# Reconfiguration problems in graphs

Paul Ouvrard

université de BORDEAUX

LABORATOIRE BORDELAIS DE RECHERCHE EN INFORMATIQUE LaBRI

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

**DOCTEUR**

**DE L'UNIVERSITÉ DE BORDEAUX**

ÉCOLE DOCTORALE MATHÉMATIQUES ET INFORMATIQUE

Spécialité Informatique

par **Paul Ouvrard**

---

# RECONFIGURATION PROBLEMS IN GRAPHS

---

**Date de soutenance :** 24 mars 2021

**Devant la commission d'examen composée de :**

| | | |
|---|---|---|
| Mathieu LIEDLOFF | Maître de Conférences, Université d'Orléans | Rapporteur |
| Laurent VIENNOT | Directeur de Recherche, Inria - Université de Paris | Rapporteur |
| Aurélie LAGOUTTE | Maître de Conférences, Université Clermont-Auvergne | Examinatrice |
| Nicolas TROTIGNON | Directeur de Recherche, ENS Lyon | Examinateur |
| Marthe BONAMY | Chargée de Recherche, CNRS - LaBRI | Encadrante |
| Paul DORBEC | Professeur, Université Caen-Normandie | Directeur |
| Cyril GAVOILLE | Professeur, Université de Bordeaux | Directeur |

**Présidée par :** Nicolas TROTIGNON

# Remerciements

Cette thèse fut une très belle aventure, parfois semée d'embûches mais tellement enrichissante. Durant ces quatre dernières années, de nombreuses personnes m'ont aidé à franchir la ligne d'arrivée et j'aimerais ici pouvoir les en remercier.

Je souhaiterais dans un premier temps remercier Mathieu Liedloff et Laurent Viennot pour la relecture attentive de ce manuscrit, surtout en ces temps difficiles. Un grand merci également à Aurélie Lagoutte et Nicolas Trotignon pour avoir accepté de faire partie de mon jury.

Cette thèse n'aurait jamais vu le jour (et encore moins abouti) sans la précieuse aide de mes trois directeurs de thèse, Marthe, Paul et Cyril. Merci pour votre soutien, vos nombreux conseils et votre bonne humeur quotidienne. J'ai pris énormément de plaisir en travaillant à vos côtés, mais j'aimerais aussi vous remercier pour l'autonomie que vous m'avez laissée et pour votre confiance. Je m'estime très chanceux de vous avoir eu comme directeurs, merci pour tout !

J'aimerais également remercier Nicolas Bousquet qui m'a tout particulièrement suivi ces deux dernières années. Je garde notamment un excellent souvenir de notre séjour au Japon et de nos nombreuses discussions, qu'elles soient scientifiques ou non.

I also would like to thank all the members of the DATCORE project. Many thanks to Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Akira Suzuki and Kunihiro Wasa. It has been a great pleasure to work and spend all this time with you! In particular, thanks to the Japanese members for your kindness and your hospitality during our stay in your wonderful country; I have excellent memories of it.

Un grand merci aux participants (que je n'ai pas remerciés jusque là) des trois *Pessac Graph Workshops* auxquels j'ai eu le plaisir d'assister : Oscar Defrain, Tereza Klimošová et Jean-Florent Raymond. Bien entendu, merci à Peppie pour son soutien sans faille, à Marthe et à la famille Simon sans qui tous ces bons moments n'auraient pas existés.

Je souhaiterais également remercier les chercheurs avec lesquels j'ai eu la chance de pouvoir travailler durant ces trois années. En particulier, merci à Cédric Chauve, Clément Dallard et Alice Joffard, j'ai beaucoup apprécié ces sessions travail (mais pas que !) avec vous. Merci également à Mikaël Rabbie, Jukka Suomela et Jara Uitto.

Durant les quatre dernières années, j'ai eu la chance de pouvoir enseigner à l'IUT, et surtout à l'UF Informatique. J'aimerais pouvoir remercier celles et ceux m'ayant fait confiance mais comme je risquerais oublier certaines personnes en donnant des noms, je préfère donc simplement remercier les membres des équipes pédagogiques d'ACSI, POO et COO pour l'IUT ; Algo des graphes, AlgoStruct', CoCa, InitInfo, Ini Prog C, POO et TAP pour l'UF.

J'aimerais également remercier tous les membres du LaBRI pour faire de ce laboratoire un environnement de travail idéal. Naturellement, merci à l'équipe CombAlgo et en particulier à tous les membres du thème Graphes et optimisation pour m'avoir accueilli à bras ouverts. Un grand merci en particulier à tous les doctorants que j'ai eu la chance de rencontrer ces quatre dernières années : Alex, Jonathan, Théo et Claire H. (la famille en premier !), mais aussi Mohammed, Thomas et Dimitri.

De manière plus générale, merci à tous les doctorants du LaBRI pour tous ces moments de convivialité et de légèreté. Je pense notamment à tous les événements organisés par l'AFoDIB comme le traditionnel pot de Noël ou les nombreuses pauses café. Merci à Karim Aderghal, Karim Alami, Tristan, Tobias, Simon, Henri, Valentin, Mathias, Julien, Nathan et Tidiane.

Je souhaiterais également remercier mes amis, du labo et d'en dehors. Merci à Lamine pour nos discussions toujours passionantes, ta culture (scientifique ou autre) m'impressionne à chaque fois. J'ai aussi beaucoup apprécié notre collaboration ! Un énorme merci à Luchito, mon chilien préféré et l'une des plus belles rencontres de cette thèse. Un tout aussi grand merci à Jason a.k.a. ShootShoot et Rohan a.k.a. Roro mes super co-bureaux mais surtout mes super amis. Merci pour tout ce qu'on a partagé ensemble (les galères comme les nombreuses bières), notre amitié a été l'un des piliers de cette thèse, mais aussi et surtout de ma vie en dehors du labo. Un immense merci à Corentin, Laura et Marie, mes amis de longue date et qui partagent avec moi (à Bordeaux et ailleurs !) cette souffrance d'être supporter des Girondins. Merci à Nico le sosie de Tom Cruise et surtout l'animateur de nos soirées ! Pour finir, merci à Dimitri et Marie pour tout. Vivement notre prochain week-end inopiné ensemble !

Merci également à ma belle-famille tout entière pour m'avoir accueilli à bras ouverts dès le départ, et me considérer comme l'un des vôtres. Merci en particulier à mes beaux oncle, tante et cousins de La Rochelle, grands-parents, sœurs et bien sûr parents. Merci pour tout.

Parce que je ne serais rien sans eux et même si ces quelques lignes ne peuvent représenter tout l'amour que je leur porte, j'aimerais remercier ma famille. Merci à tous mes oncles, tantes et cousins. J'aimerais en particulier remercier Augustin, Valentin et bien sûr Lucas avec qui je partage beaucoup, vous êtes tous les trois des cousins géniaux ! Merci aussi à mes grands-parents pour votre soutien et votre amour de tous les jours (Mamie, je pense fort à toi). Merci à Emma, ma sœur, je suis fier de toi et j'admire celle que tu es devenue. Bien sûr, un grand merci à Mehdi, je suis ravi que tu fasses partie de la famille. Merci à vous deux d'être là dans les bons comme dans les mauvais moments, et merci pour ce magnifique bébé des îles, je suis un tonton comblé. Enfin, merci à mes deux merveilleux parents. Merci pour votre soutien de tous les instants et de toute nature, merci d'avoir toujours fait en sorte que je ne manque de rien. Je suis un privilégié d'avoir des parents comme vous et je vous en serai éternellement reconnaissant.

Je ne pourrais pas terminer ces remerciements sans un grand *Miaou* à Pelote, a.k.a. Pepe le pot de colle, notre chat-chien, pour ces ronronnements et ses câlins incessants.

Mes derniers remerciements vont tout naturellement à Amandine. Je ne vais pas lister toutes les raisons que j'aurais de te dire merci - il y en aurait tellement - mais juste te remercier pour ce 19 septembre qui restera une journée inoubliable. J'espère que nous aurons l'occasion de continuer à découvrir le monde ensemble, et je suis sûr que nous saurons profiter pleinement de tous les bonheurs que la vie a à offrir.

# Contents

# Introduction (in French)

## Quelques mots sur les graphes et la domination dans les graphes

Cette thèse s'inscrit dans le cadre la théorie des graphes qui est la branche des mathématiques et de l'informatique étudiant les *graphes*. Un graphe est une structure mathématique composée de *sommets* qui peuvent être connectés deux à deux par des *arêtes*. Avec cette définition, on observe que les graphes sont des modèles très puissants qui peuvent être utilisés pour décrire n'importe quelle relation binaire sur un ensemble : chaque élément de l'ensemble correspond à un sommet, et deux éléments en relation sont reliés par une arête. L'étude des graphes a été initiée par Euler il y a près de trois cents ans, et l'intérêt pour ce domaine des mathématiques discrètes n'a cessé de croître depuis, notamment grâce à l'émergence de l'informatique.

La théorie des graphes a de nombreuses applications dans différents domaines. L'un des plus connus est probablement celui lié à la planification ; en voici un exemple très simple pour l'illustrer. Chaque étudiant d'une université suit un certain nombre de cours et doit passer un examen afin de valider le semestre. Cependant, tous les étudiants ne sont pas inscrits dans les mêmes cours et chaque étudiant ne peut pas avoir deux examens en même temps. Par conséquent, l'université doit répondre à la question suivante : comment faut-il organiser ces examens afin de minimiser le temps total nécessaire ? Ce problème peut être modélisé en tant que problème de graphe comme ceci : chaque cours est un sommet et deux cours ayant des étudiants en commun sont reliés par une arête. L'objectif est de planifier la session d'examen, c'est-à-dire d'assigner un horaire à chaque cours. Le fait que deux examens ne puissent pas être programmés en même temps est modélisé par une arête entre les deux sommets correspondants. Enfin, l'horaire d'un examen peut être représenté par une couleur ; ainsi tous les cours possédant la même couleur ne peuvent pas avoir leur examen se déroulant simultanément. La contrainte indiquant que chaque étudiant ne peut pas avoir deux épreuves se déroulant en même temps implique qu'il ne peut y avoir deux sommets adjacents possédant la même couleur. Il est maintenant clair que minimiser le temps nécessaire pour la session entière d'examens est équivalent à utiliser un nombre minimum de couleurs (voir la figure 1).

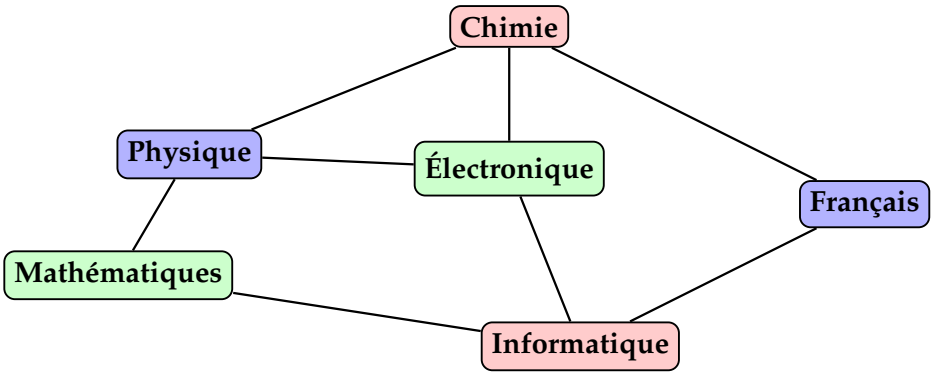



Figure 1 – Illustration du problème.

Dans la figure 1, nous avons trouvé une solution utilisant trois couleurs, et on peut se demander si ce nombre est optimal ou non. En fait, on peut observer que les examens de chimie, d'électronique et de physique ne peuvent pas se dérouler en même temps car les sommets correspondants sont tous deux à deux adjacents. Par conséquent, au moins trois couleurs sont nécessaires. Comme nous avons trouvé une solution utilisant trois couleurs, on peut en conclure que ce nombre est optimal. Plus généralement, ce type de problème peut être abordé en étudiant des propriétés structurelles du graphe sous-jacent. Par exemple, si le graphe possède un ensemble $X$ de sommets deux à deux adjacents (appelé *clique*), alors tous les sommets de $X$ doivent recevoir des couleurs différentes. Ainsi, au moins $|X|$ couleurs sont nécessaires afin de colorer tout le graphe. D'un autre côté, on peut observer que si tous les sommets sont connectés à au plus $\Delta$ autres sommets, on peut toujours colorer le graphe avec $\Delta + 1$ couleurs. Il s'ensuit que le nombre optimal de couleurs nécessaires pour colorer n'importe quel graphe est toujours au moins la taille de sa plus grande clique, mais au plus son degré maximum plus un.

Malheureusement, l'écart entre la taille d'une plus grande clique et le degré maximum peut être arbitrairement grand. Ainsi, il se peut que ces bornes ne soient pas très utiles afin de déterminer le nombre optimal de couleurs. Cela n'aide pas non plus pour effectivement trouver une solution, c'est-à-dire pour réellement planifier la session d'examens. De plus, le nombre de cours différents peut être très important rendant impossible une résolution "à la main". Pour cette raison, il est souvent très intéressant de concevoir des algorithmes rapides permettant de renvoyer une solution. Comme il se peut qu'un algorithme "rapide" trouvant une solution optimale n'existe pas, il est également très intéressant de déterminer à quel point le problème est "dur" à résoudre. Ces deux différentes approches sont précisément celles qui nous intéressent dans cette thèse : étudier les propriétés structurelles d'un problème de graphes, concevoir des algorithmes le résolvant, ou à défaut se concentrer sur sa complexité algorithmique.

Discutons maintenant d'un autre problème de la vie quotidienne pour lequel la théorie des graphes pourrait être utile. Une marque aimerait faire de la publicité afin d'augmenter les ventes de son produit phare. Pour cela, elle souhaiterait utiliser les réseaux sociaux car elle estime qu'il s'agit de la façon la plus efficace d'atteindre des clients potentiels de nos jours. Cependant, il serait très coûteux d'envoyer un message individuel à chaque membre du réseau social. Ainsi, une autre manière de procéder consiste à payer des personnes qui diffuseront elles-mêmes le message auprès de leurs amis virtuels via un poste, un tweet, une vidéo etc. Comme l'entreprise souhaite dépenser le moins d'argent possible, il est crucial de trouver les "bonnes" personnes afin de minimiser le coût total. Ces personnes choisies sont en réalité appelées *influenceurs* car elles ont de nombreux amis (sur les réseaux sociaux) ou sont suivies par de nombreuses personnes. Par souci de simplicité, on suppose que la relation est symétrique, c'est-à-dire que si une personne $A$ suit une personne $B$, alors $B$ suit également $A$. Dans ce cas, on dit que $A$ et $B$ sont amies. Ainsi, le réseau social peut être facilement modélisé par un graphe : chaque membre est un sommet et deux sommets sont adjacents si et seulement si les deux utilisateurs sont amis sur le réseau social. Par conséquent, le problème que la marque doit résoudre est le suivant : quel nombre minimum d'influenceurs faut-il payer afin que chaque personne non rémunérée soit amie avec au moins un influenceur ? Ce problème est appelé ENSEMBLE DOMINANT, et est le problème au départ de cette thèse.

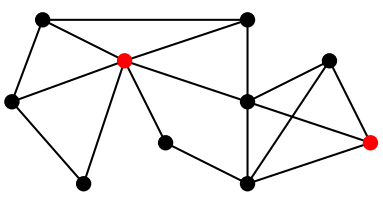
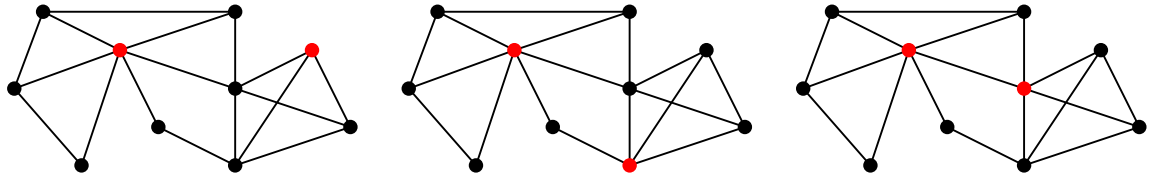


Figure 2 – Ensemble dominant minimum de taille deux d'un graphe.



(a) (b) (c)

Figure 3 – Trois nouveaux ensembles dominants de taille deux du même graphe.

