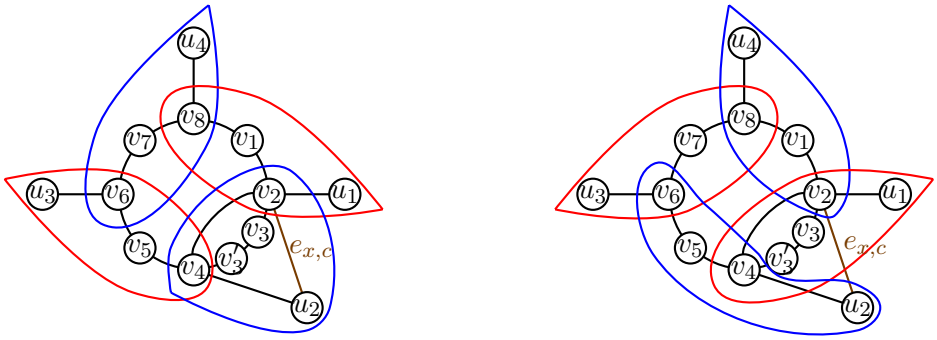
(b) An alteration of G_4.

Figure 3.13: An illustration of the transformation described above.

how many times the variable occurs in the formula. More precisely, we choose to represent a variable that occurs k times by the gearwheel G_{3k}. Optimal solutions consist of covering each gearwheel independently either with hyperedges of the form $E_i = \{v_{2i}, v_{2i+1}, v_{2i+2}, u_{i+1}\}$ (in which case we say that the variable is set to True) or of the form $E_i = \{u_i, v_{2i}, v_{2i+1}, v_{2i+2}\}$ (the variable is set to False). The edges we labelled $F_{i,x}$ and $T_{i,x}$ in the gadget graph connect vertices of the cycle to teeth of the gearwheel, the vertices c_i are teeth and the hyperedges of the optimal hypergraphs naturally induce configurations with S and U and no N or B.

We still have to find a way to ensure that UUU cost more than SSS, SSU and SUU which implies to find a way to "favour" the configuration S over U. If a variable x has a positive occurrence in a clause c, we want to favour the solution that we call True where the hyperedges are the $E_i = \{v_{2i}, v_{2i+1}, v_{2i+2}, u_{i+1}\}$. We do this by adding an edge $e_{x,c}$ between the vertex v_{2i} and the vertex u_{i+1} for a given value of i. This edge is only covered by the hyperedge E_i in the solution we call True. However, $G[E_i]$ is now a C_4 and is not co-connected. We avoid this problem by adding a vertex v'_{2i+1} between v_{2i+1} and v_{2i+2} and by connecting v_{2i} and v_{2i+2} as described above. Conversely, if the variable appears negatively, we want to favour the solution we call False and add the edge $\{u_i, v_{2i+2}\}$. In Figure 3.14, we add the edge $e_{x,c} = \{v_2, u_2\}$ (depicted in brown) to the graph of Figure 3.13b. This is an edge of the type $\{v_{2i}, u_{i+1}\}$, which denotes a positive occurrence of x in c. Hence, the solution True (depicted in Figure 3.14a) is satisfied while the solution False (Figure 3.14b) is unsatisfied. As we can see, no hyperedge covers the edge $e_{x,c}$ in the unsatisfied configuration. Covering this edge in the unsatisfied configuration would require to either add the vertex v_2 to the hyperedge that contains u_2 or to add u_2 to the hyperedge that contains v_2 (or to merge those hyperedges). In any case, it increases the cost of the solution by 1.

At this point, our graph consists of gearwheels that denote the variables of the formula and an extra edge $e_{x,c}$ for each occurrence of a variable x in a clause c. Our partial solution consists of assigning variables to True or False and choosing one of the two optimal solutions on gearwheels depending on the value of the variable. We therefore have an uncovered edge for each unsatisfied occurrence of a variable and



(a) The solution True on a positive occurrence of x.

(b) The solution False on a positive occurrence of x.

Figure 3.14: The difference between the configurations S and U.

covering this edge would increase by 1 the cost of the solution. Hence, with the current construction, the configuration UUU costs one more than SUU as intended but SUU also costs one more than SSU and two more than SSS.

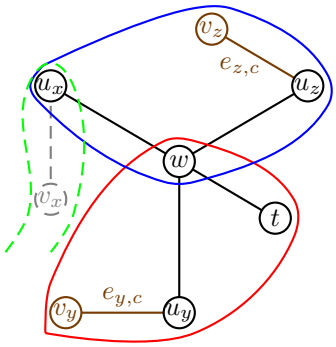
To fix this, we do the following for each clause c. Let x, y and z be the variables that appear in c. We call u_x, v_x, u_y, v_y, u_z and v_z the respective endpoints of the edges $e_{x,c}$, $e_{y,c}$ and $e_{z,c}$ where the vertices u are teeth in their respective gearwheels. We now create two new vertices w and t and connect w to t,u_x,u_y and u_z. Those four new edges are still uncovered and may also be adjacent to other uncovered edges if the clause contains unsatisfied variables. For example, if exactly one variable of the clause is satisfied by our assignment (this is the configuration we call SUU), the uncovered edges induce the graph depicted on Figure 3.15a (where x is the satisfied variable and $e_{x,c}$ is therefore already covered) and the cost of an optimal hypergraph that co-connects this graph is 4 (two hyperedges of cost 4-2=2, as illustrated in the figure).

If two of the variables are satisfied, the graph induced by the uncovered edges contains one fewer vertex and edge and can only be covered by one hyperedge that induces a co-connected graph. As illustrated in Figure 3.15b, the cost of an optimal solution is still 6-2=4.

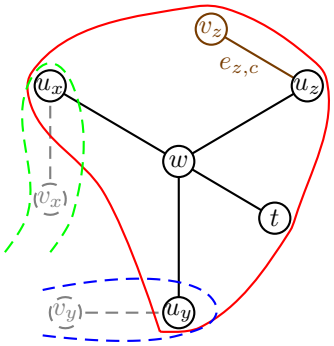
If all three variables are satisfied, the graph induced by the uncovered edges has one fewer vertex and edge than the graph of SSU and just like before, we can make one less hyperedge because of that. Indeed, the graph has a dominating vertex and thus, is not co-connected. The only way to cover those four edges is to add one of their endpoint to a hyperedge that already contains the other but since no existing hyperedge contains more than one vertex of this graph, each edge increases the cost of the solution by 1 and covering this graph still increases the cost by 4 (see Figure 3.15c).

Finally, if a clause is unsatisfied, this means that all its variables are unsatisfied and the uncovered edges therefore induce the graph depicted in Figure 3.15d. This graph has one more edge and vertex than the one we had with SUU but it is not enough to create one more hyperedge that induces a co-connected graph. An optimal

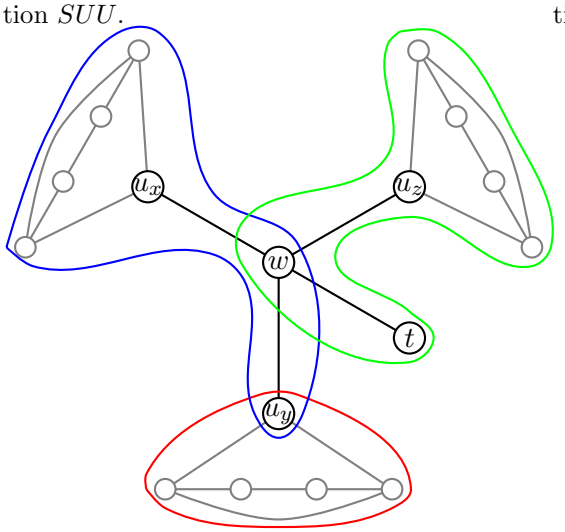
hypergraph that co-connects this graph has cost 5.



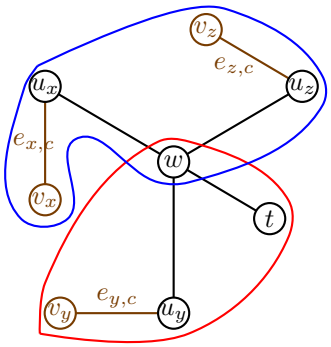
(a) An optimal solution with the configuration SUU.



(b) An optimal solution with the configuration SSU.



(c) An optimal solution with the configuration SSS.



(d) An optimal solution with the configuration UUU.

Figure 3.15: This figure depicts the edges we add to connect the gadgets that denote literals from the same clause and show how to cover them. The edges of the type $e_{x,c}$ that are not covered by the hyperedges of the clause-gadgets (unsatisfied configurations) are depicted in brown. We depict in gray the edges and vertices that are already covered by the hyperedges of the clause-gadgets.

With this construction, we use three hyperedges for each occurrence of a variable in the formula. Two of them have size 4 and therefore cost 2 while the other has cost 3 because of the additional vertex v'_{2i+1}. Each clause involves three occurrences of variables and thus, a cost of 21 plus a cost of 4 if the clause is satisfied and 5 if it is not. Hence, our solution costs 25 on the configuration SSS, SSU and SUU and 26 on the configuration UUU. We proved in the proof of Theorem 3.19 that this solution is optimal.

3.4 Conclusion

In this chapter, we studied the problem of minimum connecting transition set in graphs. Our study lead to a reformulation of **MCTS** as the problem we call optimal connecting hypergraph. This new way of seeing the problem helped us prove our other results, which include exact results on some classes of graphs and a constructive $\frac{3}{2}$-upper bound in the general case. Finally, we proved the NP-completeness of the problem on usual graphs, which also implies the NP-completeness of more general problems such as **MCTS** in graphs with forbidden transitions or the weighted version of **MCTS**. The work presented in this chapter also raises new questions and open many possibilities for future works, some of which are presented in Section 7.2.

Chapter 4

Density of sets avoiding parallelohedron distance 1

This chapter presents the results of [5], which is joint work with Christine Bachoc, Philippe Moustrou and Arnaud Pêcher.

Contents

4.1 Introduction

In this chapter, we study the density of sets of points of a normed real vector space that do not contain two points at distance exactly one from each other. This problem is closely related to a famous graph theory problem called the Hadwiger-Nelson problem.

4.1.1 Unit-distance graphs and the Hadwiger-Nelson problem

The *problem of Hadwiger-Nelson* consists of determining the *chromatic number of the plane* $\chi(\mathbb{R}^2)$, *i.e.* the smallest number of colours required to colour every point of $\mathbb{R}^2$ in such a way that two points at distance exactly one from each other receive different colours. It actually comes down to determining the chromatic number of the unit-distance graph $\mathcal{G}(\mathbb{R}^2)$.

Definition 4.1. Unit-distance graph:

Let $(\mathbb{R}^n, \|\cdot\|)$ be a normed vector space, its *unit-distance graph* denoted by $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$ is the infinite graph whose vertex set is $\mathbb{R}^n$ and whose edge set is $\{xy :$

$\|x - y\| = 1\}$. We denote simply by $\mathcal{G}(\mathbb{R}^n)$ the unit-distance graph of $\mathbb{R}^n$ equipped with the Euclidean norm.

The problem of determining $\chi(\mathbb{R}^2)$ appeared formally in the literature for the first time in [52] in 1960 but Jensen and Toft claim in [65] that Nelson studied this problem as early as 1950. Previous works by Hadwiger, such as [57] in 1944, already studied closely related problems. Erdős and de Bruijn also studied extensively the colourability of infinite graphs [17] and partitions of the space in sets avoiding certain subsets of distances [41] (such as sets avoiding rational distances or sets that do not contain two pair of points at same distance). In [32], Croft attributes to Erdős the problem of determining if $\mathbb{R}^2$ can be partitioned in four sets that do not contain two points at distance one from each other, which, in terms of graphs, comes down to asking if $\mathbb{R}^2$ is 4-colourable.

Despite considerable efforts from many researchers, very little is known about the chromatic number of the Euclidean plane. The lower bound of 4 was established by L. Moser and W. Moser in [91] by exhibiting a 4-chromatic induced subgraph of the unit-distance graph of $\mathbb{R}^2$ of 7 vertices only (illustrated in Figure 4.1). This was the best lower bound we knew until April 2018, when De Grey published in [34] a 1581-vertex 5-chromatic induced subgraph of $\mathcal{G}(\mathbb{R}^2)$. Finding smaller such graphs is currently an extremely active research field. We also know that 7 colours are enough to colour the plane, as illustrated in Figure 4.2 but this is the best upper bound we know. Hence, $5 \leqslant \chi(\mathbb{R}^2) \leqslant 7$. The problem has also been studied on other fields than $\mathbb{R}$ and Woodall proved in [113] that the rational plane $\mathbb{Q}^2$ is 2-chromatic.

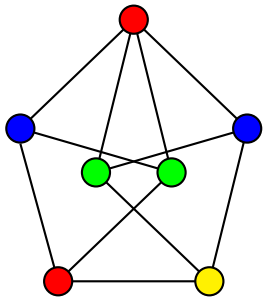


Figure 4.1: The Moser spindle is 4-chromatic. Adjacent vertices are at distance one from each other.

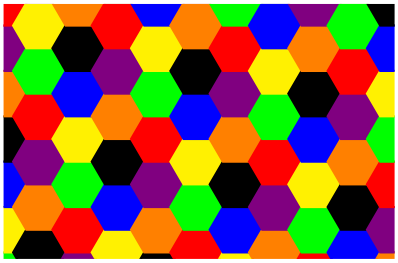


Figure 4.2: A 7-colouring of the Euclidean plane based on a tiling by open regular hexagons of diameter slightly smaller than 1.

By adding the constraint that the colour classes are measurable or using an axiomatic that does not include the axiom of choice, we define the *measurable chromatic number* $\chi_m(\mathbb{R}^n)$ of $\mathbb{R}^n$, which has also received a lot of attention in the literature. The importance of the axiomatic in the study of infinite graphs is highlighted by Erdős and de Bruijn in [17]. Determining the measurable chromatic number of the plane turns out to be a very difficult problem too but discarding unmeasurable solutions allowed to achieve the lower bound of 5 more easily. Indeed, we know since 1981 (Falconer [44]) that 5 colours are required in dimension 2. Hence, just like in

the general case, we know that $5 \leqslant \chi_m(\mathbb{R}^2) \leqslant 7$.

The Hadwiger-Nelson problem has also been studied in higher dimensions and Raiskii [98], Larman and Rogers [78], Larman [77] and finally Frankl and Wilson [51] proved successively that $\chi(\mathbb{R}^n)$ is at least linear, quadratic, cubic and exponential in n. The result of Frankl and Wilson has then been slightly improved by Raigorodskii in [96] and by combining it with an upper bound from Larman and Rogers [78], we have:

$$(1.239 + o(1))^n \leqslant \chi(\mathbb{R}^n) \leqslant (3 + o(1))^n$$

For more information on the Hadwiger-Nelson problem and its variants, we refer the reader to [104] and [97].

4.1.2 Density of sets avoiding distance 1

If the colour classes are measurable, a natural question is to try to determine the greatest density they can achieve, which leads us to the problem we study in this chapter. Here, what we try to determine is not the chromatic number of $\mathbb{R}^n$ for a given norm but the density of a maximum independent set.

Definition 4.2. Density of a set:

The *density* of a measurable set $A \subset \mathbb{R}^n$ with respect to Lebesgue measure is defined as:

$$\delta(A) = \limsup_{R \to \infty} \frac{\mathrm{Vol}(A \cap [-R, R]^n)}{\mathrm{Vol}([-R, R]^n)}$$

Definition 4.3. m_1:

A set A in a normed vector space $(\mathbb{R}^n, \|\cdot\|)$ *avoids distance* 1 if and only if $\forall x, y \in A, \|x - y\| \neq 1$.

The number $m_1(\mathbb{R}^n, \|\cdot\|)$ denotes the supremum of the *densities* of Lebesgue measurable sets $A \subset \mathbb{R}^n$ avoiding distance 1:

$$m_1(\mathbb{R}^n, \|\cdot\|) = \sup_{\substack{A \subset \mathbb{R}^n \text{measurable} \\ A \text{ avoiding } 1}} \delta(A).$$

Since the colour classes of a solution of the Hadwiger-Nelson problem avoid distance 1, we have $\chi_m(\mathbb{R}^n) \geqslant \frac{1}{m_1(\mathbb{R}^n)}$ and we can deduce lower bounds on $\chi_m(\mathbb{R}^n)$ from upper bounds for $m_1(\mathbb{R}^n)$. The problem of determining $m_1(\mathbb{R}^n, \|\cdot\|)$ has been mostly studied in the Euclidean case. The notation $m_1(\mathbb{R}^n)$ was introduced by Larman and Rogers in [78] in 1972 as a tool to study the measurable chromatic number of $\mathbb{R}^n$ but the density of sets avoiding distance 1 was already studied by L. Moser, W. Moser and Croft in [91] and [32] before the number $m_1(\mathbb{R}^n)$ was introduced and without the goal to study the Hadwiger-Nelson problem.

A natural approach to build a set avoiding distance 1 that works for any norm starts from a packing of unit balls. Let Λ be a set such that if $x, y \in \Lambda$, then the unit open balls $B(x, 1)$ and $B(y, 1)$ do not overlap. Thus, the set $A = \cup_{\lambda \in \Lambda} B(\lambda, 1/2)$ of disjoint open balls of radius $1/2$ is a set avoiding 1 and its density is $\frac{\delta}{2^n}$ where n is

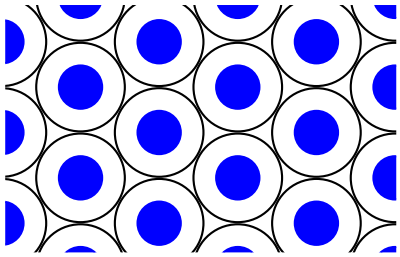


Figure 4.3: A set avoiding distance 1 built from a sphere packing.

the dimension of the space and δ is the density of the packing. This construction is illustrated in Figure 4.3.

In the Euclidean plane, the density of an optimal packing of discs of radius 1 is about 0.9069 and this approach therefore provides a lower bound of about $0.9069/4 \simeq 0.2267$ for $m_1(\mathbb{R}^2, \|\cdot\|_2)$. By refining this idea, Croft manages to build in [32] a set of density about $0.9655/4 \simeq 0.2293$ which is the best lower bound known for $m_1(\mathbb{R}^2, \|\cdot\|_2)$. His construction is a hexagonal arrangement (using the lattice A_2) of blocks defined by the intersection of a circle and a regular hexagon (illustrated in Figure 4.4).

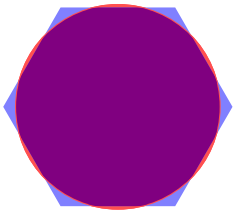


Figure 4.4: The block of Croft's construction is the intersection of a circle (depicted in red) and a regular hexagon (in blue).

Regarding upper bounds, an important objective would be to prove a conjecture by Erdős [107]:

Conjecture 4.4. *Erdős:*

$$m_1(\mathbb{R}^2) < \frac{1}{4}.$$

Prior to our work, the best upper bound was due to Keleti, Matolcsi, de Oliveira Filho and Ruzsa [70], who have shown $m_1(\mathbb{R}^2) \leqslant 0.258795$. In Chapter 5, we improve this bound to 0.256828 (Theorem 5.13).

This problem is also studied in higher dimension where the conjecture of Erdős was generalized by Moser, Larman and Rogers [78]:

Conjecture 4.5. *Larman, Moser and Rogers*

$$\forall n \geqslant 2, m_1(\mathbb{R}^n) < \frac{1}{2^n}.$$

A weaker result has been proved in [70]: a set avoiding distance 1 necessarily has a density strictly smaller than $\frac{1}{2^n}$ if it has a *block structure*, *i.e.* if it may be decomposed as a disjoint union $A = \cup A_i$ such that if x and y are in the same block A_i then $\|x - y\| < 1$ and if they are not, $\|x - y\| > 1$. For example, the construction of Croft (the densest set avoiding distance 1 that we know), as well as the colour classes of Figure 4.2 (the best colouring of the plane we know) have block structures. On the other hand, the colour classes of the 2-colouring of $\mathbb{Q}^2$ by Woodall [113] do not have a block structure and since $\mathbb{Q}^2$ is dense in $\mathbb{R}^2$, colour classes with a block structure would not allow for better colouring in $\mathbb{Q}^2$ than in $\mathbb{R}^2$: the best colouring we know where colour classes have block structure uses 7 colours instead of 2. Without the assumption that the solutions have block structure, the known upper bounds are pretty far from 2^{-n}, even asymptotically: the best asymptotic bound is $m_1(\mathbb{R}^n) \leqslant (1 + o(1))(1.2)^{-n}$ (see [78], [6]).

In the general case of an arbitrary norm, we make the remark that if the unit ball tiles $\mathbb{R}^n$ by translation (see Definition 1.53), the method described previously to build a set avoiding distance 1 from a packing provides a set of density exactly $1/2^n$, as illustrated in Figure 4.5.

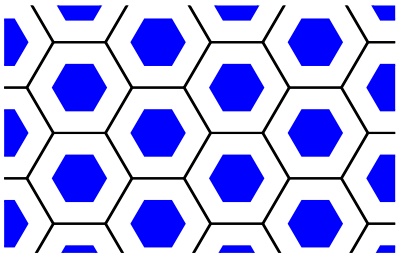


Figure 4.5: The natural packing of density $1/2^n$.

Bachoc and Robins conjecture that this construction of a set avoiding distance 1 is optimal:

Conjecture 4.6. *Bachoc-Robins:*
If $\|\cdot\|$ is a norm such that the unit ball tiles $\mathbb{R}^n$ by translation, then

$$m_1(\mathbb{R}^n, \|\cdot\|) = \frac{1}{2^n}.$$

Note that the lower bound of Croft in the Euclidean case proves that the above construction is not always optimal when the unit ball does not tile the plane but the construction of Croft still provides a bound strictly lower than $\frac{1}{2^2}$. The conjecture of Erdős implies that because the Euclidean disc does not tile the plane, the bound of $\frac{1}{4}$ cannot be achieved.

In this chapter, we prove Conjecture 4.6 in dimension 2 (Theorem 4.23).

We recall that the Voronoï region of a n-dimensional lattice (Definition 1.51) tiles $\mathbb{R}^n$ by translation and is therefore a parallelohedron (Definition 1.53). Conversely, Voronoï conjectured that all parallelohedra are, up to affine transformations, the Voronoï regions of lattices (Conjecture 1.56) and we know that this result holds in

dimension up to 4 (Theorem 1.57). Note that $m_1(\mathbb{R}^n, \|\cdot\|)$ is clearly left unchanged under the action of a linear transformation applied to the norm. So, with respect to Voronoï's conjecture, it is natural to consider in first place the polytopes that are Voronoï regions of lattices.

The most obvious family of lattices is the family of cubic lattices $\mathbb{Z}^n$, whose Voronoï regions are hypercubes. We will see that in this case, Conjecture 4.6 holds trivially (Proposition 4.14). The next families of lattices to consider are arguably the root lattices A_n and D_n (defined in Example 1.52).

We will prove Conjecture 1 for the Voronoï regions of the lattices A_n in every dimension $n \geqslant 2$ (Theorem 4.24). For the lattices D_n, we can only show the inequality

$$m_1(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}}) \leqslant \frac{1}{(3/4)2^n + n - 1} \quad \text{(Theorem 4.28)}$$

which is however asymptotically of the order $O\left(\frac{1}{2^n}\right)$.

The chapter is organized as follows: Section 4.2 presents our method and give preliminary results that we use throughout the chapter. In Section 4.3, we prove Theorem 4.23 which is the case $n = 2$ of Conjecture 4.6. Section 4.4 is dedicated to the families of lattices A_n and D_n. Finally, in Section 4.5, we discuss the chromatic number of the unit distance graph $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$.

4.2 Preliminary results and method

4.2.1 Independence ratio of a discrete graph

A set avoiding distance 1 in $\mathbb{R}^n$ is exactly an independent set in $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$. Therefore $m_1(\mathbb{R}^n, ||.||)$ is by definition the supremum of the densities achieved by independent sets. To study this value, we extend the notion of independence ratio to all discrete infinite graphs. The implication of the independence ratio of discrete graphs on the density of independent sets in uncountable graphs is examined in Subsection 4.2.2:

Definition 4.7. Independence ratio of discrete graphs:

The *independence ratio* $\overline{\alpha}(G)$ of a finite graph G is the ratio of the maximum size of an independent set and the number of vertices of the graph: $\overline{\alpha}(G) = \frac{\alpha(G)}{|V|}$.

Let $G = (V, E)$ be a discrete induced subgraph of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$. For $A \subset V$, we define the density of A in G:

$$\delta_G(A) = \limsup_{R\to\infty} \frac{|A \cap V_R|}{|V_R|}$$

where $V_R = V \cap [-R, R]^n$. Based on this notion, we extend the definition of the independence ratio to discrete graphs:

$$\overline{\alpha}(G) = \sup_{A \text{ independent set}} \delta_G(A).$$

The following lemma provides an equivalent formulation of $\overline{\alpha}(G)$. Being able to switch from a definition to the other makes $\overline{\alpha}$ easier to manipulate in the proofs of this chapter.

Lemma 4.8. *Let $G = (V, E)$ be a discrete graph with $V \subset \mathbb{R}^n$. If every $v \in V$ has finite degree, then*

$$\overline{\alpha}(G) = \limsup_{R\to\infty} \overline{\alpha}(G_R),$$

where G_R is the finite induced subgraph of G whose set of vertices is $V_R = V \cap [-R, R]^n$.

In other words, for every discrete graph G whose vertices all have finite degree,
$\sup_{A \text{ independent}} \limsup_{R\to\infty} \frac{|A \cap V_R|}{|V_R|} = \limsup_{R\to\infty} \sup_{A \text{ independent}} \frac{|A \cap V_R|}{|V_R|}$.

Note that the hypothesis that all the vertices of the graph have finite degree is weaker than requiring the graph to have finite max degree since the graph can have infinitely many vertices.

The rest of this subsection is devoted to the proof of this lemma along with a discussion on the importance of the hypothesis that all the vertices of the graph have finite degree.

Proof. Let G be a graph satisfying the assumptions of the lemma. First of all, we remark that if G is finite, there exists R such that $G_R = G$. Thus, $\overline{\alpha}(G) = \limsup_{R\to\infty} \overline{\alpha}(G_R)$ is obvious. From now on, we will assume that G has infinitely many vertices.

The inequality $\overline{\alpha}(G) \leqslant \limsup_{R\to\infty} \overline{\alpha}(G_R)$ clearly holds. Indeed, if A is an independent set of G, then $A \cap V_R$ is an independent set of G_R and so $\frac{|A\cap V_R|}{|V_R|} \leqslant \overline{\alpha}(G_R)$, leading to $\delta_G(A) \leqslant \limsup_{R\to\infty} \overline{\alpha}(G_R)$.

We will prove the reverse inequality by exhibiting a sequence of independent sets S_k such that, for all $k \geqslant 1$, $\limsup_{R\to\infty} \frac{|S_k\cap V_R|}{|V_R|} \geqslant \limsup_{R\to\infty} \overline{\alpha}(G_R) - \frac{1}{k}$.

Let r_ℓ be a strictly increasing sequence of real numbers tending to infinity and such that $\lim_{\ell\to\infty} \overline{\alpha}(G_{r_\ell}) = \limsup_{R\to\infty} \overline{\alpha}(G_R)$, and let A_{r_ℓ} be an independent subset of V_{r_ℓ} of maximal cardinality. The set S_k will be constructed from the sequence of independent sets A_{r_ℓ}; however, we will need, for reasons that will appear more clearly later, that the successive rings $V_{r_\ell} \setminus V_{r_{\ell-1}}$ are sufficiently large. In view of that, we construct a convenient subsequence of r_ℓ, with the help of a function $\varphi(\ell)$, in the following way.

Since the graph G is discrete, we know that for all R, V_R is finite and since all the vertices of the graph are of finite degree, we know that the neighbourhood $N[V_R]$ is finite too. We denote by $b(R)$ the smallest real number such that $N[V_R] \subset V_{b(R)}$. Then, we set $\varphi(0) = 0$ and, inductively for $\ell \geqslant 0$,

$$\varphi(\ell + 1) = \min\left\{ i : r_i \geqslant b(r_{\varphi(\ell)}) \text{ and } |V_{r_i} \setminus V_{r_{\varphi(\ell)}}| \geqslant |V_{r_{\varphi(\ell)}} \setminus V_{r_{\varphi(\ell-1)}}| \right\}.$$

The existence of $\varphi(\ell+1)$ at each step of the recursion holds because $\lim_{\ell\to\infty} r_\ell = +\infty$ and $V_{r_{\varphi(\ell)}} \setminus V_{r_{\varphi(\ell-1)}}$ is finite (since G is discrete). To keep the notations simple, we set $R_\ell = r_{\varphi(\ell)}$.

We will need the following property of the number of elements of the rings associated to the sequence R_ℓ:

Proposition 4.9. *For all $\ell \in \mathbb{N}$, for all $m \in \mathbb{N}^*$:*

$$|V_{R_{\ell+1}} \setminus V_{R_\ell}| \leqslant \frac{1}{m}|V_{R_{\ell+m}} \setminus V_{R_\ell}|$$

Proof. We have $|V_{R_{\ell+m}} \setminus V_{R_\ell}| = \sum_{k=0}^{m-1} |V_{R_{\ell+k+1}} \setminus V_{R_{\ell+k}}|$ and each term of the sum is larger than $|V_{R_{\ell+1}} \setminus V_{R_\ell}|$, by definition of φ. □

Now we are ready to define the sets S_k. We set, for $k \geqslant 0$,

$$S_k := \left\{v \in V : \exists i \in \mathbb{N} \text{ such that } v \in A_{R_{ik}} \text{ and } \forall j < i, v \notin N[A_{R_{jk}}]\right\}.$$

It remains to prove that S_k is an independent set and satisfies the inequality

$$\limsup_{R\to+\infty} \frac{|S_k \cap V_R|}{|V_R|} \geqslant \limsup_{R\to\infty} \overline{\alpha}(G_R) - \frac{1}{k}.$$

Proposition 4.10. *S_k is independent.*

Proof. Let v_1 and v_2 be two vertices of S_k and let i_1 and i_2 be such that $v_1 \in A_{R_{i_1 k}}$, $v_2 \in A_{R_{i_2 k}}$ and for all $j < i_1$ (respectively i_2), v_1 (resp. v_2) $\notin N[A_{R_{jn}}]$. If $i_1 = i_2$, then v_1 and v_2 both belong to $A_{R_{i_1 k}}$ which is independent. Consequently, they are not connected. If say, $i_1 > i_2$, from the very definition of S_k, $v_2 \notin N[A_{R_{i_1 k}}]$, so v_1 and v_2 are not connected either. □

Lemma 4.11. *For all $k \geqslant 1$, $i \geqslant 0$, $\frac{|S_k \cap V_{R_{ik}}|}{|V_{R_{ik}}|} \geqslant \frac{|A_{R_{ik}}|}{|V_{R_{ik}}|} - \frac{1}{k}$.*

Proof. We prove the lemma by induction on i:

- The property holds for $i = 0$ since $S_k \cap V_{R_0}$ contains A_{R_0}.
- Let $i \in \mathbb{N}$ be such that the property holds. We have:

$$\frac{|S_k \cap V_{R_{(i+1)k}}|}{|V_{R_{(i+1)k}}|} = \frac{|S_k \cap V_{R_{ik}}|}{|V_{R_{(i+1)k}}|} + \frac{|S_k \cap (V_{R_{(i+1)k}} \setminus V_{R_{ik}})|}{|V_{R_{(i+1)k}}|}. \tag{1}$$

Let us lower bound the two terms of this sum one after the other.

▶ Since $A_{R_{ik}}$ is an independent set of maximal cardinality in $V_{R_{ik}}$, we know that $|A_{R_{ik}}| \geqslant |A_{R_{(i+1)k}} \cap V_{R_{ik}}|$. Combining with the induction hypothesis, we find

$$\frac{|S_k \cap V_{R_{ik}}|}{|V_{R_{ik}}|} \geqslant \frac{|A_{R_{(i+1)k}} \cap V_{R_{ik}}|}{|V_{R_{ik}}|} - \frac{1}{k}$$

and thus:

$$\frac{|S_k \cap V_{R_{ik}}|}{|V_{R_{(i+1)k}}|} \geqslant \frac{|A_{R_{(i+1)k}} \cap V_{R_{ik}}|}{|V_{R_{(i+1)k}}|} - \frac{1}{k}\frac{|V_{R_{ik}}|}{|V_{R_{(i+1)k}}|}. \tag{2}$$

▶ By definition, S_k contains all the vertices of $A_{R_{(i+1)k}}$ except those that are in the neighbourhood of an $A_{R_{jk}}$ with $j < i+1$. Since for all $j < i$, $N[A_{R_{jk}}] \subset V_{b(R_{ik})}$,

the set $S_k \cap (V_{R_{(i+1)k}} \setminus V_{R_{ik}})$ contains $A_{R_{(i+1)k}} \setminus V_{b(R_{ik})}$. We also have by construction that $b(R_{ik}) \leqslant R_{ik+1}$. Thus,

$$|V_{b(R_{ik})} \setminus V_{R_{ik}}| \leqslant |V_{R_{ik+1}} \setminus V_{R_{ik}}| \leqslant \frac{1}{k}|V_{R_{(i+1)k}} \setminus V_{R_{ik}}|$$

where the second inequality follows from Proposition 4.9.

This leads to the following inequality:

$$\frac{|S_k \cap (V_{R_{(i+1)k}} \setminus V_{R_{ik}})|}{|V_{R_{(i+1)k}}|} \geqslant \frac{|A_{R_{(i+1)k}} \setminus V_{R_{ik}}|}{|V_{R_{(i+1)k}}|} - \frac{1}{k}\frac{|V_{R_{(i+1)k}} \setminus V_{R_{ik}}|}{|V_{R_{(i+1)k}}|} \tag{3}$$

By combining equations (1), (2) and (3), we find:

$$\frac{|S_k \cap V_{R_{(i+1)k}}|}{|V_{R_{(i+1)k}}|} \geqslant \frac{|A_{R_{(i+1)k}}|}{|V_{R_{(i+1)k}}|} - \frac{1}{k}$$

which concludes the proof of Lemma 4.11. □

Now we are ready to conclude the proof of Lemma 4.8. Indeed, for all $k \geqslant 1$, we have

$$\begin{aligned}
\overline{\alpha}(G) = \sup_S \quad \limsup_{R\to+\infty} \frac{|S \cap V_R|}{|V_R|} &\geqslant \limsup_{R\to+\infty} \frac{|S_n \cap V_R|}{|V_R|} \\
&\geqslant \lim_{i\to\infty} \frac{|S_k \cap V_{R_{ik}}|}{|V_{R_{ik}}|} \\
&\geqslant \lim_{i\to\infty} \frac{|A_{R_{ik}}|}{|V_{R_{ik}}|} - \frac{1}{k} \\
&\geqslant \limsup_{R\to\infty} \overline{\alpha}(G_R) - \frac{1}{k}.
\end{aligned}$$

In the limit when $k \to \infty$, we obtain that $\overline{\alpha}(G) \geqslant \limsup_{R\to\infty} \overline{\alpha}(G_R)$. □

To conclude our discussion of Lemma 4.8, we would like to point out that the inequality $\overline{\alpha}(G) \leqslant \limsup_{R\to\infty} \overline{\alpha}(G_R)$ does not necessarily hold if G has vertices with infinite degree, by bringing out a counterexample.

Let G be the graph given by $V = \mathbb{Z}$ and $E = \{\{a, b\} | a < 0 \text{ and } b > -2a\}$.

Let $N \in \mathbb{N}$ and let $S_N = [\![-N, -\frac{N}{2}]\!] \cup [\![0, N]\!]$. One can see easily that S_N is independent in G. Hence, $\limsup_{R\to\infty} \overline{\alpha}(G_R) \geqslant \lim_{N\to\infty} \frac{|S_N|}{|V_N|} = \frac{3}{4}$.

Let S be an independent set of G. If S contains a vertex indexed by a negative integer $-k$, it cannot contain any vertex indexed by $i > 2k$ and can therefore only contain finitely many vertices indexed by positive integers. Hence, $\lim_{N\to\infty} \frac{|S\cap V_N|}{|V_N|} \leqslant \frac{1}{2}$. If S does not contain any vertex indexed by a negative integer, the inequality $\lim_{N\to\infty} \frac{|A\cap V_N|}{|V_N|} \leqslant \frac{1}{2}$ holds as well. Thus,

$$\sup_S \limsup_{N\to+\infty} \frac{|S \cap V_N|}{|V_N|} \leqslant \frac{1}{2}$$

which proves that $\overline{\alpha}(G) \neq \limsup_{R\to\infty} \overline{\alpha}(G_R)$.