La figure 2 donne un exemple d'un ensemble dominant de taille deux d'un graphe. On peut facilement prouver qu'il s'agit bien d'un ensemble dominant minimum, c'est-à-dire qu'il n'existe pas d'ensemble dominant possédant moins de sommets. En effet, le graphe ne contient pas de sommet adjacent à tous les autres. Ainsi, il n'existe pas d'ensemble dominant de taille un. Cependant, notons que le graphe admet trois autres ensembles dominants de taille deux que celui représenté dans la figure 2 ; voir la figure 3. De plus, on peut observer que sur la figure 2 ainsi que sur les figures 3a et 3b, les deux sommets qui forment l'ensemble dominant ne sont pas adjacents, contrairement à la figure 3c. Plus généralement, on peut se concentrer sur certains cas particuliers d'ensembles dominants : que se passe-t-il si nous souhaitons avoir un ensemble dominant indépendant, c'est-à-dire dont les sommets sont deux à deux non adjacents ? À l'inverse, combien cela coûte-t-il (en termes de sommets supplémentaires) d'avoir un ensemble dominant connexe comme dans la figure 3c ? Ces ensembles dominants particuliers sont intéressants pour différentes raisons. Par exemple, les ensembles dominants connexes peuvent avoir une application dans le calcul du routage pour des réseaux mobiles ad hoc. En effet, un ensemble dominant connexe peut servir de "colonne vertébrale" pour les communications : les nœuds n'appartenant pas à l'ensemble dominant connexe peuvent communiquer avec les autres en transmettant leurs messages à un de leur voisin qui appartient à l'ensemble [WL99].

## Reconfiguration combinatoire

D'un point de vue plus théorique, on peut également se demander si ces quatre ensembles dominants de taille deux sont tous équivalents, à une opération élémentaire fixée près. Le cas échéant, peut-on borner la longueur d'une transformation entre deux ensembles dominants donnés ? D'autre part, si cela n'est pas toujours possible, quelle est la complexité de déterminer si un ensemble dominant peut être transformé en un autre ? Toutes ces problématiques peuvent être résumées par cette question très simple mais générale : étant donné une configuration courante, est-il possible d'atteindre une configuration cible fixée ? Dans ce contexte, on peut observer que de nombreux jeux à un joueur comme le *Rubik's cube*, le *jeu de Taquin* ou encore *Rush Hour* peuvent être considérés comme des problèmes de reconfiguration. Ce formalisme est très large car il peut être appliqué à la plupart des problèmes combinatoires, plusieurs opérations permettant de transformer une solution peuvent être considérées et il soulève de nombreuses questions très intéressantes. Dans cette thèse, nous nous intéressons seulement à la reconfiguration de problèmes de graphes, mais nous soulignons que de nombreux problèmes comme SATISFIABILITÉ peuvent être étudiés dans le cadre de ce formalisme. Ito et al. [IDH+08] ont initié une étude systématique de la complexité de ces questions. Contrairement à ce qui se passe pour les jeux à un joueur mentionnés ci-dessus, dans la reconfiguration de problèmes de graphes, il est nécessaire que chaque étape intermédiaire entre les solutions source et cible soit également une solution du problème (par exemple nous devons nous assurer qu'il s'agit bien d'un ensemble dominant). Cette contrainte rend le problème beaucoup plus intéressant car il se peut qu'une transformation n'existe pas. Certaines solutions peuvent même être *gelées*, ce qui signifie qu'elle ne peuvent pas du tout être modifiées. Une *séquence de reconfiguration* entre deux

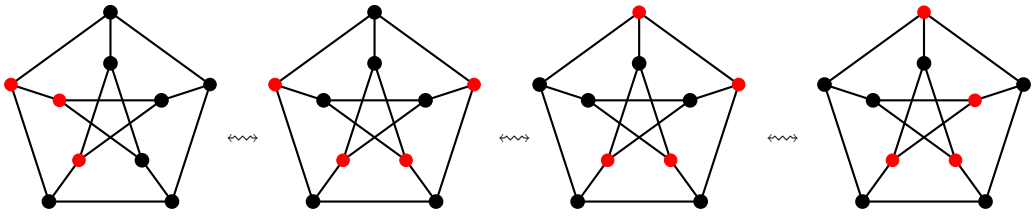
solutions $S_s$ et $S_t$ d'un problème de $\Pi$ est une séquence $\langle S_0 = S_s, S_1, \ldots, S_{\ell-1}, S_\ell = S_t \rangle$ telle que chaque $S_i$ est également une solution de $\Pi$ qui peut être obtenue à partir de $S_{i-1}$ en appliquant une seule modification appelée *règle de reconfiguration*. Notons que cette règle est unique et ne peut donc pas être changée durant la transformation. En revanche, il existe souvent plus d'un choix naturel pour celle-ci. Par exemple, si nous nous intéressons à la reconfiguration d'ensembles dominants, trois règles ont été principalement étudiées jusqu'ici :

- ajout ou suppression d'un seul sommet ;
- remplacement d'un sommet par un autre ;
- remplacement d'un sommet par l'un de ses voisins.

Il est évident que pour la première règle, nous devons ajouter une contrainte supplémentaire sur la taille de chaque solution intermédiaire sans quoi le problème devient trivial. En effet, en ajoutant d'abord chaque sommet de $S_t \setminus S_s$ puis en supprimant chaque sommet de $S_s \setminus S_t$, nous obtenons une (plus courte) transformation entre $S_s$ et $S_t$. Bien qu'il s'agisse d'un problème intéressant, la plupart des travaux dans ce domaine n'étudient pas la relation entre ces différentes règles de reconfiguration. Les questions les plus étudiées sont les suivantes :

- ACCESSIBILITÉ : existe-t-il une séquence de reconfiguration entre deux solutions ?
- CONNEXITÉ : existe-t-il une séquence de reconfiguration entre toutes les paires de solutions ?
- PLUS COURTE TRANSFORMATION: existe-t-il une séquence de reconfiguration de longueur au plus $\ell$ entre deux solutions ?

Pour tous ces problèmes, on peut s'intéresser à la complexité du problème de décision associé. Cependant, ils sont souvent PSPACE-complets, y compris lorsque le problème original peut se résoudre en temps polynomial (comme par exemple le problème du COUPLAGE PARFAIT) .



(a) $D_0 = D_s$ (b) $D_1$ (c) $D_2$ (d) $D_3 = D_t$

Figure 4 – Séquence de reconfiguration d'ensembles dominants du graphe de Petersen.

## Aperçu de la thèse

Dans cette thèse, nous nous intéressons uniquement à la reconfiguration de problèmes de graphes. Le problème au départ de cette thèse est celui de l'ENSEMBLE DOMINANT. Nous étudions notamment la relation entre un ensemble dominant et un ensemble dominant connexe dans le chapitre 2, puis nous nous focalisons sur la reconfiguration d'ensembles dominants dans le chapitre 3. Nous nous intéressons également à la reconfiguration de deux autres problèmes dans le chapitre 4 : la reconfiguration d'arbres couvrants et la $k$-recoloration. Ainsi, cette thèse se divise en quatre chapitres que nous allons maintenant présenter rapidement.

Le chapitre 1 est consacré aux définitions, notations, ainsi que certains résultats connus de la théorie des graphes que nous utiliserons. Ce chapitre contient également une introduction plus complète sur la reconfiguration combinatoire dans la section 1.5. Nous y présentons notamment quelques résultats connus dans ce domaine. Cependant, pour un aperçu plus complet, le lecteur est invité à consulter les synthèses de van den Heuvel [vdH13] et Nishimura [Nis18]. Pour chaque problème que nous étudions dans cette thèse, une introduction contenant les résultats connexes est reportée au début de la section correspondante.

La première partie du chapitre 2 est une introduction à la domination dans les graphes. Nous présentons le problème ainsi que certains résultats connus. Nous discutons également de la relation entre le nombre de domination d'un graphe (c'est-à-dire la taille minimum d'un ensemble dominant), son nombre de domination supérieure (c'est-à-dire la taille maximum d'un ensemble dominant minimal par inclusion) et d'autres paramètres de graphes. Nous nous concentrons également sur la complexité du problème ENSEMBLE DOMINANT. Nous donnons d'abord deux célèbres réductions polynomiales prouvant sa NP-complétude ; nous en utiliserons une dans le chapitre 3. Nous présentons également deux algorithmes connus permettant de calculer le nombre de domination des arbres et des graphes d'intervalles en temps linéaire. Dans la seconde partie du chapitre 2, nous étudions la relation entre le nombre de domination d'un graphe et la taille d'un ensemble dominant connexe minimum. Cette section est basée sur un travail conjoint en cours avec Marthe Bonamy, Nicolas Bousquet et Tereza Klimošová. Plus précisément, nous montrons que la taille d'un ensemble dominant minimum connexe de tout-sous graphe induit $H$ d'un graphe $G$ est au plus deux fois le nombre de domination de $H$ si et seulement $G$ ne contient pas de $P_9$, de $C_9$ ou le graphe suivant comme sous-graphe induit :

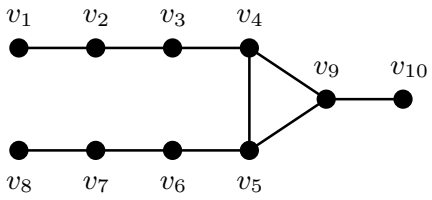



Figure 5 – Le troisième sous-graphe induit interdit.

Le chapitre 3 est entièrement consacré à la reconfiguration d'ensembles dominants par *token addition and removal* et *token sliding*. Dans la section 3.1, nous présentons un travail en commun avec Nicolas Bousquet et Alice Joffard [BJO20]. Plus précisément, nous nous concentrons sur le seuil $k$ correspondant à la taille maximum de chaque solution intermédiaire. Nous montrons que pour certaines valeurs spécifiques de $k$, il existe toujours une transformation courte (ici "courte" signifie de longueur linéaire) entre n'importe quelle paire d'ensembles dominants. De plus, nos preuves sont constructives et nous permettent de calculer une telle séquence de reconfiguration en temps polynomial (ou en temps FPT pour le résultat concernant la largeur arborescente). Dans la section 3.2, nous étudions la complexité de la question d'accessibilité d'ensembles dominants par *token sliding*. Il s'agit d'un travail en commun avec Marthe Bonamy et Paul Dorbec [BDO21]. Ce travail étant le premier à notre connaissance à considérer la reconfiguration d'ensembles dominants sous cette règle, nous commençons par introduire le problème. En particulier, nous décidons d'autoriser la superposition de jetons sur un même sommet afin d'éviter des exemples artificiels négatifs. Notre résultat principal est que le problème est PSPACE-complet, y compris lorsque l'on se restreint aux graphes bipartis par exemple. Il peut cependant se résoudre en temps polynomial dans les cographes ou les graphes dually chordaux, une classe contenant notamment les arbres et les graphes d'intervalles. Enfin, nous présentons une nouvelle variante de reconfiguration d'ensembles dominants appelée OPT-DSR récemment introduite par Ito et al. [IMNS19] pour la reconfiguration d'ensembles

indépendants. Dans ce nouveau problème, étant donné un ensemble dominant $D$ d'un graphe $G$, l'objectif est de trouver le plus petit ensemble dominant de $G$ qui est accessible depuis $D$ par *token addition and removal*. Après une rapide présentation, nous nous concentrons sur sa complexité. Nous nous intéressons principalement à sa complexité paramétrée, mais nous prouvons également qu'il est PSPACE-complet. Enfin, nous montrons que si l'objectif est de trouver le plus petit ensemble dominant indépendant accessible depuis $D$, alors le problème est également PSPACE-complet, y compris dans les graphes bipartis. Il s'agit d'un travail en commun avec Alexandre Blanché, Haruka Mizuta et Akira Suzuki [BMOS20].

Dans le chapitre 4, nous changeons de problème hôte. Ce chapitre est composé de deux parties qui sont totalement indépendantes. Dans la section 4.1, nous nous intéressons à la complexité de la reconfiguration d'arbres couvrants avec des contraintes supplémentaires sur le nombre de feuilles. Plus précisément, nous nous concentrons sur la reconfiguration d'arbres couvrants possédant respectivement au moins $k$ feuilles, et au plus $k$ feuilles. Bien qu'il soit connu qu'il existe toujours une transformation entre deux arbres couvrants [IDH+08], nous montrons que ces nouvelles contraintes rendent le problème beaucoup plus difficile. Plus précisément, nous prouvons que ces deux problèmes sont PSPACE-complets. Nous exhibons également un algorithme polynomial pour la reconfiguration d'arbres couvrants possédant au moins $n-2$ feuilles, $n$ étant le nombre de sommets du graphe. Nous expliquons ensuite comment nous pouvons utiliser ce résultat pour concevoir un algorithme polynomial pour les cographes. Il s'agit d'un travail conjoint avec Nicolas Bousquet, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Akira Suzuki and Kunihiro Wasa [BIK+20]. Enfin, dans la section 4.2, nous nous concentrons sur des problèmes de $k$-recoloration. Cependant, nous n'étudions pas ce problème dans un cadre centralisé, mais de manière décentralisée. Plus précisément, nous utilisons le modèle LOCAL d'algorithmique distribuée dans lequel chaque sommet reçoit un identifiant unique mais n'a aucune connaissance préalable du graphe. Évidemment, chaque sommet connaît également sa couleur initiale, ainsi que sa couleur cible. En outre, un sommet peut communiquer avec ses voisins et réaliser un calcul à chaque étape. Grâce à cela, il peut ainsi décider de changer sa couleur en conséquence. Comme le problème de $k$-coloration est intrinsèquement global et qu'une solution n'existe pas toujours, nous décidons d'autoriser des couleurs supplémentaires durant la transformation. Nous nous concentrons uniquement sur des instances positives, et notre objectif de minimiser à la fois le nombre de rondes de communications et le nombre maximum d'étapes nécessaires pour que chaque sommet puisse atteindre sa couleur finale. Nous présentons dans un premier temps des bornes inférieures et supérieures simples, avant de nous concentrer dans un second temps sur la 3-recoloration d'arbres et de graphes sous-cubiques. Il s'agit d'un travail en commun avec Marthe Bonamy, Mikaël Rabie, Jukka Suomela et Jara Uitto [BOR+18].

# Introduction

## A few words on graphs and domination in graphs

This thesis lies in the field of graph theory which is the branch of mathematics and computer science that studies *graphs*. A graph is a mathematical structure made up of *vertices* that may be pairwise connected by *edges*. With this definition, it is observed that graphs are very powerful models that can be used to describe any binary relation on a set: each element of the set corresponds to a vertex, and two elements that are in relation are connected by an edge. The study of graphs was pioneered by Leonhard Euler nearly three hundred years ago, and interest in this field of discrete mathematics has continued to grow ever since, thanks in particular to the emergence of computer science.

Graph theory has many applications in various fields. Probably, one of the most famous one is related to scheduling; here is a very simple example to illustrate it. Each student at a university takes a number of courses and must pass the exam in order to validate the semester. However, not all students are enrolled in the same courses and each student cannot have two exams at the same time. Hence, the university has to answer the following question: how should it organize these exams in order to minimize the amount of time needed for the whole session? Actually, this problem can be modeled as a graph problem as follows: each course is a vertex, and two courses that have students in common are connected by an edge. The goal is to schedule the exam session, i.e., to assign a timetable to each course. The fact that two exams cannot be scheduled at the same time is modeled by an edge between the two corresponding vertices. Finally, the schedule of an exam can be represented by a color; hence all the vertices with the same color cannot have their exam taking place simultaneously. The constraint that each student cannot have two exams at the same time implies that no two adjacent vertices can receive the same color. It is now clear that minimizing the amount of time needed for the whole exam session is equivalent to using a minimum number of colors (see Figure 1).

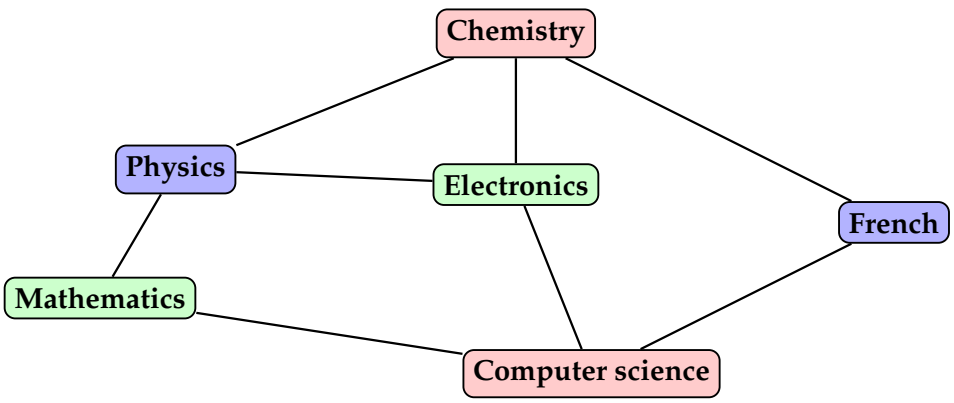



Figure 1 – Illustration of this problem.

In Figure 1, we found a solution with three colors and one can wonder whether this number of colors is optimal or not. Actually, it is observed that the exams of Chemistry, Electronics and Physics cannot occur at the same time because the corresponding vertices are pairwise adjacent. Hence, at least three colors are required. Since we found a solution using three colors, one can conclude that this number is optimal. More generally, this kind of problem can be tackled by studying structural properties of the underlying graph. For instance, if the graph has a set $X$ of pairwise adjacent vertices (called a *clique*), then all the vertices of $X$ must receive different colors and thus at least $|X|$ are needed to color the whole graph. On the other hand, one can observe that if every vertex is connected to at most $\Delta$ other vertices, then one can always color the graph with $\Delta + 1$ colors. This suggests that the optimal number of colors needed to color any graph is at least the size of its largest clique, but at most its maximum degree plus one.