4.2.2 Discretization of the problem

In the case of the chromatic number, we saw that the chromatic number of induced discrete subgraphs of a graph $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$ provides lower bounds on the chromatic number of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$ itself. For example, Figure 4.1 proves that $\chi(\mathbb{R}^2) \geqslant 4$. While this is less trivial, a similar result also holds for the independence ratio:

Lemma 4.12. *Discretization lemma:*
Let $G = (V, E)$ be a discrete subgraph induced by $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$. Then

$$m_1(\mathbb{R}^n, ||.||) \leqslant \overline{\alpha}(G).$$

Proof. By Lemma 4.8, we may assume without loss of generality that G is finite. In this case the result is well known: the proof below is included for the sake of completeness.

Let $R > 0$ be a real number, and let $X \in [-R, R]^n$ be chosen uniformly at random. For $S \subset \mathbb{R}^n$, the probability that X is in S is $\mathbb{P}(X \in S) = \dfrac{\mathrm{Vol}(S \cap [-R, R]^n)}{\mathrm{Vol}([-R, R]^n)}$. Notice that $\limsup_{R\to\infty} \mathbb{P}(X \in S) = \delta(S)$.

Let $S \subset \mathbb{R}^n$ be a set avoiding 1. We define the random variable $N = |(X+V) \cap S|$. On one hand, we have:

$$\begin{aligned}\mathbb{E}\left[\frac{N}{|V|}\right] &= \frac{1}{|V|}\mathbb{E}\left[\sum_{v\in V} \mathbb{1}_{\{X+v\in S\}}\right] \\ &= \frac{1}{|V|}\sum_{v\in V} \mathbb{P}(X \in S - v).\end{aligned}$$

For every v, we have $\limsup_{R\to\infty} \mathbb{P}(X \in S - v) = \delta(S - v) = \delta(S)$.

On the other hand, since for $v_1, v_2 \in V$, $\|(X - v_1) - (X - v_2)\| = \|v_1 - v_2\|$, and $(X + V) \cap S \subset S$, we have, for any $R > 0$,

$$\frac{N}{|V|} \leqslant \overline{\alpha}(G).$$

Thus we get,

$$\delta(S) \leqslant \overline{\alpha}(G). \qquad \square$$

Example 4.13.
The independence number of the Moser Spindle (Figure 4.1) is 2 and its independence ratio is therefore $\frac{2}{7}$. Hence, $m_1(\mathbb{R}^2) \leqslant \frac{2}{7}$.

In this chapter and the next, our strategy to find upper bounds on m_1 is to transfer the study of sets avoiding distance 1 to a discrete setting in which such sets are easier to characterize. This comes down to studying the independence ratio of appropriately chosen induced subgraphs of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$. Most of the graphs we deal with in this chapter are infinite but we look for graphs with specific structures that help us bound their independence ratio. The most difficult task here is to design a

discrete vertex set V that provides a good upper bound on $m_1(\mathbb{R}^n, \|\cdot\|)$ and at the same time is easy to analyze.

To illustrate this approach, we consider the cubic lattice, which is the easiest to deal with. If $L = 2\mathbb{Z}^n$, the Voronoï region of L is the hypercube whose vertices are the points of coordinates $(\pm 1, \pm 1, \ldots, \pm 1)$ and the associated norm is the norm $\|\cdot\|_\infty$ presented in Example 1.47.

Proposition 4.14. *For every $n \geqslant 1$, we have:*

$$m_1(\mathbb{R}^n, \|\cdot\|_\infty) = \frac{1}{2^n}$$

Proof. Let $V = \{0,1\}^n \subset \mathbb{R}^n$ and let G be the subgraph of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_\infty)$ induced by V. Following the definition of V, for every $v, v' \in V$ with $v \neq v'$, we have $\|v - v'\|_\infty = 1$. Hence, G is a complete graph and its independence number is 1. Since it has 2^n vertices, applying Lemma 4.12, we get

$$m_1(\mathbb{R}^n, \|\cdot\|_\infty) \leqslant \frac{\alpha(G)}{|V|} = \frac{1}{2^n}. \qquad \square$$

A strength of our approach that we would like to highlight is that it gives bounds directly on the chromatic number of the space and not only on the measurable chromatic number as m_1 usually does. Indeed, by determining the independence ratio $\overline{\alpha}(G)$ of a subgraph G of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$, we prove that G has chromatic number at least $\frac{1}{\overline{\alpha}(G)}$ and the chromatic number of G gives a lower bound on the chromatic number of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$.

4.3 Parallelohedron norms in the plane

In this section, we prove Theorem 4.23. We saw in Subsection 1.4.4 that the parallelohedra in dimension 2, are, up to an affine transformation, squares or Voronoï hexagons (see Figure 1.23).

We have already seen that $m_1(\mathbb{R}^2, \|\cdot\|_\infty) = \frac{1}{4}$ (Proposition 4.14). It remains to deal with Voronoï hexagons. Even though it is not true that every hexagonal Voronoï region is linearly equivalent to the regular hexagon, we first consider the regular hexagon in order to present in this basic case the ideas that will be used in the general case.

4.3.1 The regular hexagon

This subsection is devoted to the proof of the following theorem:

Theorem 4.15.
If $\mathcal{P}$ is the regular hexagon in the plane, then

$$m_1(\mathbb{R}^2, \|\cdot\|_\mathcal{P}) = \frac{1}{4}.$$

We would like to thank Dmitry Shiryaev [103] for the idea of this proof:

Proof. Let $\mathcal{P}$ be the regular hexagon in $\mathbb{R}^2$. We denote by S its set of vertices and by $\partial\mathcal{P}$ its boundary. Thus, $\|x\|_\mathcal{P} = 1$ if and only if $x \in \partial\mathcal{P}$. We label the vertices of $\mathcal{P}$ modulo 6 as described in Figure 4.6.

We call V the lattice generated by the set $\frac{1}{2}S$. Let $G_\mathcal{P}$ be the subgraph of $\mathcal{G}(\mathbb{R}^2, \|\cdot\|_\mathcal{P})$ induced by V. We shall prove that $\overline{\alpha}(G_\mathcal{P}) \leqslant 1/4$. To do so, we introduce an auxiliary graph $\tilde{G} = (\tilde{V}, \tilde{E})$ with the same set of vertices $\tilde{V} = V$ and such that for all $x, y \in V$, $(x, y) \in \tilde{E}$ if and only if $x - y \in \frac{1}{2}S$. This graph is the *Cayley graph* of V generated by $\frac{1}{2}S$. It is depicted in Figure 4.7.

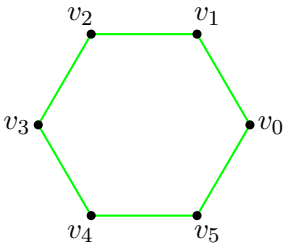


Figure 4.6: The labelling of the vertices of the regular hexagon.

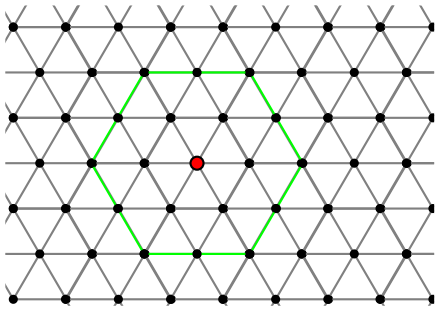


Figure 4.7: The Cayley graph $\tilde{G}$. The unit polytope is depicted in green and the origin in red.

We denote by $\tilde{d}(x, y)$ the distance between two vertices x and y in the graph $\tilde{G}$ (*i.e.* the minimum length of a path between x and y in $\tilde{G}$, which is not to be confused with the geometric distance between x and y). We define the distance $\tilde{d}(A, B)$ in $\tilde{G}$ between two subsets of vertices A and B as the minimum distance between a vertex of A and a vertex of B. This graph has a property we call property D which is crucial for our proof of Theorem 4.15:

Lemma 4.16. *Let u_1 and u_2 be two vertices of $\tilde{G}$. Then:*

$$\tilde{d}(u_1, u_2) = 2 \Rightarrow \|u_1 - u_2\|_\mathcal{P} = 1. \qquad \text{(Property D)}$$

Proof. Since $\tilde{G}$ is vertex-transitive, we may assume without loss of generality that $u_1 = \mathbf{0}$. One can see in Figure 4.7 that the vertices at graph distance 2 from $\mathbf{0}$ are points of $\partial\mathcal{P}$ and are therefore at geometric distance 1 from $\mathbf{0}$. □

It can be noted, although it will not be useful here, that the equivalence $\tilde{d}(u_1, u_2) = 2 \Leftrightarrow \|u_1 - u_2\|_\mathcal{P} = 1$ holds.

Auxiliary graphs satisfying (Property D) help us bound $m_1(\mathbb{R}^n, \|\cdot\|_\mathcal{P})$ thanks to the following lemma:

Lemma 4.17. *Let $\|\cdot\|_\mathcal{P}$ be a norm in $\mathbb{R}^n$ defined by a polytope $\mathcal{P}$. Let $\tilde{G}$ be a graph with vertex set V that satisfies* (Property D)*. Let $A \subset V$ be a set avoiding polytope distance 1. Then A can be written as the union of cliques of $\tilde{G}$: $A = \bigcup_{C \in \mathcal{C}} C$ where if $C, C' \in \mathcal{C}$ with $C \neq C'$, then $N[C] \cap N[C'] = \varnothing$.*

Proof. Since A avoids polytope distance 1, following Lemma 4.16, a connected component C of $G[A]$ cannot contain two vertices at graph distance 2 from each other.

Hence, all the vertices of C are at graph distance 1 from each other which means that C is a clique.

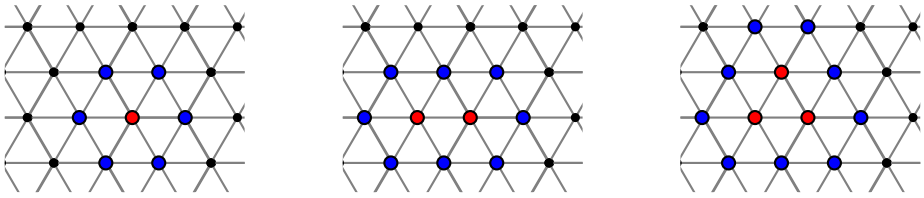
Conversely, if two different cliques C and $C' \subset A$ share a common neighbour, then $\tilde{d}(C, C') \leqslant 2$. Since C and C' are two different connected components of $G[A]$, $\tilde{d}(C, C') > 1$ and thus, $\tilde{d}(C, C') = 2$. This is impossible, since A avoids polytope distance 1. □

Now we define the local density of a clique C of $\tilde{G}$: $\delta^0(C) = \frac{|C|}{|N[C]|}$. In the next lemma, we analyze the different possible cliques of the graph $\tilde{G}$ that we constructed for the regular hexagon, and determine their local density:

Lemma 4.18. *For every clique $C \subset \tilde{G}$, $\delta^0(C) \leqslant \frac{1}{4}$.*

Proof. Let C be a clique of $\tilde{G}$. Since $\tilde{G}$ is vertex transitive, we can assume without loss of generality that $\mathbf{0} \in C$. Up to isomorphisms, there are only three possible cliques in $\tilde{G}$. Their neighbourhoods are depicted in Figure 4.8:

- $C = \{\mathbf{0}\}$: its neighbourhood is $\{\mathbf{0}\} \cup \frac{1}{2}S$ (see Figure 4.8a). Thus $\delta^0(C) = \frac{1}{7}$.
- $C = \left\{\mathbf{0}, \frac{v_0}{2}\right\}$, and $\delta^0(C) = \frac{2}{10} = \frac{1}{5}$ (see Figure 4.8b).
- $C = \left\{\mathbf{0}, \frac{v_0}{2}, \frac{v_1}{2}\right\}$, and $\delta^0(C) = \frac{3}{12} = \frac{1}{4}$ (see Figure 4.8c). □



(a) (b) (c)

Figure 4.8: The possible cliques (in red) and their neighbourhood (in blue).

We now have all the ingredients to prove that the density of a set avoiding distance 1 for distance induced by the regular hexagon cannot exceed 1/4 (Theorem 4.15):

Following Lemma 4.12, it is sufficient to prove $\overline{\alpha}(G_{\mathcal{P}}) \leqslant \frac{1}{4}$. If $A \subset V$ is a set avoiding distance 1, we know by Lemma 4.17 that it may be written as the union of cliques in $\tilde{G}$, whose neighbourhoods are disjoint. Hence, the density of A is upper bounded by the maximum local density of a clique in $\tilde{G}$. From Lemma 4.18, we conclude that $\overline{\alpha}(G_{\mathcal{P}}) \leqslant \frac{1}{4}$. □

4.3.2 General Voronoï hexagons

This subsection deals with the general hexagonal Voronoï region $\mathcal{P}$ of the plane. We prove that:

Theorem 4.19.
If $\mathcal{P}$ is an hexagonal Voronoï region in the plane, then

$$m_1(\mathbb{R}^2, \|\cdot\|_{\mathcal{P}}) = \frac{1}{4}.$$

Proof. Let $\mathcal{P}$ be the hexagonal Voronoï region of a lattice $L \subset \mathbb{R}^2$. Let $\{\beta_0, \beta_1\}$ be a basis of L such that the vectors β_0 , β_1, $\beta_2 = \beta_1 - \beta_0$, and their opposites define the faces of $\mathcal{P}$. We label v_i (with $0 \leqslant i \leqslant 5$) the vertices of $\mathcal{P}$ in such a way that $\beta_i = v_i + v_{i+1}$, where i is defined modulo 6. This labelling is illustrated in Figure 4.9.

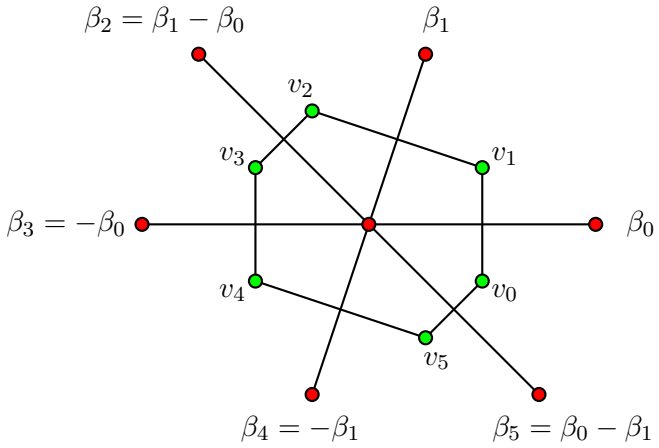


Figure 4.9: The vectors β_i and the vertices of the hexagon.

In order to prove Theorem 4.19, just like in the case of the regular hexagon, we construct a graph $G_{\mathcal{P}}$ induced by $\mathcal{G}(\mathbb{R}^2, \|\cdot\|_{\mathcal{P}})$, and prove that $\overline{\alpha}(G_{\mathcal{P}}) \leqslant \frac{1}{4}$. Unfortunately, in the general case, the vertices of the hexagons of a tiling of the plane do not form a lattice. We use a different approach to build a graph $G_{\mathcal{P}}$ and an auxiliary graph $\tilde{G}$ that satisfies a weaker version of (Property D).

To build the vertex set V of $G_{\mathcal{P}}$, we start from the lattice $\frac{1}{2}L$ and add the translates of the vertices $V_{\mathcal{P}}$ of $\mathcal{P}$ by the vectors of $\frac{1}{2}L$. We set $A = \frac{1}{2}L$ and $B = V_{\mathcal{P}} + \frac{1}{2}L$ so that $V = A \cup B$; this construction is represented in Figure 4.10 where the vertices of A are depicted in red, and those of B in green.

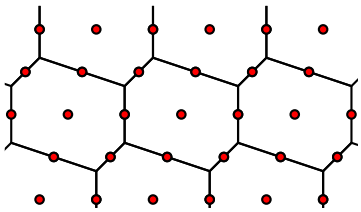
Let us note that for every i, $v_{i+2} = v_i \mod L$. Indeed,

$$v_{i+2} - v_i = v_{i+2} + v_{i+1} - (v_i + v_{i+1}) = \beta_{i+1} - \beta_i = \beta_{i+2}.$$

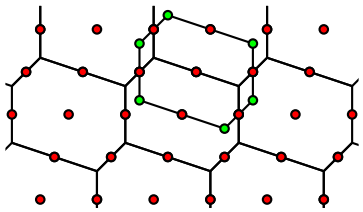
As a consequence, we may write V as the disjoint union of three translates of L:

$$V = \frac{1}{2}L \cup \left(\frac{1}{2}L + v_0\right) \cup \left(\frac{1}{2}L + v_1\right)$$

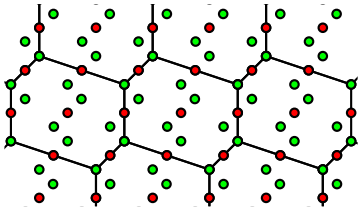
This implies that the density of B in V is twice that of A.



(a) The lattice A.



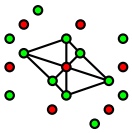
(b) The translate of $\mathcal{P}$ by $\frac{1}{2}\beta_1$.



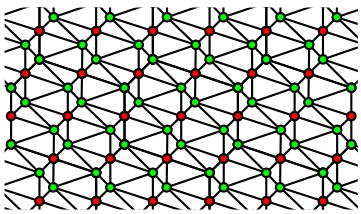
(c) The vertex set $V = A \cup B$.

Figure 4.10

The auxiliary graph $\tilde{G} = (\tilde{V}, \tilde{E})$ has the same vertex set as $G_{\mathcal{P}}$ and we define its edges as follows. By construction, there are exactly 7 vertices of V in the interior of $\mathcal{P}$: the center $\mathbf{0} \in A$, and six points of B denoted $s_0, \ldots, s_5$, with $s_i = \frac{v_{i-1}+v_{i+1}}{2}$. For every point of $a \in A$, we create the edges $(a, a + s_i)$ and $(a + s_i, a + s_{i+1})$ for all i from 0 to 5. This is illustrated in Figure 4.11.



(a) The edges we add for each vertex $a \in A$.



(b) The graph $\tilde{G}$.

Figure 4.11

Note that in the case of the regular hexagon, this construction leads to the same graph $\tilde{G}$ that we considered in Subsection 4.3.1.

Let us now describe the neighbourhood in $\tilde{G}$ of each type of vertex. By construction, a vertex of A has 6 neighbours, and they all belong to B. A vertex $a + s_i$ of B also has six neighbours. Three of them are elements of A, namely a, $a + \frac{\beta_i}{2}$ and $a + \frac{\beta_{i-1}}{2}$ and the other three are elements of B, namely, $a + s_{i-1}$, $a + s_{i+1}$ and $a + v_i$. Figure 4.12 illustrates the neighbourhoods of the vertices of $\tilde{G}$.

It should be noted that in the general case, our graph $\tilde{G}$ does not satisfy (Property D): indeed, the vertices s_0 and s_3 are at graph distance 2 in $\tilde{G}$ but not necessarily at polytope distance 1. However, this property holds for the pairs of vertices points that share a common neighbour in B. We prove this in the next

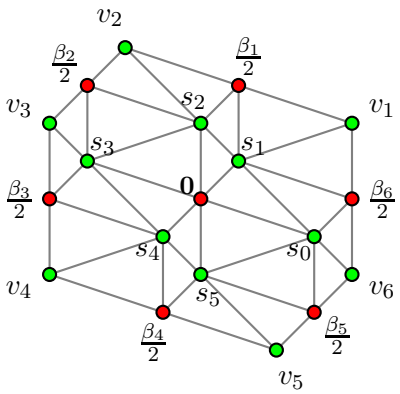


Figure 4.12: The basic pattern in $\tilde{G}$.

lemma, which plays the role of Lemma 4.16 in this subsection:

Lemma 4.20. *If two vertices $x, y \in V$ are at distance 2 from each other in $\tilde{G}$ and have a common neighbour $z \in B$, then $\|x - y\|_{\mathcal{P}} = 1$.*

Proof. **Case I: if at least one of the vertices x and y is in A.** In this case we may assume that $x \in A$ and by vertex-transitivity of A, that $x = \mathbf{0}$. Then z is one of the s_i, and since y is a neighbour of z, y must be in the set $\{\mathbf{0}, s_{i-1}, s_{i+1}, \frac{\beta_i}{2}, \frac{\beta_{i-1}}{2}, v_i\}$. Since the first three are not at graph distance 2 from $\mathbf{0}$, y is one of the last three vertices, and they all are in $\partial\mathcal{P}$. Thus, $\|x - y\|_{\mathcal{P}} = 1$.

Case II: $x, y, z \in B$. Then we may assume without loss of generality $x = s_{i-1}$, and $z = s_i$. Since z has only three neighbours in B, y can be either s_{i+1} or v_i. We have:

$$s_{i+1} - s_{i-1} = \frac{v_i + v_{i+2}}{2} - \frac{v_i + v_{i-2}}{2} = \frac{v_{i+2} - v_{i-2}}{2} = \frac{v_{i+2} + v_{i+1}}{2} = \frac{\beta_{i+1}}{2}$$

and

$$v_i - s_{i-1} = v_i - \frac{v_i + v_{i-2}}{2} = \frac{v_i - v_{i-2}}{2} = \frac{v_i + v_{i+1}}{2} = \frac{\beta_i}{2}.$$

In both cases, $\|x - y\|_{\mathcal{P}} = 1$. □

Note that an alternative statement of Lemma 4.20 is that the forbidden-transition graph obtained from $\tilde{G}$ by forbidding the transition set $F = \{uvw : v \in A\}$ observes (Property D) *i.e.* two vertices of $\tilde{G}$ are at geometric distance 1 if and only if they are connected by a compatible walk of length 2.

Let $U \subset V$ be a set of vertices avoiding polytope distance 1, let C be a connected component of $\tilde{G}[U]$ and let $N[C]$ be its closed neighbourhood. We set $N_B[C] = N[C] \cap B$ and $\delta_B^0(C) = \frac{|C|}{|N_B[C]|}$.

The following lemma is the analogue of Lemma 4.17 in this situation: we show that if C and C' are two different connected components, then $N_B[C]$ and $N_B[C']$ must be disjoint:

Lemma 4.21. *Let $U \subset V$ be a set avoiding polytope distance 1. If $C \neq C'$ are two distinct connected components of $\tilde{G}[U]$, then $N_B[C] \cap N_B[C'] = \varnothing$.*

Proof. If a vertex $z \in B$ is in both $N_B[C]$ and $N_B[C']$, then there exists $x \in C$, $y \in C'$ such that $\tilde{d}(x,z) = \tilde{d}(z,y) = 1$. Since C and C' are connected components of $\tilde{G}[U]$, we have $\tilde{d}(x,y) > 1$. Thus $\tilde{d}(x,y) = 2$ and by Lemma 4.20, $\|x-y\|_{\mathcal{P}} = 1$, which is impossible, since U avoids 1. □

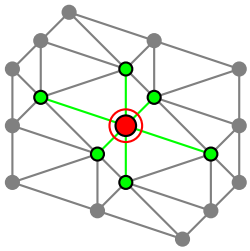
Now we study the different possible connected components:

Lemma 4.22. *Let $U \subset V$ be a set avoiding polytope distance 1. If C is a connected component of $\tilde{G}[U]$, then*

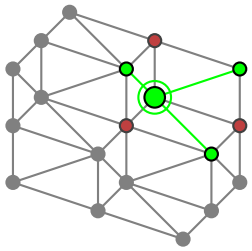
$$\delta_B^0 \leqslant \frac{3}{8}.$$

Proof. We enumerate the possible connected components of $\tilde{G}[U]$. Let us start with the isolated points. Up to translations by $\frac{1}{2}L$, the two possible connected components of size 1 are:

- $C = \{\mathbf{0}\} \subset A$. Its neighbourhood is made of six vertices which are all in B (see Figure 4.13a). Thus, $\delta_B^0(C) = 1/6$.
- $C = \{s_i\} \subset B$. We know that such a vertex has three neighbours in B (Figure 4.13b). Hence, $\delta_B^0(C) = 1/4$.



(a) The vertex $\{\mathbf{0}\}$ has six strict neighbours in B.



(b) The vertex $\{s_i\}$ has three strict neighbours in B.

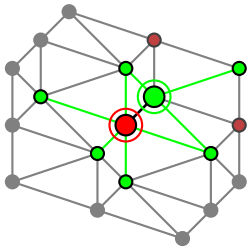
Figure 4.13: The two possible types of connected component of size 1 and their neighbours in B. Elements of C are denoted by circled vertices.

We now focus on the connected components of size 2. Since a vertex in A has all its neighbours in B, such a connected component cannot contain two elements of A. Thus, up to translation, the possibilities are:

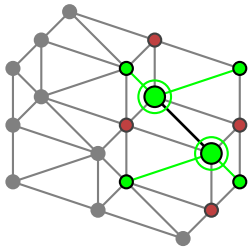
- $C = \{\mathbf{0}, s_i\}$ and the only neighbour in B that is not already a neighbour of $\mathbf{0}$ is v_i (see Figure 4.14a). Thus, $\delta_B^0 = 2/7$.
- $C = \{s_i, s_{i+1}\}$ and the neighbours in B are $s_{i-1}, v_i, s_{i+2}, v_{i+1}$ (Figure 4.14b). Thus, $\delta_B^0 = 2/6 = 1/3$.

There are two kinds of connected components of size three:

- $C = \{\mathbf{0}, s_i, s_{i+1}\}$. The only neighbour of s_{i+1} in B that is not a neighbour of $\{\mathbf{0}, s_i\}$ is v_{i+1} (Figure 4.15a). Thus, $\delta_B^0 = 3/8$.



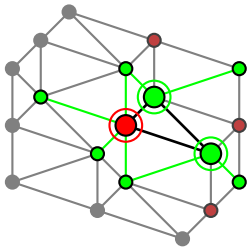
(a) The set $C = \{\mathbf{0}, s_i\}$ has six strict neighbours in B.



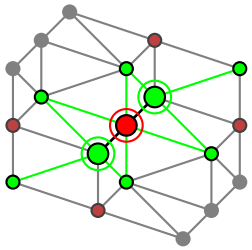
(b) The set $C = \{s_i, s_{i+1}\}$ has four strict neighbours in B.

Figure 4.14: The two possible types of connected component of size 2 and their neighbours in B.

- $C = \{\mathbf{0}, s_i, -s_i\}$. The only neighbour of $-s_i$ in B that is not a neighbour of $\{\mathbf{0}, s_i\}$ is $-v_i$ (Figure 4.15b). Thus, $\delta_B^0 = 3/8$.



(a) The set $C = \{\mathbf{0}, s_i, s_{i+1}\}$ has six strict neighbours in B.



(b) The set $C = \{\mathbf{0}, s_i, -s_i\}$ has six strict neighbours in B.

Figure 4.15: The two possible types of connected component of size 3 and their neighbours in B.

Applying Lemma 4.20, one can check that we have enumerated all the possible kinds of connected components of $\tilde{G}[U]$. □

Putting everything together, we can complete the proof of Theorem 4.19:
Let $U \subset V$ avoiding polytope distance 1. We define $\delta_B(U) = \limsup_{R\to\infty} \frac{|U \cap V_R|}{|B \cap V_R|}$ where as usual, $V_R = V \cap [-R, R]^n$. Hence, we have:

$$\delta_{G_\mathcal{P}}(U) = \delta_B(U) \times \delta_{G_\mathcal{P}}(B).$$

Since $V = A \cup B$ and B is twice as dense as A in $G_\mathcal{P}$, we find $\delta_{G_\mathcal{P}}(U) = \frac{2}{3}\delta_B(U)$.
From Lemma 4.21, we know that $\delta_B(U) \leqslant \sup_{C \text{ cc of } \tilde{G}[U]} \delta_B^0(C)$. Then Lemma 4.22 shows that

$$\delta_B(U) \leqslant \frac{3}{8}$$

and we get

$$\delta_{G_\mathcal{P}}(U) \leqslant \frac{2}{3} \times \frac{3}{8} = \frac{1}{4}.$$

□

Proposition 4.14 and Theorem 4.19 cover every parallelohedron in dimension 2. This proves that Conjecture 4.6 holds for $n = 2$.

Theorem 4.23.
If $\|\cdot\|$ is a norm such that the unit ball tiles $\mathbb{R}^2$ by translation, then

$$m_1(\mathbb{R}^2, \|\cdot\|) = \frac{1}{4}.$$

4.4 The norms associated with the Voronoï regions of the lattices A_n and D_n

4.4.1 The lattice A_n

Here, we consider the lattice A_n (see Example 1.52) for $n \geqslant 2$. We recall that $A_n = \mathbb{Z}^{n+1} \cap H$, where H is the hyperplane of $\mathbb{R}^{n+1}$ defined by $H = \{(x_1, \dots, x_{n+1}) \in \mathbb{R}^{n+1} : \sum_{i=1}^{n+1} x_i = 0\}$. We prove that:

Theorem 4.24.
For every dimension $n \geqslant 2$, if $\mathcal{P}$ is the Voronoï region of the lattice A_n, then

$$m_1(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}}) = \frac{1}{2^n}.$$

Proof. For $n = 2$, the Voronoï region of A_2 is actually the regular hexagon (see Figure 1.21b). We are going to generalize to all dimensions $n \geqslant 2$ the strategy that we used in subsection 4.3.1.

The Voronoï region $\mathcal{P}$ of A_n is described extensively in Chapter 21, Section 3 of [28]. We give a brief overview of the results that are relevant in this subsection.

We denote by p_H the orthogonal projection on H. Here, $H^\perp = \mathbb{R}(1, \dots, 1)$ and thus, $p_H(u_1, u_2, \dots, u_{n+1}) = (u_1 - \tilde{u}, u_2 - \tilde{u}, \dots, u_{n+1} - \tilde{u})$ where $\tilde{u} = \dfrac{u_1 + \cdots + u_{n+1}}{n+1}$.

The norm on H associated to the Voronoï cell of A_n is defined by $\|x\|_{\mathcal{P}} = \max_{i,j}(x_i - x_j)$. Hence, the border of $\mathcal{P}$ consists of the points $(x_1, \dots, x_{n+1})$ of H such that $\max x_i - \min x_i = 1$ and the vertices of $\mathcal{P}$ are the points H whose coordinates take exactly two different values a and b with $a - b = 1$. Hence, we find that the vertices $V_{\mathcal{P}}$ of $\mathcal{P}$ are exactly the $p_H(u)$ with $u \in V_0$ where $V_0 = \{0,1\}^{n+1} \setminus \{(0, \dots, 0), (1, \dots, 1)\}$ (with the notation introduced previously, we have $\tilde{u} = b$).

Note that for all vertex $u = (u_1, \dots, u_{n+1}) \in \mathbb{R}^{n+1}$,

$$\max_{i,j}(u_i - u_j) = \max_{i,j}((u_i - \tilde{u}) - (u_j - \tilde{u})) = \|p_H(u)\|_{\mathcal{P}}. \tag{4}$$

We consider a tiling by translation of $\mathbb{R}^n$ by the polytope $\frac{1}{2}\mathcal{P}$. Let S be the set of vertices generated by $\frac{1}{2}V_{\mathcal{P}}$ (S consists of the vertices and centers of the polytopes of the tiling) and let $G_{\mathcal{P}}$ be the subgraph of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ induced by S. The auxiliary graph $\tilde{G}$ has the same vertex set as $G_{\mathcal{P}}$ and two vertices u and v are adjacent in $\tilde{G}$ if and only if $u - v \in \frac{1}{2}V_{\mathcal{P}}$ ($\tilde{G}$ is the Cayley graph on $V_{\mathcal{P}}$ associated with the generating set $\frac{1}{2}V_{\mathcal{P}}$). These graphs generalize the graphs that we considered in subsection 4.3.1. We first show that $\tilde{G}$ satisfies the same remarkable property:

Lemma 4.25. *The graph $\tilde{G}$ satisfies* (Property D)*:*

$$\tilde{d}(u_1, u_2) = 2 \Rightarrow \|u_1 - u_2\|_{\mathcal{P}} = 1.$$

Proof. We may assume $x = \mathbf{0}$. A vertex at graph distance 2 from $\mathbf{0}$ can be written $v + v'$ with v and v' in $\frac{1}{2}V_{\mathcal{P}}$. Thus, we need to show that, for $v, v' \in \frac{1}{2}V_{\mathcal{P}}$, either $v+v'$ is not at graph distance 2 from $\mathbf{0}$ ($v+v' = \mathbf{0}$ or $v+v' \in \frac{1}{2}V_{\mathcal{P}}$) or $\|v+v'\|_{\mathcal{P}} = 1$ which comes down to saying that $v + v' \in \mathcal{V}_{\mathcal{P}}$.

Let $u, u' \in \frac{1}{2}V_0$ be such that $p_H(u) = v$ and $p_H(u') = v'$. Hence, $v + v' = p_H(u + u')$.

We know that $u + u' \in \{\mathbf{0}, \frac{1}{2}, 1\}^{n+1}$ and one of the four following situations occurs:

Case I: all the coordinates of $u + u'$ have the same value. Hence, $v + v' = p_H(u + u') = \mathbf{0}$ and is not at graph distance 2 from $\mathbf{0}$.

Case II: the coordinates of $u + u'$ consist only of 0's and $\frac{1}{2}$'s. The previous characterization of $V_{\mathcal{P}}$ states that $p_H(2(u+u')) \in V_{\mathcal{P}}$ which means that $v+v' \in \frac{1}{2}V_{\mathcal{P}}$ and is therefore at graph distance 1 from $\mathbf{0}$.

Case III: the coordinates of $u + u'$ consist only of $\frac{1}{2}$'s and 1's. We write $u + u'$ as $w + \frac{1}{2}(1, \dots, 1)$ where the coordinates of $u + u'$ consist only of 0's and $\frac{1}{2}$'s. We find that $v + v' p_H(u + u') = p_H(w)$ and we proved in the previous case that $w \in \frac{1}{2}V_{\mathcal{P}}$.

Case IV: both 0's and 1's appear in the coordinates of $u + u'$. By (4), this means that $\|v + v'\| = \|p_H(u + u')\| = 1 - 0 = 1$. □

Because $\tilde{G}$ satisfies (Property D), $\tilde{G}$ also satisfies Lemma 4.17. We can now proceed to analyze the cliques of $\tilde{G}$, and for each of them, determine its local density. Since $\tilde{G}$ is vertex transitive, we only describe the cliques containing $\mathbf{0}$. For $u \in V_0$, we define its *support* $I(u) = \{i \in \{1, \dots, n+1\} : \ u_i = 1\}$.

Lemma 4.26.
The cliques of $\tilde{G}$ containing $\mathbf{0}$ are the sets of the form $\left\{\mathbf{0}, \frac{p_H(u_1)}{2}, \dots, \frac{p_H(u_s)}{2}\right\}$ such that $I(u_1) \subsetneq I(u_2) \subsetneq \dots \subsetneq I(u_s)$.

In particular, since $s \leqslant n$, a clique cannot contain more than $n + 1$ vertices.

Proof. Let C be a clique of $\tilde{G}$, and assume $\mathbf{0} \in C$. Then the other elements of C must belong to $\frac{1}{2}V_{\mathcal{P}}$ and since C is a clique, they must be adjacent in the graph. In other words, if $\frac{v}{2}, \frac{v'}{2} \in C$, then $\frac{v-v'}{2} \in \frac{1}{2}V_{\mathcal{P}}$. Let $v \neq v' \in V_{\mathcal{P}}$, and $u, u' \in V_0$ such that $v = p_H(u)$ and $v' = p_H(u')$. For $i \in \{1, \dots, n+1\}$, the ith coordinate of $u - u'$ is:

$$\begin{cases} 1 & \text{if } i \in I(u) \setminus I(u'), \\ -1 & \text{if } i \in I(u') \setminus I(u), \\ 0 & \text{otherwise.} \end{cases}$$

If both 1 and -1 appear in the coordinates of $u-u'$, then $\|v-v'\|_{\mathcal{P}}=2$, and $\frac{v-v'}{2}\notin\frac{1}{2}V_{\mathcal{P}}$. By definition of V_0 and since $v\neq v'$, the coordinates of $u-u'$ must take two different values. Two cases remain: if $u-u'$ contains only 0's and 1's, $u-u'\in V_0$ and $v-v'\in V_{\mathcal{P}}$; and if it contains only 0's and -1's, then we can write $u-u'=w-(1,\dots,1)$, with $w\in V_0$, so that $v-v'\in V_{\mathcal{P}}$ as well.

Hence, $v-v'\in V_{\mathcal{P}}$ if and only if $I(u)\subsetneq I(u')$ or $I(u')\subsetneq I(u)$. □

Lemma 4.27.

For every clique C of $\tilde{G}$, $\delta^0(C)\leqslant\dfrac{1}{2^n}$.

Proof. Let $\left\{\mathbf{0},\frac{p_H(u_1)}{2},\dots,\frac{p_H(u_s)}{2}\right\}$ be a clique. For each $i\in[\![1,s]\!]$, we set $w_i=|I(u_i)|$. By symmetry, we may assume that for all i, $u_i=(\underbrace{1,\dots,1}_{w_i},0,\dots,0)$. We set $u_0=\mathbf{0}$ and $w_0=0$. We want to count the vertices in $N[C]=\frac{1}{2}(\{\mathbf{0},p_H(u_1),\dots,p_H(u_s)\}+V_{\mathcal{P}})$.

Since $\mathbf{0}\in C$, the set $(\{\mathbf{0},p_H(u_1),\dots,p_H(u_s)\}+V_{\mathcal{P}})$ must contain all the images of $V_0\cup\{\mathbf{0}\}$ by p_H: there are $2^{n+1}-1$ such vertices.

For each $i=1,\dots,s$, we count how many new neighbours are provided by $p_H(u_i)+V_{\mathcal{P}}$ (*i.e.* neighbours of $p_H(u_i)+V_{\mathcal{P}}$ that are not neighbours of $p_H(u_j)+V_{\mathcal{P}}$ for any $j\leqslant i$). We find that

- The vector $u_1=(\underbrace{1,\dots,1}_{w_1},0,\dots,0)$, provides $(2^{w_1}-1)(2^{n+1-w_1}-1)$ new neighbours.
- The vector $u_2=(\underbrace{1,\dots,1}_{w_1},\underbrace{1,\dots,1}_{w_2-w_1},0,\dots,0)$, provides $2^{w_1}(2^{w_2-w_1}-1)(2^{n+1-w_2}-1)$ new neighbours.
- For any $2\leqslant i\leqslant s$, the vector u_i will provide $2^{w_{i-1}}(2^{w_i-w_{i-1}}-1)(2^{n+1-w_i}-1)$ new neighbours.

By summing all the values, we get:

$$\begin{aligned}|N[C]|&=2^{n+1}-1+\sum_{i=1}^{s}2^{w_{i-1}}(2^{w_i-w_{i-1}}-1)(2^{n+1-w_i}-1)\\&=(s+1)2^{n+1}-\left(\sum_{i=1}^{s}2^{n+1-(w_i-w_{i-1})}+2^{w_s}\right).\end{aligned}$$

Since $w_s\leqslant n$ and for every i, $(w_i-w_{i-1})\geqslant 1$, we have

$$2^{w_s}+\sum_{i=1}^{s}2^{n+1-(w_i-w_{i-1})}\leqslant(s+1)2^n.$$

This implies that $|N[C]|\geqslant(s+1)2^{n+1}-(s+1)2^n=(s+1)2^n$.

Finally, the local density of C satisfies:

$$\delta^0(C) = \frac{|C|}{|N[C]|} = \frac{s+1}{|N[C]|} \leqslant \frac{1}{2^n},$$

and we may note that this bound is sharp if and only if $w_s = n$ and for every i, $w_i - w_{i-1} = 1$, that is when C is a maximal clique of the form

$$\{\mathbf{0}, (1,0,\dots,0), (1,1,0,\dots,0), \dots, (1,\dots,1,0,0), (1,\dots,1,0)\}. \qquad \square$$

We can now conclude the proof of Theorem 4.24. Indeed, following Lemma 4.27 and Lemma 4.17, we find that $\overline{\alpha}(G_{\mathcal{P}}) \leqslant \frac{1}{2^n}$, which leads to Theorem 4.24 by Lemma 4.12. $\square$

4.4.2 The lattice D_n

We apply the same method as for A_n to another classical family of lattices: D_n (see Example 1.52). We recall that D_n is defined by $D_n = \{x = (x_1, \dots, x_n) \in \mathbb{Z}^n : \sum_{i=0}^n x_i = 0 \mod 2\}$. Since the Voronoï cell of D_2 is a square like the cell of $\mathbb{Z}^2$ (see Figure 1.21) and the Voronoï cell of D_3 is a rhombic dodecahedron like the cell of A_3, we assume in this subsection that $n \geqslant 4$.

The same construction as before provides again a graph that satisfies (Property D). Unfortunately, the analysis of the neighbourhoods of the cliques does not lead to the wanted $\frac{1}{2^n}$ upper bound. Nevertheless, we can prove:

Theorem 4.28.
For every dimension $n \geqslant 4$, if $\mathcal{P}$ is the Voronoï region of the lattice D_n, then

$$m_1(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}}) \leqslant \frac{1}{(3/4)2^n + n - 1}.$$

Proof. Let us describe the Voronoï region of D_n. Again we refer to [28] for further details.

The norm on $\mathbb{R}^n$ associated to the Voronoï cell of D_n is defined by $\|x\|_{\mathcal{P}} = \max_{i \neq j}(|x_i| + |x_j|)$. The vertices of $\mathcal{P}$ are of two different types:

- **Type 1**: $\mathcal{P}$ has $2n$ vertices of the form $(0, \dots, 0, \pm 1, 0, \dots, 0)$.
- **Type 2**: $\mathcal{P}$ has 2^n vertices of the form $(\pm\frac{1}{2}, \pm\frac{1}{2}, \dots, \pm\frac{1}{2})$.

Once again, let S be the lattice generated by $\frac{1}{2}V_{\mathcal{P}}$, let $G_{\mathcal{P}}$ be the subgraph of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ induced by S and let $\tilde{G}$ be the auxiliary graph whose vertex set is S and such that u and v are adjacent in $\tilde{G}$ if and only if $(u - v) \in \frac{1}{2}V_{\mathcal{P}}$. We show that (Property D) holds again:

Lemma 4.29. *The graph $\tilde{G}$ satisfies* (Property D)*:*

$$\tilde{d}(u_1, u_2) = 2 \Rightarrow \|u_1 - u_2\|_{\mathcal{P}} = 1.$$

Proof. We use the same method as in the proof of Lemma 4.25. Let $v, v' \in \frac{1}{2}V_{\mathcal{P}}$. We distinguish three cases depending on the type of v and v':

- If both v and v' are of type 1, $v+v'$ is either $\mathbf{0}$, or, up to permutation of the coordinates, of the form $(\pm 1, 0, \dots, 0)$ or $(\pm\frac{1}{2}, \pm\frac{1}{2}, 0, \dots, 0)$, and by definition of the norm, $\|v+v'\|_{\mathcal{P}} = 1$.

- If both v and v' are of type 2, the non zero coordinates of $v+v'$ are $\frac{1}{2}$ or $-\frac{1}{2}$. If $v+v' \neq \mathbf{0}$, then either it is a vertex of $\frac{1}{2}V_{\mathcal{P}}$ of type 1, or it has at least two coordinates whose absolute values are equal to $\frac{1}{2}$, and so $\|v+v'\|_{\mathcal{P}} = 1$.

- If v is of type 1 and v' is of type 2, then $v+v'$ is either a vertex of $\frac{1}{2}V_{\mathcal{P}}$ of type 2, or, up to a permutation of coordinates, of the form $\left(\pm\frac{3}{4}, \pm\frac{1}{4}, \dots, \pm\frac{1}{4}\right)$, and $\|v+v'\|_{\mathcal{P}} = 1$. □

It remains to analyze the neighbourhoods of the cliques of $\tilde{G}$. We first determine the possible cliques of $\tilde{G}$. We may assume that they contain $\mathbf{0}$.

Lemma 4.30. *Up to symmetry, a clique of $\tilde{G}$ containing $\mathbf{0}$ must be a subset of the maximal clique*

$$C_{\max} = \left\{\mathbf{0}, \frac{v_1}{2}, \frac{v_2}{2}, \frac{v_3}{2}\right\} \text{ where } \begin{cases} v_1 = (0, \dots, 0, 1) \\ v_2 = \left(\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2}\right) \\ v_3 = \left(-\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2}\right) \end{cases}.$$

Proof. Let $v, v' \in V_{\mathcal{P}}$ such that $\frac{v-v'}{2} \in \frac{1}{2}V_{\mathcal{P}}$. The conclusion follows from the following facts:

- The vertices v and v' cannot both be of type 1, because the difference of two such vectors, is either $\mathbf{0}$ or has polytope norm 2.

- If v and v' are both of type 2, then v and v' must differ by only one coordinate, otherwise $\|v-v'\|_{\mathcal{P}} = 2$.

- If v is of type 1, say $v = (0, \cdots, 0, \underbrace{\pm 1}_{i}, 0 \cdots, 0)$, if v' is of type 2 and $\frac{v-v'}{2} \in \frac{1}{2}V_{\mathcal{P}}$, then the ith coordinate of v' must have the same sign as the ith coordinate of v. □

We now analyze the local density of the cliques:

Lemma 4.31. *For every clique of $\tilde{G}$, $\delta^0(C) \leqslant \dfrac{1}{(3/4)2^n + n - 1}$.*

Proof. By enumerating the neighbours of every element in $C_{\max}$ and by counting the intersections of the different neighbourhoods, we find that:

- If $C = \{\mathbf{0}\}$, $\delta^0(C) = \frac{1}{1+2^n+n}$.

- If $C = \left\{\mathbf{0}, \frac{v_1}{2}\right\}$, then $\delta^0(C) = \dfrac{2}{2^n + 2^{n-1} + 4n} = \dfrac{1}{(3/4)2^n + 2n}$.

  Note that for $n \geqslant 6$, this density is already greater than $\frac{1}{2^n}$.

- If C is one of the two symmetric cliques $\left\{\mathbf{0}, \frac{v_2}{2}\right\}$ and $\left\{\mathbf{0}, \frac{v_3}{2}\right\}$,

$$\delta^0(C) = \frac{2}{2 \times 2^n + 2n} = \frac{1}{2^n + n}.$$

- By symmetry, the cliques of the form $\left\{\mathbf{0}, \frac{v_i}{2}, \frac{v_j}{2}\right\}$ have the same number of neighbours. If C is one of them, then

$$\delta^0(C) = \frac{3}{2 \times 2^n + 2^{n-1} + 3n - 1} = \frac{1}{(5/6)2^n + n - 1/3},$$

  which is also greater than $\frac{1}{2^n}$.

- Finally, $\delta^0(C_{\max}) = \dfrac{4}{3 \times 2^n + 4n - 4} = \dfrac{1}{(3/4)2^n + n - 1}$, which is the highest possible value of $\delta^0(C)$. □

Like in Subsection 4.4.1, Lemma 4.29, 4.17 and 4.31 lead to Theorem 4.28. □

4.5 The chromatic number of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$

In this section, we discuss the chromatic number $\chi(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ of the unit distance graph associated with a parallelohedron. We start with the construction of a natural colouring of $\mathbb{R}^n$ with 2^n colours, leading to:

Proposition 4.32. *Let $\mathcal{P}$ be a parallelohedron in $\mathbb{R}^n$. Then $\chi(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}}) \leqslant 2^n$.*

Proof. By definition of parallelohedra, there exists a lattice Λ such that $\mathbb{R}^n$ is the union of the $(\lambda + \mathcal{P})$ for $\lambda \in \Lambda$, where only the borders of the $\lambda + \mathcal{P}$ may intersect. It is well known that one can define P' as the union of the interior of P and an appropriately chosen subset of its border so that $\mathbb{R}^n$ is the disjoint union $\cup_{\lambda\in\Lambda}(\lambda + \mathcal{P}')$. We may also write $\mathbb{R}^n$ as the disjoint union $\mathbb{R}^n = \bigcup_{\lambda \in \frac{1}{2}\Lambda} \left(\lambda + \frac{1}{2}\mathcal{P}\right)$.

If H is a coset of $\left(\frac{1}{2}\Lambda\right) \Big/ \Lambda$, then $A_H = \bigcup_{\lambda\in H} B_{\mathcal{P}}\left(\lambda, \frac{1}{2}\right)$ is a set avoiding distance 1. Hence, the points in A_H can receive the same colour. This concludes the proof since $\mathbb{R}^n$ is the disjoint union of all A_H where H runs through the 2^n cosets. □

Example 4.33.

Let $\mathcal{P}$ be a Voronoï hexagon (the most general 2-dimensional parallelohedron). We call its vertices v_i with $i \in [\![0, 5]\!]$ such that v_i and $v_{(i+1) \mod 6}$ are adjacent. One can write $\mathbb{R}^2$ as the disjoint union of translates of $\mathcal{P}'$ where $\mathcal{P}'$ is the union of the interior of the $\mathcal{P}$, the interior of the edges v_0v_1, v_1v_2 and v_2v_3 and the vertices v_1 and v_2. This tiling leads to the colouring of the plane depicted in Figure 4.17.

In order to lower bound $\chi(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$, we can take advantage of the induced subgraphs that we have constructed in previous sections. In particular, whenever

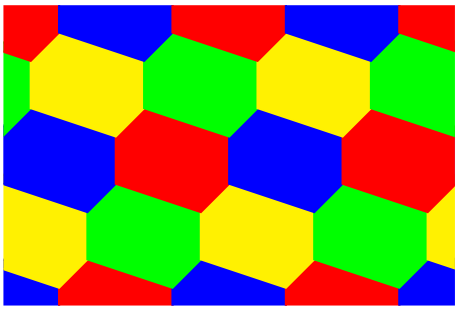


Figure 4.16: A 4-colouring of the plane with the norm induced by a Voronoï hexagon.

we have a discrete induced subgraph $G_{\mathcal{P}}$ of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ satisfying $\overline{\alpha}(G_{\mathcal{P}}) = \frac{1}{2^n}$, we obtain as an immediate consequence that

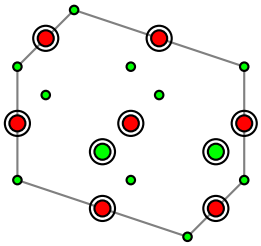
$$\chi(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}}) \geqslant \chi(G_{\mathcal{P}}) \geqslant \frac{1}{\overline{\alpha}(G_{\mathcal{P}})} = 2^n.$$

Thus we have proved:

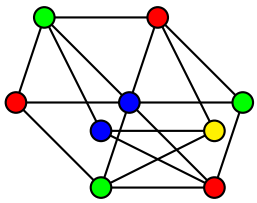
Corollary 4.34. *If $\mathcal{P}$ is a parallelohedron in $\mathbb{R}^2$, then $\chi(\mathbb{R}^2, \|\cdot\|_{\mathcal{P}}) = 4$.*

Corollary 4.35. *If $\mathcal{P}$ is the Voronoï region of the lattice A_n in $\mathbb{R}^n$, then $\chi(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}}) = 2^n$.*

Finally, we would like to point out that in dimension 2, one can find a *finite* induced subgraph of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ with chromatic number 4. Such a graph is depicted in Figure 4.17 for the norm induced by a Voronoï hexagon.



(a) We start from the graph $G_{\mathcal{P}}$ we built in Subsection 4.3.2 and only consider the set of vertices circled in the figure.



(b) The subgraph of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ that this vertex set induces is 4-chromatic.

Figure 4.17

4.6 Conclusion

This chapter studies the density of sets of points avoiding distance 1 for distances induced by several families of parallelohedra. Our approach based on the discretization

lemma (Lemma 4.12) involves bounding the independence ratio of infinite discrete graphs (Definition 4.7), which we were able to do in several cases thanks to specific properties of the auxiliary graphs we created ((Property D)). We proved the conjecture of Bachoc-Robins (Conjecture 4.6) for several families of parallelohedra, including all the parallelohedra in dimension 2 but also a family of parallelohedra of unbounded dimension (Theorem 4.24) and we were able to establish bounds of the order $O(\frac{1}{2^n})$ in other cases (Theorem 4.28). Extending our results to more families of parallelohedra and especially those of dimension 3 is an important goal of our further work. A new approach based on *discrete distribution function* is developed by Moustrou in chapter 4 of his thesis [92] and Chapter 5 of this thesis presents an approach based on finite weighted graphs.

Chapter 5

Optimal weighted independence ratio

This chapter presents a new method to find bounds on $m_1(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$, which is at the core of two ongoing projects: the first one with Christine Bachoc, Philippe Moustrou and Arnaud Pêcher on norms induced by parallelohedra in dimension higher than 2 and the second one on the Euclidean plane with Arnaud Pêcher and Antoine Sedillot. The results will be presented at ISMP 2018 and ICGT 2018 [11] [4].

Contents

5.1 Introduction

This chapter presents ongoing work on an alternative approach to find upper bounds on $m_1(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$, based on the notion of *optimal weighted independence ratio*. We present alternative proofs of results we obtained in Chapter 4 and new bounds for the Euclidean norm and the norm induced by the regular truncated octahedron. This polytope is especially interesting to us because it is the only regular parallelohedron in dimension 3 (see Subsection 1.4.4 for the classification of 3-dimensional parallelohedra) for which the Bachoc-Robins conjecture is still open; indeed, the cube is addressed by Proposition 4.14, the rhombic dodecahedron, by Theorem 4.24, and solutions for the hexagonal prism and the elongated dodecahedron have then been presented by Moustrou in Chapter 4 of [92].

As in Chapter 4, our approach uses a discretization lemma (Lemma 5.6 which is a generalization of Lemma 4.12) and we bound the independence ratio of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ by bounding the independence ratio of well-chosen subgraphs. In Chapter 4, we built infinite discrete subgraphs G of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$ and we were able to bound their independence ratio thanks to (Property D) or similar properties (Lemma 4.20).

However, given a discrete subgraph G of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$, it is not always possible to build an auxiliary graph $\tilde{G}$ that observes (Property D). This in an important obstacle to a proof of the Bachoc-Robins conjecture (Conjecture 4.6) in higher dimensions.

For example, for all the regular polytopes we dealt with in Chapter 4 (Subsections 4.3.1, 4.4.1 and 4.4.2), it was possible to build an auxiliary graph that observes (Property D) on the lattice generated by $\frac{1}{2}V_{\mathcal{P}}$ but we can see that this is no longer true with the regular truncated octahedron. Let us use the coordinate system described at the end of Subsection 1.4.4 to study the regular truncated octahedron, which is the permutohedron of order 4 (Definition 1.60). Let our norm be such that the vertices of the unit polytope are the points whose coordinate are a permutation of $\{-3,-1,1,3\}$. Hence, $\{\frac{1}{2},\frac{3}{2},-\frac{3}{2},-\frac{1}{2}\}$ and $\{\frac{3}{2},-\frac{3}{2},\frac{1}{2},-\frac{1}{2}\}$ are both vectors of $\frac{1}{2}\mathcal{P}$ but their sum, $\{2,0,-1,-1\}$ is neither $\mathbf{0}$, nor a vertex of $\frac{1}{2}\mathcal{P}$, nor a vertex of $\mathcal{P}$, which was impossible with the regular parallelohedron norms we had encountered so far. By adding $\{-\frac{3}{2},-\frac{1}{2},\frac{3}{2},\frac{1}{2}\} \in \frac{1}{2}\mathcal{P}$ to $\{2,0,-1,-1\}$, we obtain the vector $\{\frac{1}{2},-\frac{1}{2},\frac{1}{2},-\frac{1}{2}\}$, which is at geometric distance $\frac{1}{4}$ of $\mathbf{0}$ but would be at graph distance 3 with our usual auxiliary graph structure.

Since we cannot compute the independence ratio of infinite graphs in the general case, we only build finite subgraphs in this chapter but enhance their expressive power by weighting their vertices. This helps us achieve results that we were only able to achieve with infinite graphs in the unweighted case.

In Section 5.2, we define the notion of optimal weighted independence ratio, that we study throughout this chapter. We present the connection it has with the density of sets avoiding distance 1 but also with another well-known problem: fractional colouring. Sections 5.3 and 5.4 studies the notion of optimal weighted independence ratio in two different contexts: the case of a general norm in Section 5.3 and the case of a norm induced by a parallelohedron in Section 5.4. We study it both from a combinatorial and an algorithmic perspectives and present the results it provides. In the Euclidean plane, our current results include an improvement of the best upper bound known on $m_1(\mathbb{R}^2)$ (Theorem 5.13) and of the best lower bound on the fractional chromatic number of the plane (Theorem 5.14). In Section 5.4, we improve the previous algorithm in the specific case of norms induced by parallelohedra and use it to achieve the current best bound on $m_1(\mathbb{R}^3, \|\cdot\|_{\mathcal{P}})$ when P is a regular parallelohedron (Theorem 5.24).

5.2 Our approach

5.2.1 Optimal weighted independence ratio

Before presenting our approach, we need a few standard definitions on weighted graphs.

Definition 5.1. Weighted graph:

A *weighted graph* is a triplet (V, E, w) where (V, E) is a graph and $w : V \to \mathbb{R}^+$ is a *weight distribution* or a *weighting* of the vertices. For $v \in V$, $w(v)$ is called the *weight* of the vertex v. The *total weight of the graph* is defined by $w(G) = \sum_{v \in V} w(v)$.

The weight of a vertex set S is defined similarly as $w(S) = \sum_{v \in S} w(v)$.

Similarly to how we defined the vertex set of a graph as non-empty, we assume that every graph has at least one vertex of non-zero weight. Hence, every graph has strictly positive weight.

Definition 5.2. Weighted independence ratio:

The *weighted independence number* of a finite weighted graph G_w is noted $\alpha(G_w)$ and is the maximum weight of an independent set in G_w. The *weighted independence ratio* of a finite weighted graph G_w is $\overline{\alpha}(G_w) = \dfrac{\alpha(G_w)}{w(G)}$.

Let (V, E) be a discrete induced subgraph of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$, let w be a weight distribution on V and let $G_w = (V, E, w)$. For $A \subset V$, we define the density of A in G_w: $\delta_{G_w}(A) = \limsup_{R \to \infty} \dfrac{w(A \cap V_R)}{w(V_R)}$ where $V_R = V \cap [-R, R]^n$. As in Definition 4.7, we extend the definition of the independence ratio to discrete weighted graphs as $\overline{\alpha}(G_w) = \sup_{A \text{ independent set}} \delta_{G_w}(A)$.

Note that all our definitions in weighted graphs generalize the definitions in unweighted graphs, which are the specific case where every vertex has weight 1.

We are now ready to introduce the central notion of optimal weighted independence ratio.

Definition 5.3. Optimal weighted independence ratio:

We define the *optimal weighted independence ratio* $\alpha^*(G)$ of an unweighted graph $G = (V, E)$ as the infimum over all weight distribution w on V of the independence ratio of $G_w = (V, E, w)$.

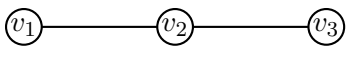
Let $G_w = (V, E, w)$ be a weighted graph and let $G = (V, E)$. If $\alpha^*(G) = \overline{\alpha}(G_w)$ we say that w is an *optimal weighting* of G and that G_w is *optimally weighted*.

Note that the colour classes of a proper k-colouring of a graph G partition $V(G)$ into k independent sets. Since the sum of the weights of those k independent sets in the total weight of G, at least one of them has weight $\frac{w(G)}{k}$ or more. Hence:

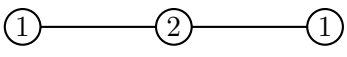
Proposition 5.4. *For every graph G, $\alpha^*(G) \geqslant \frac{1}{\chi(G)}$.*

Example 5.5.

Consider the graph P_3 depicted in Figure 5.1a. The set $\{v_1, v_3\}$ is a maximum independent set and has size 2, and its independence ratio is thus $\overline{\alpha}(P_3) = \frac{2}{3}$. However, if we set $w(v_1) = w(v_3) = 1$ and $w(v_2) = 2$ (see Figure 5.1b), we find that the two maximal independent sets on G ($\{v_1, v_3\}$ and $\{v_2\}$, which are also the colour classes in an optimal proper colouring of G) have weight 2, which proves that $\alpha^*(P_3) \leqslant \frac{1}{2}$. We know by Proposition 5.4 that this bound is optimal: $\alpha^*(P_3) = \frac{1}{2}$.



(a) The graph P_3.



(b) A weighting of P_3.

Note however that the bound provided by Property 5.4 is not always tight. For example, $\alpha^*(C_5) = \frac{2}{5}$ (reached by giving the same weight to all the vertices) while $\chi(C_5) = 3$.

5.2.2 Weighted discretization lemma

The reason why we are interested in the optimal weighted independence ratio of a graph is because it is at least as small as its independence ratio and still provides bounds on $m_1(\mathbb{R}^n, \|\cdot\|)$. The bound it provides is therefore better.

Lemma 5.6. *Weighted discretization lemma:*
Let $G = (V, E)$ be a finite subgraph induced by $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$. Then

$$m_1(\mathbb{R}^n, \|\cdot\|) \leqslant \alpha^*(G).$$

Proof. Let $G = (V, E)$, let w be a weighting of V and let $G_w = (V, E, w)$. Let $R > 0$ be a real number, and let $X \in [-R, R]^n$ be chosen uniformly at random. Like in the proof of Lemma 4.12, we have $\limsup_{R\to\infty} \mathbb{P}(X \in S) = \delta(S)$.

Let $S \subset \mathbb{R}^n$ be a set avoiding 1. We define the random variable $N = w((X + V) \cap S)$. Thus, we have:

$$\begin{aligned}\mathbb{E}\left[\frac{N}{w(V)}\right] &= \frac{1}{w(V)}\mathbb{E}\left[\sum_{v\in V} w(v) \times \mathbb{1}_{\{X+v\in S\}}\right] \\ &= \frac{1}{w(V)}\sum_{v\in V} w(v) \times \mathbb{P}(X \in S - v).\end{aligned}$$

and for every v, we have $\limsup_{R\to\infty} \mathbb{P}(X \in S - v) = \delta(S - v) = \delta(S)$. Thus, we find

$$\mathbb{E}\left[\frac{N}{w(V)}\right] = \delta(S)\frac{\sum_{v\in V} w(v)}{w(V)} = \delta(S).$$

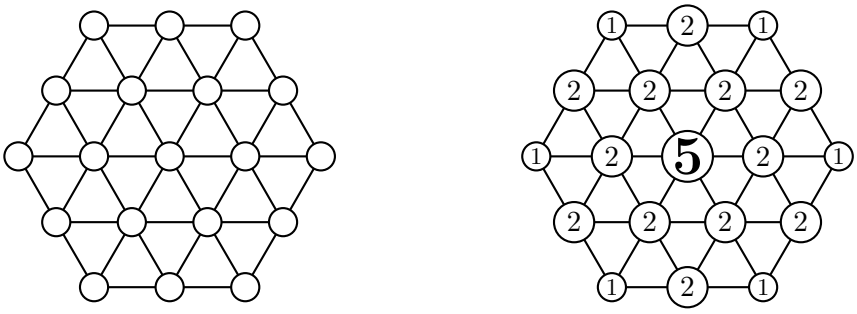
Since $(X + V) \cap S$ is contained in S which is independent, it is independent too and thus, $\frac{N}{w(V)} \leqslant \overline{\alpha}(G_w)$.

This proves that for every weighting w, $\delta(S) \leqslant \overline{\alpha}(G_w)$ and thus $\delta(S) \leqslant \alpha^*(G)$. □

Hence, we can find bounds on $m_1(\mathbb{R}^n, \|\cdot\|)$ by building a finite induced subgraph G of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$ and computing or bounding its optimal weighted independence ratio (we present an algorithm to compute it in Section 5.3).

Example 5.7.

Let $\mathcal{P}$ be the regular hexagon, and let G and $\tilde{G}$ be the graphs built in the proof of Theorem 4.15. Let H and $\tilde{H}$ be their subgraphs induced by the vertices that are contained within the unit polytope. Hence, H is still a subgraph of $\mathcal{G}(\mathbb{R}^2, \|\cdot\|_{\mathcal{P}})$ and $\tilde{H}$ (depicted in Figure 5.2a) has same vertex set and satisfies (Property D). Since H is finite, its independence number is easy to compute and we do not need the auxiliary graph. We use $\tilde{H}$ in the figures for its better readability.Recall that the



(a) The graph $\tilde{H}$.

(b) The graph $\tilde{H}_w$.

Figure 5.2

independent sets in H are exactly the sets that do not contain two vertices at graph distance 2 in $\tilde{H}$.

The graph H has independence number 6 and therefore provides a bound of $\frac{6}{19} \simeq 0.3158$ on $m_1(\mathbb{R}^2, \|\cdot\|_{\mathcal{P}})$. The two different maximum independent sets (up to isomorphism) are depicted in Figure 5.3.

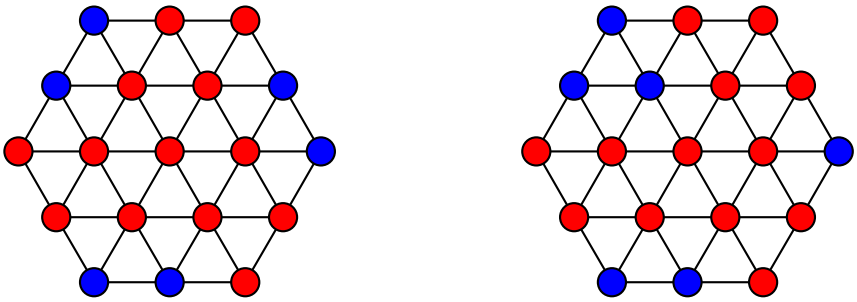


Figure 5.3: The maximum independent sets in H (in blue).

Let w be the weight distribution on $V(H)$ depicted in Figure 5.2b and let $H_w = (V(H), E(H), w)$. The maximum weighted independent sets in H_w are depicted in Figure 5.4 and have weight 9. This proves that $\alpha^*(H) \leqslant \overline{\alpha}(H_w) \leqslant \frac{9}{35} \simeq 0.2571$.

One can prove that the weighting w is optimal and $\alpha^*(H) = \frac{9}{35}$. Here, even with a weighting, the graph H is too small to reach the bound of $\frac{1}{4}$ but the weighting still allows to considerably improve the bound. Furthermore, we will see that the bound of $\frac{1}{4}$ can be reached with a finite weighted graph while we know no finite subgraph of $\mathcal{G}(\mathbb{R}^2, \|\cdot\|_{\mathcal{P}})$ that achieves an unweighted independence ratio of $\frac{1}{4}$.

5.2.3 Fractional chromatic number

Our notion of optimal weighted independence ratio is heavily related to the already well-known notion of fractional chromatic number.

Definition 5.8. b-fold colouring, fractional chromatic number:

A *b-fold colouring* of a graph $G = (V, E)$ is a function c that maps every vertex of a the graph to a set of b colours such that for every $uv \in E$, $c(u) \cap c(v) = \varnothing$.

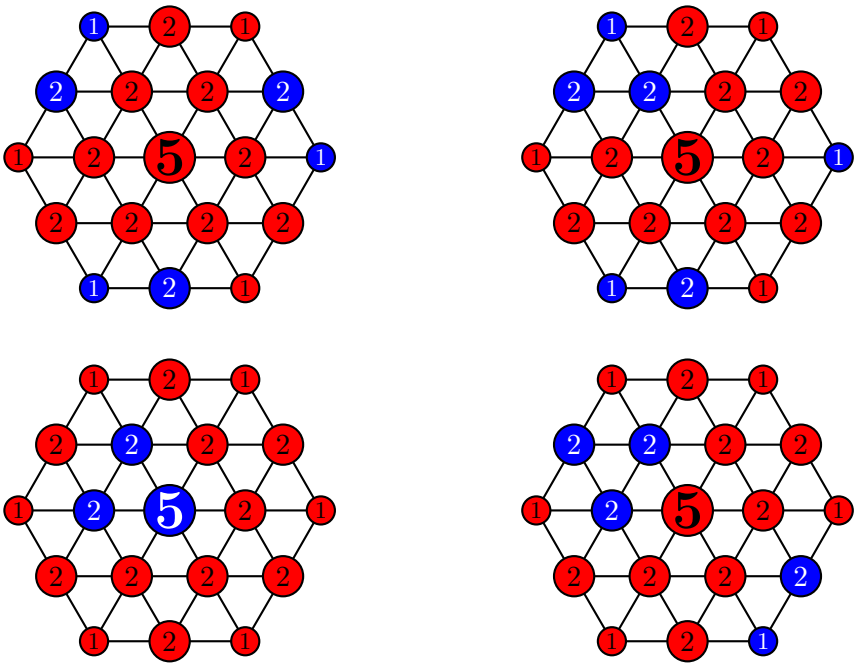


Figure 5.4: The maximum independent sets in H_w (in blue).

A b-fold colouring that uses a different colours is called an $a:b$-*colouring*. If such a colouring exists, G is $a:b$-*colourable* and if a is the smallest number for which G is $a:b$-colourable, G is $a:b$-*chromatic*. In this case, we say that a is the *b-fold chromatic number* of G, noted $\chi_b(G)$.

The *fractional chromatic number* $\chi_f(G)$ of a graph G is the smallest $\frac{a}{b}$ for which G is $a:b$-chromatic: $\chi_f(G) = \inf_b \frac{\chi_b(G)}{b}$.

One can also prove that $\chi_f(G) = \lim_{b\to+\infty} \frac{\chi_b(G)}{b}$.

Note that traditional graph colouring is the case $b = 1$ of b-fold colouring, and thus, for all graph G $\chi_f(G) \leqslant \chi(G)$.

Since the colour classes in a proper colouring of a graph are independent, the chromatic number of the graph is the smallest size of a partition of the vertices of a graph into independent sets. If we are given the set $\mathscr{S}$ of the independent sets of a graph, we can express its chromatic number by a linear program. For each $I \in \mathscr{S}$, let x_I be a binary variable that indicates whether I is a colour class of the optimal colouring we create. Our program is:

$$\begin{cases} \text{minimize } \sum_{I\in\mathscr{S}} x_I \text{ subject to} \\ \forall v \in V, \sum_{I\in\mathscr{S}:v\in I} x_I = 1 \end{cases}$$

The constraints ensure that each vertex belongs to exactly one colour class and the objective function is to use as few colours as possible.

The fractional chromatic number can be expressed by the same program where the x_I do not have to be binary and can take any value between 0 and 1. Thus, fractional graph colouring is *the linear programming relaxation* of graph colouring. However, note that this does not provide an efficient algorithm to compute the fractional chromatic number of a graph: indeed, the number of variables of the above program is the number of independent sets in G, which is exponential in its number of vertices. Hence, both the generation and the resolution of this program takes a long time. The problem of fractional chromatic number is proved NP-hard in [82].

Example 5.9.

As we know, $\chi(C_5) = \chi_1(C_5) = 3$ (see Figure 5.5a). However, by giving two colours to each vertex, we can find a 5:2-colouring of C_5 (see Figure 5.5b), which means that $\chi_f(C_5) \leqslant \frac{5}{2}$. One can prove that this bound is actually tight.

(a) An optimal 1-fold colouring of C_5. (b) An optimal 2-fold colouring of C_5.

Figure 5.5

With the notation of the above linear program, the solution depicted in Figure 5.5b is $x_{\{a\}} = x_{\{b\}} = x_{\{c\}} = x_{\{d\}} = x_{\{e\}} = 0$ and $x_{\{a,c\}} = x_{\{b,d\}} = x_{\{c,e\}} = x_{\{d,a\}} = x_{\{e,c\}} = \frac{1}{2}$.

In [54], Godsil and Royle define a *fractional clique* as a weight distribution on the vertices of the graph such that no independent set on the graph has weight more than 1 (as is the case in a unweighted clique, where the maximum independent sets are vertices and therefore have weight one). The weight of a fractional clique is defined as the total weight of the graph under the weighting that the fractional clique defines. The *fractional clique number* ω_f of a graph is the maximum weight of a fractional clique. For example, the distribution that gives weight $\frac{1}{2}$ to all the vertices of C_5 is a maximum fractional clique of weight $\frac{5}{2}$.

It follows directly from the definition of fractional clique number that for every graph G, $\omega_f(G) = \dfrac{1}{\alpha^* G}$.

As explained in [54], the fractional clique number of a graph is given by the dual of the linear program described above. By the strong duality theorem, we therefore have $\omega_f(G) = \chi_f(G)$. The relation between fractional clique number and optimal weighted independence ratio follows:

Proposition 5.10. *For every graph G,*

$$\chi_f(G) = \frac{1}{\alpha^* G}.$$

Hence, our weighted discretization lemma (Lemma 5.6) connects the density of sets avoiding distance 1 in a given space with its fractional chromatic number. The chromatic number of the Euclidean plane has already been studied and the best published lower bound, established by Cranston and Rabern in 2017 in [31], is

$$\chi_f(\mathbb{R}^2) \geqslant \frac{76}{21} \geqslant 3.61904.$$

The bound it provides on $m_1(\mathbb{R}^2)$ is $m_1(\mathbb{R}^2) \leqslant 0.276316$. In [42], Exoo and Ismailescu claim to have a proof that $\chi_f(\mathbb{R}^2) \geqslant \frac{383}{102} \geqslant 3.75491$ in a yet unpublished paper. This gives a bound on $m_1(\mathbb{R}^2)$ of 0.266319. However, none of these bounds are as good as the current best bound of 0.258795 by Keleti et al. [70]. Thus, our approach also involves to improve the bound on $\chi_f(\mathbb{R}^2)$.

5.3 General norms

This section details the implementation of our method and the results it provides for the case of a general norm. We first focus on how to find an optimal weighting of the vertices of a given graph and thus, how to determine its optimal weighted independence ratio. For example, we are looking for an algorithm that would return the weighting w depicted in Figure 5.2b when given the graph H in Figure 5.2a. Subsection 5.3.1 presents notions that we use to design the algorithm in Subsection 5.3.2. Finally, Subsection 5.3.3 outlines how to build interesting graphs and presents the results of our method in the Euclidean plane.