Unfortunately, the gap between the size of a largest clique and the maximum degree can be arbitrarily large, and thus it might not be very helpful to find the optimal number of colors. And it does not help us to actually find a solution, i.e., to schedule the exam session. Moreover, the number of different courses may be huge and thus, making impossible to solve it by hand. Therefore, one might be interested to design fast algorithms to find a solution. Because a "fast" algorithm finding an optimal solution might not exist, it is also very interesting to determine how hard it might be to solve it. These two different approaches are precisely the ones that interest us in this thesis: studying structural properties of a graph problem, designing algorithms to solve it, or focusing on its computational complexity.

Let us now discuss about another daily-life problem where graph theory might be useful. A brand company would like to advertise in order to increase the sales of its flagship product. To this end, it would like to use social media because it believes this is the most effective way to reach potential clients nowadays. However, it would be very costly to send an individual message to each member of the social media. Therefore, another possible way is to pay some people to spread the message to their virtual friends themselves; it could be via a post, a tweet, a video and so on. Because the company wants to spend as little money as possible while reaching all the members of the social media, it is crucial to choose the right people so that the company minimizes the number of paid people. These people are usually called social media influencers, because they have many friends (on the social media) or are followed by a lot of people. For the sake of simplicity, suppose that the relation is symmetric, i.e., if a person $A$ follows a person $B$, then $B$ also follows $A$. In that case, we say that $A$ and $B$ are *friends*. Thus, the social media can be easily modeled by a graph: each member is a vertex, and two vertices are adjacent if and only if the two users are friends on the social media. Hence, the problem that the company must solve is the following: what is the minimum number of influencers it has to pay so that each non-chosen people is friend with at least one chosen influencer? This problem is the so-called DOMINATING SET problem, which is the original problem of this thesis.

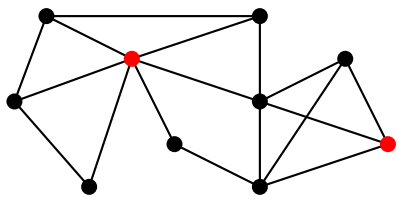
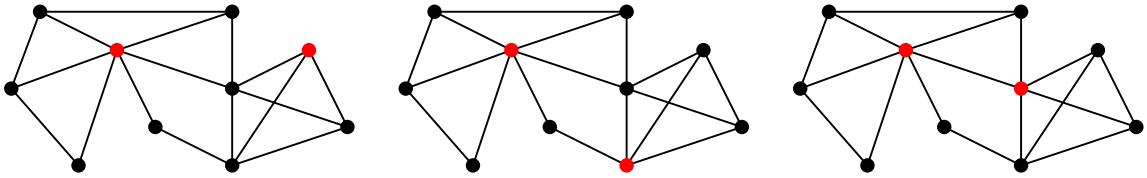


Figure 2 – Minimum dominating set of size two of a graph.

Figure 2 gives a example of a dominating of size two of a graph. One can easily prove that this is actually a minimum dominating set, i.e., there is no dominating set with less vertices. Indeed, the graph does not contain a vertex adjacent to all the others, hence there is no dominating set of size one. However, note that the graph admits three more dominating sets of



(a) (b) (c)

Figure 3 – Three different dominating sets of size two of the same graph.

size two different from the one depicted in Figure 2 (see Figure 3). Moreover, one can observe that on Figure 2 as well as Figures 3a and 3b, the two vertices that form the dominating set are not adjacent while they are neighbors in Figure 3c. More generally, one can focus on some special cases of dominating sets: what happens if we want an independent dominating set, i.e., a dominating set where the vertices are pairwise non-adjacent? On the opposite, how does it cost (in terms of extra vertices) to have a connected dominating set like in Figure 3c? These special dominating sets are interesting for various reasons. For instance, connected dominating sets may find application in the computation of routing for mobile ad hoc networks. Indeed, a connected dominating set can be used as a backbone for communications: nodes that are not in the connected dominating set can communicate with the others by passing messages through a neighbor in the set [WL99].

## Combinatorial reconfiguration

From a more theoretical point of view, one may also wonder whether these four dominating sets of size two are all equivalent up to a fixed elementary operation. If so, can we bound the length of a transformation between two given dominating sets? On the other, if this is not always possible, what is the complexity of determining if one dominating set can be transformed into another one? All these questions can be embedded into a more general framework called *combinatorial reconfiguration*, and can be summarized by this very simple but general question: given a current configuration, is it possible to reach a fixed target? In that context, one can observe that a lot of one-player games like *Rubik's cube*, the *15-puzzle* or *Rush hour* for instance, can be considered as reconfiguration problems. This framework is very large as it can be applied to most combinatorial problems, several operations that allow us to modify a solution can be considered and it raises a lot of very interesting questions. In this thesis, we are only interested in the reconfiguration of graph problems but we stress that many problems such as SATISFIABILITY can be studied through the lens of this framework. Ito et al. [IDH+08] initiated a systematic study of the complexity of these problems. Unlike what happens for the one-player games mentioned above, in the reconfiguration of graph problems it is required that each intermediate step between the source and the target solutions is also a feasible solution of the problem, e.g., we must ensure that it is a dominating set. This constraint makes the problem much more interesting as it might be possible that a transformation does not exist at all, and some solution may even be *frozen* meaning that there is no way to modify them. A *reconfiguration sequence* between two feasible solutions $S_s$ and $S_t$ of a problem $\Pi$ is a sequence $\langle S_0 = S_s, S_1, \ldots, S_{\ell-1}, S_\ell = S_t \rangle$ such that each $S_i$ also is a feasible solution of $\Pi$ and it can be obtained from $S_{i-1}$ by a single move called *reconfiguration rule*. Note that this rule is unique, hence it cannot be changed during a transformation. However, there is often more than one natural choice for this rule. For instance, if we focus on the reconfiguration of dominating set, three different ones have been studied so far:

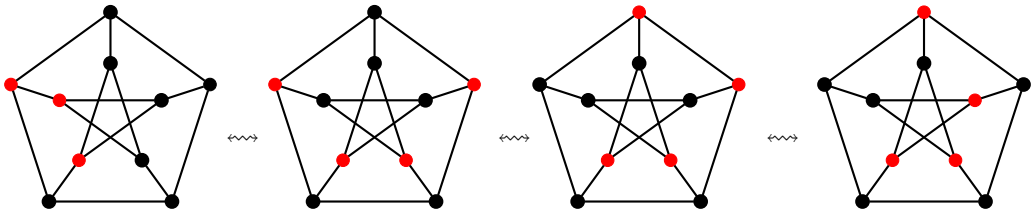
- addition or removal of a single vertex;

- the replacement of a vertex by any other one;
- the replacement of a vertex by one of its neighbors.

Obviously, for the first one, one need to add a constraint on the size of each intermediate solution otherwise the problem would become trivial. Indeed, first adding each vertex in $S_t \setminus S_s$ and then removing every vertex in $S_s \setminus S_t$ yields a (shortest) transformation between $S_s$ and $S_t$. While this is an interesting problem, most work on this field do not study the relation between these three different reconfiguration rules. The most studied questions are the following:

- REACHABILITY: does there exist a reconfiguration sequence between two given solutions?
- CONNECTIVITY: does there exist a reconfiguration sequence between any pair of feasible solutions?
- SHORTEST TRANSFORMATION: does there exist a reconfiguration sequence of length at most $\ell$ between two feasible solutions?

For all of those problems, one might be interested in the complexity of the associated decision problem. However, they are usually PSPACE-complete, even when the original problem is polynomial-time solvable, like PERFECT MATCHING.



(a) $D_0 = D_s$ (b) $D_1$ (c) $D_2$ (d) $D_3 = D_t$

Figure 4 – Reconfiguration sequence between two dominating sets of the Petersen graph.

## Overview of the thesis

In this thesis, we are only interested in reconfiguration problems on graphs. The original problem is DOMINATING SET and we study the relation between a dominating set and a connected dominating set in Chapter 2 as well as the reconfiguration of dominating sets in Chapter 3. We also focus on the reconfiguration of two other problems, namely spanning trees and $k$-colorings in Chapter 4. Therefore, the thesis is divided in four different chapters; let us introduce them.

Chapter 1 is devoted to definitions, notations and some known results of graph theory that we will use throughout the thesis. This chapter also contains in Section 1.5 a more detailed introduction to combinatorial reconfiguration. We present some known results on this field. However, for a more complete overview, we refer the reader to the surveys of van den Heuvel [vdH13] and Nishimura [Nis18]. For each problem that we study in this thesis, the introduction containing related results is deferred to the beginning of the corresponding section.

The first part of Chapter 2 is an introduction to domination in graphs. We present the problem and give some known results. We also discuss the relation between the domination number of a graph (i.e., the minimum size of a dominating set), its upper domination number (i.e., the maximum size of an inclusion-wise minimal dominating set) and other graph parameters. We then focus on the computational complexity of the DOMINATING SET problem. We first give two well-known polynomial-time reductions to prove its NP-completeness; we will use

one of them in Chapter 3. We also present some well-known algorithms that can be used to solve the problem on restricted graph classes like trees and interval graph in linear time. In the second part of Chapter 2, we study the relation between minimum dominating sets and connected dominating sets of a graph. This section is based on an ongoing joint work with Marthe Bonamy, Nicolas Bousquet and Tereza Klimošová. More precisely, we prove that for any induced subgraph $H$ of a graph $G$, the minimum size of a connected dominating set of $H$ is at most twice the size of a minimum dominating set of $H$ if and only if $G$ does not contain $P_9$, $C_9$ or the following graph as an induced subgraph:

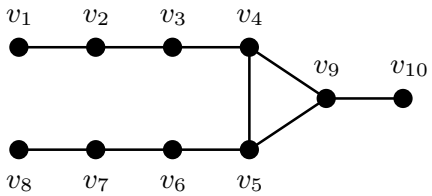



Figure 5 – The third forbidden induced subgraph.

Chapter 3 is entirely devoted to the reconfiguration of dominating sets under the token addition and removal and token sliding rules. In Section 3.1, we present a joint work with Nicolas Bousquet and Alice Joffard [BJO20]. More precisely, we focus on the threshold $k$ corresponding to the maximum size of each intermediate solution. We show that for some specific values of $k$, one can always find a short transformation (here "short" means of linear length) between any two dominating sets. Moreover, our proofs are constructive and such a reconfiguration sequence can be found in polynomial time (or in time FPT for the result regarding the treewidth). In Section 3.2, we investigate the computational complexity of the reachability of dominating sets reconfiguration under token sliding. This part is based on a joint work with Marthe Bonamy and Paul Dorbec [BDO21]. Because our work is the first one to consider the reconfiguration of dominating sets under this rule to the best of our knowledge, we start by introducing the problem. In particular, we decide to allow the superposition of tokens to avoid some artificial negative examples. Our main result is that the problem is PSPACE-complete even when restricted to bipartite graphs for instance, while it is polynomial-time solvable on cographs or dually chordal graphs, a class containing trees and interval graphs. Finally, we present a new optimization variant of dominating sets reconfiguration called OPT-DSR recently introduced by Ito et al. [IMNS19] for the reconfiguration of independent sets. In this new problem, we are given a dominating set $D$ of a graph $G$ and we want to find the smallest dominating set of $G$ that is reachable from $D$ under the token addition and removal rule. After presenting it and making some useful observations, we study its complexity. We mainly focus on its parameterized complexity, but we also prove that it is PSPACE-complete. Finally, we show that if the goal is to find a smallest independent dominating set reachable from $D$, then the problem is also PSPACE-complete even when restricted to bipartite graphs. This is joint work with Alexandre Blanché, Haruka Mizuta et Akira Suzuki [BMOS20].

In Chapter 4, we change the host problem. This chapter contains two different parts, that are completely independent. In Section 4.1, we investigate the complexity of the reconfiguration of spanning trees with additional constraints on the number of leaves. More precisely, we focus on the reconfiguration of spanning trees with respectively at least $k$ leaves, and at most $k$ leaves. While it is known that there always exists a transformation between two spanning trees [IDH+08], we show that these new constraints make the problem much harder. More precisely, we prove that these two problems are PSPACE-complete. We also provide a polynomial-time algorithm for spanning trees with at least $n-2$ leaves, and explain how to use this result to design a polynomial-time algorithm for cographs. This is joint work with Nicolas Bousquet,

Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Akira Suzuki and Kunihiro Wasa [BIK+20]. Finally, in Section 4.2, we consider $k$-recoloring. However, we do not study this problem in the "centralized setting" but in a decentralized way. More precisely, we use the LOCAL model in Distributed Computing where each vertex has a unique identifier but has no knowledge about the graph. Obviously, each vertex knows its initial color and its target color. Moreover, it can communicate with its neighbors at any step, and it can perform a computation. Thanks to this, it can then decide to change its color accordingly. Because the $k$-recoloring problem is inherently global and solutions do not always exist, we decide to allow extra colors during the transformation. We only focus on positive instances, and our goal is to minimize both the number of communication rounds and the maximum number of steps needed so that each vertex can reach its target color. We present first some simple lower and upper bounds. We then focus on 3-recoloring trees and subcubic graphs. This is based on joint work with Marthe Bonamy, Mikaël Rabie, Jukka Suomela and Jara Uitto [BOR+18].

# 1 Preliminaries

**Contents**



## 1.1 Basic definitions

### 1.1.1 A few words on finite sets

**Cardinality, subset, complement.** Let $S$ be a finite set. We denote by $|S|$ the cardinality (or size) of $S$, i.e., the number of elements in $S$. A $k$-subset of $S$ is a set $S' \subseteq S$ containing exactly $k$ distinct elements of $S$. The number of $k$-subsets of a finite set $S$ on $n$ elements is $\binom{n}{k}$. Let $X$ be a subset of $S$, i.e., $X \subseteq S$. We denote by $\overline{X}$ the complement of $X$ in $S$, that is $\overline{X} = S \setminus X$.

**Bijection between two sets.** Let $S_1$ and $S_2$ be two sets, and let $f : S_1 \mapsto S_2$ be a mapping. The mapping $f$ is *injective* if for every element $x \in S_2$, there is at most one element $x' \in S_1$ such that $f(x') = x$. The mapping $f$ is *surjective* if for every element $x \in S_2$, there is at least one element $x' \in S_1$ such that $f(x') = x$. The mapping $f$ is *bijective* if it is both injective and surjective. The inverse mapping of $f$ is a bijective mapping from $S_2$ to $S_1$. In particular, if $S_1$ and $S_2$ are two finite sets such that there exists a bijection between them, then $|S_1| = |S_2|$.

**Symmetric difference.** Let $S_1$ and $S_2$ be two finite sets. We denote by $S_1 \triangle S_2$ the *symmetric difference* of $S_1$ and $S_2$, that is $S_1 \triangle S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1) = (S_1 \cup S_2) \setminus (S_1 \cap S_2)$ .

**Partition of a set.** Let $S$ be a finite set. A *partition* of $S$ is a set of subsets of $S$ denoted by $S_1, S_2, \ldots, S_k$ satisfying the following:

(i) each element of $S$ belongs to some subset $S_i$, i.e., $\cup_{i=1}^{k} S_i = S$;

(ii) the subsets are pairwise disjoint, i.e., $S_i \cap S_j = \emptyset$;

(iii) every subset $S_i$ is non empty, i.e., $S_i \neq \emptyset, 1 \leq i \leq k$.

**Maximal/minimal set and maximum/minimum set.** Let $P$ be a property, and let $S$ be a set satisfying $P$. We say that $S$ is inclusion-wise *minimal* for $P$ if every proper subset $S' \subset S$ does not satisfy $P$. We say that $S$ is *minimum* for property $P$ if there is no set $S'$ such that $|S'| < |S|$ satisfying $P$ (note that $S'$ is not necessarily a subset of $S$). It follows that every minimum set is minimal. However, the converse is often false.

We say that $S$ is inclusion-wise *maximal* for $P$ if every proper superset $S'$ of $S$ (i.e., $S' \supset S$) does not satisfy $P$. We say that $S$ is *maximum* for $P$ if there is no set $S'$ such that $|S'| > |S|$ satisfying $P$. Here again, each maximum set is also maximal, but the converse may not be true.

### 1.1.2 Basic definitions on graphs

A *graph* is an object made from points called *vertices*, with lines (called *edges*) connecting some pairs of points. A graph is *simple* if there is neither a line connecting a point to itself (i.e., we say that the graph is *loopless*) nor two parallel lines connected the same pair of points (i.e., there is no *multiple edges*). More formally, a simple graph $G$ is an ordered pair $(V, E)$ where $V$ is a non-empty finite set of vertices and $E$ is a set of two-subsets of $V$. Hence, the number of edges of a simple graph on $n$ vertices is at most $\binom{n}{2} = \frac{n \cdot (n-1)}{2}$. We simply denote by $uv$ the edge $\{u, v\}$. The vertices $u$ and $v$ are the *endpoints* of the edge $uv$. We say that the edge $uv$ is *incident* to $u$ and $v$, and that $u$ and $v$ are *adjacent*.