5.3.1 Preliminary study

While combinatorial bounds and the complexity of the fractional chromatic number have been studied on a variety of classes of graphs, very few algorithms have yet been developed to compute it.

The algorithm we design in this Section uses two linear programs. The first one is used to find a weighting that minimizes the maximum weights of a collection of independent sets.

Let $G = (V, E)$, let $V = \{v_1, \ldots, v_n\}$ and let $S_1, \ldots, S_k$ be independent sets of G. For every vertex v, we create a variable w_v that indicates the weight of v. Since the independence ratio of a graph is left unchanged by the multiplication of the weight of the vertices by a constant, we may set that the total weight of the graph is 1. We create a variable M that indicates the maximum weight of a set S_i.

$$\begin{cases} \text{minimize } M \\ \displaystyle\sum_{v \in V} w_v = 1 \\ \forall i \in [\![1, k]\!], \displaystyle\sum_{v \in S_i} w_v \leqslant M \end{cases}$$

The variables of this linear program do not have to be integers, which allows to solve this program efficiently in practice. However, generating a suitable set $\mathscr{S}$ of

independent sets $S_1, \ldots, S_k$ takes much more time.

The relation between optimal weighted independence number and fractional chromatic and clique numbers and the linear program we have presented in Subsection 5.2.3 suggest to run the above program on the entire set $\mathscr{S}$ of independent sets of G. However, the time required to generate a set $\mathscr{S}$ is at least $\sum_{S\in\mathscr{S}} |S|$. Since they are exponentially many, generating all the independent sets or even all the maximal independent sets of a graph is only feasible on very small graphs. We will thus look for smaller suitable sets $\mathscr{S}$. Our criteria is that by minimizing the weight of the maximum-weight set of $\mathscr{S}$, we want to minimize the weight of the maximum-weight independent set of the entire graph.

The first important observation we make is that there always exists an optimal weighting of the graph such that all the vertices of a same orbit (see Definition 1.16) have the same weight. Indeed, let $\mathrm{Aut}(G)$ be the set of automorphisms on a graph G. The image of an independent set by an automorphism σ is still an independent set and the composition of an optimal weighting w with σ is therefore still an optimal weighting. One can prove that the average of the $w \circ \sigma$ for $\sigma \in \mathrm{Aut}(G)$ gives the same weight to all the vertices of a same orbit and is still an optimal weighting. We call a weighting that gives the same weight to all the vertices of a same orbit a *symmetric weighting*.

Because of this observation, our problem is now to find a collection of independent sets $\mathscr{S} = \{S_1, \cdots, S_k\}$ that contains a maximum-weight independent set under any symmetric weighting w. In particular, if two sets are images of each other by an automorphism, $\mathscr{S}$ only needs to contain at most one of them. Consider the graph C_6 depicted in Figure 5.6, where all the vertices belong to the same orbit O_1. The sets $\{v_2, v_5\}$ and $\{v_3, v_6\}$, although maximal, are the images of $\{v_1, v_4\}$ by an automorphism. Thus, $\mathscr{S}$ only needs to contain at most one of them. Furthermore, note that even if $\{v_1, v_4\}$ is contained in no strictly bigger independent set than itself, it only contains two vertices of O_1 while the set $\{v_1, v_3, v_5\}$ contains three. The set $\{v_1, v_3, v_5\}$ is therefore heavier than $\{v_1, v_4\}$ for every symmetric weighting of C_6.

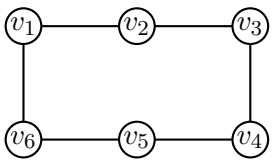


Figure 5.6: The graph C_6.

More generally, let G be a graph, let $O_1, \ldots, O_p$ be its orbits, let S be an independent set of G and let n_i be the number of vertices of O_i that S contains. If an independent set S' contains at least n_i vertices of O_i for all i, then we know that S' is heavier than S for every symmetric weighting w. This observation provides a new partial order on the vertex set of a graph, which is consistent with the inclusion and for which fewer pairs of sets are incomparable. Thus, there are fewer maximal sets. For example, the only two maximal sets in C_6 for this relation are $\{v_1, v_3, v_5\}$ and $\{v_2, v_4, v_6\}$, and by symmetry, we know that $\{v_1, v_3, v_5\}$ is maximal for any symmet-

ric weighting of G. However, while it is easy to ensure that an independent set S is strictly contained in no other (it comes down to checking whether S is dominating), it is much harder to determine if S is maximal for our new order. Furthermore, while being maximal for this order is necessary, it is still not sufficient to ensure that there exists a symmetric weighting w for which S has maximum weight.

Indeed, consider the graph G depicted in Figure 5.7. It has two orbits that we call A and B. Up to isomorphism, there are three maximal independent sets for the previous partial order:
- sets that contain three vertices of A and no vertex of B, such as $S_1 = \{a_1, a_2, a_3\}$;
- sets that contain no vertex of A and three vertices B, such as $S_2 = \{b_1, b_3, b_5\}$;
- sets that contain one vertex of each orbit, such as $S_3 = \{a_1, b_3\}$.

However, one can see that for every symmetric weighting w, $w(S_3) = \frac{w(S_1)+w(S_2)}{3}$ and the weight of S_3 is therefore smaller than at least one of S_1 and S_2.

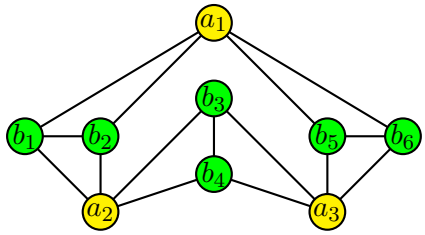


Figure 5.7: A graph G. Its two orbits of vertices A and B are depicted in yellow and green respectively.

The above examples suggest that determining if there exists a symmetrical weighting w for which a given independent S has maximum weight is a difficult task. On the other hand, it also suggests that such sets are much fewer than maximal independent sets. Since the time-complexity of the above linear program depends heavily on the size of $\mathscr{S}$, the algorithms we present in the rest of this chapter try to keep its size small.

5.3.2 The algorithm

Let $G = (V, E)$ be a graph and let $O_1, \ldots, O_p$ be its orbits of vertices. We suppose that we have a function named orbit that returns the number of the orbit of a given vertex v. Our algorithm uses the two following linear programs.

The first one, that we call P_1, is an improved version of the program presented in the previous subsection, that takes the symmetry of the graph into account and looks for symmetric weightings. Let $\mathscr{S} = \{S_1, \ldots, S_k\}$ be a collection of independent sets and let $n_{i,j} = |S_i \cap O_j|$. Our variables are the $w_1, \ldots, w_p$ where w_j indicates the weight of the vertices of O_j.

$$\begin{cases} \text{minimize } M \\ \sum_{j=1}^{p} w_j |O_j| = 1 \\ \forall i \in [\![1,k]\!], \sum_{j=1}^{p} n_{i,j} w_j \leqslant M \end{cases} \tag{$P_1(\mathscr{S})$}$$

The second is an integer linear program close to the one presented in Example 1.76. It returns an independent set of maximum weight for a given symmetric weighting $w = (w_1, \ldots, w_p)$ defined by the weight it gives to each orbit. For each vertex v, we use a binary variable x_v that indicates whether v belongs to the maximum-weight independent set S we create.

$$\begin{cases} \text{maximize } \sum_{v \in V} w_{\text{orbit}(v)} x_v \\ \forall uv \in E, x_u + x_v \leqslant 1 \end{cases} \tag{$P_2(w)$}$$

Our algorithm is described in Algorithm 1.

Algorithm 1: Computing an optimal weighting of a graph G.

```
1 Let O_1, ..., O_p be the orbits of vertices of G
2 Let 𝒮 = {} and for all j ∈ [[1,p]], w_j = 1/|V|
3 Let ub = 1 and lb = 0.
4 while ub ≠ lb do
5     Let S be the independent set returned by (P_2(w)).
6     𝒮 = 𝒮 ∪ {S}.
7     ub = w(S)
8     Let w be the weighting returned by (P_1(𝒮)) and let lb be the objective
      value.
9 return the w_j and ub.
```

At each step of the algorithm, ub is an upper bound on $\alpha^*(G)$ and lb is a lower bound. Indeed, after line 7, $(P_2(w))$ ensures that for any symmetric weight distribution w, there exists an independent set in $\mathscr{S}$ whose weight is at least ub. Furthermore, after line 8, $(P_1(\mathscr{S}))$ ensures that there exists a weighting (namely, w) that reaches the independence ratio of lb.

At each iteration of the **while** loop except the last one, our algorithm returns an independent set S for which there exists a symmetric weight distribution such that S is strictly heavier than every set of $\mathscr{S}$. The last iteration happens when our set $\mathscr{S}$ contains a maximum-weight independent set for every symmetric weighting. However, there might exist smaller such sets $\mathscr{S}$ than the one our algorithm returns. Indeed, as long as there exists an orbit O_j such that no set of $\mathscr{S}$ contains a vertex of O_j, $(P_1(\mathscr{S}))$ can return a weighting where every set of $\mathscr{S}$ has weight 0 by giving weight 0 to all the orbits in $\mathscr{S}$. Then, if an orbit has weight 0, the set S returned by $(P_2(w))$ may not be maximal: for example, if a graph contains two obrits O_1 and

O_2 but $w_2 = 0$, then ($P_2(w)$) may return a set that contains two vertices of O_1 and no vertex of O_2 even if there exist independent sets that contain two vertices of O_1 and one of O_2. This problem can be avoided by adding the constraint $w_j \geqslant \varepsilon$ in ($P_1(\mathscr{S})$), at least during the first few iterations of the **while** loop.

At the end of the execution of the algorithm, the w_j describes an optimal weighting and $\text{lb} = \text{ub} = \alpha^*(G)$.

The running time of our algorithm is hard to analyze because of the difference between the theoretical and the practical running times of linear programs (see Section 1.6 for a discussion of the efficiency of linear programs). Determining the orbits of a graph is a difficult problem in the general case but is easy on the geometric graphs that we study in this chapter and can even be done by hand on most of them. The running time of ($P_1(\mathscr{S})$) is also much smaller than the running time of ($P_2(w)$). Thus, the total running time of the algorithm comes from the running time of ($P_2(w)$) and the number of iterations of the **while** loop (which is equal to the size of the final set $\mathscr{S}$). On the geometric graphs that we study in this chapter, the size of the set $\mathscr{S}$ seems to be empirically linear in the number of orbits of the graph. The theoretical bounds in the general case are most likely much worse and would surely be interesting to study. The running time of ($P_2(w)$) is exponential in $|V|$ but also depends strongly on the symmetry and the independence ratio of the graph. In practice, we are able to handle graphs up to around 600 vertices. Note that even if we are given an optimal weighting of the graph, P_2 is the fastest way we know in the general case to compute the associated independence ratio. Since this problem is NP-complete, we have little hope to find an algorithm that would allow to compute the optimal weighted independence ratio of much bigger graphs than the ones we are currently able to handle.

When running a linear program on a big graph, an important limiting factor is also the memory that the solvers require. However, the size limit for our algorithm to compute the optimal weighted independence ratio is the same as the size limit for simply computing the unweighted independence ratio of a graph with P_2 and the bound provided with by the optimal weighted independence ratio is at least as good and often significantly better than the unweighted independence ratio.

Finally, note that even if the computation is interrupted before completion (because of a too high time- or space-complexity), the values of ub obtained at each iteration of the **while** loop still provide upper bounds on $\alpha^*(G)$ and therefore on $m_1(\mathbb{R}^n, \|\cdot\|)$.

5.3.3 The Euclidean plane

We now show how to use Algorithm 1 to find bounds on $m_1(\mathbb{R}^n, \|\cdot,\|)$ in the case of a general norm (we make no assumption on whether the unit ball tiles the plane). We then use the method we develop to study the density of sets avoiding distance 1 in the Euclidean plane.

As we said previously, for a given norm $\|\cdot\|$, our approach consists of deducing upper bounds on $m_1(\mathbb{R}^n, \|\cdot\|)$ from the optimal weighted independence ratio of finite subgraphs of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$. How to build interesting graphs will be discussed extensively in the rest of this chapter. In the literature on $m_1(\mathbb{R}^n)$ or on

the Hadwiger-Nelson problem, one can already find many interesting subgraphs of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$ (especially in the Euclidean plane) that have been designed to optimize criteria that are close to ours (namely, the unweighted independence ratio or the chromatic number). Before going into details on how to build relevant graphs, we would like to point out how our weighted approach makes this problem easier. An important advantage of our method that we have not explicitly mentioned yet comes from the following property:

Proposition 5.11. *If $G = (V, E)$ is a graph and H is an induced subgraph of G, then $\alpha^*(G) \leqslant \alpha^*(H)$.*

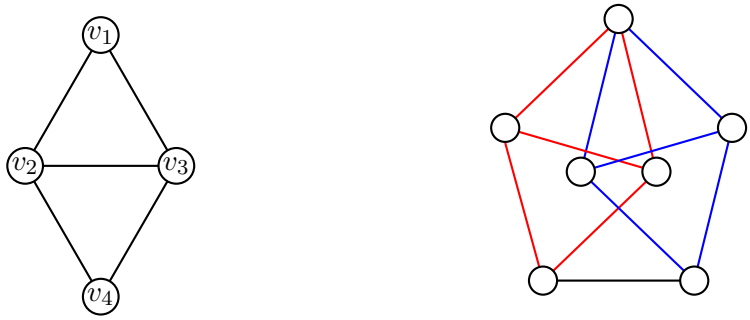
Proof. Let w_H be an optimal weighting of the vertices of H. Let w_G be a weighting of V such that $w_G(v) = w_H(v)$ if $v \in V(H)$ and $w_G(v) = 0$ otherwise and let $G_w = (V, E, w_G)$. Then, $\alpha^*(G) \leqslant \overline{\alpha}(G_w) = \alpha^*(H)$. □

This means that by computing the optimal weighted independence ratio of a given subgraph G of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|)$, we find a better bound than the ones that any subgraph of G would provide.

Note that this property does not hold with the unweighted independence ratio, which leaves us with many more graphs to study. For example, $\overline{\alpha}(P_3) = \frac{2}{3} \geqslant \frac{1}{2} = \overline{\alpha}(P_2)$. Of course, while the unweighted independence ratio of the subgraphs H of G can provide a better bound than $\overline{\alpha}(G)$, they cannot provide better bounds that $\alpha^*(G)$ since $\alpha^*(G) \leqslant \alpha^*(H) \leqslant \overline{\alpha}(H)$.

A good way to build graphs with small independence ratio (weighted or not) is to arrange copies of smaller graphs in such a way that the optimal independent sets of each copies are not compatible with each other. For example, the graph G depicted in Figure 5.8a is 3-chromatic but v_1 and v_4 must have the same colour in a 3-colouring of G. It also has unweighted independence ratio $\frac{1}{2}$ but its only optimal independent set is $\{v_1, v_4\}$. The Moser spindle (Figure 5.8b), which is the smallest 4-chromatic subgraph of $\mathcal{G}(\mathbb{R}^2)$, contains two copies of G (whose edges are depicted in red and blue respectively) combined in such a way that there is at least one of them in which v_1 and v_4 do not have the same colour. Its chromatic number is therefore 4 and its unweighted independence ratio is $\frac{2}{7}$. Similar processes have also been used repeatedly by de Grey in [34] in order to build a 5-chromatic subgraph of $\mathcal{G}(\mathbb{R}^2)$.

This process is harder to illustrate in the weighted case. Indeed, if a graph is optimally weighted, then for every vertex v of degree at least 1, there exists an optimal independent set that contains v (otherwise, we could make the weighted independence ratio of the graph decrease by increasing the weight of v, which is impossible) and one that does not (since there exists an optimal independent set that contains its neighbour). However, given an optimally-weighted graph G, what we can still do is look for pairs of vertices (u, v) such that every maximum-weight independent set that contains u also contains v. We then rotate G around u until the vertex v of the new copy of G is at distance 1 of the vertex v of the original graph G (this process is illustrated in Figures 5.8a and 5.8b with $u = v_1$ and $v = v_4$). This process can then be iterated on the new graph we obtain.



(a) A graph G.

(b) The Moser spindle, built from two copies of G whose vertices v_4 are at distance 1.

Figure 5.8

By Proposition 5.11, we know that adding vertices to a graph can only decrease its optimal weighted independence ratio. However, we are limited by our computation power and can only handle graphs up to a certain size. When building a subgraph G of $\mathcal{G}(\mathbb{R}^n \| \cdot \|)$, it is therefore important to determine which vertices are the most useful to decrease $\alpha^*(G)$. The weights of the vertices in an optimal weighting gives a bound on the impact that the removal of a vertex can have on the optimal weighted independence ratio of the graph.

Proposition 5.12.

Let $G = (V, E)$ be a graph, let $e \geqslant 0$, let w be an optimal weighting of the vertices of G and let $V' \subset V$ be such that $\frac{w(V')}{w(V)} \leqslant e$. Then,

$$\frac{1-e}{1}\alpha^*(G \setminus V') \leqslant \alpha^*(G).$$

Proof. Since every independent set of $G \setminus \{V'\}$ is independent in G, $\alpha(G_w \setminus \{V'\}) \leqslant \alpha(G_w)$. Hence

$$\begin{aligned}\alpha^*(G \setminus V') \leqslant \overline{\alpha}(G_w \setminus \{V'\}) &= \frac{\alpha(G_w \setminus \{V'\})}{w(V) - w(V')} \\ &\leqslant \frac{\alpha(G_w)}{w(V)} = \frac{w(V)}{w(V) - w(V')}\frac{\alpha(G_w)}{w(V)} = \frac{1}{1-e}\alpha^*(G)\end{aligned}$$ □

Hence, when we reach the limit of our computation power, we can remove the vertices of lowest weight and try to replace them by more useful vertices, for example by combining our new graph with a copy of itself or of a smaller graph.

Our current best bound on $m_1(\mathbb{R}^2)$ has surpassed the former best bound of 0.258795 by Keleti et al. [70]. It is interesting to note that the authors concluded their paper by saying that better bounds could probably be achieved by their methods but that they did not think it would allow to get below 0.257 which we did with ours.

Theorem 5.13.

$$m_1(\mathbb{R}^2) \leqslant 0.256828$$

Proof. A subgraph G of $\mathcal{G}(\mathbb{R}^2)$ that achieves $\alpha^*(G) \leqslant 0.256828$ is described in Appendix A. □

A lower bound on the fractional chromatic number of the plane follows, that improves the previous best bounds of 3.61904 by Cranston and Rabern [31] and of 3.75491 by Exoo and Ismailescu.

Theorem 5.14.

$$\chi_f(\mathbb{R}^2) \geqslant 3.89366.$$

5.4 Parallelohedron norms

In this section, we study the specific case of norms induced by parallelohedra and how to compute the optimal weighted independence ratio of the associated geometric graphs.

5.4.1 Λ-classes and k-regularity

Let $\mathcal{P}$ be a parallelohedron in $\mathbb{R}^n$, let Λ be the lattice associated to a face-to-face tiling by translation of $\mathbb{R}^n$ by $\mathcal{P}$ and let $G = (V, E)$ be a finite induced subgraph of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$. We know that $\Lambda + \frac{1}{2}\mathring{\mathcal{P}}$ (where $\mathring{\mathcal{P}}$ denotes the interior of $\mathcal{P}$) is a set avoiding distance 1 of density $\frac{1}{2^n}$ (see Figure 4.5). Thus, by the proof of Lemma 5.6, we know that for any weighting w of G, there exists $k \in \mathbb{R}^n$ such that $V \cap (k+\Lambda+\frac{1}{2}\mathring{\mathcal{P}})$ has a weight of at least $\frac{w(G)}{2^n}$.

To build a graph G of optimal weighted independence ratio $\frac{1}{2^n}$, we proceed in two steps:

- we first ensure that all the sets of the form $V \cap (k + \Lambda + \frac{1}{2}\mathring{\mathcal{P}})$ have weight at most $\frac{w(G)}{2^n}$;
- then, we ensure that no other independent set has a higher weight than the sets of the form $V \cap (k + \Lambda + \frac{1}{2}\mathring{\mathcal{P}})$.

We start by studying the vertex sets of the form $V \cap (k + \Lambda + \frac{1}{2}\mathring{\mathcal{P}})$.

Definition 5.15. Λ-equivalence, Λ-classes:

We define the relation $\sim_\Lambda$ on $\mathbb{R}^n$ as follows: for u and v in V, $u \sim_\Lambda v$ if and only if $u - v \in \Lambda$. Since Λ is a lattice, it is closed under addition. Thus, $\sim_\Lambda$ is transitive and is therefore an equivalence relation that we call Λ-*equivalence*. We call the equivalence classes of a vertex set V under this relation the Λ-*classes*.

Example 5.16.

Let $\mathcal{P}$ be the regular hexagon, let Λ be the lattice associated with a tiling of the plane by $\mathcal{P}$ (here, $\Lambda = A_2$, see Example 1.52) and let G and $\tilde{G}$ be the graph we built in the proof of Theorem 4.15. Let H and $\tilde{H}$ be their subgraph induced by the set V of vertices that are contained within the unit polytope. Like, in Example 5.7, we are only interested in the independent sets in H but since $\tilde{H}$ is easier to read, we depict $\tilde{H}$ in the figures (the independent sets in H being the set of vertices that avoid distance 2 in $\tilde{H}$).

We label the vertices of V as depicted in Figure 5.9. There are 12 Λ-classes in G, which all have at least one elements in the unit polytope, as illustrated in Figure 5.10. We label them $c_1, \ldots, c_{12}$. To make the vectors of Λ more apparent, we depict the Λ-classes on a bigger vertex set in Figure 5.11.

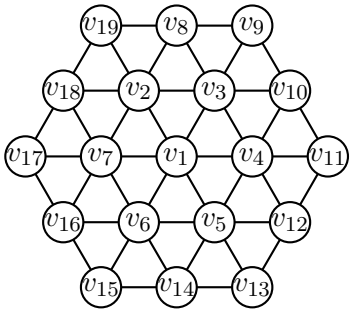


Figure 5.9: The graph $\tilde{H}$.

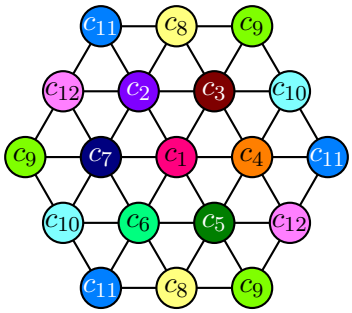


Figure 5.10: The Λ-classes in V.

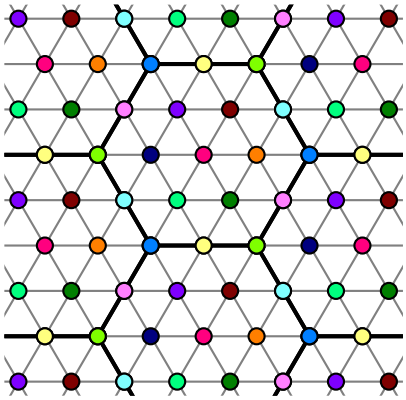


Figure 5.11: The Λ-classes in G.

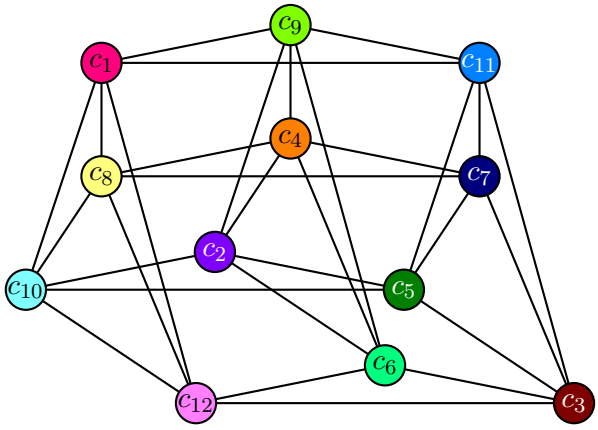


Figure 5.12: Incompatibility between the Λ-classes of G.

Definition 5.17. Compatibility:

We say that two Λ-classes c_1 and c_2 are *compatible* if and only if $\forall u \in c_1, \forall v \in c_2, uv \notin E$.

Example 5.18.

The Λ-classes c_6 and c_9 (see Figures 5.9 and 5.10) are incompatible because $v_6 \in c_6$, $v_{13} \in c_9$ and v_6 and v_{13} are at distance 1.

The Λ-classes c_2 and c_{11} are compatible.

Since a set of the form $(k + \Lambda + \frac{1}{2}\mathring{\mathcal{P}})$ is left unchanged by translation by a vector of Λ, if $S = V \cap (k + \Lambda + \frac{1}{2}\mathring{\mathcal{P}})$ contains a vertex u, it also contains every vertex v

of the Λ-class of u. Hence, if S is independent, it can only contains two vertices if they are from compatible Λ-classes.

Proposition 5.19. *If an independent vertex set S can be written under the form $V \cap (k + \Lambda + \frac{1}{2}\mathring{\mathcal{P}})$, then S is a union of compatible Λ-classes.*

Let us investigate the possible unions of compatible classes in our graph G. They are exactly the independent sets in the incompatibility graph of the Λ-classes of G depicted in Figure 5.12. Since the graph can be partitioned into three cliques $(\{c_1, c_8, c_{10}, c_{12}\}, \{c_9, c_4, c_2, c_6\}, \{c_{11}, c_7, c_5, c_3\})$, no independent set has size greater than 3. An independent set of the form $V \cap (k + \Lambda + \frac{1}{2}\mathring{\mathcal{P}})$ is therefore the union of at most three Λ-classes. This means that if all the Λ-classes have the same weight, the sets of the form $V \cap (k + \Lambda + \frac{1}{2}\mathring{\mathcal{P}})$ will have weight at most $\frac{3}{12} = \frac{1}{4}$, which is our goal.

The graph H has four orbits which are depicted in Figure 5.13. Since no Λ-class of H contains vertices of two different orbits (see Figure 5.10), the weighting w depicted in Figure 5.14 is the only symmetric weighting of H (up to multiplication by a constant) where all the Λ-classes have same weight.

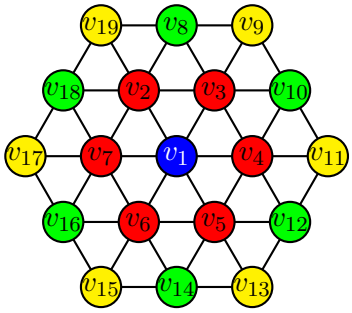


Figure 5.13: The orbits of H.

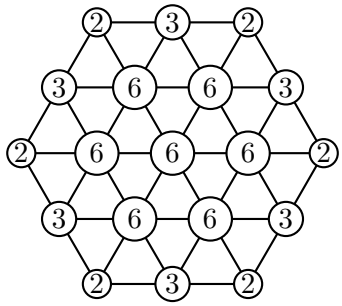


Figure 5.14: The weighting w.

We have now ensured that all the independent sets of the form $V \cap (k+\Lambda+\frac{1}{2}\mathring{\mathcal{P}})$ in H_w have weight at most 18 ($= \frac{w(H)}{4}$ since the graph has total weight 72). However, there are independent sets of weight up to 20 in H_w, one such example is depicted in Figure 5.15.

Taking a closer look at this independent set, we realize that it contains entirely the Λ-classes c_5, c_6 and c_8 but also contains the vertex $v_{19} \in c_{11}$ while c_{11} and c_5 are compatible (for example, v_5 and v_{15} are at geometric distance 1). Indeed, even if c_{11} and c_5 are incompatible, $v_{19} \in c_{11}$ is at distance 1 of no vertex of c_5 in H. This is the reason why H does not achieve an optimal weighted independence ratio of $\frac{1}{4}$, as explained in Example 5.7. We address this problem by adding to H vertices of c_5 that are at geometric distance 1 of v_{19}, as depicted in Figure 5.16. It remains to choose the weight of the new vertices.

Since c_5 and c_{11} are incompatible, we weight the vertices of our graph in such a way that no independent set on $c_5 \cup c_{11}$ is heavier than c_5. Since $\{v_5, v_{19}\}$ is independent and $w(v_{19}) = 2$ (which we cannot change because we have not added any vertex in c_{11}) we set $w(v_5) = 4$. Thus, to preserve $w(c_5) = 6$, we give a weight

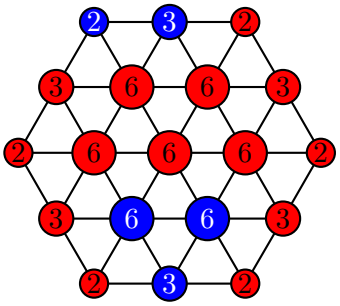


Figure 5.15: A maximum-weight independent set in H_w (in blue).

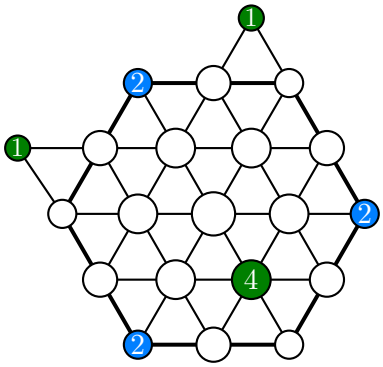


Figure 5.16: Two vertices we have to add to H. The colours denote the Λ-classes c_5 and c_{11}.

of 1 to our two new vertices. The two new vertices form with v_{11} and v_{15} another maximal independent set but it also has weight $6 = w(c_5)$.

By iterating this process on every vertex of the orbit of v_5, we build the weighted graph depicted in Figure 5.17 whose weighted independence ratio is $\frac{1}{4}$. Hence, this graph gives an alternative proof of Theorem 4.15, which states that $m_1(\mathbb{R}^2, \|\cdot\|_{\mathcal{P}}) = \frac{1}{4}$ if $\mathcal{P}$ is the regular hexagon.

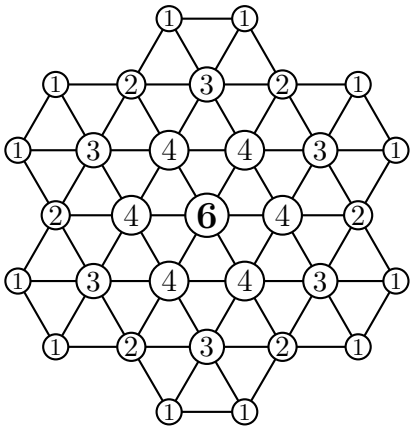


Figure 5.17: A weighted subgraph of G of weighted independence ratio $\frac{1}{4}$.

Our objective is now to generalize the method we have used with the regular hexagon to any parallelohedron norm. This requires first to describe more formally what we have done so far. To this end, we introduce the notion of k-regularity of an independent set:

Definition 5.20. k-regularity:

Let $\mathcal{P}$ be a n-dimensional parallelohedron, let Λ be a lattice associated to a tiling of $\mathbb{R}^n$ by $\mathcal{P}$ and let G be a subgraph of $\mathcal{G}(\mathbb{R}^n, \|\cdot\|_{\mathcal{P}})$. An independent set S in G is *k-regular* if and only if there exist k Λ-classes $c_1, \ldots, c_k$ in G such that:

- for all $i \in [\![1, k]\!]$, S contains all the vertices of c_i;
- for all $i \in [\![1, k]\!]$, S contains no vertex of any class incompatible with c_i.

Example 5.21.

The independent set S depicted in Figure 5.15 contains all the vertices of c_5, c_6 and c_8 but also contains a vertex of c_{11}, which is incompatible with c_5. It is therefore only 2-regular.

The set $S \setminus \{v_{19}\}$ is 3-regular. The most regular independent sets are always those of the form $V \cap (k + \Lambda + \frac{1}{2}\mathring{\mathcal{P}})$.

Since the incompatibility graph of the Λ-classes of G has independence number 3 (see Figure 5.12), we know that a 3-regular independent set is the union of three of the twelve Λ-classes of G and cannot contain any other vertex. By setting that all the Λ-classes must have the same weight, we therefore ensure that no 3-regular independent set has weight ratio greater than $\frac{1}{4}$. Note that this is not the only way to do so: any optimal weighting of the graph depicted in Figure 5.12 indicates a possible weighting of the Λ-classes such that no 3-regular independent set has weight ratio greater than $\frac{1}{4}$.

Note that the sets $S_1 = c_1 \cup c_2 \cup c_3$, $S_2 = c_4 \cup c_{11} \cup c_{12}$, $S_3 = c_7 \cup c_9 \cup c_{10}$ and $S_4 = c_5 \cup c_6 \cup c_8$ are independent and form a partition of G. Therefore, for any subgraph H of G of weighted independence ratio $\frac{1}{4}$, we must have $w(S_1) = w(S_2) = w(S_3 = w(S_4) = \frac{w(H)}{4}$. The same goes for the sets $S_5 = c_2 \cup c_3 \cup c_8$, $S_6 = c_1 \cup c_5 \cup c_6$, $S_7 = c_4 \cup c_{10} \cup c_{11}$ and $S_8 = c_7 \cup c_9 \cup c_{12}$. Since $w(S_1) = w(S_5)$, we find $w(c_1) = w(c_8)$. Similarly, we find $w(c_1) = w(c_8) = w(c_{10}) = w(c_{12})$, $w(c_2) = w(c_4) = w(c_6) = w(c_9)$ and $w(c_3) = w(c_5) = w(c_7) = w(c_{11})$ (see Figure 5.18). This condition is equivalent to the fact that no 3-regular independent set of H has weight ratio greater than $\frac{1}{4}$ and is weaker than our previous condition (all the Λ-classes must have the same weight). Figure 5.19 depicts a weighted subgraph of $\mathcal{G}(\mathbb{R}^2, \|\cdot\|_{\mathcal{P}})$ of weighted independence ratio $\frac{1}{4}$ where all the Λ-classes do not have the same weight.

However, weighted graphs that satisfy this property can still admit 2-regular independent set of weight ratio greater than $\frac{1}{4}$, as we saw in Figure 5.15. We avoid this problem by asking that for every incompatible Λ-classes c and c', no independent set on $c \cup c'$ has weight greater than $w(c)$. We will see why this worked in the case of the regular hexagon but we will also see that this condition is neither necessary (Example 5.22) nor sufficient (Example 5.23) in the general case.

Example 5.22.

In the weighted graph depicted in Figure 5.19, the classes c_1 and c_{11} are not compatible and c_{11} is an independent set on $c_1 \cup c_{11}$ of weight $6 > w(c_1) = 2$. However, this does not help build independent sets of weight ratio greater than $\frac{1}{4}$ because every union of three compatible Λ-classes that contains c_1 also contains another class that is not compatible with c_{11} and that makes it impossible to replace c_1 by an independent set on $c_1 \cup c_{11}$ of weight greater than $w(c_1)$. Note that in the graph depicted in Figure 5.14, both the sets $c_6 \cup c_8 \cup c_5$ and $c_6 \cup c_8 \cup c_{11}$ were independent and had weight ratio $\frac{1}{4}$ and this is why we could replace c_5 by any independent set of $c_5 \cup c_{11}$.

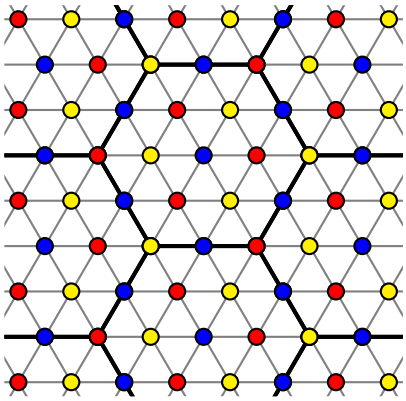


Figure 5.18: The lattice generated by the $\frac{1}{2}\mathcal{P}$ can be partitioned into three cosets as depicted here. Only the Λ-classes within a same coset must have the same weight.

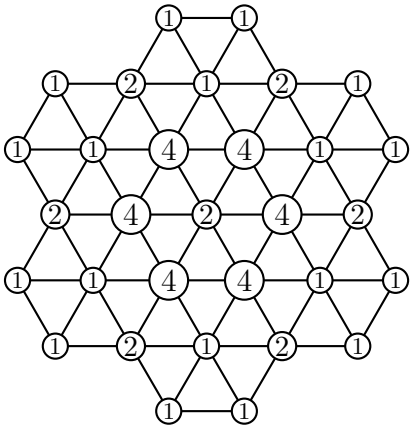


Figure 5.19: Here, all the Λ-classes within the coset depicted in blue in Figure 5.18 have weight 2 while the other have weight 6 but the graph still has a weighted independence ratio of $\frac{1}{4}$.

Example 5.23.

Let $\mathcal{P}'$ be an irregular Voronoï hexagon and let G' and $\tilde{G}'$ be the graph we built in the proof of Theorem 4.19. The auxiliary graph $\tilde{G}'$ has the same combinatorial structure than in the case of a regular hexagon but (Property D) does not hold anymore. We recall (Lemma 4.20) that the vertices of the graph can be partitioned into two sets A and B and that two vertices are at distance 1 from each other if there are at distance 2 in $\tilde{G}'$ and have a common neighbour in B. In Figure 5.20, vertices of A are depicted by squares and vertices of B are depicted by circles. Thus, the compatibility graph between the Λ-classes of G' can be obtained from the compatibility graph of G (Figure 5.12) by removing the edges c_9c_{11}, c_4c_7, c_2c_5 and c_6c_3. This graph still has independence number 3.

In the weighted subgraph of G' depicted in Figure 5.20 (which is the analogous of the graph we built in Figure 5.17), the Λ-classes all have same weight and for any two incompatible classes c and c', there is no independent set on $c \cup c'$ of weight greater than $w(c)$. However, Figure 5.20 depicts a 2-regular independent set of weight ratio greater than $\frac{1}{4}$.

The independent set S depicted in Figure 5.20 contains the entirety of c_9 and c_{11} and contains vertices from the classes c_8, c_{10} and c_{12}, which are compatible with c_9 and c_{11} but are pairwise incompatible. The classes c_8, c_{10} and c_{12} all have weight 6 and their respective intersections with S all have weight 3. As one can notice, there is no independent set on the union of two of these classes of weight greater than 6. However, note that a 3-regular set can contain at most of one these three classes and cannot contain any vertex from the other, which amounts to a weight of 6, while S contains an independent set of $c_8 \cup c_{10} \cup c_{12}$ of weight 9. This is a problem because the sets $c_9 \cup c_{11} \cup c_8$, $c_9 \cup c_{11} \cup c_{10}$ and $c_9 \cup c_{11} \cup c_{12}$ are all independent and we can therefore replace c_8 in the first one by any independent set on $c_8 \cup c_{10} \cup c_{12}$.

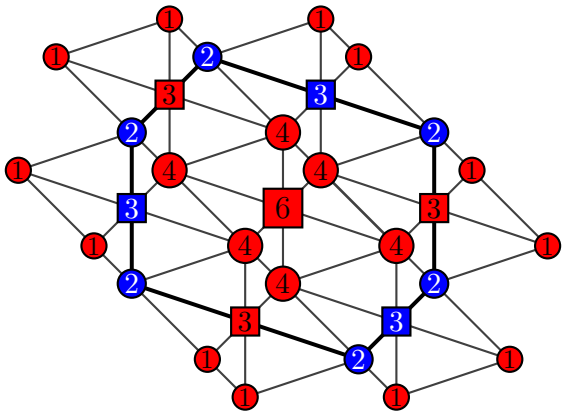


Figure 5.20: The set depicted in blue is independent and has weight ratio $\frac{21}{72} > \frac{1}{4}$.

Thus, if the incompatibility graph of our Λ-class has independence number 3, the condition on our weighting to ensure that no 2-regular independent set has weight ratio greater than $\frac{1}{4}$:
• that no 3-regular independent set has weight ratio greater than $\frac{1}{4}$ and
• that for all set of classes $C = \{c_1, \ldots, c_k\}$ such that there exist c and c' such that for all i, $c \cup c' \cup c_i$ is independent, there is no independent set on the union of the classes of C of weight greater than $w(c_1)$. We saw in Example 5.22 that not all pairs of incompatible classes satisfy this condition and in Example 5.23 that in the case of the irregular hexagon, this condition can be satisfied by sets C of size 3. In the case of the regular hexagon, the only such sets C are the pairs of Λ-classes from the same coset (see Figure 5.18).

At this point, we have found a weighting for which no 2-regular independent set has weight ratio greater than $\frac{1}{4}$. We then look for 1- and for 0-regular independent sets and find that their weight ratio is also upper bounded by $\frac{1}{4}$. Hence, we have found an optimal weighting of G.

5.4.2 The algorithm

This subsection presents an algorithm for the optimal weighted independence ratio (Algorithm 2) that uses the notion of k-regularity to improve Algorithm 1 in the case of a norm induced by a general parallelohedron.

We replace the linear program ($P_2(w)$) (which returns a maximum-weight independent set for a given weighting) by the following program ($P_3(k, w)$), which returns a maximum-weight k-regular independent set for a given integer k and weighting w.

Let $c_1, \ldots, c_\ell$ be the Λ-classes of our graph. For each vertex v, we use a binary variable x_v that indicates whether v belongs to the maximum-weight independent set S we create. For each Λ-class c_i, we use a binary variable E_i that indicates whether S contains a vertex of c_i and a variable C_i that indicates whether S contains all the vertices of c_i and no vertex from a class incompatible with c_i.

$$\begin{cases} \text{maximize } \sum_{v\in V} w_{\text{orbit}(v)} x_v \\ \forall uv \in E, x_u + x_v \leqslant 1 \\ \forall i \in [\![1,\ell]\!], \forall u \in c_i, E_i \geqslant x_u \\ \forall i,j \in [\![1,\ell]\!], \text{ if } c_j \text{ is incompatible with } c_i, C_i \leqslant 1 - E_j \\ \forall i \in [\![1,\ell]\!], \forall u \in c_i, C_i \leqslant x_u \\ \sum_{i=1}^{\ell} C_i \geqslant k \end{cases} \qquad (P_3(k,w))$$

The second line of the program ensures that S is independent, the third is the definition of E_i, fourth and fifth are the definition of C_i, the sixth ensures that S is k-regular and the objective function maximizes the weight of S.

Let k_0 be the independence number of the incompatibility graph of the Λ-classes. Hence, k_0 is the greatest value such that there exist k_0-regular independent sets. In all the graphs we study, $k_0 = \frac{\ell}{2^n}$ where ℓ is the number of Λ-classes and n is the dimension of our space. Like we did in Subsection 5.4.1, our algorithm consists of minimizing the maximum weight of k-regular independent sets for $k = k_0$ down to 0. For each value of k, we compute the optimal weighting with an algorithm similar to Algorithm 1 (with $(P_3(k,w))$ instead of $(P_2(w))$) but we do not reset the set of independent sets $\mathcal{S}$ when k decreases. Our algorithm is described in Algorithm 2.

Algorithm 2: Computing an optimal weighting of a geometric graph G for a parallelohedron norm.

```
Let O_1, ..., O_p be the orbits of vertices of G.
Let c_1, ..., c_ℓ be the Λ-classes of G.
Let 𝒮 = {} and for all j ∈ [[1,p]], w_j = 1/|V|. Let lb = 0.
for k = k_0 down to 0 do
    Let ub = 1.
    while ub ≠ lb do
        Let S be the independent set returned by (P_3(k,w)).
        𝒮 = 𝒮 ∪ {S}.
        ub = w(S)
        Let w be the weighting returned by (P_1(𝒮)) and let lb be the
        objective value.
return the w_j and ub.
```

At any step of the algorithm, lb and ub provide lower and upper bound on the minimum over all weightings of the graph of the maximum weight ratio reached by k-regular independent set. Since k-regular independent sets are just a specific case of independent sets, this weight ratio is smaller than $\alpha^*(G)$. Hence, lb gives a lower bound on $\alpha^*(G)$ at any step of the execution but ub only gives upper bounds on $\alpha^*(G)$ when $k = 0$ and needs to be reset each time k decreases.

Here again, the running time of the algorithm is lower bounded by the time it takes to find a maximum-weight independent set on G for a given optimal weighting w. Therefore, the maximal size of graph we can handle is not much higher than with Algorithm 1. However, for a given graph, Algorithm 2 is still significantly faster than Algorithm 1. Let us compare the two algorithms.

Algorithm 2 requires to compute the Λ-classes of the graph. However, this can be done in linear time in $O(n \times \ell)$ where n is the number of vertices of the graph and ℓ the number of Λ-classes: for each new vertex u, for each Λ-class c_i, we pick an element v of c_i and check whether $u - v \in \Lambda$; if it is the case, then $u \in c_i$; if u belongs to none of the c_i, we create a new Λ-class.

Algorithm 2 creates a bigger set $\mathscr{S}$ than Algorithm 1 and therefore uses $(P_3(k, w))$ more than Algorithm 1 uses $(P_2(w))$. Indeed, we saw in Subsection 5.3.1 that Algorithm 1 adds a vertex S to $\mathscr{S}$ only if there exists a symmetric weighting for which S is maximum, while Algorithm 2 adding a set S only means that S is maximum among the k-regular independent set for the current value of k. However, empirically, the difference between the sizes of $\mathscr{S}$ at the end of the two algorithms is never really important. Furthermore, as k decreases, the set of k-regular independent sets differs less and less from the sets of all independent sets. Thus, most of the useless sets in $\mathscr{S}$ are added for big values of k. This brings us to the most important difference between the two algorithms: the running time of $(P_3(k, w))$ is much higher for the small values of k than for the large ones. Note that for $k = 0$, $(P_3(k, w))$ has to find an maximum independent set in a graph of size n while it only has to find a maximum independent set in a graph of size l (the incompatibility graph between the Λ-classes) for $k = k_0$. The time required by $(P_3(k, w))$ is only relevant for very small values of k but the large majority of the sets in $\mathscr{S}$ are added for large values of k. For example, we saw in Subsection 5.4.1 that in the case of the regular hexagon, we had an optimal weighting after optimizing for $k = 3$ and $k = 2$ only. Thus, Algorithm 2 would have added sets to $\mathscr{S}$ for $k = 3$ and $k = 2$ and would only have needed one iteration of $(P_3(k, w))$ for $k = 1$ and $k = 0$, to ensure that the current weighting was already optimal. On the other hand, all the sets that Algorithm 1 adds to $\mathscr{S}$ are found by $(P_2(w))$ which is the case $k = 0$ of $(P_3(k, w))$.

Since Algorithm 2 creates a bigger set $\mathscr{S}$ than Algorithm 1, Algorithm 2 also requires more iteration of $(P_1(\mathscr{S}))$ than Algorithm 1 but the running time of $(P_1(\mathscr{S}))$ appears to be negligible in relation to $(P_2(w))$ and $(P_3(k, w))$ for small values of k.

We would like to conclude this subsection by pointing out that when the graph has optimal weighted independence ratio higher than $\frac{1}{2^n}$, the optimal weightings are not necessarily optimal among the k-regular independent sets for large values of k. For example, let H be the graph depicted in Figure 5.2a. The weighting depicted in Figure 5.2b is optimal while the weighting depicted in Figure 5.14 is not but the first one already has weight ratio strictly greater than $\frac{1}{4}$ with 3-regular independent sets while the latter does not. The purpose of optimizing the weighting for high values of k in the first steps of the algorithm is not to directly optimize the weighting for smaller values of k but to find quickly constraints (elements of $\mathscr{S}$) that will help us do so. This is also why we keep all the elements of $\mathscr{S}$ in memory and not the intermediate values of w.

5.4.3 Building finite graphs

In this subsection, we assume that we are given an infinite discrete graph G like the graph we built in the proof of Theorem 4.15 and we want to build a finite subgraph H of G with optimal weighted independence ratio as small as possible.

Proposition 5.11 indicates that in order to have a low optimal weighted independence ratio, the graph H that we create should contain as many vertices as possible. We assume in this subsection that we cannot compute the optimal weighted independence ratio of a graph of more than a given number of vertices that we call n_0 and we thus look for subgraph of G of size smaller than n_0 and of small optimal weighted independence ratio. We also assume that the graph G and n_0 are such that all the subgraphs H of n_0 vertices have geometric diameter significantly bigger than 1. To minimize the size of the independent sets, we want the graph to contain as many edges as possible, which is done by picking vertices that are close to each other. A method that works well in practice is to choose the set V_{d_0} of all the vertices that are at geometric distance less than d_0 from the origin, where d_0 is the highest value possible such that $|V_{d_0}| \leqslant n_0$. For example, the graph H we considered in Example 5.16 was induced by the set of vertices V_1. Such graphs also have the advantage of having several symmetries and thus, have few orbits compared to their number of vertices. However, especially when the vertex set of G has high density (*i.e.* many vertices in a small portion of space), there are cases where removing an orbit from H and replacing it by vertices further away from the origin can decrease $\alpha^*(H)$. This situation happens frequently in the 3-dimensional graphs we deal with in Subsection 5.4.4.

In the example we gave in Subsection 5.4.1, we started from a small graph H and added vertices to it (Figure 5.16) as we realized that we needed them to keep $\alpha^*(H)$ at $\frac{1}{4}$. This is not the approach that we choose in this subsection. Since G is infinite, we need to choose from the start a finite number of vertices that we may add to the graph we start from. This comes down to having those vertices in the graph in the first place, but with zero-weight. Algorithm 2 naturally starts by giving non-zero weight to as few vertices as possible (only one orbit in each Λ-class, as is the case in Figure 5.14) and then increases the number of weighted vertices if it helps decrease $\alpha^*(H)$ (this is what happens in Figure 5.16). Furthermore, we recall that the time that $(P_3(k, w))$ takes depends heavily on k. Thus, the running time of Algorithm 2 is decided by the number of vertices of the graphs on which we run $(P_3(k, w))$ for small values of k. Hence, unlike in Subsection 5.4.1, we would like to decrease the size of H as k decreases.

For $k \in [\![0, k_0]\!]$, let n_k be the maximum size of graphs for which we can run $(P_3(k, w))$ in practice. We also define d_k as the highest value such that $|V_{d_k}| \leqslant n_k$. We introduce the program $(P_4(\mathscr{S}, n_k))$ which, given a graph $G = (V, E)$, a collection $\mathscr{S} = \{S_1, \dots, S_\ell\}$ of independent sets and a maximum size n_k determines the best weighting of V where at most n_k vertices have non-zero weight. Let $O_1, \dots, O_p$ be the orbits of G and let $n_{i,j} = |S_i \cap O_j|$. For each orbit O_j, we create a variable w_j that indicates the weight of the vertices of O_j and a binary variable P_j that indicates if the vertices of O_j have non-zero weight. Our program is the following:

$$\begin{cases} \text{minimize } M \text{ subject to} \\ \qquad \sum_{j=1}^{p} w_j |O_j| = 1 \\ \forall i \in [\![1, \ell]\!], \sum_{j=1}^{p} n_{i,j} w_j \leqslant M \\ \qquad \forall j \in [\![1, p]\!], P_j \geqslant w_i \\ \qquad \sum_{j=1}^{p} |O_j| P_j \leqslant n_k \end{cases} \qquad (P_4(\mathscr{S}, n_k))$$

The objective and the first two constraints are the same as in $(P_1(\mathscr{S}))$. The third constraint is the definition of P_j and the fourth is the constraint on the number of the vertices of non-zero weight of the solution.

Given a infinite discrete graph G, Algorithm 3 builds a subgraph H of G of size smaller than n_0 and of small optimal weighted independence ratio. Note that this algorithm is a heuristic and we do not claim that no subgraph of G of size smaller than n_0 can have better optimal weighted independence ratio than the graph H returned by Algorithm 3.

Algorithm 3: Given a infinite discrete graph G and the numbers $n_0, \ldots, n_{k_0}$, looks for a subgraph of G of small optimal weighted independence ratio.

```
Let H be the graph induced by V_{d_{k_0}}.
Let O_1, ..., O_p be the orbits of V
Let c_1, ..., c_ℓ be the Λ-classes of G.
Let 𝒮 = {} and for all j ∈ [[1, p]], w_j = 1/|V|. Let lb = 0.
for k = k_0 down to 0 do
    if lb > 1/2^n then
        Let w be the weighting returned by (P_4(𝒮, n_k))
        Remove all the vertices of weight 0 from H
    else
        V(H) = V_{d_k}
    Let ub = 1.
    while ub ≠ lb do
        Let S be the independent set returned by (P_3(k, w)).
        𝒮 = 𝒮 ∪ {S}.
        ub = w(S)
        Let w be the weighting returned by (P_1(𝒮)) and let lb be the
        objective value.
return the w_j and ub.
```

The idea is to start from a large set of vertices V, that we can only deal with for large values of k. After each step of the **for** loop, we can use our current set

$\mathscr{S}$ to have a lower bound on the optimal weighted independence ratio of a graph by looking at the weighting that minimizes the sets of $\mathscr{S}$. This lower bound is the best we are able to provide on a graph of size n_k. We then use this estimation to find the best subset of V whose size will be manageable at the next step of the algorithm. The algorithm ends when V is small enough ($V = n_0$) to compute its optimal weighted independence ratio exactly.

Note that as long as there are weightings for which the k-regular independent sets have weight ratio $\frac{1}{2^n}$, there are generally too many weightings that can lead to the optimal ratio and some of them they might need very few vertices (for example, in the case of the regular hexagon, we only need to give non-zero weights to the vertices of the orbit depicted in blue and green in Figure 5.13 to ensure that no 3-regular independent set has weight ratio higher than $\frac{1}{4}$). We do not want to let $(P_4(\mathscr{S}, n_k))$ choose one of them arbitrarily and take the risk to remove vertices that are important to optimize k-regular independent sets for smaller values of k. In this situation, we choose to keep the vertices that are closest to the origin and therefore set $V(H) = V_{d_k}$.

Also note that our algorithm preserve the symmetries of the original set $V_{d_{k_0}}$.

5.4.4 The truncated octahedron

Finally, we discuss in this subsection how to build the infinite discrete graph G when the unit polytope is a truncated octahedron. We recall that the truncated octahedron is a generalization of all the parallelohedra in dimension 3 (see Subsection 1.4.4) and while the regular truncated octahedron is not a generalization of the other regular parallelohedra, it still appears to be the most difficult to deal with and is the only regular one for which the Bachoc-Robins conjecture is still open [92].

As in Chapter 4, the graphs we build in this subsection are unions of Λ-classes of $\mathbb{R}^3$. We choose in this subsection a finite number of Λ-classes to consider, and the algorithm of the previous subsection then builds a finite graph by choosing orbits among those Λ-classes.

Our aim is to generalize the constructions we used in Section 4.3 to prove the Bachoc-Robins conjecture in dimension 2. Let $\mathcal{P}$ be the regular truncated octahedron, let $V_\mathcal{P}$ be its vertex set and let Λ be the lattice of a tiling of $\mathbb{R}^3$ by $\mathcal{P}$. We recall that $\mathcal{P}$ is the permutohedron of order 4 (see Definition 1.60) and throughout this subsection, we use the coordinate system we described at the end of Subsection 1.4.4 to describe its vertices. We refer the reader to this part of the thesis for a description of the edges and faces of $\mathcal{P}$ with this coordinate system. We assume that the vertices of $\mathcal{P}$ are the permutation of $\{-3, -1, 1, 3\}$ so that $\mathcal{P}$ is centered at $\mathbf{0}$ and all its edges have same length.

As we did in Section 4.3, we can define the vertex set of our graph G as $V^{32} = \frac{1}{2}\Lambda + (V_\mathcal{P} \cup \{\mathbf{0}\})$. The set $\frac{1}{2}\Lambda$ contains eight Λ-classes and we then translate them by the vectors of $(V_\mathcal{P} \cup \{\mathbf{0}\})$. The set $(V_\mathcal{P} \cup \{\mathbf{0}\})$ contains the origin and the 24 vertices of the permutohedron, which are split between 6 Λ-classes. This set therefore contains seven Λ-classes but what matters here is that it only contains four $(\frac{1}{2}\Lambda)$-classes: indeed, if v and v' are two opposite vertices of a square face, they do not belong to the same Λ-class but their difference is a vector of $\frac{1}{2}\Lambda$ and thus, $v + \frac{1}{2}\Lambda = v' + \frac{1}{2}\Lambda$.

Hence, the set V^{32} contains 32 Λ-classes. It is also interesting to note that V^{32} is the lattice generated by the vertices of $\mathcal{P}$.

Unfortunately, we can prove that the graph induced by V^{32} has independence ratio $\frac{1}{4}$ and not $\frac{1}{8}$ as we hoped. Note that V^{32} contains the center of the faces of the polytopes of the tiling, but unlike the graphs that we built in the plane, it does not contain the middle of the edges of the polytopes. In the proof of Theorems 4.15, 4.24 and 4.28, we defined our vertex set as the lattice generated by $\frac{1}{2}V_{\mathcal{P}}$. This lattice contains the middle of the edges of the polytopes of the tiling and is much more likely to induce a graph of independence ratio $\frac{1}{8}$, but it contains $32 \times 2^3 = 256$ Λ-classes and is very difficult to manage.

Let $E_{\mathcal{P}}$ be the set of the middles of the edges of $\mathcal{P}$. The set $E_{\mathcal{P}}$ contains 36 vertices and 12 Λ-classes (since each edge belongs to 3 polytopes of the tiling). The set $V_{\mathcal{P}} \cup \{\mathbf{0}\} \cup E_{\mathcal{P}}$ thus contains 16 different Λ-classes which also define 16 different $(\frac{1}{2}\Lambda)$-classes. The set $V^{128} = (V_{\mathcal{P}} \cup \{\mathbf{0}\} \cup E_{\mathcal{P}}) + \frac{1}{2}\Lambda$ therefore has 128 Λ-classes. In practice, it takes significantly fewer vertices to achieve the same bounds with the graphs we build from V^{128} than with those we build from V^{256}.

Let $H = \{(a, b, c, d) \in \mathbb{R}^4 : a + b + c + d = 0\}$ be the 3-dimensional hyperplane of $\mathbb{R}^4$ that $\mathcal{P}$ is defined on. We note that the vertices of V^{32} are defined as sums of vertices whose coordinates all have same parity and therefore observe this property too. Hence, $V^{32} = H \cap ((2\mathbb{Z})^4 \cup (2\mathbb{Z} + 1)^4)$. The set V^{128} is actually $H \cap \mathbb{Z}^4$. We can build an intermediate set V^{64} by adding to $H \cap \mathbb{Z}^4$ the constraint that the first and second coordinates of the vertices must have same parity: $V^{64} = H \cap (((2\mathbb{Z})^2 \cup (2\mathbb{Z} + 1)^2)^2)$. An equivalent definition that is easier to generalize to non-regular case is $V^{64} = (V_{\mathcal{P}} \cup \{\mathbf{0}\} \cup E'_{\mathcal{P}}) + \frac{1}{2}\Lambda$ where $E'_{\mathcal{P}}$ is the union of the Λ-classes of the edges of one of the square (or parallelogram) face. The set V^{64} is the smallest that we have found for which we conjecture that the induced subgraph has independence ratio $\frac{1}{8}$. In practice, V^{64} and V^{128} are the vertex sets that provide the best results.

Our current best bound is achieved by the graph induced by the vertices of V^{128} at distance 1.5 or less from $\mathbf{0}$.

Theorem 5.24. *If $\mathcal{P}$ is the regular truncated octahedron,*

$$m_1(\mathbb{R}^3, \|\cdot\|_{\mathcal{P}}) \leqslant 0.130443$$

Since $\mathcal{P}$ is the last regular 3-dimensional parallelohedron for which the Bachoc-Robins conjecture is open, the bound of Theorem 5.24 holds for any regular 3-dimensional parallelohedron. Since the bound is smaller than $\frac{1}{7}$, this also implies that the chromatic number of $\mathbb{R}^3$ equipped with a regular parallelohedron norm is 8 (achieved by a colouring similar to the one described in Section 4.5 and depicted in Figure 4.16).

5.5 Conclusion

This chapter extends the work presented in Chapter 4 with a new method based on the notion of optimal weighted independence ratio. The flexibility of our approach and the efficiency of our algorithms allow us to significantly improve the best

known bounds on m_1 and the fractional chromatic number of several spaces, for both Euclidean and parallelohedron norms.

All the results presented in this chapter are very recent and the projects are still ongoing. Possibilities for future works are presented in Section 7.3.

Chapter 6

Complexity of locally-injective homomorphisms to tournaments

This chapter presents the results of [7], which is joint work with Stefan Bard, Christopher Duffy, Gary MacGillivray and Feiran Yang.

Contents

6.1 Introduction

6.1.1 Our problem

Given two graphs G and H, a homomorphism $f : G \to H$ is locally-injective if, for every $v \in V(G)$, it is injective when restricted to the neighbourhood of v. Depending on how we define the neighbourhood of a vertex, there exist several variants of this definition, especially in the oriented case.

The problem of determining the existence of a locally-injective homomorphism between two graphs has been widely studied in the cases of both undirected and of directed graphs. Locally-bijective and locally-injective homomorphisms are also widely studied under the names of *graph cover* and *partial graph cover* respectively. The algorithmic and complexity aspects of locally-injective homomorphisms for undirected graphs have been examined by a variety of authors and in a variety of contexts in [46], [47], [48], [49], [84] or [100] among others. The notions of injective colouring and injective chromatic number also derive from locally-injective homomorphism and have been studied in [26], [38] and [58] for example. Locally-injective homomorphisms of graphs find application in a range of areas including bio-informatics [16] [43] [45] and coding theory [58].

In this chapter, we consider locally-injective homomorphisms of oriented graphs (see Definition 1.2). We mentioned in Subsection 1.1.2 that the problem of graph

homomorphism is trivial if the target graph G has a loop on a vertex v since the function that maps all the vertices of F to v is a homomorphism from F to G. However, this homomorphism is not injective and the existence of a locally-injective homomorphism between two graphs is non-trivial even if the target graph is reflexive.

To define locally-injective homomorphisms of oriented graphs formally, one must choose the neighbourhood(s) on which the homomorphism must be injective. Up to symmetry, there are four natural choices:

(1) $N^-(v)$ (we can choose $N^+(v)$ instead but it leads to an equivalent problem);

(2) $N^+(v)$ and also $N^-(v)$;

(3) $N^+(v) \cup N^-(v)$;

(4) $N^+[v] \cup N^-[v] = N^+(v) \cup N^-(v) \cup \{v\}$.

If the target is irreflexive, (2), (3) and (4) are equivalent. Under (4), adjacent vertices must always be assigned different colours, and hence whether or not the target contains loops is irrelevant. Therefore, we may assume that targets are irreflexive when considering (4). Then, a locally-injective homomorphism to an irreflexive target satisfying (4) is equivalent to a locally-injective homomorphism to the same irreflexive target under either (2) or (3). As such, we need not consider (4). We are left with three possible definitions of locally-injective homomorphisms depending on whether we take (1), (2) or (3) as our injectivity requirement.

Definition 6.1. Locally-injective homomorphisms:
Given two graphs G and H, a homomorphism $f : G \to H$ is

(1) *in-injective* if and only if for all vertex v of G, it is injective on $N^-(v)$;

(2) *ios-injective* (for "in and out separately") if and only if for all vertex v of G, it is injective on $N^+(v)$ and on $N^-(v)$;

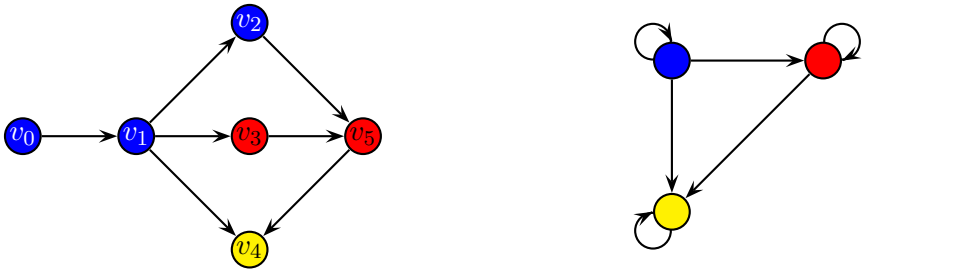
(3) *iot-injective* (for "in and out together") if and only if for all vertex v of G, it is injective on $N^+(v) \cup N^-(v)$.

In-injective, *ios-injective* and *iot-injective* n*-colourings* of a graph G are respectively in-injective, ios-injective and iot-injective homomorphisms between a graph G and a tournament T on n vertices.

Example 6.2.
Let us look for ios- and iot-injective colourings of the graph G from Example 1.13.

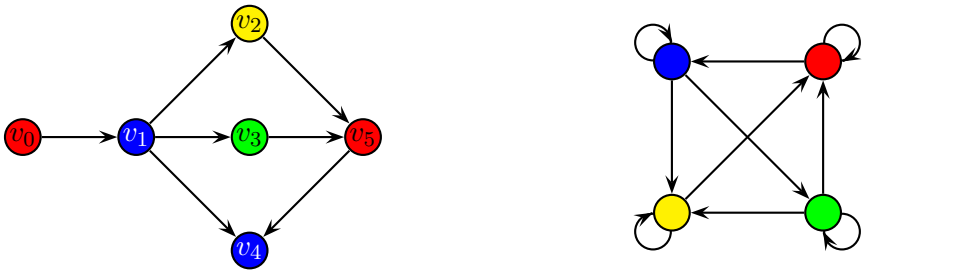
We can pick arbitrarily the colour of v_1, say blue. By injectivity, the vertices v_2, v_3 and v_4 must all receive different colours but since the target is reflexive, one of them can be blue.

In an ios-injective colouring of G, the vertex v_0 can be blue too since there is no other blue vertex in the in-neighbourhood of v_1. Figure 6.1 depicts a 3-ios-injective colouring of G and since $N^+(v_1)$ has size 3, this colouring is optimal.



(a) A 3-ios-injective colouring of G. (b) The associated target tournament.

Figure 6.1



(a) A 4-iot-injective colouring of G. (b) An associated target tournament.

Figure 6.2

In an iot-injective colouring of G, the vertex v_0 cannot be blue if there already is a blue vertex in the out-neighbourhood of v_1. Figure 6.2 depicts a 4-iot-injective colouring of G and since $N^+(v_1) \cup N^-(v_1)$ has size 4, this colouring is optimal.

When studying the complexity of a problem such as homomorphism to given graphs, the ideal objective would be to establish in which case the problem is polynomial and in which case it is not. However, since the problem is always in NP, establishing that it is not polynomial would require to prove that P $\neq$ NP, which seems out of reach with our current knowledge. Our objective is thus to establish what we call a *dichotomy theorem*. A dichotomy theorem consists of partitioning the problems in two sets and proving that the problems of the first set are polynomial on that those of the second are NP-complete.

The problem of in-injective homomorphism has been examined by MacGillivray, Raspaud, and Swarts in [83] and [84]. They give a dichotomy theorem for the problem of in-injective homomorphism to reflexive oriented graphs; and one for the problem of in-injective homomorphism to irreflexive tournaments. The problem of in-injective homomorphism to irreflexive oriented graphs H is shown to be NP-complete when the maximum in-degree of H, $\Delta^-(H)$, is at least 3, and Polynomial when $\Delta^-(H) = 1$. For the case $\Delta^-(H) = 2$ they show that every instance of directed graph homomorphism polynomially transforms to an instance of in-injective homomorphism to a target with maximum in-degree 2. As such the restriction of in-injective homomorphism to targets H so that $\Delta^-(H) = 2$ constitutes a rich class

of problems.

The remaining problems, ios-injective homomorphism and iot-injective homomorphism, are considered by Campbell, Clarke and MacGillivray in [19], [20] and [21] and their main results are presented in Subsection 6.1.2. In this chapter, we extend their results to provide dichotomy theorems (Theorem 6.14 and 6.26) for the restriction of the problems of iot-injective homomorphism and ios-injective homomorphism to reflexive tournaments.

6.1.2 Known results

For a fixed undirected graph H, the problem of determining whether an undirected graph G admits a homomorphism to H (i.e., *the H-colouring problem*) admits a well-known dichotomy theorem.

Theorem 6.3. *Hell-Nešetřil (1990) [61]*

Let H be an undirected graph.

- *If H is irreflexive and non-bipartite, then H-colouring is NP-complete.*
- *If H has a loop, or is bipartite, then H-colouring is Polynomial.*

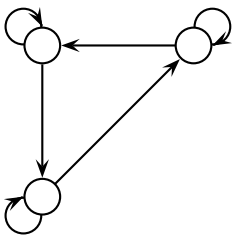
A dichotomy theorem for the complexity of H-colouring of directed graphs is given by Bulatov [18] and Zhuk [114].

For fixed small reflexive tournaments T, Campbell, Clarke and MacGillivray give the following result for the complexity of ios-injective T-colouring and iot-injective T-colouring.

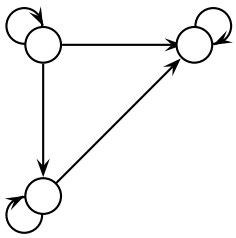
Theorem 6.4. *Campbell, Clarke and MacGillivray (2009) [19, 20, 21]*

If T is a reflexive tournament on 2 or fewer vertices, then ios-injective T-colouring and iot-injective T-colouring are Polynomial. If T is a reflexive tournament on 3 vertices, then ios-injective T-colouring and iot-injective T-colouring are NP-complete.

There are two reflexive tournaments on three vertices which are depicted in Figure 6.3.



(a) The oriented cycle of three vertices C_3.



(b) The transitive tournament on three vertices TT_3.

Figure 6.3: The two reflexive tournaments on three vertices.

Note that Theorem 6.4 only solves the complexity of a finite number of targets and prior to our work, no dichotomy results were known on infinite classes of tournaments.

In Section 6.2, we show that ios-injective homomorphism is NP-complete for any reflexive target tournament on 4 vertices or more and we thereby establish a dichotomy theorem for the complexity of ios-injective homomorphisms to reflexive tournaments (Theorem 6.14). In Section 6.3, we show that iot-injective homomorphism is also NP-complete for reflexive tournaments on 4 or more vertices which leads to a similar dichotomy theorem (Theorem 6.26).

6.2 Ios-injective homomorphisms

In this section we prove a dichotomy theorem for ios-injective T-colouring, where T is a reflexive tournament. In Subsection 6.2.1 and 6.2.2, we show respectively that ios-injective T_4-colouring and ios-injective T_5-colouring are NP-complete (Theorems 6.7 and 6.9) where T_4 and T_5 are the tournaments depicted in Figures 6.4 and 6.9. We then show in Subsection 6.2.3 that any instance of ios-injective T-colouring, where T is a reflexive tournament on at least 4 vertices, polynomially reduces to an instance of ios-injective T'-colourings, where T' is C_3, TT_3, T_4 or T_5 (see Figures 6.3, 6.4 and 6.9). The dichotomy theorem follows from combining these results with the result in Theorem 6.4.

6.2.1 Ios-injective T_4-colouring

We begin with a study of ios-injective T_4-colouring where T_4 is the graph depicted in Figure 6.4.

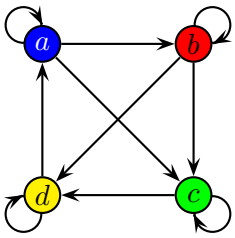


Figure 6.4: T_4, the only strongly connected reflexive tournament on four vertices.

To show that ios-injective T_4-colouring is NP-complete we provide a transformation from 3-edge-colouring (see Definition 1.14) subcubic graphs. A *subcubic* graph is an undirected graph whose maximum degree is 3. Holyer proves in [62] that 3-edge-colouring is NP-complete even when restricted to subcubic graphs.

We construct an oriented graph H from a graph G so that G has a 3-edge-colouring if and only if H admits an ios-injective homomorphism to T_4. The key ingredients in this construction are a pair of oriented graphs, H_x and H_e, given in Figures 6.5 and 6.6, respectively.

We first establish a few preliminary results on T_4-colourings of our gadgets.

Lemma 6.5. *In any ios-injective T_4-colouring of H_x (depicted in Figure 6.5):*

1. *the vertices* $3, 13$ *and* 23 *are coloured a;*

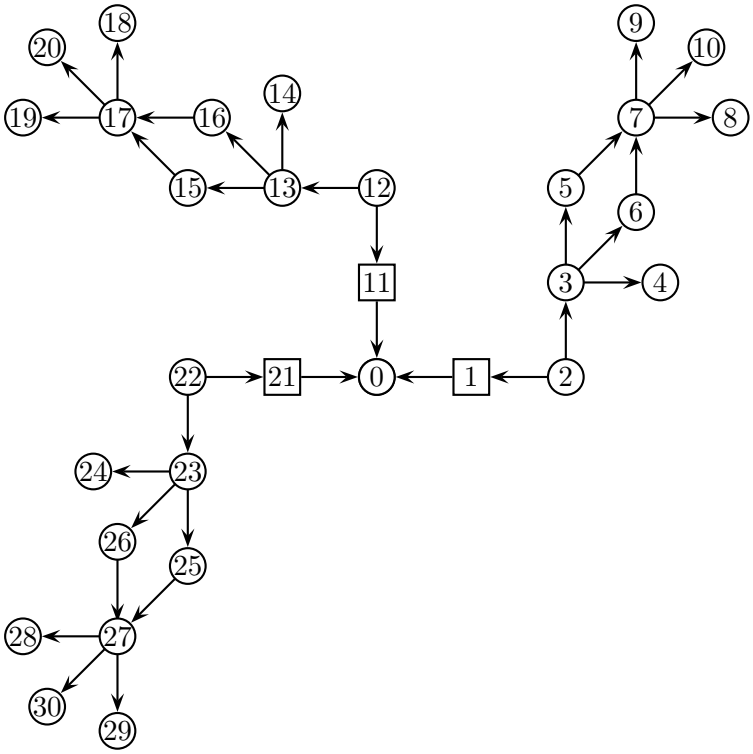


Figure 6.5: H_x.