In the remaining of this thesis, we only consider finite graphs. Let $G = (V, E)$ be a finite graph. We denote by $V(G)$ and $E(G)$ the vertex set and edge set of $G$, respectively. If there is no ambiguity, we simply denote them by $V$ and $E$. The *order* of $G$ is its number of vertices and we denote it by $n(G)$ (or simply $n$ if there is no ambiguity), i.e., $n(G) = |V(G)|$. Similarly, we denote by $m(G)$ (or simply $m$ if the context is clear) its number of edges, i.e., $m(G) = |E(G)|$.

**Degree.** The *degree* of a vertex $u \in V$ is the number of edges incident to $u$. We denote it $\deg_G(u)$, or simply $\deg(u)$ when it is clear from the context. A vertex of degree zero is an *isolated vertex*, a vertex of degree one is a *leaf* (or *pendant vertex*), and a *universal vertex* is a vertex adjacent to all the vertices. The *minimum degree* (resp. maximum degree) of $G$, denoted by $\delta(G)$ (resp. $\Delta(G)$), is the minimum (resp. maximum) over all the degrees of the vertices of $G$. More formally, we have:

$$\delta(G) = \min_{u \in V} \deg(u) \text{ and } \Delta(G) = \max_{u \in V} \deg(u).$$

**Neighborhood.** The *open neighborhood* of a vertex $u$, denoted by $N_G(u)$, is the set of vertices which are adjacent to $u$, i.e., $N_G(u) = \{v \mid uv \in E\}$. Hence, $|N_G(u)| = \deg_G(u)$. The *closed neighborhood* of $u$, denoted by $N_G[u]$ is the set $N_G(u) \cup \{u\}$. When there is no ambiguity, we simply denote by $N(u)$ and $N[u]$ the open neighborhood and closed neighborhood of $u$, respectively.

Let $S \subseteq V(G)$ be a subset of vertices. One can define the closed neighborhood of $S$, denoted by $N[S]$ as the set $\bigcup_{u \in S} N_G[u]$. The open neighborhood of $S$ is $N[S] \setminus S$, i.e., the set of vertices which have a neighbor in $S$ but that are not in $S$.

**Graph isomorphism.** Let $H$ be a graph. The graphs $G$ and $H$ are *isomorphic* if there exists a bijection between the vertex set of $G$ and the one of $H$ that preserves the adjacency relation. More formally, let $f : V(G) \mapsto V(H)$ be a bijective function. Then, $G$ and $H$ are isomorphic if for any two vertices $u, v \in V(G)$, $uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$. If $G$ and $H$ are isomorphic, we write $G \simeq H$.

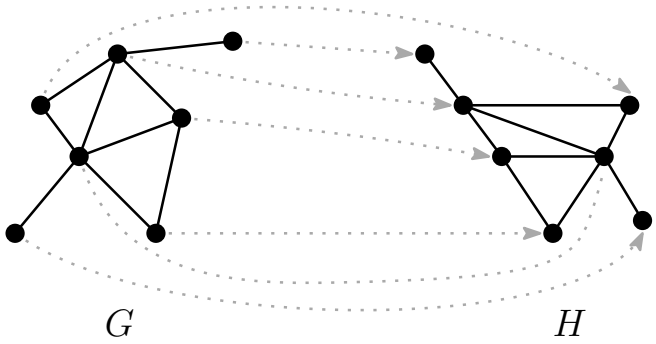



Figure 1.1 – Isomorphism between two graphs $G$ and $H$.

**Subgraphs.** Let $S \subseteq V$. The *subgraph of $G$ induced by $S$*, denoted by $G[S]$, is the graph with vertex set $S$ and which can be obtained from $G$ by keeping all the edges of $G$ with both endpoints in $S$. More formally, $V(G[S]) = S$ and $E(G[S]) = \{uv \in E \mid u, v \in S\}$. If $S = V \setminus \{u\}$ for some vertex $u \in V$, we sometimes denote by $G - u$ the graph obtained from $G$ by removing $u$ and all the edges incident to $u$. In other words, $G - u$ denotes the graph $G[V \setminus \{u\}]$.

A graph $H$ is an *induced subgraph* of $G$ if there exists a subset $S \subseteq V$ such that $G[S]$ is isomorphic to $H$. In other words, the graph $H$ can be obtained from $G$ by a vertex-removals sequence, i.e., we iteratively remove a vertex and all the edges incident to it and repeat this process in the resulting graph. If $H$ indeed is an induced subgraph of $G$, we say that $G$ *contains $H$ as an induced subgraph*. On the other hand, we say that $G$ if *$H$-free* if it does not contain $H$ as an induced subgraph.

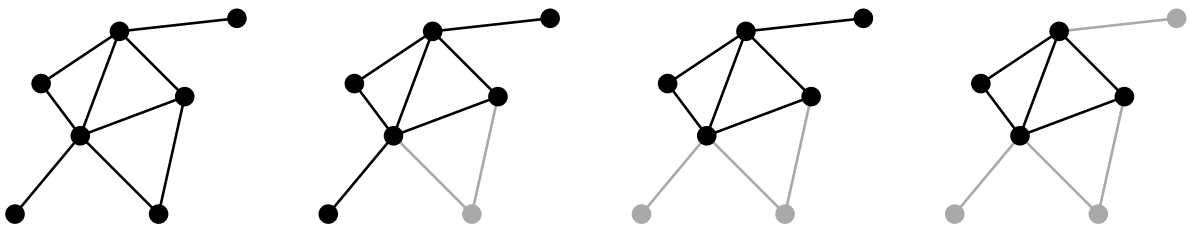


Figure 1.2 – The rightmost graph is an induced subgraph of the leftmost one.

We say that $H$ is a *subgraph* of $G$ if $H$ can be obtained from $G$ by a sequence of removals of vertices and/or edges. More formally, $H$ is a subgraph of $G$ if $V(H) \subseteq V(G)$, and $E(H) \subseteq$

$\{uv \in E(G) \mid u, v \in V(H)\}$. It follows that any induced subgraph is a subgraph, but the converse might not be true.

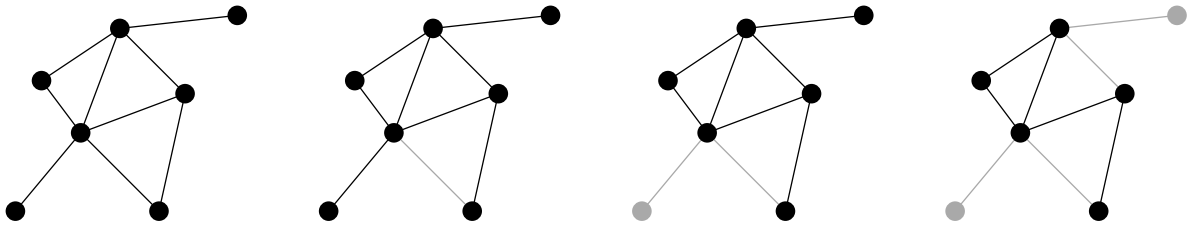


Figure 1.3 – Sequence of removals of vertices and edges: the rightmost graph is a subgraph of the leftmost one, but not an induced one.

If $S \subseteq E$, the subgraph induced by $S$ is the graph obtained from $G$ by removing each edge in $E \setminus S$. More formally, $G[S]$ is the graph with vertex set $V(G[S]) = V(G)$ and edge set $E(G[S]) = S$. In that case, we say that $G[S]$ is a *partial graph* of $G$.

An *edge contraction* is an operation that removes an edge $uv$ from $G$, and merge the two vertices $u$ and $v$ into a new one $z_{uv}$. The neighborhood of $z_{uv}$ is the union of the neighborhoods of $u$ and $v$. A graph $H$ is a *minor* of $G$ if it can be obtained from $G$ by a sequence of vertex and/or edge removals and/or edge contractions.

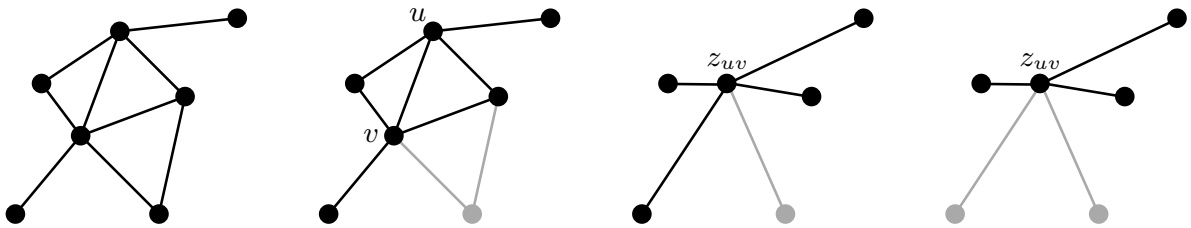



Figure 1.4 – The rightmost graph is a minor of the leftmost one. The third graph in this sequence is obtained from the second by contracting the edge $uv$.

**Clique and independent set.** An *independent set* (or *stable set*) of $G$ is a subset of vertices $S \subseteq V$ such that all the vertices in $S$ are pairwise non-adjacent. In other words, the graph $G[S]$ is $K_2$-free, i.e., it does not contain any edge. On the other hand, $S$ is a *clique* if all the vertices in $S$ are pairwise adjacent. In other words, $S$ is an independent set in the complement graph $\overline{G}$, that is the graph with vertex set $V(\overline{G}) = V(G)$, and for any two vertices $u \neq v$, $uv \in E(\overline{G})$ if and only if $uv \notin E(G)$.

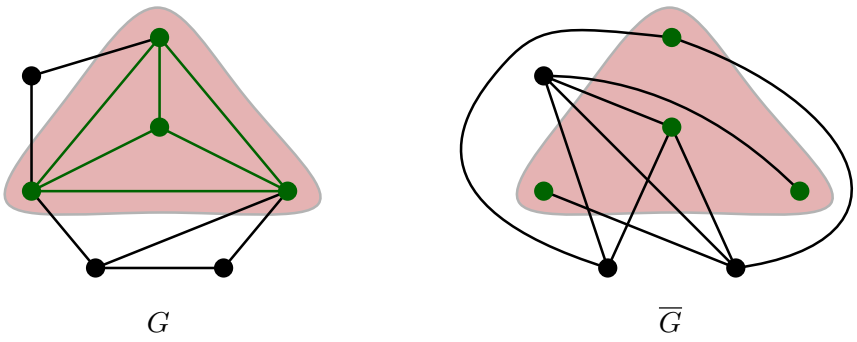



Figure 1.5 – Graph $G$ and its complement $\overline{G}$. The graph induced by the green vertices is a clique of size four in $G$, and an independent set in $\overline{G}$.

**Path, cycle, connectivity.** A *path* in $G$ between two vertices $u_0$ and $u_k$ is a sequence of pairwise distinct vertices $(u_0, u_1, \ldots, u_{k-1}, u_k)$ such that $u_i u_{i+1} \in E(G)$, for every $0 \leq i < k$. For the sake of simplicity, we denote by $u_0 u_1 \ldots u_k$ such a path. Note that if we do not require the vertices in the sequence $(u_0, u_1, \ldots, u_{k-1}, u_k)$ to be pairwise distinct, then the sequence is a *walk* between $u_0$ and $u_k$. The vertex $u_0$ (respectively $u_k$) is called the *first* (respectively *last*) vertex, and we say that $u_0$ and $u_k$ are the *extremities* of the path $u_0 u_1 \ldots u_k$.

The graph $G$ is *connected* if there exists a path between any pair of vertices of $G$. If $G$ is not connected, a *connected component* $C$ of $G$ is an inclusion-wise maximal subset of vertices $C \subseteq V$ such that $G[C]$ is connected.

A *cycle* is a sequence of pairwise distinct vertices $(u_0, u_1, \ldots, u_{k-1}, u_k)$ such that $u_0 u_1 \ldots u_k$ is a path, and $u_0 u_k \in E(G)$. Since we consider simple graphs, we must have $k \geq 2$, that is a cycle must contain at least three vertices. A cycle on three vertices (and thus with three edges) is sometimes called a *triangle*.

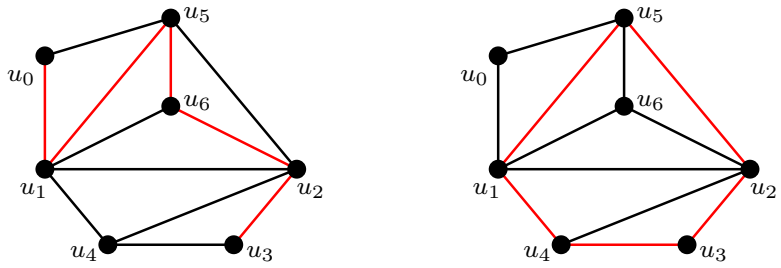



Figure 1.6 – Connected graph with a path $u_0 u_5 u_6 u_2 u_3$, and a cycle $u_1 u_4 u_3 u_2 u_5 u_1$.

**Distance, radius, diameter.** The *length* of a path is its number of edges. A *shortest path in* $G$ between two vertices $u$ and $v$ (if it exists) is a path with a minimum length among all the paths connecting $u$ to $v$. The *distance* between $u$ and $v$ is the length of a shortest path between $u$ and $v$ if it exists, it is infinity otherwise. We denote by $\text{dist}_G(u, v)$ (or $\text{dist}(u, v)$ when it is clear from the context) the distance between $u$ and $v$ in $G$.

Let $u \in V$ be a vertex of $G$. The *eccentricity of* $u$ denoted by $\epsilon(u)$ is the greatest distance between $u$ and any other vertex of $G$, i.e., $\epsilon(u) = \max_{v \in V} \text{dist}_G(u, v)$. The *radius* $r$ of $G$ is the minimum eccentricity of any vertex of $G$, i.e., $r = \min_{u \in V} \epsilon(u)$. Hence, if the radius of $G$ is $r$, then there exists a vertex $u$ at distance at most $r$ from all the other vertices of $G$. Similarly, the *diameter* $D$ of $G$ is the maximum eccentricity of any vertex of $G$, i.e., $D = \max_{u \in V} \epsilon(u)$. Hence, if the diameter of $G$ is $D$, then the distance in $G$ between any pair of vertices is at most $D$.

The radius of the graph in Figure 1.6 is two, as there is no universal vertex. On the other hand, $u_0 u_1 u_2 u_3$ is a shortest path of length three between $u_0$ and $u_3$. One can observe that the distance between any pair of vertices is at most three, and so the diameter of the graph in Figure 1.6 is three.

## 1.2 Some graph classes

### 1.2.1 Simple graph classes

**Forests and trees.** The graph $G$ is a *forest* if it does not contain a cycle as a subgraph. If, in addition, $G$ is connected, then $G$ is a *tree*. Let $T = (V, E)$ be a tree on at least two vertices, and let $u \in V$. We often refer $u$ as a *node*. The node $u$ is a *leaf* if $\deg_T(u) = 1$, it is an *internal node* otherwise. Note that since $T$ is connected and has at least two nodes, $T$ does not contain a vertex of degree zero.

**Proposition 1.1** (Folklore). *Let $T$ be a tree on $n$ nodes. The following conditions are equivalent:*

*(i)* *$T$ is connected and does not contain any cycle;*

*(ii)* *$T$ is connected and $T$ has $n-1$ edges;*

*(iii)* *for any pair of vertices $u \neq v$, there exists exactly one path between $u$ and $v$;*

*(iv)* *adding one edge creates a cycle, and removing an edge disconnects $T$.*

Let $G$ be a connected graph on $n$ vertices which is not a tree. A *spanning tree of* $G$ is a connected subgraph $T$ of $G$ such that $V(T) = V(G)$ and $|E(T)| = n - 1$. Hence, $T$ is a tree.

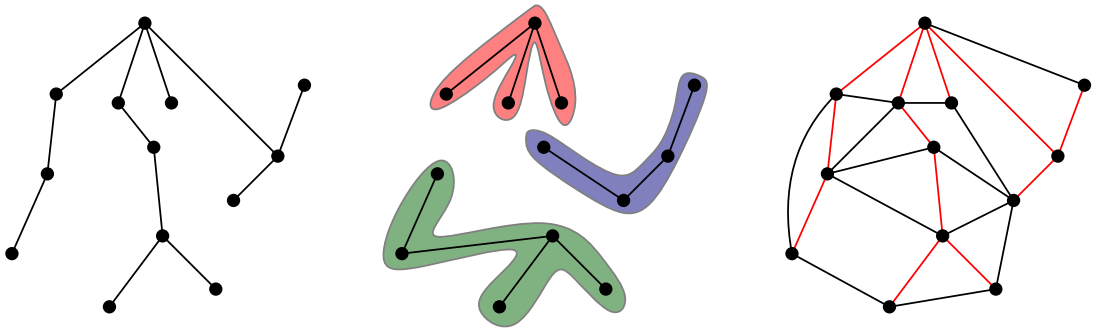



Figure 1.7 – A tree $T$ and a forest $F$ with three connected components, each of them being a tree. The red edges of the graph $G$ induce a spanning tree of $G$.