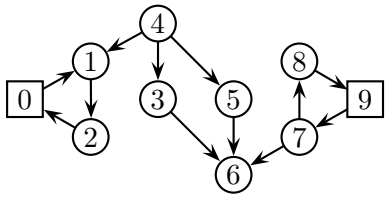


Figure 6.6: H_e.

2. vertex 0 *is coloured* d.

Proof. (1) By symmetry, it suffices to show the claim for vertex 3. Let us first note that the vertices 3 and 7 have out-degree 3 and can therefore only be coloured a or b, as these are the only vertices of out-degree 3 in T_4. If vertex 7 is coloured a, then its two in-neighbours, vertices 5 and 6, are coloured d and a. However, this is impossible as no vertex of out-degree three in T_4 has both d and a as out-neighbours. Hence, vertex 7 is coloured b. If vertex 3 is coloured b, then vertices 5 and 6 would be both in- and out-neighbours of vertices coloured b. Thus, each of vertices 5 and 6 are coloured b. This is a violation of the injectivity requirement. Therefore, vertex 3 (and by symmetry, the vertices 13 and 23) must be coloured a.

(2) Notice that the square vertices in the graph H_x (vertices 1, 11 and 21) cannot be coloured a; they each have an in-neighbour that already has an out-neighbour coloured a. These square vertices have a common out-neighbour and so must receive distinct colours by the injectivity requirement. As none is coloured a, these three vertices are coloured b, c and d, in some order. The only vertex that is an out-neighbour of b, c and d in T_4 is d. And so, the common out-neighbour of vertices 1, 11 and 21 (i.e., vertex 0) has colour d. □

Lemma 6.6. *Let H'_e be an oriented graph formed from a copy of H_e (Figure 6.6)and two copies of H_x (Figure 6.5)by identifying vertex 0 in H_e with any square vertex in one copy of H_x and identifying vertex 9 in H_e with any square vertex in the other copy of H_x. In any ios-injective T_4-colouring of H'_e, the vertices 0 and 9 in the subgraph induced by H_e have the same colour.*

Proof. Let H'_e be constructed as described. Consider an ios-injective T_4-colouring of H'_e. We examine the colours of the vertices in the subgraph induced by the copy of H_e. By Lemma 6.5 and the construction of H'_e, vertices 0 and 9 each have an in-neighbour that has an out-neighbour coloured a. By the injectivity requirement, neither vertex 0 nor 9 is coloured a. We proceed in cases to show that vertices 0 and 9 receive the same colour.

Case I: vertex 0 is coloured b. Vertex 1 cannot be coloured d as vertex 0 already has an out-neighbour coloured d (vertex 0 in a copy of H_x). Vertex 1 cannot be coloured c as no 3-cycle of T_4 contains both a vertex coloured b and a vertex coloured c. Thus, vertex 1 must be coloured b. The vertex 2 is both an in-neighbour and an out-neighbour of vertices coloured b and is therefore coloured b. The vertex 4 is an in-neighbour of vertex 1, and so cannot be coloured b as vertex 1 already has an in-neighbour coloured b. The vertex 4 must thus be coloured a. By injectivity, the out-neighbours of vertex 4 must receive distinct colours that are out-neighbours of a in T_4. Therefore, vertices 3 and 5 are coloured a and c in some order, as vertex 1 is coloured b. The only common out-neighbour of a and c in T_4 is c. As such, the vertex 6 must be coloured c. By injectivity, each of the in-neighbours of vertex 6 must receive distinct colours that are in-neighbours of c in T_4. And so vertex 7 must be coloured b. As vertex 9 cannot be coloured a and it has an out-neighbour coloured b, namely vertex 7, we have that vertex 9 must be coloured b. Thus, vertices 0 and 9 have the same colour.

Case II: vertex 0 is coloured c. Vertex 1 cannot be coloured d as vertex 0 already has an out-neighbour coloured d (vertex 0 in a copy of H_x). Since the out-neighbours of c in T_4 are c and d, vertex 1 is coloured c.

The vertex 4 has an out-neighbour coloured c, and so must be coloured a or b or c. Since vertex 0 is coloured c, vertex 4 cannot be coloured c without violating injectivity. We claim vertex 4 is coloured b.

By way of contradiction, suppose that vertex 4 is coloured a. Then by injectivity, vertices 3 and 5 are coloured a and b, in some order. The only out-neighbour of a

and c in T_4 that has in-degree 3 is c. As such vertex 6 is coloured c. The vertex c in T_4 has three in-neighbours – a, b, and c. As vertex 6 has in-neighbours coloured a and b (namely, vertices 3 and 5), then by injectivity the third in-neighbour of vertex 6 (namely, vertex 7) is coloured c. In T_4, c has two out-neighbours: c and d. Since vertex 7 is coloured c and already has an out-neighbour coloured c, vertex 8 must be coloured d. The vertex 9 has an in-neighbour coloured d. Only vertices a and d in T_4 have d as an in-neighbour. Therefore, vertex 9 is coloured with a or d. However, we have shown previously that vertex 9 cannot be coloured a. This implies, that vertex 9 is coloured d. However, vertex 9 has an out-neighbour coloured c. Since c is not an out-neighbour of d in T_4, we arrive at a contradiction. Thus, vertex 4 is not coloured a. Therefore, vertex 4 is coloured b.

Since vertex 4 is coloured b, vertices 3 and 5 are coloured b and d, in some order. The only common out-neighbour of b and d in T_4 is d. Therefore, vertex 6 is coloured d. Hence, by injectivity, the vertex 7 is coloured c. Since vertex 7 has an out-neighbour coloured d, vertex 8, another out-neighbour of vertex 7 must be coloured c. Since 9 has both an in-neighbour and an out-neighbour coloured c, the vertex 9 must be coloured c. Thus, vertices 0 and 9 have the same colour.

Case III: Vertex 0 is coloured d. Vertex 1 cannot be coloured d as vertex 0 already has an out-neighbour coloured d (vertex 0 in a copy of H_x). Vertex d has two out-neighbours in T_4: a and d. Therefore, vertex 1 is coloured a.

The vertex 4 has out-degree 3 and an out-neighbour coloured a. Vertex a is the only vertex in T_4 to have out-degree 3 and have a as an out-neighbour. Therefore, vertex 4 is coloured a. By injectivity the vertices 3 and 5, the remaining out-neighbours of vertex 4, are coloured b and c. The vertex 7 cannot be coloured d since 9 already has an out-neighbour coloured d (vertex 37 in a copy of H_x). Moreover, vertex 7 is an in-neighbour of 6, which already has in-neighbours coloured c and b. Hence, the vertex 7 must be coloured a. In T_4 the only in-neighbours of a are a and d. Thus, vertex 9 is coloured a or d. Since vertex 9 has an out-neighbour coloured d, it cannot be coloured a. Therefore, vertex 9 is coloured d, as required. □

We can now prove the main theorem of this subsection.

Theorem 6.7.
The problem of ios-injective T_4-colouring is NP-complete.

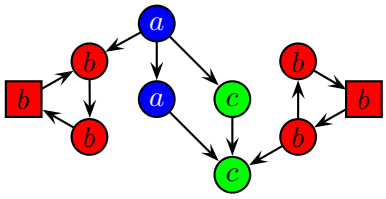
Proof. The transformation is from 3-edge-colouring of subcubic graphs.

Let G be a graph with maximum degree at most 3 and let $\tilde{G}$ be an arbitrary orientation of G. We create an oriented graph H from $\tilde{G}$ as follows. For every $x \in V(G)$ we add H_x, a copy of the oriented graph given in Figure 6.5, to H. For every arc $e \in E(\tilde{G})$ we add H_e, a copy of the oriented graph given in Figure 6.6, to H. To complete the construction of H, for each arc $e = uv \in E(\tilde{G})$ we identify the vertex 0 in H_e with one of the three square vertices (i.e., vertices 1, 11, or 21) in H_u and identify the vertex 9 in H_e with one of the three square vertices in H_v. We identify these vertices in such a way that each square vertex in a copy of H_x is identified with at most one square vertex from a copy of H_e. We note that this is always possible as vertices in G have degree at most three.

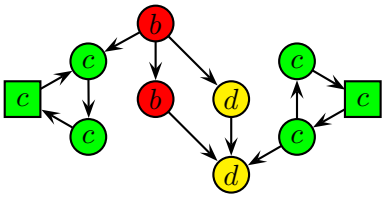
We claim G has a 3-edge-colouring if and only if H has an ios-injective T_4-colouring. The proof is in two parts:

• Suppose that an ios-injective T_4-colouring of H is given. This ios-injective T_4-colouring induces a 3-edge-colouring of G: the colour of an edge in $e \in E(G)$ is given by colour of vertices 0 and 9 in corresponding copy of H_e contained in H. By Lemma 6.6, this colour is well-defined. Recall that for each copy of H_x, the vertices 1, 11 and 21 are respectively identified with either vertex 0 or vertex 9 in some copy of H_e. Therefore by Lemma 6.5, each of the edges incident with any vertex receive different colours and no more than 3 colours, namely b, c, and d, are used on the edges of G.

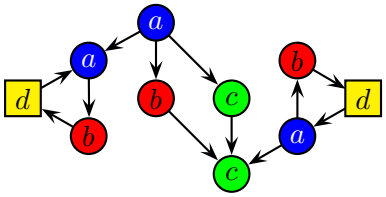
• Suppose that a 3-edge-colouring of G, $f : E(G) \to \{b, c, d\}$ is given. For each $e \in E(G)$, we colour H_e using one of the ios-injective T_4-colourings given in Figure 6.7. We choose the colouring of each copy of H_e so that vertices 0 and 9 in that copy are assigned the colour $f(e)$. To complete the proof, we show that such a colouring can be extended to all copies of H_x contained in H.



(a) The colouring of H_e when $f(e) = b$.



(b) The colouring of H_e when $f(e) = c$.



(c) The colouring of H_e when $f(e) = d$.

Figure 6.7

The vertices 1, 11 and 21 of each copy of H_x may be identified with either vertex 0 or vertex 9 of some copy of H_e. Since f is a 3-edge-colouring of G, if we colour each copy of H_e using Figure 6.7, the vertices 1, 11 and 21 of a same copy of H_x all receive distinct colours from the set $\{b, c, d\}$. By symmetry of H_x, we can assume without loss of generality that for all vertex x, an edge incident to x and coloured b uses the vertex 1 of H_x, an edge coloured c uses the vertex 11 and an edge coloured d uses the vertex 21. The ios-injective T_4-colouring given in Figure 6.8 extends a pre-colouring of the vertices 1, 11 and 21 with colours b, c, and d, respectively, to an ios-injective T_4-colouring of H_x. Therefore, G has a 3-edge-colouring if and only if H admits an ios-injective T_4-colouring

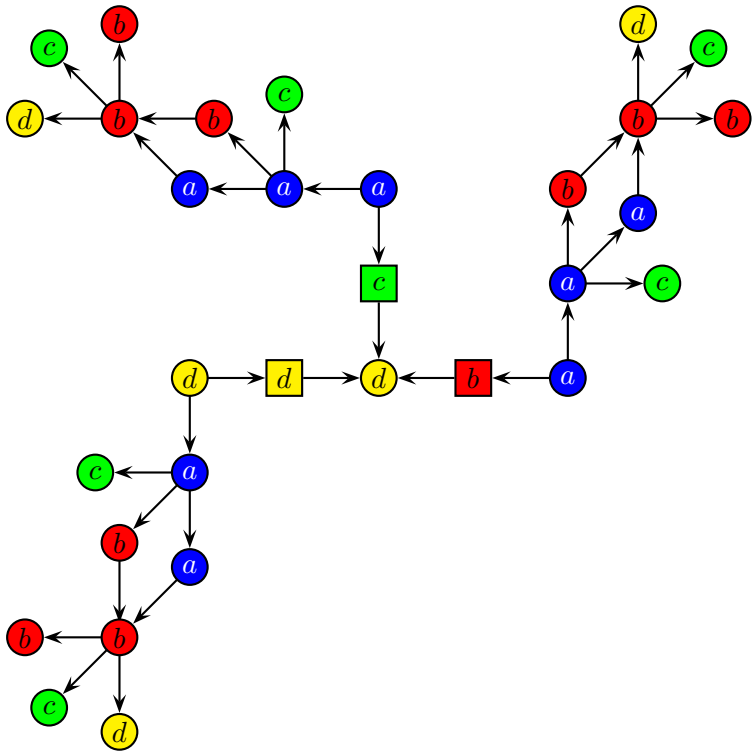


Figure 6.8: A colouring of H_x.

Since the construction of H can be carried out in polynomial time, ios-injective T_4-colouring is NP-complete □

6.2.2 Ios-injective T_5-colouring

We now prove the NP-completeness of ios-injective T_5-colouring where T_5 is the graph depicted in Figure 6.9.

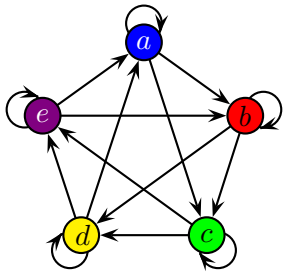


Figure 6.9: T_5, the only reflexive tournament on five vertices where all the vertices have in-degree and out-degree three.

The transformation is from ios-injective C_3-colouring (see Theorem 6.4). We construct an oriented graph J from a graph G so that G admits an ios-injective

homomorphism to C_3 if and only if J admits an ios-injective homomorphism to T_5. The key ingredient in this construction is the oriented graph J_v, given in Figure 6.10.

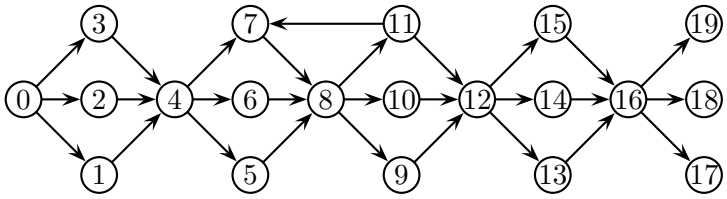


Figure 6.10: J_v.

For each $n > 0$ we construct an oriented graph J_n from n copies of J_v, say $J_{v_0}, J_{v_1}, \ldots, J_{v_{n-1}}$, by letting vertices 17, 18 and 19 of J_{v_i} be in-neighbours of vertex 0 in $J_{v_{i+1 \pmod n}}$ for all $i \in [\![0, n-1]\!]$.

Like previously, we start be studying the ios-injective T_5-colourings of J_n.

Lemma 6.8.

For any positive integer n, in an oriented ios-injective T_5-colouring of J_n, each of the vertices labelled 0 (respectively, 4, 8, 12 and 16) receive the same colour.

Proof. Since T_5 is vertex-transitive, assume without loss of generality that vertex 0 in J_{v_0} receives colour a. If 0 is coloured a, then the vertices 1, 2 and 3 must be coloured a, b and c in some order since these vertices are the only out-neighbours of a in T_5. Since vertex c is the only common out-neighbour of vertices a, b and c in T_5, vertex 4 must be coloured c. Since the automorphism of T_5 that maps a to c also maps c to e, we conclude by a similar argument that vertex 8 is coloured e. Similarly, we conclude that vertex 12 is coloured b and vertex 16 is coloured d.

Since vertex 16 is coloured d in J_{v_0}, vertices $17, 18, 19$ are coloured, in some order, a, e, d, as these are the only out-neighbours of d in T_5. The only common out-neighbour of a, e and d in T_5 is a. Therefore, vertex 0 in J_{v_1} is coloured a. Repeating this argument, we conclude that vertices 4, 8, 12 and 16 in J_{v_2} receive colours c, e, b and d, respectively. Continuing in this fashion gives that in an oriented ios-injective T_5-colouring of J_n each of the vertices labelled 0 (respectively, 4, 8, 12 and 16) receive the same colour. □

Theorem 6.9.

The problem of ios-injective T_5-colouring is NP-complete.

Proof. We prove the NP-completeness of ios-injective T_5-colouring by reducing ios-injective C_3-colouring, which is proved NP-complete in [20] (see Theorem 6.4).

Let G be a graph with vertex set $\{v_0, v_1, \ldots, v_{|V(G)|-1}\}$. Let $\nu_G = |V(G)|$. We construct J from G by first adding a copy of J_{ν_G} to G and then, for each $i \in [\![1, \nu_G]\!]$, adding an arc from vertex 11 in J_{v_i} to v_i.

We show that J has an ios-injective T_5-colouring if and only if G has an ios-injective C_3-colouring.

Consider an ios-injective T_5-colouring of J. Since T_5 is vertex-transitive we can assume without loss of generality that the vertex 8 in each copy of J_v is coloured a.

Therefore, in each J_{v_i}, vertices 9, 10 and 11 are coloured, in some order, with colours a, b, c; and vertex 12 is coloured c.

We claim that v_i is coloured with b, d or e for all $i \in [\![0, \nu_G - 1]\!]$. If v_i is coloured a, then vertex 11 in J_{v_i} has both an in-neighbour and an out-neighbour coloured a and is therefore coloured a. Thus, vertex 7 in J_{v_i} also has both an in- and an out-neighbour coloured a and must be coloured a. Since vertex 11 already has an out-neighbour coloured a, this contradicts the injectivity requirement. If v_i has colour c, then vertex 11 in J_{v_i} has two out-neighbours coloured c, which is also a violation of the injectivity requirement.

Therefore, v_i is coloured with one of b, d or e for each $i \in [\![0, \nu_G - 1]\!]$. Since vertices b, d and e of T_5 induce a copy of C_3 in T_5, restricting an ios-injective T_5-colouring of J to the vertices of G yields an ios-injective C_3-colouring of G.

Let β be an ios-injective C_3-colouring of G using colours b, d and e. We extend such a colouring to be an ios-injective T_5-colouring of J by assigning to the vertices of each J_{v_i} colours based upon $\beta(v_i)$ as shown in Figure 6.11. Every vertex $v_i \in V(G)$ is the out-neighbour of vertex 11 of J_{v_i} but this cannot lead to a violation of the injectivity requirement since vertex 11 of J_{v_i} is always coloured either a or c while the vertices of G are coloured b, d or e.

Therefore, J has an ios-injective T_5-colouring if and only if G has an ios-injective C_3-colouring. Since J can be constructed in polynomial time, ios-injective T_5-colouring is NP-complete. □

6.2.3 Dichotomy theorem

We now present a reduction to instances of ios-injective T-colouring for when T has a vertex v of out-degree at least four. This reduction allows us to polynomially transform an instance of ios-injective T-colouring to an instance of ios-injective T'-colouring, where T' is T_4, T_5, C_3 or TT_3.

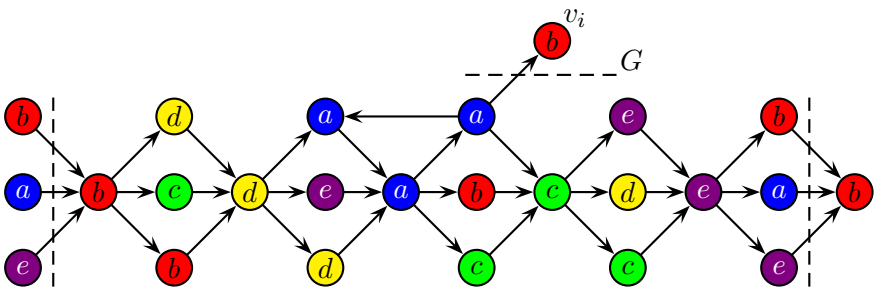
Lemma 6.10. *If T is a reflexive tournament on n vertices with a vertex v of out-degree at least four, then ios-injective homomorphism to T' polynomially transforms to ios-injective homomorphism to T, where T' is the tournament induced by the strict out-neighbourhood of v.*

Proof. Let T be a reflexive tournament on n vertices with a vertex v of out-degree at least four. Let G be an oriented graph with vertex set $\{w_0, w_1, \ldots, w_{|V(G)|-1}\}$. Let $\nu_G = |V(G)|$. We construct H from G by adding to G

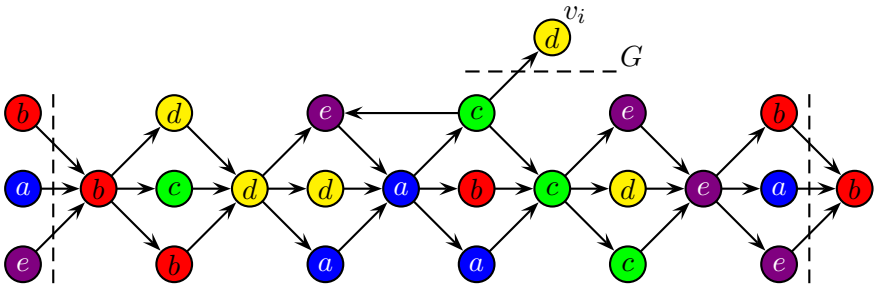
- vertices $x_0, x_1, \ldots, x_{\nu_G - 1}$;
- an arc from x_i to w_i for all $i \in [\![0, \nu_G - 1]\!]$;
- ν_G irreflexive copies of T, labelled T_i, for all $i \in [\![0, \nu_G - 1]\!]$.

Let $v_i \in T_i$ be the vertex corresponding to $v \in V(T)$. We complete our construction by adding the arcs $v_i x_i$ and $x_i v_{i+1 \pmod{\nu_G}}$ for all i. This construction is illustrated in Figure 6.12.

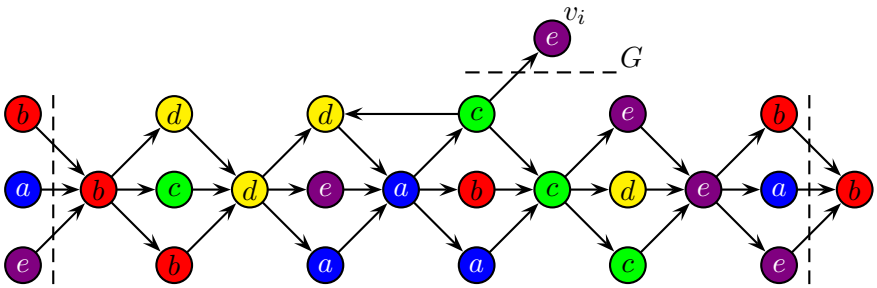
Proposition 6.11. *In an ios-injective T-colouring of H no two vertices of T_i have the same colour.*



(a) A colouring of the vertices of J_{v_i} when $\beta(v_i) = b$.



(b) A colouring of the vertices of J_{v_i} when $\beta(v_i) = d$.



(c) A colouring of the vertices of J_{v_i} when $\beta(v_i) = e$.

Figure 6.11

Proof. Since T has a vertex of out-degree at least 4 we observe that T has at least 4 vertices. By way of contradiction, assume that two vertices x and y receive the same colour c and let z be a third vertex of T_i. By injectivity, x and y cannot both be in- or out-neighbours of z. Thus, z has an in-neighbour and an out-neighbour coloured c which is only possible if z itself is coloured c. Let w be a fourth vertex of T_i. Since x, y and z are all neighbours of w, w has three strict neighbours with the same colour, which is impossible in an ios-colouring of T_i. □

Proposition 6.12. *In an ios-injective T-colouring of H, every vertex of $\{x_0, x_1, \ldots, x_{\nu_G-1}\} \cup \{v_0, v_1, \ldots, v_{\nu_G-1}\}$ receives the same colour.*

Proof. By Proposition 6.11, all the colours of T are used exactly once in each T_i.

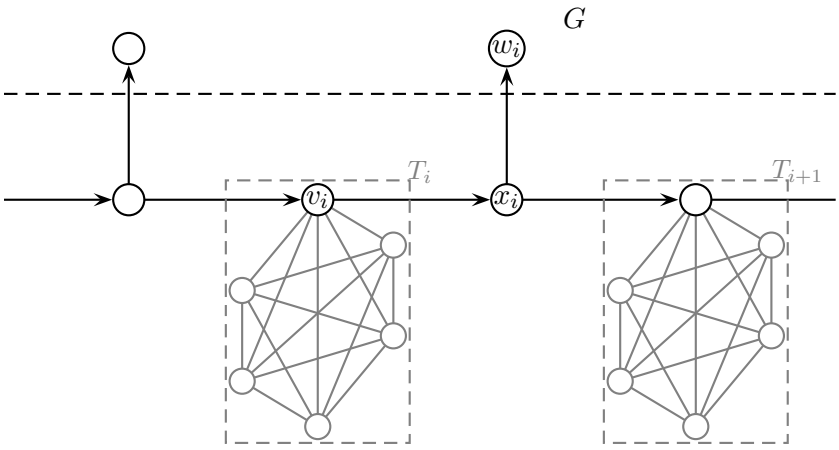


Figure 6.12: The construction of H in Lemma 6.10.

Therefore, the only possible colour for an out-neighbour or an in-neighbour of v_i outside of T_i is $\varphi(v_i)$. Thus, for each i, we have $\varphi(v_i) = \varphi(x_i) = \varphi(x_{i+1 \pmod{\nu_G}})$. The proposition follows. □

Since T_0 is a copy of T, we know that an ios-injective homomorphism φ maps v_0 to a vertex of the same orbit as v. We may assume without loss of generality that $\varphi(v_0) = v$. Thus, by Proposition 6.12, $\varphi(v_i) = v$ for all $i \in [\![0, \nu_G - 1]\!]$.

Let T' be the reflexive tournament induced by the strict out-neighbourhood of v. Note that T' is a reflexive tournament on at least 3 vertices and an induced subgraph of T. We show that H has an ios-injective T-colouring if and only if G has an ios-injective T'-colouring.

Let φ be an ios-injective T-colouring of H. By our previous claim, each x_i has an in-neighbour and an out-neighbour with colour v, namely $v_i \in V(T_i)$ and $v_{i-1} \in V(T_{i-1})$. Therefore, $\varphi(w_i)$ is an out-neighbour of v in T. That is, $\varphi(w_i) \in V(T')$. Therefore, the restriction of φ to the vertices of G yields an ios-injective T'-colouring of G.

Let β be an ios-injective T'-colouring of G. For all $i \in [\![0, \nu_G - 1]\!]$ and all $u \in V(T)$. Let $u_i \in V(T_i)$ be the vertex corresponding to $u \in V(T)$.

We extend β to be an ios-injective T-colouring of H as follows:

- $\beta(x_i) = \beta(v_i) = v$ for all $i \in [\![0, \nu_G - 1]\!]$;
- for all $u_i \in T_i$, let $\beta(u_i) = u$.

Hence, ios-injective T'-colouring of G can be polynomially reduced to ios-injective T-colouring of H. □

If T is a reflexive tournament with a vertex of in-degree at least 4, a similar argument holds. We modify the construction by reversing the arc between x_i and w_i in the construction of H.

Lemma 6.13. *If T is a reflexive tournament on n vertices with a vertex v of in-degree at least four, then ios-injective homomorphism to T' polynomially transforms*

to ios-injective homomorphism to T, where T' is the tournament induced by the strict in-neighbourhood of v.

Our results compile to give a dichotomy theorem.

Theorem 6.14.
Let T be a reflexive tournament. If T has at least 3 vertices, then the problem of deciding whether a given oriented graph G has an ios-injective homomorphism to T is NP-complete. If T has 1 or 2 vertices, then the problem is Polynomial.

Proof. If T is a reflexive tournament on no more than three vertices, the result follows by Theorem 6.4. Suppose then that T has four or more vertices. If $T = T_4$, or if $T = T_5$, then the result follows from Theorem 6.7 or Theorem 6.9. Up to isomorphism, there are 16 distinct reflexive tournaments on 4 or 5 vertices. By inspection, tournaments T_4 and T_5 respectively are the only reflexive tournaments on 4 and 5 vertices respectively with no vertex of out-degree or in-degree four. Since the average out-degree of a reflexive tournament on $n > 5$ vertices is $\frac{n-1}{2} + 1 > 3$, every reflexive tournament on at least six vertices has a vertex with out-degree at least four. Therefore, if T has at least four vertices, $T \neq T_4$ and $T \neq T_5$, then T has a vertex with either in-degree or out-degree at least four. By repeated application of Lemma 6.10 and Lemma 6.13, an instance of ios-injective homomorphism to T polynomially transforms to instance of either ios-injective homomorphism to T_4; ios-injective homomorphism to T_5 or ios-injective homomorphism to a target on 3 vertices. □

6.3 Iot-injective homomorphisms

As is the case with ios-injectivity, it has been proven in [19, 20] that the problem is Polynomial if T has two vertices or fewer and that the problem is NP-complete if T is one of the two reflexive tournaments on three vertices. To extend this result, we employ similar methods as the ones used in the case of ios-injective T-colouring in the previous section. Again, we show respectively in Subsection 6.3.1 and 6.3.2 that iot-injective T_4-colouring and iot-injective T_5-colouring are NP-complete (Theorems 6.18 and 6.20). In Subsection 6.3.3, we are able to prove a lemma analagous to Lemma 6.10. Dichotomy theorem 6.26 follows.

6.3.1 Iot-injective T_4-colouring

We begin with a study of iot-injective T_4-colouring (where T_4 is the graph depicted in Figure 6.4). To show iot-injective T_4-colouring is NP-complete we reduce 3-edge-colouring. We construct an oriented graph F from a graph G so that G has a 3-edge-colouring if and only if F admits an iot-injective homomorphism to T_4. The key ingredients in this construction are a pair of oriented graphs; F_x and F_e given in Figures 6.13 and 6.14, respectively.

Lemma 6.15. *In any iot-injective T_4-colouring of F_x (Figure 6.13), vertex 0 is coloured b and vertex 4 is coloured a.*

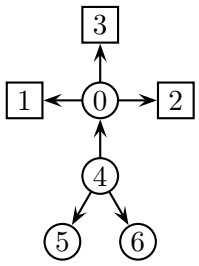


Figure 6.13: F_x.

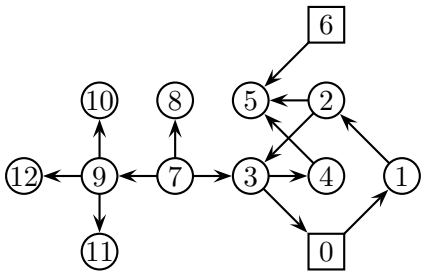


Figure 6.14: F_e.

Proof. Consider some iot-injective T_4-colouring of F. Vertex 0 of F_x has out-degree 3. Since each vertex of T_4 has at most three out-neighbours (including itself), vertex 0 must have the same colour as one of its out-neighbours. To satisfy the injectivity constraint, if a colour appears on an out-neighbour of vertex 0, that colour cannot appear on an in-neighbour of vertex 0. Therefore, vertex 4 does not have the same colour as vertex 0. Both vertices 4 and 0 have out-degree 3, and there is an arc from 4 to 0. Vertex a in T_4 is the only vertex to have out-degree 3 and have a strict out-neighbour with out-degree 3. Hence, vertex 4 is coloured a and vertex 0 is coloured b. □

Lemma 6.16. *In any iot-injective T_4-colouring of F_e (Figure 6.14), vertex 7 is coloured a and vertex 9 is coloured b.*

Proof. This proof is similar to the proof of Lemma 6.15 since the subgraph of F_e induced vertices 3, 7, 8, 9, 10, 11 and 12 is isomorphic to F_x. □

Lemma 6.17. *Let F'_e be the oriented graph formed from a copy of F_e and two copies of F_x by identifying vertex 0 in the copy of F_e with any square vertex in one copy of F_x and identifying vertex 6 in the copy of F_e with any square vertex in the other copy of F_x. In any iot-injective T_4-colouring of F'_e, vertices 0 and 6 in the subgraph induced by F_e have the same colour, and are coloured with one of b, c or d.*

Proof. Let F'_e be constructed as described. Consider an iot-injective T_4-colouring of F'_e. We examine the colours of the vertices in the subgraph induced by the copy of F_e. By Lemma 6.15 and the construction of F'_e, vertices 0 and 6 in F_e have an in-neighbour coloured b – vertex 0 in a copy of F_x. Since b is not an in-neighbour of a in T_4, vertex 0 in the copy of F_e must receive one of the colours b, c, or d. We proceed in cases based on the possible colour of vertex 0 in copy of F_e.

Case I: Vertex 0 is coloured b. Since the vertex 0 already has a neighbour coloured b (vertex 0 in a copy of F_x), vertex 3, an in-neighbour of vertex 0 in F_e, cannot be coloured b. Since $b \in V(T_4)$ has only a and b as in-neighbours, vertex 3 must be coloured a. By Lemma 6.16 vertex 3 has a neighbour coloured a – namely vertex 7. By the injectivity constraint, this colour cannot appear on any other neighbour of vertex 3. As such, vertices 2 and 4 are coloured d and c respectively. The only common out-neighbour of d and c in T_4 is d. Therefore, vertex 5 has colour

d. In T_4, vertex d has three in-neighbours – b, c and d. Since c and d both appear in the in-neighbourhood of vertex 5, vertex 6 must be coloured b.

Case II: Vertex 0 is coloured c. Vertex c in T_4 has three in-neighbours: a, b and c. Since vertex 0 has an in-neighbour coloured b, namely the vertex 0 in a copy of F_x, vertex 3 in F_e must have either colour a or colour c.

By way of contradiction, assume that vertex 3 is coloured c. In this case, the injectivity constraint implies that vertex 1 is not coloured c. Since c and d are the only out-neighbours of c in T_4, vertex 1 must be coloured d. Vertex 2, an in-neighbour of vertex 3, is coloured with one of a, b or c, the in-neighbours of c in T_4. By Lemma 6.16, vertex 3 has a neighbour coloured a – vertex 7. By assumption, vertex 0 has colour c. Therefore, by injectivity, vertex 2 has colour b. This is a contradiction, as the arc between vertex 1 and vertex 2 does not have the same direction as the arc between vertex c and vertex b in T_4. Therefore, vertex 3 is coloured a.

In T_4, the in-neighbours of a are a and d, and the out-neighbours of a are a, b and c. Since vertex 7 is coloured a, no other neighbour of vertex 3 can be coloured a. Therefore, vertex 2, an in-neighbour of vertex 3, must have colour d. Since vertex 0 is coloured c, vertex 4, an out-neighbour of vertex 3 must have colour b. Vertex 5, a common in-neighbour of vertices 2 and 4, must be coloured with a common in-neighbour of d and b in T_4. The only such vertex in T_4 is d. Hence, vertex 5 has colour d.

Vertex d in T_4 has three in-neighbours: b, c and d. Since vertex 5 already has in-neighbours coloured b and d (vertices 4 and 2 respectively), vertex 6 must be coloured c, as required.

Case III: Vertex 0 is coloured d. Recall by Lemma 6.16 that vertex 3 has a neighbour coloured a – vertex 7. Since vertex 0 is coloured d, vertex 3 is coloured with a vertex that is an out-neighbour of a and an in-neighbour of d in T_4. The only such vertices are b and c. However, vertex 0 has a neighbour coloured b (vertex 0 in a copy of F_x). Therefore, vertex 3 has colour c. Vertex 4 must have a colour that is an out-neighbour of c in T_4. The only such colours are c and d. Since vertex 0, an out-neighbour of vertex 3, is coloured d, vertex 4 must be coloured c. Vertex 2 must have a colour that is an in-neighbour of c in T_4. The only such colours a, b and c. Vertex 7, an in-neighbour of vertex 3, has colour a. Vertex 4, a neighbour of vertex 3, has colour c. Therefore, by injectivity vertex 2 has colour b. Vertex 5 must be coloured with a common out-neighbour of b and c in T_4. The only such colours are c and d. Since vertex 3, an out-neighbour of vertex 2, has colour c, we have by injectivity that vertex 5 has colour d. The in-neighbours of vertex 5 must be coloured with the in-neighbours of d in T_4. Vertex d has three in-neighbours in T_4 – b, c and d. Since vertex 2 has colour b and vertex 4 has colour c, we have by injectivity that vertex 6 has colour d. □

Theorem 6.18. *The problem of iot-injective T_4-colouring is NP-complete.*

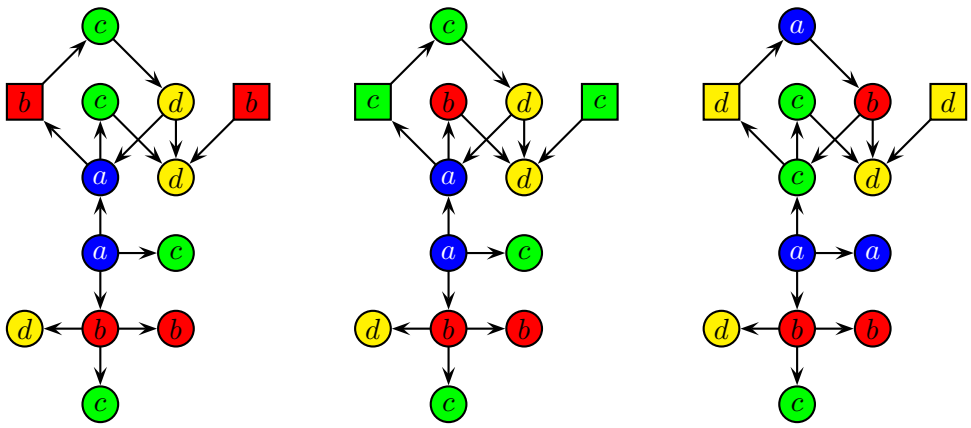
Proof. The reduction is from 3-edge-colouring subcubic graphs.