**Rooted tree, breadth-first search algorithm.** Let $T$ be a tree. One can choose a particular node $r$ of $T$ called the *root* and obtain a *tree rooted in* $r$, denoted by $T_r$. Then, the edges of $T_r$ can be oriented in two natural ways: either away from $r$, or towards $r$. In the first case, the resulting tree is called an *out-tree*; in the latter case it is called an *in-tree*.

Let $G = (V, E)$ be a graph, and let $u \in V$. A *breadth-first search* algorithm can be used to traverse $G$, starting from $u$ and exploring all vertices of $G$ by increasing distance from $u$. In particular, the first nodes that are explored (except $u$) are the neighbors of $u$, and then the neighbors of the neighbors of $u$ (i.e., the nodes at distance two from $u$), and so on.

**Algorithm 1** Breadth-first search algorithm (BFS)

**Require:** A graph $G = (V, E)$, a vertex $u \in V$.
**Ensure:** A set of oriented edges $S$

```
Label all vertices UNDISCOVERED
S = ∅
Let Q be an empty queue (i.e., a FIFO data structure)
Add u to Q
Label u with DISCOVERED
while Q is not empty do
    Let v be the top vertex of Q
    for every vertex x ∈ N(v) do
        if x is UNDISCOVERED then
            Add x to Q
            Label x with DISCOVERED
            Orient the edge e = vx ∈ E from v to x and add it to S
return S
```

Note that if $G$ is connected, all the nodes are explored. Thus, the complexity of this algorithm is $O(|V| + |E|)$ since each vertex and each edge of $G$ is visited at most once.

Suppose that $G$ is connected. Let $u \in V$, and let $S$ be the set of edges returned by the BFS algorithm performed from $u$. The set $S$ is of size $|V(G)| - 1$, and induces an in-tree $T_u$ rooted in $u$. If we omit the orientation of the edges in $S$, then $T_u$ is a spanning tree of $G$. Moreover, each path in $T_u$ from $u$ to a vertex $v$ corresponds to a shortest path of $G$ between $u$ and $v$, assuming that $G$ is an unweighted graph.

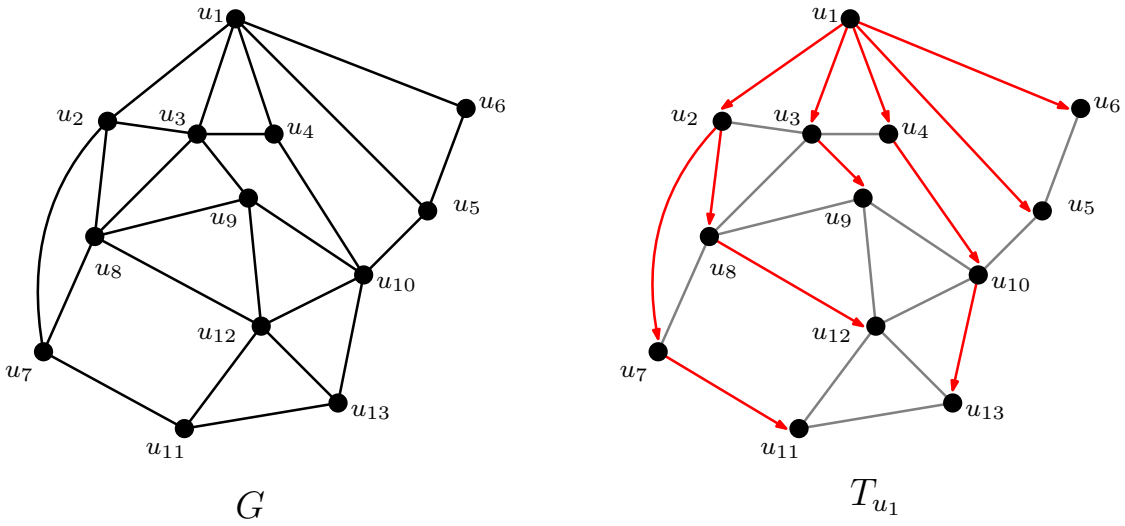



Figure 1.8 – The set of edges $S$ obtained by running a BFS from $u_1$ is depicted by the red edges. The vertices are visited by lexicographic order: if $u_i$ and $u_j$ are two vertices adjacent to $u_k$ with $i < j$, the vertex $u_i$ is visited before $u_j$. $G[S]$ is a spanning tree of $G$.

**$k$-regular graphs.** Let $k$ be a non-negative integer. The graph $G$ is $k$-regular if all the vertices of $G$ have degree exactly $k$. A well-studied class of $k$-regular is the class of 3-regular graphs. Such a graph is also called a *cubic* graph. A graph with maximum degree three (but that is not necessarily 3-regular) is called a *subcubic* graphs.

Suppose that $G$ is connected and has $n$ vertices. If $G$ is a $(n-1)$-regular graph, then $G$ is called the *complete graph on $n$ vertices*, and it is denoted by $K_n$. Observe that it corresponds to a clique on $n$ vertices. If $G$ is a 2-regular graph on $n$ vertices, then $G$ is the *cycle on $n$ vertices*, and it is denoted by $C_n$. Finally, if all the vertices of $G$ but exactly two have degree two, then $G$ is the *path on $n$ vertices*, and it is denoted by $P_n$.

**Bipartite and split graphs.** A graph $G$ is *bipartite* if its vertex set can be partitioned into two disjoint subsets $U$ and $V$ such that $G[U]$ and $G[V]$ are two independent sets. In other words, all the edges of $G$ are between a vertex in $U$, and a vertex in $V$. Since $G[U]$ and $G[V]$ are two independent sets, $G$ has no cycle of odd length. Actually, the class of bipartite is precisely the class of $\{C_k \mid \text{for any odd } k \geq 3\}$-free graphs.

The class of bipartite graphs contains all the trees (and forests by extension). Indeed, let $T$ be a tree and root $T$ on an arbitrary node $r$. Then, we divide the vertex set of $T$ into two parts: $U$ is the set of nodes at even distance from $r$, and $V$ the set of nodes at odd distance. We then observe that both $U$ and $V$ induce two independent sets.

Let $G = (U \cup V, E)$ be a bipartite graph, with $|U| = n$ and $|V| = m$. If $U$ is complete to $V$ (i.e., if for every vertex $u \in U$ and every vertex $v \in V$, $uv \in E$), then $G$ is the *complete bipartite graph* and it is denoted by $K_{n,m}$.

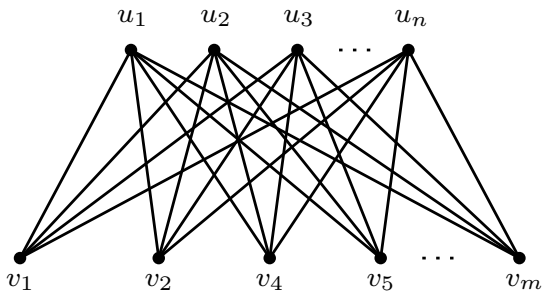



Figure 1.9 – Complete bipartite graph $K_{n,m}$.

### 1.2.2 Chordal graphs and related subclasses

In this subsection, we focus on a well-studied graph class, namely the one of chordal graphs. We will then discuss about some subclasses of chordal graphs that interest us in this thesis.

Let $G = (V, E)$ be a graph. $G$ is a *chordal* graph if every cycle of $G$ of length at least four has a *chord*, i.e., an edge whose endpoints are two non-consecutive vertices of the cycle. In other words, there is no induced cycle of length at least four. There exist several characterizations of chordal graphs. One of them is based on the existence of a special ordering of the vertices of $G$, namely a perfect elimination ordering. Let $v \in V$ be a vertex. We say that $v$ is a simplicial vertex if the neighbors of $v$ are pairwise adjacent, that is $G[N(v)]$ is a clique, and thus so is $G[N[v]]$. A *perfect elimination ordering* (or *peo* for short) of $G$ is an ordering $v_1, v_2, \ldots, v_n$ on the vertices such that the vertex $v_i$ is a simplicial vertex in the graph $G[\{v_i, v_{i+1}, \ldots, v_n\}]$, for every $1 \leq i \leq n$.

**Theorem 1.2** ([FG65]). *$G$ is chordal if and only if it has a peo.*

A chordal graph can be recognized in linear time by using a refinement of the breadth-first search algorithm, namely the *Lexicographic BFS* (or LexBFS for short) algorithm [RT75]. This algorithm can be used to compute a perfect elimination ordering of a chordal graph.

Another characterization of chordal graphs is closely related to the notion of tree decomposition that we will see in Section 1.4.2. Let $G = (V, E)$ be a chordal graph, and let $\mathcal{K}$ be the set of maximal cliques of $G$. Given a vertex $u \in V$, we denote by $\mathcal{K}_u$ the set of maximal cliques of $G$ containing $u$. A *clique tree* of $G$ is a tree $T$ whose vertex set is $\mathcal{K}$ and such that $T[\mathcal{K}_u]$ is a connected subtree of $T$, for any $u \in V$ (see Figure 1.10). Then, we have the following characterization of chordal graphs due to Gavril:

**Theorem 1.3** ([Gav74]). *$G$ is a chordal graph if and only if it has a clique tree.*

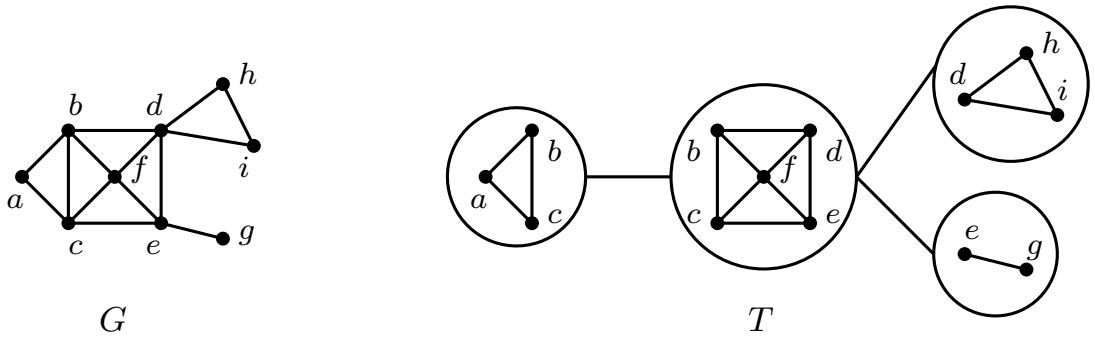



Figure 1.10 – Chordal graph $G$ and a clique tree $T$ of $G$.

**Split graphs.** A first subclass of chordal graphs that we consider in this thesis is the one of split graphs. A graph $G$ is a *split* graph if its vertex set can be partitioned into two disjoint subsets of vertices $K$ and $I$ such that $G[K]$ is a clique, and $G[I]$ is an independent set. Note that the partition is not unique: for instance in Figure 1.11, $K \cup \{v\}$ and $I \setminus \{v\}$ is also a partition of $V(G)$ into a clique and an independent set.

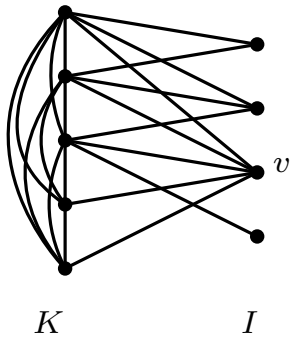



Figure 1.11 – Split graph $G$.

It is not difficult to see that any split graph is a chordal graph. Indeed, any vertex $u \in I$ satisfies $N(u) \subseteq K$, and thus $N(u)$ induces a clique. Hence, we first remove one by one each vertex in $I$ in an arbitrary order. Thus, the resulting graph is a clique. Since any induced subgraph of a clique is also a clique, it is clear that any vertex is simplicial. It follows that this elimination ordering is a perfect elimination ordering, and thus that $G$ is chordal by Theorem 1.2. Note moreover that the complement of a split graph is also a split graph. Hence, the complement of a split graph is also chordal. Actually, Foldes and Hammer proved that these are the only chordal graphs whose complement is also chordal, i.e., $G$ is a split graph if and only if it is chordal and its complement $\overline{G}$ is chordal [FH77]. They also proved that $G$ is a split graph if and only if $G$ is $(C_4, C_5, 2K_2)$-free. Note that excluding $2K_2$ as an induced subgraph is enough to avoid any induced cycle of length at least six.

Finally, the class of split graphs is of special interest as "almost all chordal graphs are split graphs". More formally, Bender, Richmond and Wormald proved that the fraction of chordal graphs on $n$ vertices which are also split graphs tends to one, as $n$ tends to infinity [BRW85].

**Interval graphs.** We end this subsection with the class of interval graphs. An interval graph is the intersection graph of a family of intervals on the real line. In other words, let $\{I_1, I_2, \ldots, I_n\}$ be a set of intervals on the real line. Each interval $I_i$ can be represented by its extremities with $\ell(I_i) \leq r(I_i) \in \mathbb{R}$. We call these values respectively $\ell$-value and $r$-value (for left and right). The corresponding interval graph $G = (V, E)$ is the following:

- $V = \{I_1, I_2, \ldots, I_n\}$;
- $I_iI_j \in E \Leftrightarrow I_i \cap I_j \neq \emptyset$ i.e., $\ell(I_j) \leq r(I_i)$ and $\ell(I_i) \leq r(I_j)$.

We now order the vertices of $G$ with respect to their $r$-value, i.e., $v_i < v_j$ if and only if $r(I_i) < r(I_j)$ (or $r(I_i) = r(I_j)$ and $\ell(I_i) < \ell(I_j)$). Then, we get the following useful property:

**Observation 1.4.** *Let $v_i$ and $v_j$ be two vertices of $G$ such that $v_i < v_j$. If $v_iv_j \in E$, then for any $v_k$ such that $v_i < v_k < v_j$, we have $v_kv_j \in E$.*

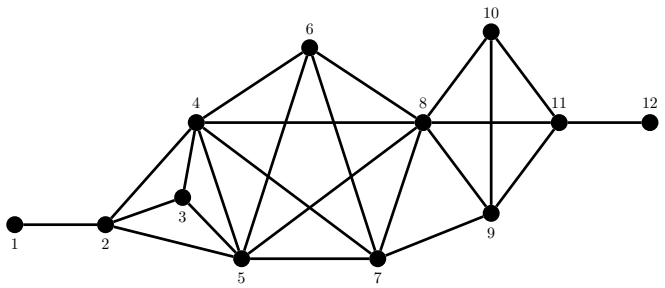
*Proof.* Since $v_i < v_k < v_j$, we have $r(I_i) \leq r(I_k) \leq r(I_j)$. Since $v_iv_j$ is an edge, $\ell(I_j) \leq r(I_i)$. Thus, we get that $\ell(I_j) \leq r(I_k)$. Adding that $\ell(I_k) \leq r(I_k) \leq r(I_j)$, the conclusion follows. □

It immediately follows from Observation 1.4 that any interval graph is chordal. Indeed, let us consider a cycle $C = v_i v_j v_k, \dots v_\ell v_i$ of length at least four, such that $v_i < v_j < v_k < v_\ell$. Since $v_i v_\ell \in E$, $v_\ell$ is adjacent to all the vertices of $C$. Hence, $C$ is not an induced cycle.

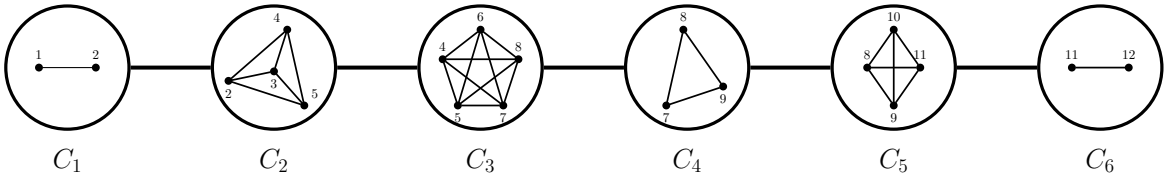
Actually, interval graphs are the chordal graphs whose complement graph is a comparability graph, that is a graph which is transitively orientable [GH64]. Let $G = (V, E)$ be a graph, and let $uv \in E$ be an edge. We denote by $\overrightarrow{uv}$ the orientation of the edge $uv$ from $u$ to $v$. Thus, $G$ is a comparability graph if its edges can be oriented in such a way that if we have $\overrightarrow{ab}$ and $\overrightarrow{bc}$, then we must have $\overrightarrow{ac}$.

Another characterization of interval graphs is related to the notion of clique tree seen previously. More precisely, $G$ is an interval graph if and only if its maximal cliques can be linearly ordered such that, for every vertex $u$ of $G$, the maximal cliques containing $u$ occur consecutively [GH64]. This characterization can be reformulated as follows: $G$ is an interval graph if and only if there is a path $\mathcal{P}$ whose vertex set corresponds to the maximal cliques of $G$ and such that the subgraph of $\mathcal{P}$ induced by the cliques containing any particular vertex is connected. In other words, the clique tree of an interval graph is actually a *clique path*.

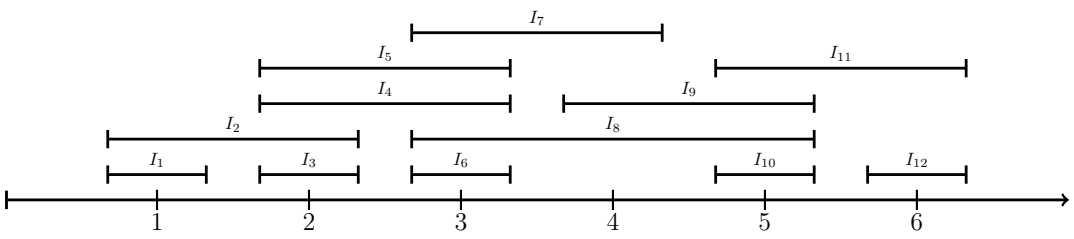
This last characterization is of special interest since one can compute an interval representation of an interval graph from its clique path. Indeed, suppose that a vertex $v$ is contained in the maximal cliques $C_i, C_{i+1}, \dots, C_j$. We define the corresponding interval $I_v$ with $r(I_v) = i - \epsilon$ and $\ell(I_v) = j + \epsilon$, for some $\epsilon > 0$. By choosing different epsilon values for each interval, we can moreover assume that all the intervals start and end at distinct points of the line. Habib et al. [HMPV00] gave a linear-time algorithm for interval graphs recognition. More precisely, they provided an algorithm that, given a graph $G = (V, E)$, returns a clique path if $G$ is an interval graph. This algorithms runs in time $O(|V| + |E|)$.




(a) An interval graph $G$.




(b) Clique path of $G$.




(c) A possible interval representation of $G$.

Figure 1.12 – Interval graph and an interval representation obtained from the clique path of $G$.

### 1.2.3 Planar graphs and cographs

We end this section with the definition of two graphs classes that we consider in this thesis in various chapters, namely the classes of planar graphs and the class of cographs.

**Planar graphs.** The class of planar graphs is probably one of the most famous, and thus one of the most studied class in graph theory. A *planar drawing* of a graph $G$ is a drawing of $G$ in the plane such that no two edges cross each other. A *planar graph* is a graph that admits a planar drawing. Let $G = (V, E)$ be a graph. A *subdivision of the edge* $uv \in E$ is an operation that consists in removing the edge $uv$ from $G$, and adding a new degree-two vertex $w$ adjacent to both $u$ and $v$. A *subdivision of* $G$ is a graph $G'$ that can obtained from $G$ by edge subdivisions. Then, we have the following characterization of planar graphs due to Kuratowski:

**Theorem 1.5** ([Kur30]). *A finite graph is planar if and only if it has no subgraph isomorphic to a subdivision of $K_5$ or $K_{3,3}$.*

Given a planar drawing of a planar graph $G$, the edges of $G$ partition the plane into several regions, called *faces*. The only unbounded face is called the *outer face*. The *size* of a face is the number of edges on its boundary. Assume now that $G$ is connected and let us consider a planar drawing of $G$. We denote by $f$ the number of faces of $G$. There is a relation between the number of edges, vertices and faces of $G$, as pointed out by the following formula:

**Theorem 1.6** (Euler's formula). *Any planar graph $G$ satisfies the formula $|V| - |E| + f = 2$.*

*Proof.* By induction on the number of edges of $G$. Let $n = |V|, m = |E|$, and let $T$ be a spanning tree of $G$. Observe that $T$ is drawn in a planar way since it has been obtained from $G$ by edge removals. Then, $T$ has $n$ vertices, $n - 1$ edges and only one face (the outer face) since it has no cycle. Hence, $n - (n - 1) + 1 = 2$. Now, suppose that the statement holds for any connected partial graph $G'$ of $G$ on $k$ edges, with $n - 1 < k < m$. By assumption, $G'$ satisfies Euler's formula. Now, observe that the addition of a single edge creates exactly one new face since it partitions a face into two new faces. Hence, any partial subgraph of $G$ with $k + 1$ edges also satisfies Euler's formula. □

Any face of a planar graph has size at least three, and an edge belongs to at most two faces. Hence, $2|E| \geq 3|F|$. Euler's formula implies that $|E| \leq 3|V| - 6$. Hence, planar graphs are *sparse* graphs in the sense that they have $O(|V|)$ edges. The graph in the Figure 1.13 below satisfies the formula since it has fifteen vertices, twenty edges and seven faces.

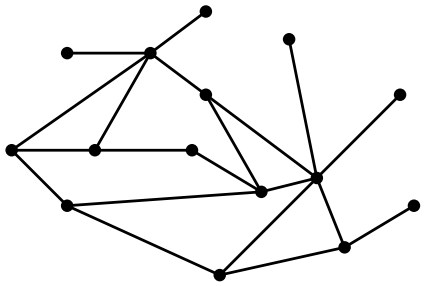


Figure 1.13 – A planar graph with a planar drawing.

**Cographs.** We end this subsection with the definition of cographs. Let us first define the two following operations on graphs. Let $G$ and $H$ be two graphs. The *disjoint union* $G \cup H$ of $G$ and $H$

yields a graph with vertex set $V(G) \cup V(H)$, and edge set $E(G) \cup E(H)$. Informally, the resulting graph is obtained by making two copies of $G$ and $H$. The second operation is the *join* $G + H$ of $G$ and $H$. Informally, the resulting graph is obtained by (i) making the disjoint union $G \cup H$, and (ii) adding all the possible edges between $V(G)$ and $V(H)$. More formally, $G + H$ is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H) \cup \{uv \mid u \in V(G),\ v \in V(H)\}$.

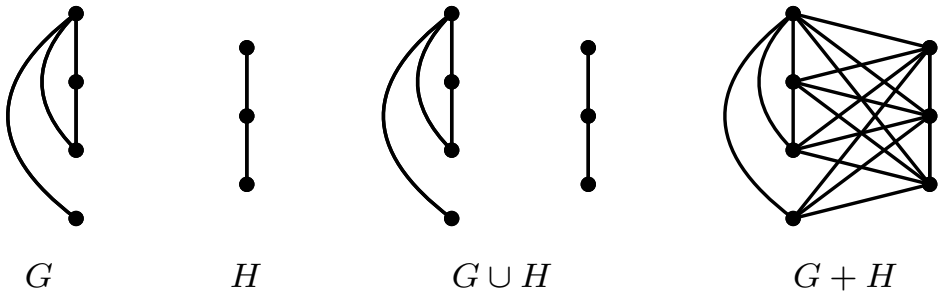



Figure 1.14 – Graphs $G$ and $H$, and their disjoint union $G \cup H$ and join $G + H$.

We are now ready to define the family of *cographs* which corresponds to the family of $P_4$-free graphs, or equivalently graphs that can be obtained with the following recursive construction:

- $K_1$ is a cograph;
- for $G$ and $H$ any two cographs, the disjoint union $G \cup H$ is a cograph;
- for $G$ and $H$ any two cographs, the join $G + H$ is a cograph.

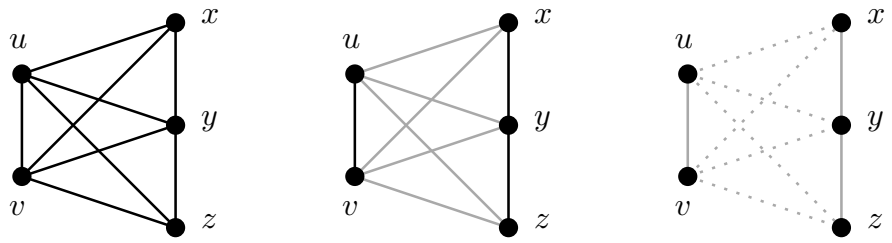



Figure 1.15 – Cograph $G$ obtained by the join of two cographs: $G[\{u, v\}]$ is the join of two $K_1$ and $G[\{x, y, z\}]$ is obtained by the join of $G[\{y\}]$ and $G[\{x, z\}]$.

There exist several characterizations of cographs [CLB81]. Recall that for instance, $G$ is a cograph if and only if $G$ is $P_4$-free. Hence, the diameter of any connected component of a cograph is at most two. Finally, note that cographs can be recognized in linear time, see e.g., [HP05].

## 1.3 Computational complexity

In this section, we give some definitions on computational complexity. For further information on this subject, we refer the reader to the following books [Pap94, CFK+15].

### 1.3.1 Decision problems and Turing machines

David Hilbert and Wilhelm Ackermann asked in 1928 whether there exists a "mechanical process" (i.e., an algorithm) that considers, as input, a mathematical statement and answers "yes" or "no" according to whether the statement is universally valid, i.e., valid in every structure

satisfying the axioms. This kind of mathematical problem is called a *decision problem*. More formally, a *decision problem* $\Pi$ is a problem where we are given a set of input values on which we ask a question whose answer is either yes or no. For instance, the following problem is a decision problem:

**Graph isomorphism**

**Instance:** Two graphs $G$ and $H$.
**Question:** Are $G$ and $H$ isomorphic?

Alan Turing introduced in 1936 in his seminal paper "On Computable Numbers, With An Application To The Entscheidungsproblem" [Tur36] an abstract model (called a Turing machine) of what such a mechanical process could be, in an attempt to answer the aforementioned question of Hilbert and Ackermann.

Suppose that we are given a one-way infinite and one-dimensional tape, divided into cells. Each cell can contain a symbol. It is supplied with a "head" that can be positioned on a cell in order to scan the symbol written on that cell, or to write a symbol at this position. Then, a *Turing machine* is a 6-tuple $(Q, q_0, F, \Gamma, \Sigma, \delta)$ where:

- $Q$ is a finite, non-empty set of states;
- $q_0$ is the initial state;
- $F \subseteq Q$ is a subset of states called final (or accepting) states;
- $\Gamma$ is a finite, non-empty set of tape alphabet symbols, with $\square \in \Gamma$ the blank symbol;
- $\Sigma \subseteq \Gamma \setminus \{\square\}$ is the input alphabet;
- $\delta : (Q \setminus F) \times \Gamma \mapsto Q \times \Gamma \times \{\leftarrow, \rightarrow, -\}$ is the transition function.

The set of transitions represents the algorithm. Indeed, given the current state and the symbol scanned by the head, the transition function indicates:

- the symbol that has to be written on the tape by the head;
- the new state; and
- the head shift on the tape: "$\leftarrow$" (resp. "$\rightarrow$") means "move to the left (resp. right) cell", and "$-$" means "stay stationary".

Note that if $\delta$ is not defined on the current state and the current tape symbol, then the machine halts. The machine also halts if it reaches a final state. Note also that there exist several definitions (e.g., with more than one tape, an infinite two-way tape and so on), which are all equivalent.

A *computable function* is a function that can be computed by an algorithm that runs in finite time. Unfortunately, there exist functions which are not computable by Cantor's diagonal argument. What makes Turing machines of special interest is the Church-Turing thesis. It asserts that if a function is computable, then there exists a Turing machine that can compute this function as well.

A Turing machine is *deterministic* if given a non final state $q$ and a symbol $a$, there is at most one triple $(q', a', s)$ with $q' \in Q$, $a' \in \Gamma$, $s \in \{\leftarrow, \rightarrow, -\}$ such that $\delta(q, a) = (q', a', s)$. In other words, the next configuration of the Turing machine is completely determined by the

current one. In contrast, a *nondeterministic* Turing machine allows more than one triple, i.e., the next configuration is not necessarily fully determined by the current one. One might think that nondeterministic Turing machines are more powerful than the deterministic ones. In fact this is not true, i.e., each nondeterministic Turing machine can be simulated with a deterministic Turing machine. However, it is believed that the time complexity might be different.

### 1.3.2 Some complexity classes and completeness

We herein focus on computational problems, i.e., problems that can be solved by an algorithm in finite time. The goal of computational complexity is to classify problems according to the amount of resources needed to solve them. This amount of resources is usually measured with respect to the input size, and involved time and memory requirements.

**Classical complexity classes.** We first present the three different complexity classes that interest us in this thesis, namely P, NP and PSPACE.

**Definition 1.7** (P, NP, PSPACE)**.** *We denote by P the set of decision problems that can solved by a deterministic Turing machine in polynomial time with respect to the input size.*

*NP is the set of decision problems that can be solved by a nondeterministic Turing machine in polynomial time. Alternatively, NP is the set of decision problems that can be verified in polynomial time by a deterministic Turing machine. In other words, one can check in polynomial time using a deterministic Turing machine that a given solution is a feasible solution.*

*Finally, we denote by PSPACE the set of decision problems that can be solved by a deterministic Turing machine using a polynomial amount of space.*

Similarly to NP, one can define NPSPACE as the set of decision problems that can be solved by a nondeterministic Turing machine using a polynomial amount of space. However, Savitch's theorem states that PSPACE $=$ NPSACE, i.e., these two classes are equivalent [Sav70]. Moreover, these definitions imply the following inclusions:

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE}$$

The first inclusion is trivial because any deterministic Turing machine is also a nondeterministic Turing machine. For the second inclusion, a nondeterministic Turing machine that solves a problem in NP can only visit a polynomial space since it runs in polynomial time. Hence, $\mathrm{P} \subseteq \mathrm{NPSPACE}$, and thus $\mathrm{P} \subseteq \mathrm{PSPACE}$ by Savitch's theorem. However, the two following questions are still open:

**Question 1.8.** Do we have $\mathrm{NP} \subseteq \mathrm{P}$?

**Question 1.9.** Do we have $\mathrm{PSPACE} \subseteq \mathrm{NP}$?

**Polynomial-time reductions and completeness.** Given a decision problem $\Pi$ and an instance $\mathcal{I}$ of $\Pi$, we say that $\mathcal{I}$ is a yes-*instance* if $\mathcal{I}$ has a solution. Let $\Pi_1$ and $\Pi_2$ be two decision problems. A *polynomial-time reduction* from $\Pi_1$ to $\Pi_2$ is a function that transforms in polynomial time an instance $\mathcal{I}_1$ of $\Pi_1$ to an instance $\mathcal{I}_2$ of $\Pi_2$ such that $\mathcal{I}_1$ is a yes-instance if and only if $\mathcal{I}_2$ is a yes-instance. In that case, we say that $\Pi_1$ is *(polynomially) reducible* to $\Pi_2$.

Let $\mathcal{C}$ be a complexity class, and let $\Pi$ be a decision problem. If all the problems in $\mathcal{C}$ are reducible to $\Pi$, we say that $\Pi$ is $\mathcal{C}$-hard. Intuitively, this means that $\Pi$ is at least as hard to solve as any problem in $\mathcal{C}$. If moreover $\Pi \in \mathcal{C}$, then $\Pi$ is $\mathcal{C}$-complete.

It follows from the definition that if a problem $\Pi_1$ is polynomially reducible to another polynomial-time solvable problem $\Pi_2$, then $\Pi_1$ in polynomial-time solvable as well. This observation is highly related to the above open question: is P equal to NP? Indeed, if one can find a polynomial-time algorithm solving an NP-complete problem $\Pi$, then any NP-complete problem can be solved is polynomial-time as well since it is reducible to $\Pi$ by definition. On the other hand, if P $\neq$ NP, then there is no polynomial-time algorithm solving a problem in NP.

**Conjunctive normal form and SAT.** Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of boolean variables, that is variables which can be either True (1, or $\top$) or False (0, or $\bot$). A *boolean formula* is a logical formula built from variables and logical operators. There exist several logical operators, but we only present the three we need: the logical "and" called *conjunction* and denoted by $\wedge$, the logical "or" (*disjunction*) $\vee$, and finally the logical "not" (*negation*) $\neg$. Note that $\wedge$ and $\vee$ are both binary operators, while $\neg$ is an unary operator. Let $\phi$ be a boolean formula. A *literal* is either a variable ($x_i$), called *positive literal*, or the negation of a variable ($\neg x_i$), called *negative literal*. A *clause* is a disjunction of literals, e.g., $x_1 \vee \neg x_2 \vee x_3$. A formula $\phi$ is in *conjunctive normal form* (or CNF for short) if it is a conjunction of clauses, i.e., $\phi$ can be written $C_1 \wedge C_2 \wedge \ldots \wedge C_m$, where every $C_i$ is a clause. Any propositional formula can be transformed into a new formula in CNF that preserves satisfiability thanks to Tseitin transformation [Tse83]. The size of this new formula is linear in the size of the original one. However, the two formulas are not necessarily logically equivalent, as they may not have the same truth tables.

> If we want to build an equivalent formula, then it possible thanks to the double negation elimination, De Morgan's laws, and the distributive law. However the size of the new formula might be exponential in the size of the original one. For instance, transforming the following formula $(x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee \ldots \vee (x_n \wedge y_n)$ into CNF results in a new formula with $2^n$ clauses.