Let G be a graph with maximum degree at most 3 and let $\tilde{G}$ be an arbitrary orientation of G. We create an oriented graph F from $\tilde{G}$ as follows. For every $v \in V(G)$ we add F_v, a copy of the oriented graph given in Figure 6.13, to F. For every arc $uv \in E(\tilde{G})$, we add F_{uv}, a copy of the oriented graph given in Figure 6.14, to F. To complete the construction of F, for each arc $uv \in E(\tilde{G})$ we identify the vertex 0 in F_{uv} with one of the three square vertices (i.e., vertices 1, 2, or 3) in F_u and identify the vertex 6 in F_{uv} with one of the three square vertices in F_v. We identify these vertices in such a way that each square vertex in a copy of F_x is identified with at most one square vertex from a copy of F_e. We note that this is always possible as vertices in G have degree at most three.

We claim that G has a 3-edge-colouring if and only if F has an iot-injective T_4-colouring.

Suppose an iot-injective T_4-colouring of F is given. This iot-injective T_4-colouring induces a 3-edge-colouring of G: the colour of an edge in $uv \in E(G)$ is given by colour of vertices 0 and 6 in corresponding copy of F_{uv} contained in F. By Lemma 6.17, this colour is well-defined, and is one of b, c, or d. Recall that for each copy of F_x, the vertices 1,2 and 3 are respectively each identified with either vertex 0 or vertex 6 in some copy of F_e. By Lemma 6.15, vertices 1, 2 and 3 in a copy of F_x cannot be coloured a. By injectivity, vertices 1, 2 and 3 in a copy of F_x all are assigned different colours. Therefore, each of the edges incident with any vertex receive different colours and no more than 3 colours, namely b, c, and d, are used on the edges of G. By definition, this induces a 3-edge-colouring of G.

Suppose that a 3-edge-colouring of G, $f : E(G) \to \{b, c, d\}$ is given. For each $uv \in E(G)$, we colour F_{uv} using one of the iot-injective T_4-colourings given in Figure 6.15. We choose the colouring of each F_{uv} so that vertices 0 and 6 are assigned the colour $f(e)$.



(a) The colouring of F_e when $f(e) = b$.

(b) The colouring of F_e when $f(e) = c$.

(c) The colouring of F_e when $f(e) = d$.

Figure 6.15

To complete the proof, we show that such colouring can be extended to all copies

of F_x contained in F. Recall that for each copy of F_x, the vertices 1, 2 and 3 are all identified with either vertex 0 or vertex 9 of some copy of F_e. Since f is a 3-edge-colouring of G, for each $x \in V(G)$, each of the vertices 1, 2 and 3 in F_x are coloured with distinct colours from the set $\{b, c, d\}$ when we colour each copy of F_e using Figure 6.15.

By symmetry of H_x, we can assume without loss of generality that for all vertex x, an edge incident to x and coloured b uses the vertex 1, an edge coloured c uses the vertex 2 and an edge coloured d uses the vertex 3.

The iot-injective T_4-colouring given in Figure 6.16 extends a pre-colouring of the vertices 1, 2 and 3 with colours b, c, and d, respectively, to an iot-injective T_4-colouring of F_x.

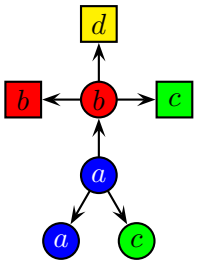


Figure 6.16: A colouring of F_x.

Thus, F has an iot-injective T_4-colouring.

Since the construction of F can be carried out in polynomial time, iot-injective T_4-colouring is NP-complete. □

6.3.2 Iot-injective T_5-colouring

We now prove the NP-completeness of iot-injective T_5-colouring. The reduction is from iot-injective C_3-colouring. We construct an oriented graph D from a graph G so that G admits an iot-injective homomorphism to C_3 if and only if D admits an iot-injective homomorphism to T_5. The key ingredient in the construction is the oriented graph, D_v, given in Figure 6.17.

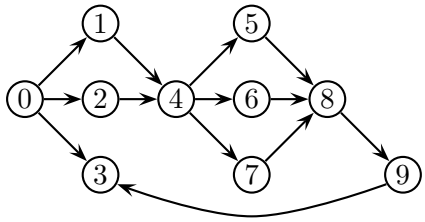


Figure 6.17: D_v.

For each $n > 0$, let D_n be the oriented graph constructed from n disjoint copies of D_v, say $D_{v_0}, D_{v_1}, \ldots, D_{v_{n-1}}$, by letting vertex 8 of D_{v_i} be an in-neighbour of vertex 0 in $D_{v_{i+1 \pmod n}}$ for all $i \in [\![0, n-1]\!]$.

Lemma 6.19.

For any positive integer n, up to automorphism, in an oriented iot-injective T_5-colouring of D_n each of the vertices labelled 0 receive the colour d, each of the vertices labelled 4 receive the colour a, and each of the vertices labelled 8 receive the colour c.

Proof. Since T_5 is vertex-transitive, assume without loss of generality that vertex 0 in D_{v_0} receives colour d in some iot-injective T_5-colouring of D_n. Observe that vertex 4 has three out-neighbours. Since each vertex of T_5 has at most three out-neighbours (including itself), vertex 4 must have the same colour as one of its out-neighbours. To satisfy the injectivity constraint, no in-neighbour of vertex 4 has the same colour as vertex 4. Furthermore, vertex 4 has two in-neighbours, vertices 1 and 2, that are out-neighbours of a vertex coloured d. Only vertices a and b in T_5 have two in-neighbours that are out-neighbours of d. Therefore, vertex 4 has colour a or b.

By way of contradiction, assume that vertex 4 has colour b. Then vertices 1 and 2 are coloured with the vertices of T_5 that are out-neighbours of d and in-neighbours of b. The only such vertices in T_5 that satisfy these criteria are a and e. Therefore, vertices 1 and 2 are coloured with a and e, in some order. Vertex d has three out-neighbours in $T_5 - a, d$ and e. Since vertices 1 and 2 are coloured with a and e, in some order, the third out-neighbour of vertex 0, vertex 3, is coloured with d. Vertex b in T_5 has three out-neighbours – b, c and d. Therefore, the out-neighbours of vertex 4, vertices $5, 6$ and 7, are coloured, in some order, with these colours. Vertices b, c and d in T_5 have only d as a common out-neighbour. Hence, the common out-neighbour of vertices $5, 6$ and 7, vertex 8, is coloured d. This is a contradiction, as now vertex 9 has two vertices coloured d in its neighbourhood. Therefore, vertex 4 has colour a.

Vertex a in T_5 has three out-neighbours – a, b and c. Thus, the out-neighbours of vertex 4 are coloured with a, b and c, in some order. The only common out-neighbour of a, b and c in T_5 is c. Therefore, vertex 8 has colour c. This implies that vertex 9 in D_{v_0} and 0 in D_{v_1} have colours from the set $\{c, d, e\}$, the out-neighbours of c in T_5. Since vertex 8 has a neighbour coloured c, neither vertex 0 in D_{v_1} nor 9 (in D_{v_0}) can have this colour. Furthermore, since vertex 3 has a neighbour coloured d, vertex 9 has cannot be coloured d. Thus, vertex 9 in D_{v_0} has colour e and vertex 0 in D_{v_1} has colour d.

Repeating this argument implies that every vertex labelled 0 has colour d. □

Theorem 6.20. *The problem of iot-injective T_5-colouring is NP-complete.*

Proof. The reduction is from iot-injective C_3-colouring.

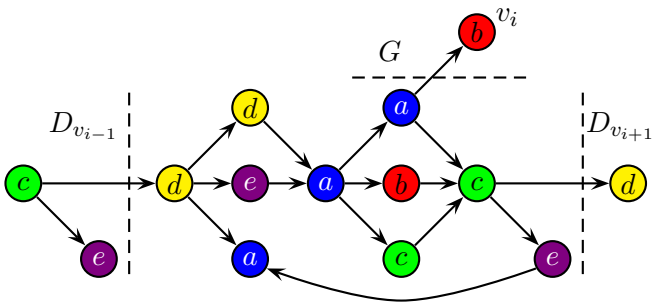
Let G be an oriented graph with vertex set $\{v_0, v_1, \ldots, v_{|V(G)|-1}\}$. Let $\nu_G = |V(G)|$. We construct D from G by first adding a copy of D_{ν_G} to G and then, for each $i \in [\![0, \nu_G - 1]\!]$, adding an arc from vertex 5 in D_{v_i} to v_i.

We show that D has an iot-injective T_5-colouring if and only if G has an iot-injective C_3-colouring.

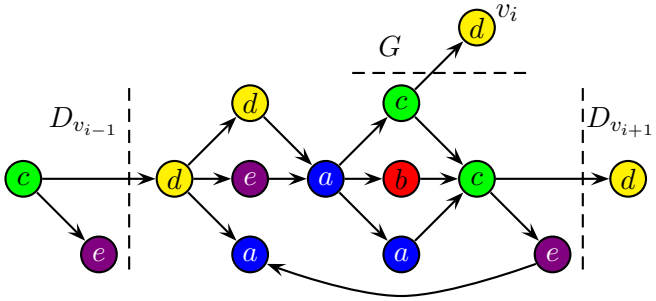
Let φ be an iot-injective T_5-colouring of D. Since T_5 is vertex-transitive, we may assume that vertex 0 in D_{v_0} has colour d. By Lemma 6.19, for all $i \in [\![0, \nu_G - 1]\!]$, the vertex in D_{v_i} labelled 0 has colour d, the vertex labelled 4 has colour a and the

vertex labelled 8 has colour c. By the injectivity requirement, the neighbours of the vertex labelled 5 in each copy of D_v have distinct colours. Since the vertices 4 and 8 have colours a and c, respectively, only colours b, d and e can appear at v_i, for all $i \in [\![0, \nu_G - 1]\!]$. Since b, d and e induce a copy of C_3 in T_5, we conclude that the restriction of φ to the vertices of G is indeed an iot-injective C_3-colouring.

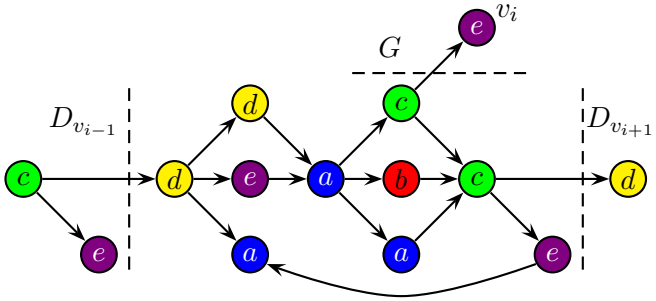
Let β be an iot-injective C_3-colouring of G using colours b, d and e.We extend such a colouring to be an iot-injective T_5-colouring of D by assigning to the vertices of each D_{v_i} colours based upon $\beta(v_i)$ as shown in Figure 6.18.



(a) A colouring of D_{v_i} when $\beta(v_i) = b$.



(b) A colouring of D_{v_i} when $\beta(v_i) = d$.



(c) A colouring of D_{v_i} when $\beta(v_i) = e$.

Figure 6.18

Note that vertex 5 of D_{v_i} is always coloured either a or c. Since the other neighbours of v_i are vertices of G which are thus coloured b, d or e, this colouring observes the injectivity requirement.

Therefore, D has an iot-injective T_5-colouring if and only if G has an iot-injective C_3-colouring. As D can be constructed in polynomial time, iot-injective T_5-colouring is NP-complete. □

6.3.3 Dichotomy theorem

We now present a reduction to instances of iot-injective T-colouring for when T has a vertex v of out-degree at least four. This reduction allows us to polynomially transform an instance of iot-injective T-colouring to an instance of iot-injective T'-colouring, where T' is T_4, T_5, C_3 or TT_3.

Lemma 6.21. *If T is a reflexive tournament on n vertices with a vertex v of out-degree at least four, then iot-injective homomorphism to T' polynomially transforms to iot-injective homomorphism to T, where T' is the tournament induced by the strict out-neighbourhood of v.*

Proof. Let T be a reflexive tournament on n vertices with a fixed vertex v of out-degree four or more. Let $T^\star$ be the graph obtained by removing from T all the arcs with their tail at v.

Let G be an oriented graph with vertex set $\{w_0, w_1, \ldots, w_{|V(G)|-1}\}$. Let $\nu_G = |V(G)|$. We construct C from G by adding to G

- ν_G disjoint irreflexive copies of $T : T_0, T_1, \ldots, T_{\nu_G-1}$;
- ν_G disjoint irreflexive copies of $T^\star : T_0^\star, T_1^\star, \ldots, T_{\nu_G-1}^\star$;
- and for all $u \in V(T)$ where $u \neq v$, an arc from the vertex corresponding to u in $T_{i-1}^\star$ to the vertex corresponding to u in T_i, for all $i \in [\![0, \nu_G - 1]\!]$.

Let v_i and $v_i^\star$ be the vertices corresponding to v in T_i and $T_i^\star$, respectively. We complete the construction of C by adding an arc from v_i to $v_i^\star$ for all $i \in [\![0, \nu_G - 1]\!]$. This construction is illustrated in Figure 6.19.

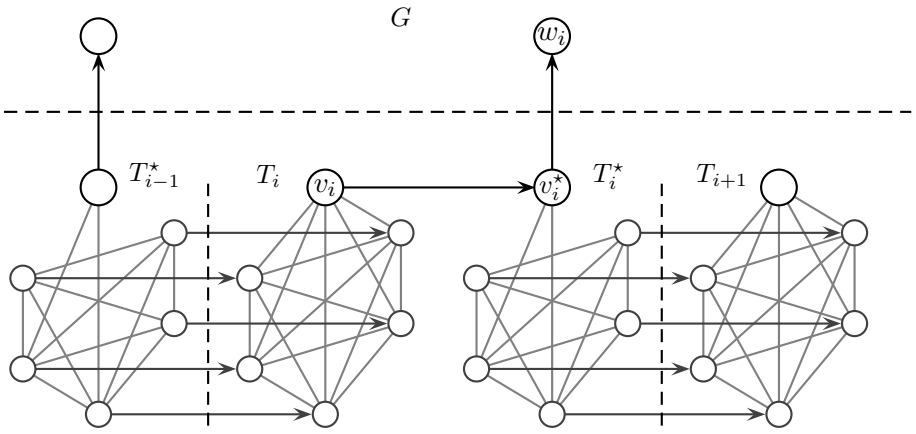


Figure 6.19: The construction of C in Lemma 6.21.

We first need to prove a few preliminary results.

Proposition 6.22.
In an iot-injective T-colouring of C, no two vertices of T_i have the same colour.

Proof. If two vertices of T_i are assigned the same colour, then a common neighbour of such vertices in T_i has a pair of neighbours with the same colour. This is a violation of the injectivity requirement. Therefore, no two vertices of T_i are assigned the same colour. □

Proposition 6.23.
In an iot-injective T-colouring of C, v_i and $v_i^\star$ have the same colour.

Proof. Since v_i has n neighbours, in any iot-injectve T-colouring of C, v_i is assigned the same colour as one of its neighbours. By the previous claim, the neighbour of v_i that has the same colour as v_i must be $v_i^\star$. □

Proposition 6.24.
In an iot-injective T-colouring of C, v_i and v_{i+1} have the same colour.

Proof. If v has out-degree n in T, then, by construction, v_i and v_{i+1} each have out-degree n in C. Since no two out-neighbours of any vertex can receive the same colour, and since there can be at most one vertex of out-degree n in T, it must be that v_i and v_{i+1} have colour v.

Otherwise, v has at least one strict in-neighbour distinct from itself, say y, in T. Let $y_i^\star$ be the vertex corresponding to y in $T_i^\star$. Let u_{i+1} be a vertex in $T_{i+1} \setminus \{v_{i+1}\}$, and let $u_i^\star$ be the vertex of $T_i^\star$ which has u_{i+1} as an out-neighbour. By the first claim, no two vertices in T_{i+1} share a colour, and since u_{i+1} has $n-1$ neighbours in T_{i+1}, it must be that u_{i+1} and $u_i^\star$ share a colour. This implies that no two vertices of $T_i^\star \setminus \{v_i^\star\}$ have the same colour, and the colours used to colour $T_i^\star \setminus \{v_i^\star\}$ are the same colours as those used to colour $T_{i+1} \setminus \{v_{i+1}\}$. The vertex $y_i^\star$ has $v_i^\star$ as an out-neighbour, and each colour except the colour of v_{i+1} is used to colour a vertex distinct from $v_i^\star$ which is a neighbour of $y_i^\star$. Therefore, $v_i^\star$ must have the same colour as v_{i+1}. The result now follows from Proposition 6.23. □

Let T' be the reflexive tournament induced by the strict out-neighbourhood of v. We show that G has an iot-injective T'-colouring if and only if C has an iot-injective T-colouring.

Let φ be an iot-injective T-colouring of C. Since all the vertices of T_i are assigned distinct colours, v_i must be coloured with a vertex in the same orbit as v in T. We can assume without loss of generality that $\varphi(v_0) = v$ and thus, by Proposition 6.24, $\forall i \in [\![1, \nu_G - 1]\!]$, $\varphi(v_i^\star) = \varphi(v_i) = v$.

For all $i \in [\![1, \nu_G - 1]\!]$, since w_i is an out-neighbour of $v_i^\star$, $\varphi(w_i)$ must be contained in the out-neighbourhood of v in T. Hence, $\forall i \in [\![1, \nu_G - 1]\!], \varphi(w_i) \in V(T')$. The restriction of φ to G is therefore an iot-injective homomorphism to T'.

Conversely, let β be an iot-injective T'-colouring of G. We extend β to be an iot-injective T-colouring of C as follows. For each $z \in V(T)$, let z_i and $z_i^\star$ be the corresponding vertices in T_i and $T_i^\star$, respectively. We extend β so that $\beta(z_i) = \beta(z_i^\star) = z$. It is easily verified that β is an iot-injective T-colouring of C. □

The construction of C can be modified to give the corresponding result for reflexive tournaments T with a vertex of in-degree at least four.

Lemma 6.25. *If T is a reflexive tournament on n vertices with a vertex v of in-degree at least four, then iot-injective homomorphism to T' polynomially transforms to iot-injective homomorphism to T, where T' is the tournament induced by the strict in-neighbourhood of v.*

Similar to the case of ios-injective colouring, our results compile to give a dichotomy theorem.

Theorem 6.26.
Let T be a reflexive tournament. If T has at least 3 vertices, then the problem of deciding whether a given oriented graph G has an iot-injective homomorphism to T is NP-complete. If T has 1 or 2 vertices, then the problem is Polynomial.

6.4 Conclusion

In this chapter, we have studied and established dichotomy theorems for the complexity of locally-injective homomorphisms under two definitions of local injectivity, in the case when the target is a reflexive tournament. Our results also imply a dichotomy theorem for the complexity of ios- and iot- injective k-colouring:

Theorem 6.27. *Ios- and iot-injective k-colouring of directed graphs is NP-complete if $k \geqslant 3$ and polynomial otherwise.*

Possible continuations of this work are discussed in Section 7.4.

Chapter 7

Conclusion and further work

This chapter gives a brief overview of the main results of this thesis and presents some of the open problems that they raise. This chapter aims at giving ideas of directions for future reserach.

Contents

7.1 Separation on languages and traffic monitoring

Chapter 2 focuses on the problem of traffic monitoring. In Section 2.3, we introduced a new model of separation based on languages that allows to study traffic monitoring with tools coming from the theory of separating codes, such as separating sets, and language theory such as regular expressions.

We then highlighted several subproblems (Sections 2.4, 2.5 and 2.6) that are relevant for practical applications and used the model we developed in the beginning of the chapter to outline algorithms to address these problems. We notably established a reduction theorem (Theorem 2.31) that allows to solve the problem on specific infinite instances.

This study and the notions we introduced raise many open problems that could be at the core of future work. We present some of them in the rest of this section.

7.1.1 Reducible languages

We saw in Section 2.5 and 2.6 that the reduction theorem (Theorem 2.31) holds for reachable languages and even for the more general class of FTG-reachable languages, but it also holds for other languages that do not belong to those classes, such as for example all the finite languages (the theorem is obvious since finite languages are equal to their restriction). However, it does not hold for the more general class of rational languages. For example, $L = (ab)^* + ababa$ is a rational language but is

neither FTG-reachable nor reachable. Indeed, we proved in the proof of Proposition 2.14 that a language that contains *ababa* and satisfies Lemma 2.13 has to contain *aba* and *abababa*. A restriction of L is $\overline{L} = \varepsilon + ab + abab + ababa$. One can see that the alphabet $A' = \{a\}$ separates $\overline{L}$. However, both *ababa* and *ababab* belong to L and have the same image under $p_{\{a\}}$.

Thus, natural questions are to determine which languages satisfy the reduction theorem and how to separate infinite languages that do not.

Rational languages can be infinite because their definition uses the Kleene star that denotes an infinite sum. The restriction that we presented in Definition 2.15 consists of truncating all the infinite sums at $k = 2$. A natural generalization of the restriction follows:

Definition 7.1. k-restriction of a rational language:

Given a regular expression of a rational language L, we define a *k-restriction* of L and we denote by $\overline{L}$ the language built inductively as follows:

- $\overline{\varnothing} = \varnothing$
- $\forall a \in A^*, \overline{\{a\}} = \{a\}$
- $\overline{L_1 + L_2} = \overline{L_1} + \overline{L_2}$
- $\overline{L_1 L_2} = \overline{L_1}\ \overline{L_2}$
- $\overline{L^*} = \sum_{i=0}^{k} L^k$

Definition 7.2. k-reducibility:

A language $L \in A^*$ is *k-reducible* if and only if for every k-restriction $\overline{L}$ of L, the subalphabets $\mathcal{C} \subset A$ that separate L are exactly those that separate $\overline{L}$.

An alternative statement of Theorem 2.22 and 2.31 is that reachable and FTG-reachable languages are 2-reducible.

At the end of Subsection 2.2.2, we showed that separating the elementary cycles starting on the vertex u in the graph depicted in Figure 2.2 was not enough to separate all the cycles starting on u. This proves that reachable languages and therefore the more general class of FTG-reachable languages are not 1-reducible since the set of elementary cycles is a 1-restriction of the set of the cycles on u (see Example 2.23 for a description of this language).

The language $L = (ab)^* + ababa$ that we used to prove that the reduction theorem does not hold for rational languages is actually 3-reducible. However, for all $k \in \mathbb{N}$, the language $(ab)^* + (ab)^k a$ is rational and not k-reducible which proves that our generalization of reduction is not strong enough to deal with rational languages.

This generalization may however be useful to deal with a generalization of FTGs where we can forbid a given set of subwalks. This problem is more general since forbidden transitions are simply forbidden subwalks of length 2. Forbidden walks of length 2 are the ones that make the most sense in regards to the highway code but forbidding longer subwalks can probably help discard walks that, while not forbidden, are very unlikely in practice.

The reduction theorem does not hold on graphs with forbidden subwalks. For example, let $G = (V, A)$ be the graph depicted in Figure 7.1a and let the walks ac, abc and $abbc$ be forbidden. The set of permitted walks leading from u or v to w is $L = ab^3b^*c + b^*c$. Its reduction is $c + bc + bbc + ab^3c + ab^4c + ab^5c$ and is separated by $\mathcal{C} = \{b\}$. However, the words $abbbc$ and $bbbc$ both denote walks from a starting point to a destination and have the same projection on $\mathcal{C}$. Here, the language L is 3-reducible but we can build counter-examples for any k. Indeed, let $k \in \mathbb{N}$, the language of walks from u or v to w that do not contain any of the walks ab^ic with $i \leqslant k$ is not k-reducible.

(a) (b)

Figure 7.1: Examples of graph with forbidden subwalks that are not 2-reducible.

However, if the set of forbidden walks is finite, there exists k such that no forbidden walk has length strictly greater than k. In this case, there may exist $\ell \in \mathbb{N}$ such that $\mathcal{W}$ is ℓ-reducible and determining such values of ℓ would help reduce the separation problem to a finite language.

Conjecture 7.3. *There exists a function f in $O(k)$ such that all the languages that denote sets of walks leading from a starting point to a destination in a graph with forbidden subwalks of length $\leqslant k$ are $f(k)$-reducible.*

For example, if $k = 2$, Theorem 2.31 states that the language is 2-reducible. The above example proves that $f(k) \geqslant k + 1$.

Finally, we would like to point out that constraints on the graph may allow to decrease the function f and therefore to reduce the problem to separation on a smaller language. For example, if the graph we work on is irreflexive, our counter-example depicted in Figure 7.1a does not work. In Figure 7.1b, if we forbid the subwalks $a(bc)^id$ for $i \leqslant k$, the set of walks leading from u or v to w is not k-reducible but we use forbidden walks of length $2k + 2$ (instead of $k + 2$ in the reflexive case). Hence, in this case, we only know that $f(k) \geqslant k/2$. We do not know if this counter-example is minimal.

Graphs with forbidden subpaths have also been studied for their applications in optical networks in [1]. Here again, our previous counter-examples do not work since the forbidden walks we use are not elementary. Thus, the function f may also be smaller in this case than in the general case.

Determining how to reduce separation on an infinite language to a finite language as small as possible is an interesting language theory problem and could be useful in the practical applications of traffic monitoring. This is probably a very wide area

to explore since there are undoubtedly different models that we can take advantage of differently.

7.1.2 Planar instances

A strong property of the graphs that arise from the modelling of road networks is that they are planar or at the very least, can be embedded in the plane with very few crossing edges. Furthermore, note that if we can forbid transitions, if two edges cross, we can get back to a planar graph by adding only one vertex as illustrated in Figure 7.2. The study we present in this Chapter does not make any hypothesis on the planarity of our instances but planarity has already been used to design efficient algorithms for problems that share similarities with traffic monitoring. For example, in [13], Bonamy et al. present the notion of connectivity patterns and use them to design a dynamic programming algorithm for feedback vertex sets in directed planar graphs. A feedback vertex set is a set of vertex such that every directed cycle contains at least one vertex of the feedback vertex set. Notice that a necessary condition on a solution of complete traffic monitoring is to monitor at least one arc in each cycle that can be reached from a starting point and from which a destination can be reached (otherwise, we cannot keep track of how many times the object walking in the graph has used this cycle). Works like this one could be an interesting starting point to study traffic monitoring on planar instances.

(a) A graph with crossing edges.

(b) A planar graph with a similar set of possible walks.

Figure 7.2

Other important topics for future work include the study of the ILP our reduction leads to and divide-and-conquer algorithms, that could drastically decrease the computation time and lead to good approximations, especially on planar instances.

7.2 Minimum connecting transition sets

In Chapter 3, we studied the problem of Minimum Connecting Transition Set in undirected graphs. We introduce a reformulation of the problem that helped design a polynomial $\frac{3}{2}$-approximation and prove the NP-completeness of the problem. The rest of this section presents potential directions of future research.

7.2.1 Sparse graphs

Our results suggest that the density of the graph has an impact on the complexity of **MCTS**. We know for example that the problem is trivial in trees (Lemma 3.2) and more generally, in graphs with cut vertices (Proposition 3.13). Similarly, our counter-examples to the tightness of the bound presented in Theorem 3.5 and our proof of NP-completeness involve very dense graphs. Consequently, it would be interesting to study the complexity of this problem on sparser graphs such as planar graphs or graphs with bounded maximum average degree, which generalize planar graphs. Since the problem is easy on trees, we could also study the case of graph with bounded treewidth or try to design general algorithm parametrized by the treewidth of the graph.

7.2.2 Stretch of the solution

Another interesting continuation of this work would also be the study of low-stretch connecting transition sets, a problem that is already well-studied for minimum spanning trees [95]. Intuitively, it consists on looking for a subset of transitions T such that the shortest T-compatible path between two vertices is not much longer than the shortest path with no forbidden transitions, which is also an important criteria of robustness.

If G is a tree, the construction we exhibited in the proof of Lemma 3.2 achieves a worst-case stretch of 3 if we pick $f(v)$ arbitrarily. Indeed, if the shortest walk between u and v is $(u, u_1, u_2, \dots, u_k, v)$ and uses none of the $f(u_i)$, the shortest T-compatible path with $T = \{uvf(v) : v \in V(G), u \in N(v) \setminus \{f(v)\}\}$ is $(u, u_1, f(u_1), u_1, u_2, f(u_2), u_2, \dots, u_k, f(u_k), u_k, v)$ and has length $3k + 1$ instead of $k + 1$. However, let us pick a root r in the tree and let $f(v)$ be the parent of v for $v \neq r$ and any neighbour of r if $v = r$. Let u and v be vertices of G and let w be their first common ancestor. The shortest path between u and v in G is the concatenation of the path between u and w and the path between w and v, let ℓ be its length. Note that the concatenation of the path between u and w, of the path $(w, f(w), w)$ and of the path between w and v is T-compatible and has length $\ell + 2$. Hence, there exist minimum connecting transition sets of asymptotic stretch 1 on trees and we can combine this with minimum-stretch spanning tree to obtain a heuristic in general graphs but it is not known whether this construction is optimal.

In the general case, the construction in the proof of Lemma 3.2 provides a connecting transition set of stretch 3 but of size $2|E(G)| - |V(G)|$, which is not necessarily minimum.

Another expected continuation of this work is to study **MCTS** in directed graphs, which are more suitable for many practical applications.

7.3 Sets avoiding distance 1

In Chapters 4 and 5, we studied the density of sets avoiding distance 1 and developed methods based on the independence ratio of infinite graphs or of optimally weighted finite graphs to establish bounds on m_1 in different normed spaces. Our main results

are the proof of the Bachoc-Robins conjecture in dimension 2 (Theorem 4.23) and for a class of polytopes of unbounded dimension (Theorem 4.24), a bound of the order $O(\frac{1}{2^n})$ for another infinite class (Theorem 4.28), the best known upper bound on $m_1(\mathbb{R}^2)$ (Theorem 5.13), the best known lower bound on $\chi_f(\mathbb{R}^2)$ (Theorem 5.14) and bounds on $m_1(\mathbb{R}^3, \|\cdot\|_{\mathcal{P}})$ for any regular parallelohedron P (Theorem 5.24). This project is still ongoing are there are a lot of open problems to explore. We present some of them in this section.

7.3.1 The Euclidean plane and Erdős' conjecture

De Grey's proof of the existence of a 5-chromatic graph has many implications for our method and problem. Indeed, the existence of 5-chromatic graph makes it plausible to find graphs of optimal weighted independence ratio strictly smaller than $\frac{1}{4}$ and to prove Erdős' conjecture. However, the first 5-chromatic graph exhibited by De Grey had more than 20.000 vertices and is clearly out of reach of our algorithm. Considerable efforts have since been made by many mathematicians to find smaller 5-chromatic graphs and this problem is notably at the core of the collaborative math project Polymath16. The current smallest known 5-chromatic subgraph of $\mathcal{G}(\mathbb{R}^2)$ was found by Marijn Heule and has 610 vertices. However, if a 5-chromatic graph is minimal, this means that removing any of its vertex, even the one of smallest weight, makes the graph 4-chromatic again and thus, that all its other vertices can be partitioned into 4 independent sets. Such graphs are extremely unlikely to have optimal weighted independence ratio smaller than $\frac{1}{4}$. So far, all the subgraphs of $\mathcal{G}(\mathbb{R}^2)$ that have provided interesting results for our problems were 4-chromatic.

Improving our results will probably require a deep understanding of $\mathcal{G}(\mathbb{R}^2)$ and of the optimal weighted independence ratio to find ways to adapt to our problem the ideas that have emerged recently to design graphs of small chromatic number.

Another important problem would be to develop tools to find bounds other than computationally. Indeed, even if there may exist subgraphs of $\mathcal{G}(\mathbb{R}^2)$ of optimal weighted independence ratio smaller than $\frac{1}{4}$, we do not know their size and even the smallest of them might very well be far beyond the reach of our algorithms. For example, future works could investigate how to derive non-trivial bounds on the optimal weighted independence ratio of large graphs from those of some of its subgraphs.

7.3.2 Parallelohedra and Bachoc-Robins' conjecture

Now that we have proved the conjecture of Bachoc-Robins in dimension 2, the next natural step is the study of dimension 3. The conjecture is now solved for every regular 3-dimensional parallelohedron except the truncated octahedron, for which bounds are presented in this thesis. However, the only non-regular 3-dimensional parallelohedra for which the conjecture is solved are the cuboid (implied by Example 4.14) and the hexagonal prism [92]. The 3 remainings are, in increasing order of difficulty, the rhombic dodecahedron, the elongated dodecahedron and finally, the truncated octahedron, which generalizes every 3-dimensional parallelohedron.

Our study of Λ-classes, of k-regular independent sets and our algorithm to compute the optimal weighted independence ratio of graphs still work well in the case

of norms induced by irregular parallelohedra but our construction of the infinite discrete graph G is not straightforward to generalize to the irregular truncated octahedron.

Indeed, in the case of the regular truncated octahedron $\mathcal{P}$, the set of vectors between the center of $\mathcal{P}$ and the centers of its square faces are exactly the set of vectors between the opposite vertices of a square face (these vectors are the permutation of $(-2,-2,2,2)$). Hence, if two vertices v and v' are opposite vertices of a square face, $v - v' \in \frac{1}{2}\Lambda$ and thus, $v + \frac{1}{2}\Lambda = v' + \frac{1}{2}\Lambda$. For example, if we define $V^{32} = (V_{\mathcal{P}} \cup \{\mathbf{0}\}) + \frac{1}{2}\Lambda$ like we did in Subsection 5.4.4, the vertex v of coordinates (-3,-1,1,3) belongs to V^{32} both as a vertex of $\mathcal{P}$ and as the translate of another vertex of $\mathcal{P}$ (the vertex of coordinates (-1,-3,3,1)) by the vector $(-2,2,-2,2) \in \frac{1}{2}\Lambda$. However, in the irregular case, those two definitions of v are not equivalent anymore, which increases the number of vertices and of Λ-classes in the graph and make their distribution less homogeneous (the maximum number of Λ-classes that an independent set may contain is no longer one eighth of the total number of Λ-classes).

We avoid this problem by partitioning the vertices of $\mathcal{P}$ in 3 $(\frac{1}{2}\Lambda)$-classes (two opposite vertices of a square face belong to different Λ-classes but to the same $\frac{1}{2}\Lambda$-class) and by picking only one Λ-class in each of them. We call $V'_{\mathcal{P}}$ the union of the three chosen Λ-classes and we define our vertex set as $V^{32} = (V'_{\mathcal{P}} \cup \{\mathbf{0}\}) + \frac{1}{2}\Lambda$. By replacing $V_{\mathcal{P}}$ by $V'_{\mathcal{P}}$, we can also generalize the definition of the vertex sets V^{64}, V^{128} which are the most interesting to us.

Building a vertex set by selecting all the vertices at a given distance from the origin was useful in Algorithm 3 and in the proof of Theorem 5.24 but here again, it can only be done in the case of a regular parallelohedron. Indeed, the distance between all the vertices of our graph is easy to determine in the case of the regular parallelohedron but it cannot be done in the general case. For example, we know that the edges of a regular truncated octahedron all have length $\frac{2}{3}$ for the distance it induces but in the general case, the length of the edges can be anywhere between 0 and 2. However, our generalization of the construction of V^{64} and V^{128} makes it easy to define an isomorphism f between the vertex set V we build in the regular case and the vertex set V' we build in the general case. While we do not know which vertices of V' are at distance smaller than d from $\mathbf{0}$, we can still define V'_d as the image by f of the vertices of V_d.

However, the unit-distance subgraph that we build in the irregular case have much fewer edges and the bounds we obtain on their independence ratio are far from as good as in the regular case. Proving the conjecture of Bachoc-Robins for general 3-dimensional parallelohedra or even for the regular truncated octahedron seems out of reach of our current algorithm but a more reasonable objective could be to establish an upper bound smaller than $\frac{1}{7}$, which would prove that the chromatic number of the space equipped by any parallelohedron norm is 8.

7.3.3 Power and limitation of weighted subgraphs

Another interesting topic to study would be the bound we can achieve on m_1 with the independence ratio of unweighted and weighted graphs. For example, in the case of

the regular hexagon, we know a finite weighted graph of independence ratio $\frac{1}{4}$ (Figure 5.17) but we do not know if this bound can be reached by an unweighted graph. Similarly, there are several parallelohedra for which we do not know if the bound of $\frac{1}{2^n}$ can be reached by finite graphs, even with a weighting. These parallelohedra include the irregular hexagon for which we know that the Bachoc-Robins conjecture holds. Our proofs of Theorems 4.15 and 4.19 suggest that we could find finite graphs of independence ratio arbitrarily close to $\frac{1}{4}$ but do not imply that these bounds can be reached.