Let $\phi$ be a boolean formula with variable set $X$. A *boolean assignment* (or simply *assignment*) is a function $\phi : X \mapsto \{0, 1\}$, i.e., a function that sets each variable of $\phi$ to 1 (True) or 0 (False). The formula $F$ is *satisfiable* if there exists an assignment of its variables such that $\phi$ is True. We are now ready to define the SATISFIABILITY problem (or SAT for short):

SAT

**Instance:** A formula $\phi$ in conjunctive normal form.
**Question:** Is $\phi$ satisfiable?

> Actually, the SATISFIABILITY problem can take as input any boolean formula, and not necessarily a formula in CNF. However, we assume in this thesis that the input formula is in conjunctive normal form.

**Example 1.10.** Let $x_1, x_2, x_3$ and $x_4$ be four boolean variables, and let us consider the following formula:

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee \neg x_4)$$

The assignment $x_1 = \top$ , $x_2 = \top$, $x_3 = \top$ and $x_4 = \bot$ satisfies the formula $\phi$ and therefore $\phi$ is satisfiable.

The SAT problem plays a key role in computational complexity due to the following result proved independently by Stephen Cook in 1971 and Leonid Levin in 1973:

**Theorem 1.11** ([Coo71, Lev73]). *SAT is NP-complete.*

Let $\phi$ be a formula in conjunctive normal form, and let $k \geq 1$ be an integer. If we require each clause of $\phi$ to contain exactly $k$ literals, then we say that $\phi$ is in $k$-conjunctive normal form (or $k$-CNF). Similarly, one can define the $k$-SAT problem as follows:

*k*-SAT

**Instance:** A formula $\phi$ in $k$-conjunctive normal form.
**Question:** Is $\phi$ satisfiable?

The 3-SAT problem is one of Karp's 21 NP-complete problems [Kar72]. Note that by adding new clauses and literals, one can prove that $k$-SAT is NP-complete, for every $k \geq 3$. On the other hand, it is known that 1-SAT and 2-SAT are in P, and can even be solved in linear time [APT79], yielding a complete dichotomy for the complexity of $k$-SAT.

### 1.3.3 Parameterized complexity

The parameterized complexity has been introduced by Downey and Fellows in the 1990s (see, e.g., [DF99]). It is a branch of computational complexity whose goal is to classify the complexity of computational problems according to some parameter (or even parameters). Let us first define what is a fixed-parameter tractable problem.

**Definition 1.12** (Fixed-parameter tractable). *A decision problem $\Pi$ with parameter $k$ is fixed-parameter tractable (or FPT for short) if it can be solved in time $f(k) \cdot n^{O(1)}$, for some computable function $f$ and where $n$ is the size of the instance of $\Pi$.*

**Example 1.13.** Given a graph $G = (V, E)$, a subset of vertices $C \subseteq V$ is a *vertex cover* of $G$ if for any edge $uv \in E$, $\{u, v\} \cap C \neq \emptyset$. In other words, the graph $G[V \setminus C]$ is edgeless. We claim that deciding whether $G$ admits a vertex cover of size at most $k$ can be solved in time $O(2^k \cdot |G|)$. In other words, the problem is FPT with respect to the natural parameter $k$. Let us denote by $(G, k)$ the corresponding instance.

The key observation is that in any vertex cover of $G$, at least one endpoint of each edge must be in the vertex cover. And they are at most two possibilities to cover the edge. Let $uv \in E$. Then, we observe that $G$ has a vertex cover of size $k$ if and only if $G - u$ or $G - v$ has a vertex cover of size $k - 1$.

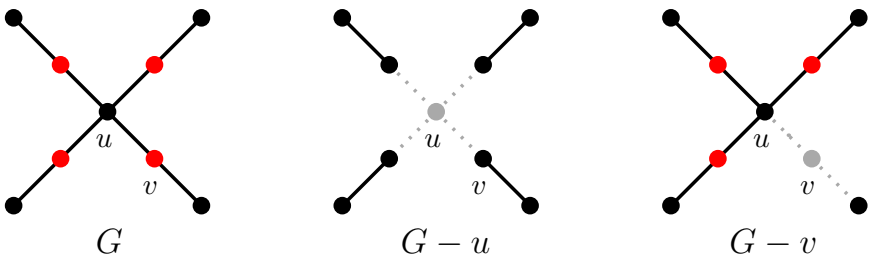



Figure 1.16 – $G$ has a vertex cover of size four depicted by the red vertices but $G - u$ has no vertex cover of size three. However, $G - v$ has a vertex cover of size three.

Following this idea, let us try to build a vertex cover $C$ of size at most $k$ as follows. First, we choose arbitrarily an edge $uv \in E$. Then, we recurse on $G - u$ and $G - v$ by creating two new instances: $(G - u, k - 1)$ and $(G - v, k - 1)$. There are two base cases: if the resulting graph is

edgeless, then $(G, k)$ is a yes-instance. If the graph has at least one edge but $k = 0$, then $(G, k)$ is a no-instance. In any other case, we recurse again. Note that this process ends since whenever we create a new instance, we decrease by one the size of the desired vertex cover and we remove at least one edge from $G$. Moreover, each instance creates at most two new instances. Hence, the number of recursive calls is at most $2^k$. Finally, it is clear that detecting whether $G$ is edgeless or not, or removing a vertex from $G$ can be done in time $O(|G|)$. It follows that the time complexity of this algorithm is $O(2^k \cdot |G|)$.

**Reduction rules and kernelization.** Let $\Pi$ be a parameterized problem, and let $(\mathcal{I}, k)$ be an instance of $\Pi$ with parameter $k$. In order to design an FPT algorithm with respect to $k$ for $\Pi$, one can try to reduce the size of the instance $\mathcal{I}$ so that $\mathcal{I}$ is "small enough" and can be solved by a brute-force algorithm for instance. This idea yields the concept of *kernelization*:

**Definition 1.14** (Kernelization). *A kernelization is an algorithm* A *that transforms the instance* $(\mathcal{I}, k)$ *of* $\Pi$ *into a new instance instance* $(\mathcal{I}', k')$ *such that:*

*(i)* A *runs in polynomial-time with respect to* $|\mathcal{I}|$*;*

*(ii)* $(\mathcal{I}, k)$ *is a* yes*-instance if and only if* $(\mathcal{I}', k')$ *is a* yes*-instance;*

*(iii)* $|\mathcal{I}'| \leq f(k)$ *for some computable function* $f$*; and*

*(iv)* $k' \leq g(k)$ *for some computable function* $g$*.*

The instance $(\mathcal{I}', k')$ obtained from the kernelization of $(\mathcal{I}, k)$ is called a *kernel*. Hence, "small enough" means that the size of $\mathcal{I}'$ only depends on $k$, the original parameter. It is clear from Definition 1.14 that if $(\mathcal{I}, k)$ has a kernel, then $\mathcal{I}$ can be solved in time FPT with respect to $k$. Moreover, it is known that if $\mathcal{I}$ can be solved in FPT time with respect to $k$, then $\mathcal{I}$ has a kernel:

**Proposition 1.15** (Folklore). *An instance* $(\mathcal{I}, k)$ *of* $\Pi$ *has a kernel if and only if* $\Pi$ *is FPT in* $k$*.*

In order to reduce the size of $\mathcal{I}$ and obtain a kernel, one needs to use some reductions rules. Let us illustrate this concept with an example.

**Example 1.16.** We again consider the VERTEX COVER problem presented in Example 1.13. Let $(G, k)$ be an instance. Our goal is to determine whether $G$ has a vertex cover of size at most $k$ or not. Let us consider the two followings rules:

*Rule 1*: if there is an isolated vertex $u$, we consider the instance $(G - u, k)$.

*Rule 2*: if there is a vertex $u$ such that $\deg(u) > k$, we consider the instance $(G - u, k - 1)$.

Note that applying a rule can be done in linear time in $|G|$. We apply iteratively as many times as possible the two aforementioned rules on the resulting graph. Let $(G', k')$ be the resulting instance, and note that it can be obtained in polynomial-time since whenever a rule is applied, exactly one vertex is removed. In order to prove that $(G', k')$ is a yes-instance if and only if $(G, k)$ is a yes-instance, one needs to prove that these two rules are "safe", meaning that applying them does not change the existence of a solution. Since an isolated vertex cannot cover any edge, *Rule 1* is safe. On the other hand, if a vertex $u$ satisfies $\deg(u) > k$, then $u$ must be in any vertex cover of $G$ of size at most $k$. Indeed, if $u$ is not in a vertex cover, all of its neighbors must be in the vertex cover. It follows that *Rule 2* is also safe. So any vertex $u$ of $G'$ satisfies $1 \leq \deg_{G'}(u) \leq k$. If $|E(G')| > k^2$, then $(G', k')$ is a no-instance since any vertex of $G'$ can cover at most $k$ edges (because it has degree at most $k$). Otherwise, we have $|V(G')| \leq 2k^2$. Indeed, each edge has two endpoints, and $G'$ has no isolated vertex. Hence, $(G, k)$ has a kernel of size $O(k^2)$.

**FPT reductions.** There is an analogous notion to the one of polynomial reduction, namely *FPT reduction* (or *parameterized reduction*).

**Definition 1.17** (FPT reduction)**.** *Let $\Pi_1$ and $\Pi_2$ be two parameterized problems, and let $(\mathcal{I}_1, k_1)$ be an instance of $\Pi_1$ with parameter $k_1$. An FPT reduction from $\Pi_1$ to $\Pi_2$ is a function $f$ that transforms $(\mathcal{I}_1, k_1)$ into an instance $(\mathcal{I}_2, k_2)$ of $\Pi_2$ such that:*

*(i) The running time of $f$ is $h(k_1) \cdot |\mathcal{I}_1|^{O(1)}$, for some computable function $h$;*

*(ii) $k_2 \leq g(k_1)$ for some computable function $g$; and*

*(iii) $(\mathcal{I}_1, k_1)$ is a* yes*-instance if and only if $(\mathcal{I}_2, k_2)$ is a* yes*-instance.*

Note that since we require the parameter $k_2$ to be bounded by a function of $k_1$, not every polynomial-time reduction is an FPT reduction. On the other hand, the running time of an FPT reduction can be more than polynomial, and thus not every FPT reduction is a polynomial-time reduction. Hence, FPT reductions and polynomial reductions are incomparable. However, they can be used to transfer fixed-parameter tractability results:

**Theorem 1.18** ([CFK+15])**.** *If there exists an FPT reduction from $\Pi_1$ to $\Pi_2$, and $\Pi_2$ is FPT, then $\Pi_1$ is FPT as well.*

**W-hierarchy.** Even if there exists an FPT reduction from a parameterized problem $\Pi_1$ to a parameterized problem $\Pi_2$, the converse might not be true. Hence, this suggests that there is a hierarchy of hard parameterized problems. For this reason, Downey and Fellows introduced the W-hierarchy in an attempt to capture more precisely the complexity of hard parameterized problems. In order to define the W-hierarchy, we first need to define boolean circuits. A *boolean circuit* is a directed acyclic graph (DAG) with four kinds of nodes:

- the nodes of indegree zero are the *input nodes*;
- the nodes of indegree one are the *negation nodes*;
- the nodes of indegree at least two are either *and-node* or *or-node*;
- the only node of outdegree zero is the *output node*.

Hence, any boolean formula $\phi$ in CNF on $n$ variables and $m$ clauses can be represented as a boolean circuit with $n$ input nodes, $m$ or-nodes, and one and-node which is the output node. Since any boolean formula can be transformed into an equivalent CNF formula, it is enough to restrict the nodes of a boolean circuit to and-nodes, or-nodes and negation nodes. The *depth* of a boolean circuit is the maximum length of a path from an input node to the output node.

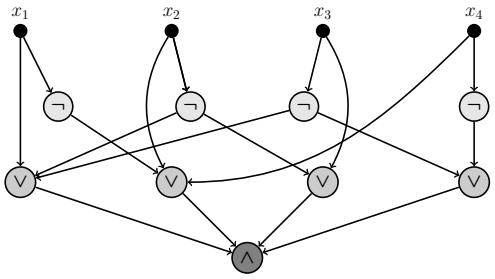



Figure 1.17 – Boolean circuit corresponding to the boolean formula $\phi$ of Example 1.10.

By assigning to each input node a weight in $\{0, 1\}$, one can obtain the weight of every node of the circuit in the natural way. In particular, one can obtain a weight for the output node. Given an assignment $\nu$ of the input nodes, we say that $\nu$ *satisfies* the boolean circuit if the output node has weight one. It is clear that deciding whether there exists an assignment satisfying a boolean circuit is NP-complete since any 3-SAT formula can be represented as a boolean circuit as discussed above. The *weight* of an assignment is the sum of the weights of the input nodes. One can now define the following problem:

WEIGHTED CIRCUIT SATISFIABILITY (WCS)

**Instance:** A boolean circuit $C$, an integer $k \geq 1$.
**Question:** Is there an assignment of $C$ of weight exactly $k$ satisfying $C$?

We need a last definition that allows us to distinguish the nodes. Given a boolean circuit, a *small node* is a node with indegree at most two, and a *large node* is a node which is not small. The *weft* of a boolean circuit is the maximum number of large nodes in a path from an input node to the output node. We denote by $\mathcal{C}_{t,d}$ the class of boolean circuits with weft at most $t$ and depth at most $d$. We are now ready to define the W-hierarchy:

**Definition 1.19.** *For any $t \geq 1$, a parameterized problem $\Pi$ is in W[t] if there exists an FPT reduction from $\Pi$ to* WEIGHTED CIRCUIT SATISFIABILITY *restricted to $\mathcal{C}_{t,d}$, for some $d \geq 1$.*

We have FPT=W[0] and $W[i] \subseteq W[j]$ for any $i \leq j$. Similarly as for NP-completeness or PSPACE-completeness, a problem $\Pi$ is W[$t$]-complete (for some $t \geq 1$) if the following two conditions are satisfied:

(i) there exists an FPT reduction from a W[t]-complete problem; and

(ii) $\Pi \in$ W[$t$], i.e., there exists an FPT reduction from $\Pi$ to WCS restricted to $\mathcal{C}_{t,d}$, for some $d$ by Definition 1.19.

Hence, one needs an analogous theorem to Cook-Levin's theorem. We say that a boolean formula is $t$-normalized if it is the conjunction of disjunctions of conjunctions of disjunctions, ... , alternating for $t$ levels. For instance, CNF formulas are 2-normalized formulas. Let us consider the more formal definition proposed in [CFK+15]. Let us first define both $\Delta_0$ and $\Gamma_0$ to be the set of boolean formulas consisting of only a single literal. Then, for $t \geq 1$, we define $\Delta_t$ (respectively $\Gamma_t$) to contain formulas that are the disjunctions of an arbitrary number of $\Gamma_{t-1}$ (resp. $\Delta_{t-1}$) formulas. Hence, $\Gamma_t$ is exactly the set of $t$-normalized formulas. Then, one can define the WEIGHTED $t$-NORMALIZED SATISFIABILITY problem as the weighted satisfiability problem corresponding to boolean circuits representing $t$-normalized formulas. We are now ready to introduce the following theorem:

**Theorem 1.20** ([DF99])**.** *For any $t \geq 1$, the* WEIGHTED $t$-NORMALIZED SATISFIABILITY *problem is W[t]-complete.*

As they mention in [DF99], Downey and Fellows believe that "WEIGHTED $t$-NORMALIZED SATISFIABILITY is strictly easier than WEIGHTED $(t+1)$-NORMALIZED SATISFIABILITY", meaning that $W[i] \subsetneq W[j]$ for every $1 \leq i < j$. Moreover, it is widely believed that FPT $\neq$ W[1]. In other words, any W[$t$]-hard problem (for $t \geq 1$) is believed not to be FPT. Indeed, if NP = P, then any problem in NP must be in FPT, hence W[1] = FPT. It follows that W[1] $\neq$ FPT would imply that P $\neq$ NP, providing a negative answer to Question 1.8. The converse direction (i.e.,

does W[1] = FPT imply P = NP?) is not known but Downey and Fellows [DF99] observed that FPT = W[1] would imply the failure of the *Exponential Time Hypothesis* (ETH) which states that SAT cannot be solved in subexponential time, roughly speaking.

## 1.4 Some graph problems

In this section, we define some graph problems that we mention throughout this thesis. Note that we mainly speak about the decision problem. Let $G = (V, E)$ be a simple graph, and let $k$ be a non-negative integer.

### 1.4.1 Independent set, vertex cover, $k$-coloring

**Independent set.** One of the most studied problems in graph theory due to its many connections with other graph problems is probably the INDEPENDENT SET problem. Recall that an independent set (or stable set) is a subset $I \subseteq V$ of pairwise non-adjacent vertices. Then, the INDEPENDENT SET problem is simply defined as follows:

INDEPENDENT SET

**Instance:** A graph $G = (V, E)$, a non-negative integer $k$.
**Question:** Does $G$ have an independent set of size *at least* $k$?

This problem is known to be NP-complete in the general case. Indeed, the CLIQUE problem which asks for a subset of pairwise adjacent vertices of size at least $k$ is one of the Karp's 21 NP-complete problems [Kar72]. Since a clique of size $k$ in $G$ is an independent set in $\overline{G}$, and that $\overline{G}$ can be computed from $G$ in polynomial-time, we obtain a polynomial-time reduction from the CLIQUE problem to the INDEPENDENT SET problem. Hence, this problem is NP-hard. Besides, one can also check in polynomial-time whether a given subset is an independent set, or not. Hence, the problem is NP-complete. Actually, these two problems are polynomially equivalent, because this reduction also works in the converse direction. Note that this is also an FPT reduction. Moreover, the INDEPENDENT SET problem is W[1]-complete when parameterized by $k$ [DF99].