We currently have very few tools to prove that such graphs do not exist. A first step would be to answer those questions for a given discrete infinite graph G: we investigate the existence of finite subgraphs H of G whose independence ratio reaches $\frac{1}{2^n}$. For example, in Subsection 5.4.1, in the case of the regular hexagon, we looked for properties that a weighted subgraph H of the graph G depicted in Figure 4.7 had to observe to have independence ratio $\frac{1}{4}$. We saw that if there exist classes C and C' of a same coset (see Figure 5.18) such that an independent set in $C \cup C'$ has higher weight than C or C', we can build a 2-regular independent set of weight ratio greater than $\frac{1}{4}$. This implies that in a graph of independence ratio $\frac{1}{4}$, no subset c of C can be heavier than its neighbourhood in C' since $C' \setminus N[c] \cup c$ is independent. This property can be satisfied by small weighted graphs but is a strong constraint in the unweighted case. We also recall that all the classes of a same coset must have same weight and thus, in the unweighted case, same cardinality. Still, we were able to build unweighted subgraph of G that observes this constraint (for example, a graph where all three cosets induce the graph depicted in Figure 7.3) but they all had independence ratio much higher than $\frac{1}{4}$, even if it could only be reached by at most 1-regular set.

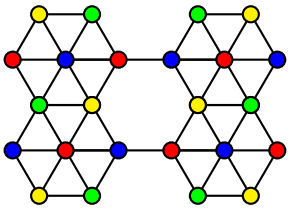


Figure 7.3: The colours denote the Λ-classes of a same coset and the edges denote geometric distance 1. All the Λ-classes have 6 vertices and no independent set on the union of two classes has size more than 6.

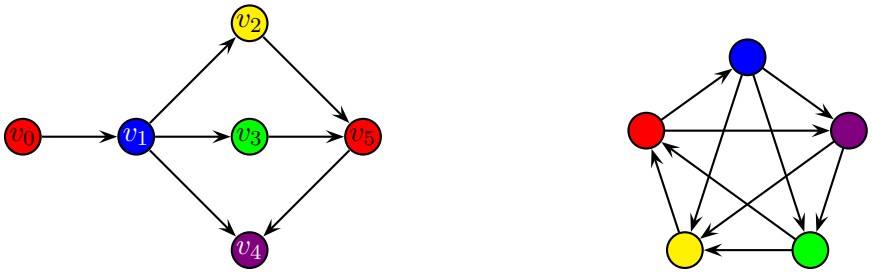
In the case of the irregular hexagon, the requirement is even stronger: we saw in Example 5.23 that no independent set on $c_8 \cup c_{10} \cup c_{12}$ can have a higher weight than one of these three classes. As mentioned previously, we do not know if the bound of $\frac{1}{4}$ for the irregular hexagon can be reached by finite weighted graphs. The best bound we have reached so far with finite graphs is 0.250951 with a graph of size 487.

7.4 Locally-injective directed homomorphisms

In Chapter 6, we established dichotomy theorems for the complexity of locally-injective homomorphisms to reflexive tournaments, for two definitions of local injec-

tivity, namely ios- and iot-injectivity. These theorems imply a dichotomy theorem for the complexity of the associated colouring problems. A long-term objective would be to establish a dichotomy theorem for the complexity of locally-injective homomorphism to any directed graphs and starting with complete targets is often a good starting point because of the diversity of the configuration it involves. For example, in the case of homomorphism to undirected graph, the dichotomy theorem established by Hell and Nešetřil ([61], see Theorem 6.3) depends entirely on the colourability of the target of the homomorphism.

A more immediate goal would be to study of the complexity of locally-injective homomorphism to irreflexive tournaments, for which ios- and iot-injectivity are equivalent since neither in- nor out-neighbours of a vertex v can have the same colour as v. For example, if we try to colour the graph of Example 6.2 with an irreflexive tournament, we notice that five colours are necessary since $|N[v_1]| = 5$ and sufficient, as illustrated in Figure 7.4.



(a) A 5-irreflexive-injective colouring of G. (b) An associated target tournament.

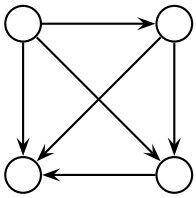
Figure 7.4

The results of [20, 19] tell us that the problem is not only Polynomial for the irreflexive tournaments on two vertices or less but also for the irreflexive tournaments on three vertices.

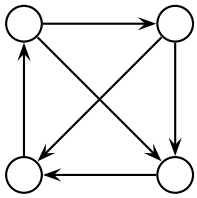
Furthermore, we were able to prove that locally-injective homomorphism is also polynomial to two of the four irreflexive tournaments on four vertices, namely, T_4 and the transitive tournament TT_4 (see Figure 7.5). However, we proved that the problem is NP-complete on the two other, which consist respectively of an oriented cycle of length 3 with a dominating vertex, and an oriented cycle of length 3 with a dominated vertex (see Figure 7.6). Our preliminary work on this matter also allows us to establish the NP-completeness of locally-injective homomorphism to several other irreflexive tournaments, including at least ten of the twelve irreflexive tournaments on five vertices. The problem has not been proven polynomial on any irreflexive tournament on five vertices or more, which lead us to the following conjecture:

Conjecture 7.4.

Locally-injective homomorphism is polynomial to irreflexive tournament on 3 vertices or less as well as to the irreflexive version of T_4 and TT_4 (see Figure 7.5) and NP-complete on the two other irreflexive tournament on four vertices (see Figure 7.6) as well as to every irreflexive tournament on 5 vertices or more.

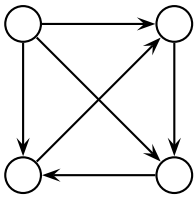


(a) TT_4.

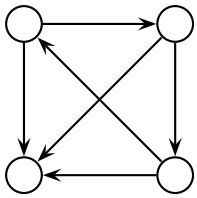


(b) T_4.

Figure 7.5: Locally-injective homomorphism to these two tournaments is polynomial.



(a) This tournament consists of an induced C_3 and a dominating vertex.



(b) This tournament consists of an induced C_3 and a dominated vertex.

Figure 7.6: Locally-injective homomorphism to these two tournaments is NP-complete.

For now, no equivalent of Lemma 6.10 or 6.21 has emerged that could help us establish a dichotomy theorem on the infinitely many irreflexive tournaments. However, even the equivalent of such a theorem would not be sufficient to prove Conjecture 7.4.

Indeed, in our dichotomy Theorems 6.14 and 6.26, the set $S_{\mathrm{NP-C}}$ of reflexive tournaments to which locally-injective homomorphism is NP-complete problem are the tournaments on three vertices or more. Our proof works by induction where we define $S_{\mathrm{NP-C}}$ as C_3, TT_3, T_4, T_5 (Figure 6.3, 6.4 and 6.9) or tournaments that have a vertex whose open in- or out-neighbourhood induces another tournament of $S_{\mathrm{NP-C}}$. This definition involves 4 base cases for which we were able to prove case-by-case that the problem is NP-complete (see Theorem 6.4 [19], Theorems 6.7, 6.9, 6.18 and 6.20).

In the case of an irreflexive target, an equivalent of Lemma 6.10 or 6.21 would leave us to deal with all the tournaments that have no vertex whose open in- or out-neighbourhood has size five or induces one of the tournaments depicted in Figure 7.6. This includes many tournaments since this description is satisfied by tournaments of size up to 9 and the number of tournaments on n vertices increases extremely fast (there are 4 tournaments on 4 vertices, 12 on 5 vertices,... 191,536 on 9 vertices). Extending our dichotomy theorems to irreflexive tournaments will require finding new techniques of proof and would be an interesting continuation of this work.

Appendix A

Computational bound in the Euclidean plane

This appendix describes our subgraph G of $\mathcal{G}(\mathbb{R}^2)$ that achieves the bound of $\alpha^*(G) \leqslant 0.256828$ announced in Theorem 5.13.

Our starting point is a graph W introduced by De Grey as an intermediate graph toward his construction of a 5-chromatic subgraph of $\mathcal{G}(\mathbb{R}^2)$ (see Figure 8 in [34]). This graph has 301 vertices, 1230 edges and was useful to De Grey because of the restrictive properties that 4-colourings of W must observe. While those properties are not directly important to us, the low number of optimal colourings is because it makes it easier to find weightings that balance the colour classes in all of them.

We then iterated on W the processes of optimization of the weighting, removal of the vertices of lowest weight and combination of the resulting graph with copy of itself that we described in Subsection 5.3.3.

The last step of our construction was the combination of six copies of an intermediate graph of 282 vertices obtained by rotating the graph of $\frac{k\pi}{3}$ for $k \in [\![0,5]\!]$ around the point of coordinate (1,0). This implies that the final graph is invariant by rotation of $\frac{k\pi}{3}$ around (1,0). Thus, even though it has 487 vertices, we can describe it by giving only the coordinates of the 82 vertices x such that $x-(1,0)$ has polar angle in $[0, \frac{\pi}{3}[$. Hence, the entire vertex set of the graph can be generated by rotating those vertices. The coordinates of our vertices are elements of $\mathbb{Q}[\sqrt{3}][\sqrt{11}]$. For simplicity, we denote by (a,b,c,d) the number $a+b\sqrt{3}+c\sqrt{11}+d\sqrt{33}$. The vertices are:

$((\frac{1}{2}, 0, 0, \frac{1}{6}), (0, \frac{1}{3}, 0, 0))$, $((\frac{3}{4}, 0, 0, \frac{1}{12}), (0, -\frac{1}{4}, \frac{1}{4}, 0))$, $((\frac{3}{4}, 0, 0, \frac{1}{12}), (0, \frac{7}{12}, -\frac{1}{4}, 0))$, $((\frac{11}{12}, 0, 0, \frac{1}{12}), (0, \frac{1}{12}, \frac{1}{12}, 0))$, $((1, 0, 0, 0), (0, 0, 0, 0))$, $((1, 0, 0, \frac{1}{6}), (0, \frac{1}{6}, 0, 0))$, $((1, 0, 0, \frac{1}{6}), (0, \frac{1}{2}, 0, 0))$, $((1, 0, 0, \frac{1}{6}), (0, \frac{5}{6}, 0, 0))$, $((\frac{5}{4}, 0, 0, \frac{1}{12}), (0, -\frac{5}{12}, \frac{1}{4}, 0))$, $((\frac{5}{4}, 0, 0, \frac{1}{12}), (0, -\frac{1}{12}, \frac{1}{4}, 0))$, $((\frac{5}{4}, 0, 0, \frac{1}{12}), (0, \frac{1}{4}, \frac{1}{4}, 0))$, $((\frac{5}{4}, 0, 0, \frac{1}{12}), (0, \frac{3}{4}, -\frac{1}{4}, 0))$, $((\frac{5}{4}, 0, 0, \frac{1}{12}), (0, \frac{13}{12}, -\frac{1}{4}, 0))$, $((\frac{17}{12}, 0, 0, \frac{1}{12}), (0, \frac{7}{12}, \frac{1}{12}, 0))$, $((\frac{3}{2}, 0, 0, 0), (0, \frac{1}{6}, 0, 0))$, $((\frac{3}{2}, 0, 0, 0), (0, \frac{1}{2}, 0, 0))$, $((\frac{3}{2}, 0, 0, \frac{1}{6}), (0, \frac{1}{3}, 0, 0))$, $((\frac{3}{2}, 0, 0, \frac{1}{6}), (0, \frac{2}{3}, 0, 0))$, $((\frac{3}{2}, 0, 0, \frac{1}{6}), (0, 1, 0, 0))$, $((\frac{19}{12}, 0, 0, \frac{1}{12}), (0, \frac{7}{12}, -\frac{1}{12}, 0))$, $((\frac{5}{3}, 0, 0, 0), (0, \frac{1}{2}, -\frac{1}{6}, 0))$, $((\frac{7}{4}, 0, 0, -\frac{1}{12}), (0, -\frac{5}{12}, \frac{1}{4}, 0))$, $((\frac{7}{4}, 0, 0, -\frac{1}{12}), (0, \frac{3}{4}, -\frac{1}{4}, 0))$, $((\frac{7}{4}, 0, 0, \frac{1}{12}), (0, -\frac{1}{4}, \frac{1}{4}, 0))$, $((\frac{7}{4}, 0, 0, \frac{1}{12}), (0, \frac{1}{12}, \frac{1}{4}, 0))$, $((\frac{7}{4}, 0, 0, \frac{1}{12}), (0, \frac{5}{12}, \frac{1}{4}, 0))$, $((\frac{7}{4}, 0, 0, \frac{1}{12}), (0, \frac{7}{12}, -\frac{1}{4}, 0))$, $((\frac{7}{4}, 0, 0, \frac{1}{12}), (0, \frac{3}{4}, \frac{1}{4}, 0))$, $((\frac{7}{4}, 0, 0, \frac{1}{12}), (0, \frac{11}{12}, -\frac{1}{4}, 0))$,

$((\frac{11}{6}, 0, 0, 0), (0, 0, \frac{1}{6}, 0)), ((\frac{23}{12}, 0, 0, -\frac{1}{12}), (0, -\frac{1}{12}, \frac{1}{12}, 0)), ((\frac{23}{12}, 0, 0, \frac{1}{12}), (0, \frac{1}{12}, \frac{1}{12}, 0)), ((\frac{23}{12}, 0, 0, \frac{1}{12}), (0, \frac{13}{12}, \frac{1}{12}, 0)), ((2, 0, 0, 0), (0, \frac{1}{3}, 0, 0)), ((2, 0, 0, 0), (0, \frac{2}{3}, 0, 0)), ((2, 0, 0, 0), (0, 1, 0, 0)), ((2, 0, 0, \frac{1}{6}), (0, \frac{1}{6}, 0, 0)), ((2, 0, 0, \frac{1}{6}), (0, \frac{1}{2}, 0, 0)), ((2, 0, 0, \frac{1}{6}), (0, \frac{5}{6}, 0, 0)), ((2, 0, 0, \frac{1}{6}), (0, \frac{7}{6}, 0, 0)), ((\frac{13}{6}, 0, 0, 0), (0, 1, -\frac{1}{6}, 0)), ((\frac{9}{4}, 0, 0, -\frac{1}{12}), (0, -\frac{1}{4}, \frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, -\frac{1}{12}), (0, \frac{1}{12}, \frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, -\frac{1}{12}), (0, \frac{7}{12}, -\frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, -\frac{1}{12}), (0, \frac{11}{12}, -\frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, \frac{1}{12}), (0, -\frac{5}{12}, \frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, \frac{1}{12}), (0, -\frac{1}{12}, \frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, \frac{1}{12}), (0, \frac{1}{4}, \frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, \frac{1}{12}), (0, \frac{7}{12}, \frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, \frac{1}{12}), (0, \frac{3}{4}, -\frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, \frac{1}{12}), (0, \frac{11}{12}, \frac{1}{4}, 0)), ((\frac{9}{4}, 0, 0, \frac{1}{12}), (0, \frac{13}{12}, -\frac{1}{4}, 0)), ((\frac{7}{3}, 0, 0, 0), (0, \frac{1}{2}, \frac{1}{6}, 0)), ((\frac{29}{12}, 0, 0, -\frac{1}{12}), (0, \frac{5}{12}, \frac{1}{12}, 0)), ((\frac{29}{12}, 0, 0, \frac{1}{12}), (0, \frac{7}{12}, \frac{1}{12}, 0)), ((\frac{5}{2}, 0, 0, -\frac{1}{6}), (0, \frac{1}{3}, 0, 0)), ((\frac{5}{2}, 0, 0, 0), (0, \frac{1}{6}, 0, 0)), ((\frac{5}{2}, 0, 0, 0), (0, \frac{1}{2}, 0, 0)), ((\frac{5}{2}, 0, 0, 0), (0, \frac{5}{6}, 0, 0)), ((\frac{5}{2}, 0, 0, \frac{1}{6}), (0, \frac{1}{3}, 0, 0)), ((\frac{5}{2}, 0, 0, \frac{1}{6}), (0, \frac{2}{3}, 0, 0)), ((\frac{31}{12}, 0, 0, -\frac{1}{12}), (0, \frac{5}{12}, -\frac{1}{12}, 0)), ((\frac{31}{12}, 0, 0, \frac{1}{12}), (0, \frac{7}{12}, -\frac{1}{12}, 0)), ((\frac{8}{3}, 0, 0, 0), (0, \frac{1}{2}, -\frac{1}{6}, 0)), ((\frac{11}{4}, 0, 0, -\frac{1}{12}), (0, -\frac{5}{12}, \frac{1}{4}, 0)), ((\frac{11}{4}, 0, 0, -\frac{1}{12}), (0, -\frac{1}{12}, \frac{1}{4}, 0)), ((\frac{11}{4}, 0, 0, -\frac{1}{12}), (0, \frac{1}{4}, \frac{1}{4}, 0)), ((\frac{11}{4}, 0, 0, -\frac{1}{12}), (0, \frac{7}{12}, \frac{1}{4}, 0)), ((\frac{11}{4}, 0, 0, \frac{1}{12}), (0, -\frac{1}{4}, \frac{1}{4}, 0)), ((\frac{11}{4}, 0, 0, \frac{1}{12}), (0, \frac{1}{12}, \frac{1}{4}, 0)), ((\frac{11}{4}, 0, 0, \frac{1}{12}), (0, \frac{5}{12}, \frac{1}{4}, 0)), ((\frac{11}{4}, 0, 0, \frac{1}{12}), (0, \frac{7}{12}, -\frac{1}{4}, 0)), ((\frac{17}{6}, 0, 0, 0), (0, 0, \frac{1}{6}, 0)), ((3, 0, 0, -\frac{1}{6}), (0, \frac{1}{6}, 0, 0)), ((3, 0, 0, 0), (0, \frac{1}{3}, 0, 0)), ((3, 0, 0, 0), (0, 1, 0, 0)), ((3, 0, 0, \frac{1}{6}), (0, \frac{1}{6}, 0, 0)), ((\frac{13}{4}, 0, 0, -\frac{1}{12}), (0, \frac{1}{12}, \frac{1}{4}, 0)), ((\frac{13}{4}, 0, 0, -\frac{1}{12}), (0, \frac{5}{12}, \frac{1}{4}, 0)), ((\frac{13}{4}, 0, 0, \frac{1}{12}), (0, -\frac{5}{12}, \frac{1}{4}, 0)), ((\frac{13}{4}, 0, 0, \frac{1}{12}), (0, -\frac{1}{12}, \frac{1}{4}, 0))$ and $((\frac{7}{2}, 0, 0, 0), (0, \frac{1}{2}, 0, 0))$.

Index

Bibliography

[1] Mustaq Ahmed and Anna Lubiw. Shortest paths avoiding forbidden subpaths. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, pages 63–74, 2009.

[2] Dean N. Arden. Delayed-logic and finite-state machines. In *Proceedings of the 2Nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*, FOCS '61, pages 133–151, Washington, DC, USA, 1961. IEEE Computer Society.

[3] Gabriela R. Argiroffo, Silvia M. Bianchi, and Annegret Katrin Wagler. Polyhedra associated with identifying codes. *Electronic Notes in Discrete Mathematics*, 44:175–180, 2013.

[4] Christine Bachoc, Thomas Bellitto, Philippe Moustrou, and Arnaud Pêcher. Weighted independence ratio of geometric distance graphs. *ICGT 2018*.

[5] Christine Bachoc, Thomas Bellitto, Philippe Moustrou, and Arnaud Pêcher. On the density of sets avoiding parallelohedron distance 1. *CoRR*, abs/1708.00291, 2017.

[6] Christine Bachoc, Alberto Passuello, and Alain Thiery. The density of sets avoiding distance 1 in Euclidean space. *Discrete Comput. Geom.*, 53(4):783–808, 2015.

[7] Stefan Bard, Thomas Bellitto, Christopher Duffy, Gary MacGillivray, and Feiran Yang. Complexity of locally-injective homomorphisms to tournaments. *CoRR*, abs/1710.08825, under revision for DMTCS, 2018.

[8] Cristina Bazgan, Bruno Escoffier, and Vangelis Th. Paschos. Completeness in standard and differential approximation classes: Poly-(d)apx- and (d)ptas-completeness. *Theor. Comput. Sci.*, 339(2-3):272–292, 2005.

[9] Thomas Bellitto. Separating codes and traffic monitoring. *Theoretical Computer Science*, 717:73 – 85, 2018. Selected papers presented at the 11th International Conference on Algorithmic Aspects of Information and Management (AAIM 2016).

[10] Thomas Bellitto and Benjamin Bergougnoux. On minimum connecting transition sets in graphs. *WG 2018*.

[11] Thomas Bellitto and Arnaud Pêcher. Optimal weighting to minimize the independence ratio of a graph. *ISMP 2018*.

[12] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

[13] Marthe Bonamy, Lukasz Kowalik, Jesper Nederlof, Michal Pilipczuk, Arkadiusz Socala, and Marcin Wrochna. On directed feedback vertex set parameterized by treewidth. *CoRR*, abs/1707.01470, 2017.

[14] Koen M. J. De Bontridder, Bjarni V. Halldórsson, Magnús M. Halldórsson, Cor A. J. Hurkens, Jan Karel Lenstra, R. Ravi, and Leen Stougie. Approximation algorithms for the test cover problem. *Math. Program.*, 98(1-3):477–491, 2003.

[15] Karl Heinz Borgwardt. Probabilistic analysis of simplex algorithms. In *Encyclopedia of Optimization, Second Edition*, pages 3073–3084. 2009.

[16] Gaëlle Brevier, Romeo Rizzi, and Stéphane Vialette. Pattern matching in protein-protein interaction graphs. In *Fundamentals of Computation Theory, 16th International Symposium, FCT 2007, Budapest, Hungary, August 27-30, 2007, Proceedings*, pages 137–148, 2007.

[17] N. G. De Bruijn and P. Erdős. A colour problem for infinite graphs and a problem in the theory of relations, 1951.

[18] Andrei Bulatov. A dichotomy theorem for nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, 2017.

[19] Russell J. Campbell. Reflexive injective oriented colourings. *M.Sc. Thesis, Department of mathematics and Statistics, University of Victoria, Canada*, 2009.

[20] Russell J. Campbell, Nancy E. Clarke, and Gary MacGillivray. Injective homomorphisms to small tournaments. *Unpublished*, 2016.

[21] Russell J. Campbell, Nancy E. Clarke, and Gary MacGillivray. Obstructions to some injective oriented colourings. *Unpublished*, 2016.

[22] Emmanuel Charbit, Irène Charon, Gérard D. Cohen, Olivier Hudry, and Antoine Lobstein. Discriminating codes in bipartite graphs: bounds, extremal cardinalities, complexity. *Adv. in Math. of Comm.*, 2(4):403–420, 2008.

[23] Irène Charon, Gérard D. Cohen, Olivier Hudry, and Antoine Lobstein. Discriminating codes in (bipartite) planar graphs. *Eur. J. Comb.*, 29(5):1353–1364, 2008.

[24] Irène Charon, Olivier Hudry, and Antoine Lobstein. Minimizing the size of an identifying or locating-dominating code in a graph is np-hard. *Theor. Comput. Sci.*, 290(3):2109–2120, 2003.

[25] C. C. Chen and David E. Daykin. Graphs with hamiltonian cycles having adjacent lines different colors. *J. Comb. Theory, Ser. B*, 21(2):135–139, 1976.

[26] Min Chen, Gena Hahn, André Raspaud, and Weifan Wang. Some results on the injective chromatic number of graphs. *J. Comb. Optim.*, 24(3):299–318, 2012.

[27] J. H. Conway and N. J. A. Sloane. Low-dimensional lattices. VI. Voronoĭ reduction of three-dimensional lattices. *Proc. Roy. Soc. London Ser. A*, 436(1896):55–68, 1992.

[28] J. H. Conway, N. J. A. Sloane, and E. Bannai. *Sphere-packings, Lattices, and Groups*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[29] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158, 1971.

[30] Bruno Courcelle. The monadic second order logic of graphs VI: on several representations of graphs by relational structures. *Discrete Applied Mathematics*, 54(2-3):117–149, 1994.

[31] Daniel W. Cranston and Landon Rabern. The fractional chromatic number of the plane. *Combinatorica*, 37(5):837–861, 2017.

[32] H. T. Croft. Incidence incidents. *Eureka (Cambridge)*, 30:22–26, 1967.

[33] George B. Dantzig. Inductive proof of the simplex method. *IBM Journal of Research and Development*, 4(5):505–506, 1960.

[34] Aubrey D.N.J. de Grey. The chromatic number of the plane is at least 5. *CoRR*, abs/1804.02385, 2018.

[35] B. Delaunay. Sur la partition régulière de l'espace à 4 dimensions. I, II. *Bull. Acad. Sci. URSS*, 2:79–110, 1929.

[36] Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2016.

[37] Dietmar Dorninger. Hamiltonian circuits determining the order of chromosomes. *Discrete Applied Mathematics*, 50(2):159–168, 1994.

[38] Alain Doyon, Gena Hahn, and André Raspaud. Some bounds on the injective chromatic number of graphs. *Discrete Mathematics*, 310(3):585–590, 2010.

[39] Zdenek Dvorak. Two-factors in orientated graphs with forbidden transitions. *Discrete Mathematics*, 309(1):104–112, 2009.

[40] R. M. Erdahl. Zonotopes, dicings, and Voronoi's conjecture on parallelohedra. *European J. Combin.*, 20(6):527–549, 1999.

[41] Paul Erdős. Some remarks on set theory. *Proc. Am. Math. Soc.*, 1:127–141, 1950.

[42] Geoffrey Exoo and Dan Ismailescu. The chromatic number of the plane is at least 5 - a new proof. *CoRR*, arXiv:1805.00157v1, 2018.

[43] Isabelle Fagnot, Gaëlle Lelandais, and Stéphane Vialette. Bounded list injective homomorphism for comparative analysis of protein-protein interaction graphs. *J. Discrete Algorithms*, 6(2):178–191, 2008.

[44] K.J Falconer. The realization of distances in measurable subsets covering rn. *Journal of Combinatorial Theory, Series A*, 31(2):184 – 189, 1981.

[45] Guillaume Fertin, Romeo Rizzi, and Stéphane Vialette. Finding exact and maximum occurrences of protein complexes in protein-protein interaction graphs. In *Mathematical Foundations of Computer Science 2005, 30th International Symposium, MFCS 2005, Gdansk, Poland, August 29 - September 2, 2005, Proceedings*, pages 328–339, 2005.

[46] Jirí Fiala and Jan Kratochvíl. Complexity of partial covers of graphs. In *Algorithms and Computation, 12th International Symposium, ISAAC 2001, Christchurch, New Zealand, December 19-21, 2001, Proceedings*, pages 537–549, 2001.

[47] Jirí Fiala and Jan Kratochvíl. Partial covers of graphs. *Discussiones Mathematicae Graph Theory*, 22(1):89–99, 2002.

[48] Jirí Fiala and Jan Kratochvíl. Locally injective graph homomorphism: Lists guarantee dichotomy. In *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*, pages 15–26, 2006.

[49] Jirí Fiala, Jan Kratochvíl, and Attila Pór. On the computational complexity of partial covers of theta graphs. *Discrete Applied Mathematics*, 156(7):1143–1149, 2008.

[50] Herbert Fleischner and Bill Jackson. Compatible path-cycle-decompositions of plane graphs. *J. Comb. Theory, Ser. B*, 42(1):94–121, 1987.

[51] Peter Frankl and Richard M. Wilson. Intersection theorems with geometric consequences. *Combinatorica*, 1(4):357–368, 1981.

[52] Martin Gardner. *The Second Scientific American Book of Mathematical Puzzles and Diversions*. University Of Chicago Press, 1961, reeditied in 1987.

[53] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[54] Christopher D. Godsil and Gordon F. Royle. *Algebraic Graph Theory*. Graduate texts in mathematics. Springer, 2001.

[55] George Grätzer. *Lattice Theory: Foundation*. Springer Basel, 1st edition. edition, 2011.

[56] Gregory Gutin and Eun Jung Kim. Properly coloured cycles and paths: Results and open problems. In *Graph Theory, Computational Intelligence and Thought, Essays Dedicated to Martin Charles Golumbic on the Occasion of His 60th Birthday*, pages 200–208, 2009.

[57] H. Hadwiger. Ein Überdeckungssatz für den euklidischen raum. *Portugal. Math.*, 4:140–144, 1944.

[58] Gena Hahn, Jan Kratochvíl, Jozef Sirán, and Dominique Sotteau. On the injective chromatic number of graphs. *Discrete Mathematics*, 256(1-2):179–192, 2002.

[59] Magnús M. Halldórsson and Jaikumar Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18(1):145–163, 1997.

[60] Teresa W. Haynes, Debra J. Knisley, Edith Seier, and Yue Zou. A quantitative analysis of secondary rna structure using domination based parameters on trees. *BMC Bioinformatics*, 7:108, 2006.

[61] Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990.

[62] Ian Holyer. The np-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718–720, 1981.

[63] Iiro S. Honkala, Tero Laihonen, and Sanna M. Ranto. On strongly identifying codes. *Discrete Mathematics*, 254(1-3):191–205, 2002.

[64] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[65] Tommy R Jensen and Bjarne Toft. Graph coloring problems, 1995.

[66] Mamadou Moustapha Kanté, Christian Laforest, and Benjamin Momège. An exact algorithm to check the existence of (elementary) paths and a generalisation of the cut problem in graphs with forbidden transitions. In *SOFSEM 2013: Theory and Practice of Computer Science, 39th International Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 26-31, 2013. Proceedings*, pages 257–267, 2013.

[67] Mamadou Moustapha Kanté, Fatima Zahra Moataz, Benjamin Momège, and Nicolas Nisse. Finding paths in grids with forbidden transitions. In *Graph-Theoretic Concepts in Computer Science - 41st International Workshop, WG 2015, Garching, Germany, June 17-19, 2015, Revised Papers*, pages 154–168, 2015.

[68] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103, 1972.

[69] Mark G. Karpovsky, Krishnendu Chakrabarty, and Lev B. Levitin. On a new class of codes for identifying vertices in graphs. *IEEE Transactions on Information Theory*, 44(2):599–611, 1998.

[70] Tamás Keleti, Máté Matolcsi, Fernando Mário de Oliveira Filho, and Imre Z. Ruzsa. Better bounds for planar sets avoiding unit distances. *Discrete & Computational Geometry*, 55(3):642–661, 2016.

[71] Jeong Han Kim, Oleg Pikhurko, Joel H. Spencer, and Oleg Verbitsky. How complex are random graphs in first order logic? *Random Struct. Algorithms*, 26(1-2):119–145, 2005.

[72] Victor Klee and George J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York, 1972.

[73] S. C. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, 1956.

[74] William Klostermeyer and Gary MacGillivray. Homomorphisms and oriented colorings of equivalence classes of oriented graphs. *Discrete Mathematics*, 274(1-3):161–172, 2004.

[75] Anton Kotzig. Moves without forbidden transitions in a graph. *Matematický časopis*, 18(1):76–80, 1968.

[76] Moshe Laifenfeld, Ari Trachtenberg, Reuven Cohen, and David Starobinski. Joint monitoring and routing in wireless sensor networks using robust identifying codes. *MONET*, 14(4):415–432, 2009.

[77] D. G. Larman. A note on the realization of distances within sets in euclidean space. *Commentarii Mathematici Helvetici*, 53(1):529–535, Dec 1978.

[78] D. G. Larman and C. A. Rogers. The realization of distances within sets in Euclidean space. *Mathematika*, 19:1–24, 1972.

[79] Kwangil Lee and Mark A. Shayman. Optical network design with optical constraints in IP/WDM networks. *IEICE Transactions*, 88-B(5):1898–1905, 2005.

[80] L. A. Levin. Universal sequential search problems. *Probl. Peredachi Inf.*, 9:115–116, 1973.

[81] David Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[82] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 286–293, 1993.

[83] Gary MacGillivray, André Raspaud, and Jacobus Swarts. Obstructions to locally injective oriented improper colourings. *European J. Combin.*, 35:402–412, 2014.

[84] Gary MacGillivray and Jacobus Swarts. The complexity of locally injective homomorphisms. *Discrete Mathematics*, 310(20):2685–2696, 2010.

[85] S. Maheshwari. Traversal marker placement problem are np-complete. In *Research report no CU-CS-092-76*, Dept. of Computer Science, University of Colorado at Boulder, 1976.

[86] P. McMullen. Convex bodies which tile space by translation. *Mathematika*, 27(1):113–121, 1980.

[87] Gernot Metze, Donald R. Schertz, Kilin To, Gordon Whitney, Charles R. Kime, and Jeffrey D. Russell. Comments on "derivation of minimal complete sets of test-input sequences using boolean differences. *IEEE Trans. Computers*, 24(1):108, 1975.

[88] Philippe Meurdesoif, Pierre Pesneau, and François Vanderbeck. Meter installation for monitoring network traffic. In *International Network Optimization Conference (INOC)*, Spa, Belgium, 2007.

[89] Hermann Minkowski. Allgemeine lehrsätze über die convexen polyeder. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1897:198–220, 1897.

[90] B. Moret and H. Shapiro. On minimizing a set of tests. *SIAM Journal on Scientific and Statistical Computing*, 6(4):983–1003, 1985.

[91] L. Moser and W. Moser. Solution to problem 10. *Canadian mathematical bulletin*, 4:187–189, 1961.

[92] Philippe Moustrou. *Geometric distance graphs, lattices and polytopes.* Theses, Université de Bordeaux, December 2017.

[93] Patrenahalli M. Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Trans. Computers*, 26(9):917–922, 1977.

[94] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[95] David Peleg. Low stretch spanning trees. In *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, pages 68–80, 2002.

[96] A. M. Raigorodskii. On the chromatic number of a space. *Russian Mathematical Surveys*, 55(2):351, 2000.

[97] A. M. Raigorodskii. The Erdős-Hadwiger problem and the chromatic numbers of finite geometric graphs. *Sbornik: Mathematics*, 196(1):115, 2005.

[98] D. E. Raiskii. Realization of all distances in a decomposition of the space rn into n+1 parts. *Mathematical notes of the Academy of Sciences of the USSR*, 7(3):194–196, Mar 1970.

[99] Michaël Rao. Exhaustive search of convex pentagons which tile the plane. *CoRR*, abs/1708.00274, 2017.

[100] Pawel Rzazewski. Exact algorithm for graph homomorphism and locally injective graph homomorphism. *Inf. Process. Lett.*, 114(7):387–391, 2014.

[101] Suk Jai Seo and Peter J. Slater. Open neighborhood locating-dominating in trees. *Discrete Applied Mathematics*, 159(6):484–489, 2011.

[102] Suk Jai Seo and Peter J. Slater. Open neighborhood locating-domination for infinite cylinders. In *ACM Southeast Regional Conference*, pages 334–335, 2011.

[103] Dmitry Shiryaev. Personal communication.

[104] Alexander Soifer. *The mathematical coloring book: Mathematics of coloring and the colorful life of its creators*. Springer Science & Business Media, 2008.

[105] Benny Sudakov. Robustness of graph properties. *CoRR*, abs/1610.00117v1, 2016.

[106] Stefan Szeider. Finding paths in graphs avoiding forbidden transitions. *Discrete Applied Mathematics*, 126(2-3):261–273, 2003.

[107] L. A. Székely. Erdős on unit distances and the Szemerédi-Trotter theorems. In *Paul Erdős and his mathematics, II (Budapest, 1999)*, volume 11 of *Bolyai Soc. Math. Stud.*, pages 649–666. János Bolyai Math. Soc., Budapest, 2002.

[108] Rachanee Ungrangsi, Ari Trachtenberg, and David Starobinski. An implementation of indoor location detection systems based on identifying codes. In *INTELLCOMM*, pages 175–189, 2004.

[109] B. A. Venkov. On a class of Euclidean polyhedra. *Vestnik Leningrad. Univ. Ser. Mat. Fiz. Him.*, 9(2):11–31, 1954.

[110] Gueorgui Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs. *J. Reine Angew. Math.*, 134:198–287, 1908.

[111] W. R. Willcox and S. P. Lapage. Automatic construction of diagnostic tables. *Comput. J.*, 15(3):263–267, 1972.

[112] W. R. Willcox, S. P. Lapage, and B. Holmes. A review of numerical methods in bacterial identification. *Antonie van Leeuwenhoek*, 46(3):233–299, 1980.

[113] Douglas R. Woodall. Distances realized by sets covering the plane. *J. Comb. Theory, Ser. A*, 14(2):187–200, 1973.

[114] Dmitry Zhuk. An algorithm for constraint satisfaction problem. In *2017 IEEE 47th International Symposium on Multiple-Valued Logic—ISMVL 2017*, pages 1–6. IEEE, New York, 2017.