A *maximal independent set* $I \subseteq V$ is an inclusion-wise maximal independent set, meaning that the addition of any vertex to $I$ would break the independence property. A maximal independent set can be computed in polynomial time by a greedy algorithm. A *maximum independent set* $I$ is a maximal independent set of largest size, meaning that $G$ does not have any independent set of size strictly larger than $|I|$. The *independence number* of $G$, denoted by $\alpha(G)$, is the size of a maximum independent set of $G$. The *clique number* of $G$, denoted by $\omega(G)$, is the size of a maximum clique of $G$. Hence, $\alpha(G) = \omega(\overline{G})$.

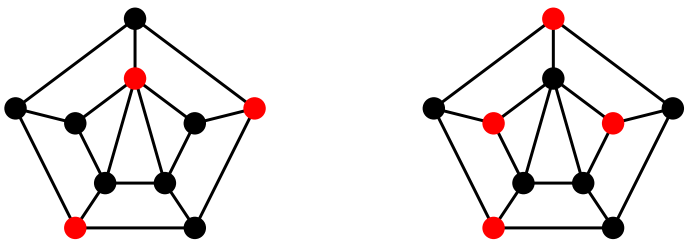


Figure 1.18 – A maximal independent set of size three and a maximum independent set of size four of the same graph.

**Vertex cover.** Recall that a vertex cover is a subset of vertices $C \subseteq V$ such $\{u, v\} \cap C \neq \emptyset$ holds for every edge $uv \in E$. Let $I \subseteq V$. Then, we have the following useful observation:

**Observation 1.21.** *$I$ is an independent set of $G$ if and only if $V \setminus I$ is a vertex cover of $G$.*

*Proof.* Suppose first that there is an edge $uv$ which is not covered by $V \setminus I$. This means that both $u$ and $v$ are in $I$, and thus $I$ is not an independent set, a contradiction. Suppose now that $V \setminus I$ is a vertex cover of $G$. Assume by contradiction that $I$ is not an independent set of $G$. Then, there exist two adjacent vertices $u, v \in I$. Thus, $u, v \notin V \setminus I$. It follows that the edge $uv \in E$ is not covered, a contradiction. □

The VERTEX COVER problem is defined similarly as the INDEPENDENT SET problem: it takes as input a graph $G$ and a non-negative integer $k$, and it asks whether $G$ admits a vertex cover of size *at most* $k$ or not. Let $(G, k)$ be an instance of INDEPENDENT SET. By Observation 1.21, $(G, k)$ is a yes-instance if and only if $(G, n - k)$ is a yes-instance for VERTEX COVER. Hence, VERTEX COVER is NP-hard. It is not difficult to see that VERTEX COVER is in NP, and thus is NP-complete. Unfortunately, this reduction is not an FPT reduction because $n - k$ is not bounded by a function which only depends on $k$.

$C$ is a *minimal vertex cover* if and only if the removal of any vertex in $C$ leaves at least one edge uncovered. A *minimum vertex cover* is the minimum size of a vertex cover; we denote by $\tau(G)$ the size of a minimum vertex cover of $G$. As an immediate corollary of Observation 1.21, we obtain that $\alpha(G) + \tau(G) = n(G)$ holds for any graph $G$.

> If we want a subset of vertices covering the other vertices of the graph and not its edges, then we are interested in a different problem, namely the DOMINATING SET problem. Given a graph $G = (V, E)$, a dominating set of $G$ is a subset $D \subseteq V$ of vertices such that $N[D] = V$. A more complete introduction of this problem is given in Section 2.1.

**$k$-coloring.** A $k$-coloring of $G$ is a function $f : V \mapsto \{1, 2, \ldots, k\}$ such that for any two adjacent vertices $u, v$ of $G$, we have $f(u) \neq f(v)$. Hence, the set of vertices with color $i$ induces an independent set of $G$, for every $1 \leq i \leq k$. Hence, one can think of a $k$-coloring as a partition of $V$ into at most $k$ independent sets, which are called *color classes*.

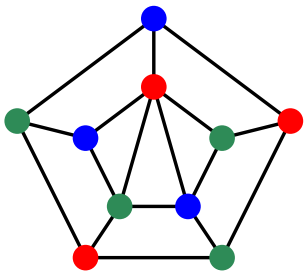


Figure 1.19 – A 3-coloring of a graph $G$. Note that $\chi(G) = 3$ since $\alpha(G) = 4$ and $|V(G)| = 10$.

We say that a graph is *$k$-colorable* if it admits a $k$-coloring. Then, the $k$-COLORING problem asks whether $G$ is $k$-colorable or not. This problem is polynomial for $k \in \{1, 2\}$. Indeed, a graph is 2-colorable if and only if it is bipartite. Hence, it is sufficient to determine if $G$ has an odd cycle to determine if it is 2-colorable. However, it is NP-complete for any $k \geq 3$ [Kar72]. This problem is one of the most studied in graph theory, and the Four Colors Theorem is probably one of the best known results in this field. It states that any planar graph is 4-colorable. It was proved by Appel and Haken in 1976 [AH77, AHK77] but it was a long outstanding question,

initiated by Francis Guthrie in 1852. The smallest integer $k$ such that a graph $G$ is $k$-colorable is called the chromatic number of $G$, and it is denoted by $\chi(G)$. A great deal of work has been done on both structural and algorithmic aspects of the chromatic number of a graph. However, it is beyond the scope of this thesis. We only mention the following inequality implied by the fact that a $\chi(G)$-coloring is a partition into $\chi(G)$ independent set, each of them being of size at most $\alpha(G)$: $|V(G)| \leq \chi(G) \times \alpha(G)$. In Section 4.2.4, we also consider a generalization of $k$-COLORING, namely the LIST COLORING problem.

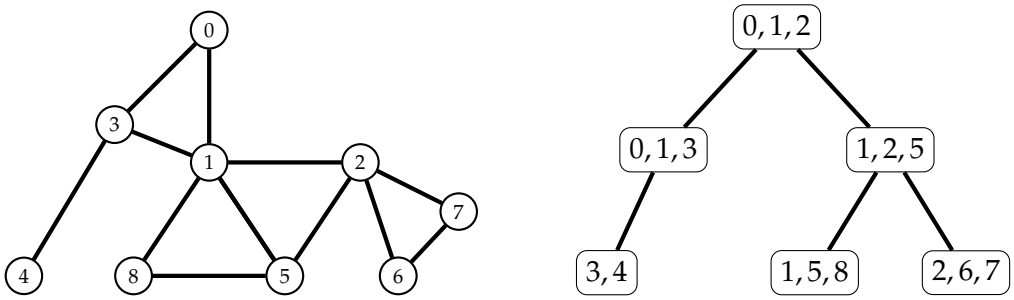
### 1.4.2 Treewidth, pathwidth and graph bandwidth

**Tree decomposition and treewidth.** The tree decomposition is a method of decomposition into subsets of vertices initially introduced by Bertelè and Brioschi in 1972 [BB72]. It was then rediscovered by Robertson and Seymour in its current form in 1984 [RS84]. It was used in their seminal work on the *Graph Minor Theory*. Let $G = (V, E)$ be a graph. A *tree decomposition* is an ordered pair $(X, T)$ where $X$ is a set of subsets of vertices of $G$ called *bags* and $T$ is a tree whose vertex set is $X$, and that satisfies the three following points:

(i) for any vertex $v \in V(G)$, $v$ belongs to at least one bag of $X$;

(ii) for any edge $uv \in E(G)$, there exists a bag that contains both $u$ and $v$; and

(iii) for any vertex $v \in V$, the set of bags containing $v$ forms a subtree of $T$.




(a) Graph $G$ (b) Tree decomposition of $G$

Figure 1.20 – Graph $G$ and a tree decomposition of $G$ of width two.

Let $G = (V, E)$ be a graph, and let $(X, T)$ be a tree decomposition of $G$. The *width* of $(X, T)$ is defined as the maximum size of a bag of $X$, minus one. For instance, the width of the tree decomposition in Figure 1.20b is two since the largest bag contains three elements. The *treewidth* of $G$, denoted by $tw(G)$ is the minimum width among all possible tree decompositions of $G$. Informally, the treewidth measures how tree-like a graph is. For instance, trees and forests are the only graphs of treewidth one, and the complete graphs on $n$ vertices has treewidth $n - 1$. The tree decomposition in Figure 1.20b is optimal, i.e., the treewidth of the graph $G$ in Figure 1.20a is two, as it contains triangles and is therefore not a tree.

Given a graph $G = (V, E)$ and an integer $k \geq 1$, the TREEWIDTH problem asks whether $tw(G) \leq k$ or not. This problem is NP-complete [ACP87]. However, it is FPT with respect to $k$. More precisely, Bodlaender [Bod96] showed the following theorem:

**Theorem 1.22** ([Bod96]). *Let $G = (V, E)$ be a graph on $n$ vertices, and let $k \geq 1$ be an integer. There exists an algorithm that determines whether $tw(G) \leq k$ and, if so, finds a tree decomposition of width at most $k$. This algorithm runs in time $O(2^{O(k^3)} \cdot n)$.*

The complexity of finding a tree decomposition of a given graph on $n$ vertices can be improved if one does not require this tree decomposition to be optimal. For instance, one can compute a tree decomposition of $G$ of width $O(tw(G) \cdot \sqrt{\log tw(G)})$ in time $O(n)$ [FHL05]. If we want to find a tree decomposition which is closer to an optimal one, then this is possible at the expense of a worse time complexity. For instance, one can compute a tree decomposition of width at most $4.5k$ in time $O(8^k \cdot k^{1.5} \cdot n^2)$ [Ami01], or of width at most $3k+2$ in time $O(81^k \cdot k^2 \cdot n \log n)$ [Ree92].

The tree decomposition of graphs is of special interest to solve combinatorial optimization problems on graphs. Indeed, the idea is to use a bottom-up traversal starting from the leaves of the tree decomposition to iteratively solve the problem on a set of bags, and then try to extend the partial solution to a superset of bags. Moreover, Courcelle showed the following metatheorem on graphs of small treewidth:

**Theorem 1.23** ([Cou90])**.** *Every graph property definable in the monadic second-order logic of graphs can be decided in linear time on graphs of bounded treewidth.*

The *monadic second order logic* (MSO) is a fragment of the second-order logic where we are allowed to quantify over sets of variables. In terms of formal language theory, the Büchi-Elgo-Trakhtenbrot theorem states that MSO is exactly the class of regular languages, i.e., languages that can be recognized by a deterministic finite automaton [Bü60, Elg61, Tra61].

**Example 1.24.** Let $G = (V, E)$ be a graph. The 3-colorability of $G$ can be defined by the following MSO formula:

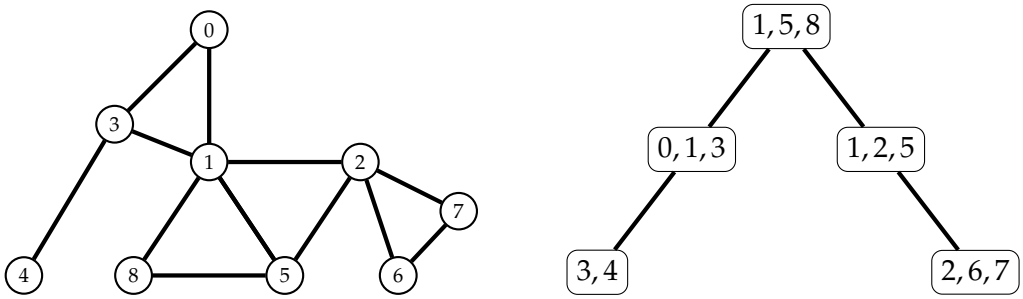
$$\exists X, Y, Z, (\forall u \in V,\ (u \in X \vee u \in Y \vee u \in Z) \wedge \tag{1.1}$$
$$X \cap Y = \emptyset \wedge X \cap Z = \emptyset \wedge Y \cap Z = \emptyset \wedge \tag{1.2}$$
$$\forall u, v \in V,\ (((u \in X \wedge v \in X) \vee (u \in Y \wedge v \in Y) \vee (u \in Z \wedge v \in Z)) \tag{1.3}$$
$$\Rightarrow uv \notin E))$$

Lines 1.1 and 1.2 state that there exist three pairwise disjoint sets $X, Y$ and $Z$ that cover all the vertices of $G$. In other words, $X, Y, Z$ is a partition of the vertex set of $G$. Line 1.3 states that any pair of vertices that belong to the same set are not adjacent in $G$, and thus that $X, Y, Z$ induce three independent sets of $G$.

By Courcelle's theorem, 3-colorability is linear-time solvable on bounded treewidth graphs. However, there is a constant which is a tower of exponentials whose height depends on the number of alternations of existential and universal quantifiers of the underlying MSO formula. Unfortunately, the height of this tower of exponentials is in general unbounded, making the algorithm inapplicable in practice.

**Path decomposition and pathwidth.** A *path decomposition* $(X, T)$ of a given graph $G$ is a special case of tree decomposition for which the underlying tree $T$ is actually a path. Similarly, one can define the width of a given path decomposition to be the maximum size of a bag of $X$ minus one. The *pathwidth* of $G$, denoted by $pw(G)$, is the minimum width among all possible path decompositions of $G$ (see Figure 1.21a).

As for the treewidth problem, it is NP-complete to compute the pathwidth of a given graph $G$, even if $G$ is a planar graph of bounded maximum degree [MS88] or is bipartite [KBMK93]. However, the problem is FPT when parameterized by $k$: there is a linear-time algorithm that tests whether $G$ has pathwidth at most $k$ and, if so, outputs a path decomposition of $G$ of




(a) Graph $G$ (b) Path decomposition of $G$

Figure 1.21 – Path decomposition of the graph $G$ seen in Figure 1.20a of width two.

width at most $k$ [Bod96]. However, the constant factor of this algorithm is very large, "much too large for practical purposes" as said by the author. If we restrict to interval graphs, then the problem can be solved in time $O(|V| + |E|)$. Moreover, the pathwidth corresponds to the size of a maximum clique of $G$ minus one. Indeed, a clique path of $G$ is a path decomposition of $G$ since any edge (and isolated vertex) is contained in at least one maximal clique. Since the cliques containing a vertex $u \in V$ occur consecutively in the clique path by definition, it is indeed a path decomposition of $G$. Following this observation, there exists an alternative definition of pathwidth, involving interval graphs. Let $G = (V, E)$ be a graph. An *interval completion of* $G$ is an interval graph $H = (V, F)$ with $E \subseteq F$. In other words, $H$ is an interval graph obtained from $G$ by iteratively adding edges. The pathwidth of $G$ is defined as $\min\{\omega(H) - 1 \mid H \text{ is an interval completion of } G\}$, or equivalently as the minimum pathwidth of an interval completion of $G$ among all possible interval completions.

Any graph $G$ satisfies $tw(G) \leq pw(G)$ since any path decomposition also is a tree decomposition. For interval graphs, these two parameters are equal. However, this is not true in general. For instance, for any forest $F$ on $n$ vertices, we have $tw(F) = 1$ but $pw(F) = O(\log n)$ [KS93], and this bound is tight. For instance, the complete binary tree of height $h$ has pathwidth $\lceil \frac{h}{2} \rceil$ (see, e.g., [MW15]).

**Graph bandwidth.** Let us consider a labeling $\ell : V \mapsto \{1, 2, \ldots, n\}$ of the vertices of $G$ (with $|V| = n$), where no two vertices receive the same label. The graph bandwidth problem aims to minimize the maximum absolute difference of the labels of two adjacent vertices. More formally, the GRAPH BANDWIDTH problem is defined as follows:

GRAPH BANDWIDTH

**Instance:** A graph $G = (V, E)$, a non-negative integer $k$.
**Question:** Does there exist a labeling $\ell : V \mapsto \{1, 2, \ldots, n\}$ such that $|\ell(u) - \ell(v)| \leq k$ for every edge $uv \in E$?

This problem is NP-complete [Pap76] and Bodlaender et al. [BFH94] showed that it is hard for W[$t$] for every $t \geq 1$, even when restricted to trees. This latter result therefore indicates that the problem is unlikely to be FPT, unless the W-hierarchy collapses. We denote by $bw(G)$ the bandwidth of $G$, that is the smallest integer $k$ such that $(G, k)$ is yes-instance. It is easy to see that the complete graph on $n$ vertices $K_n$ has bandwidth $n - 1$, while the path on $n$ vertices $P_n$ has bandwidth one. More generally, if $G$ has bandwidth $k$, then any vertex can have at most $k$ neighbors with a lower label, and at most $k$ neighbors with a larger label. Hence, if $bw(G) \leq k$, then the maximum degree $\Delta(G)$ is at most $2k$.

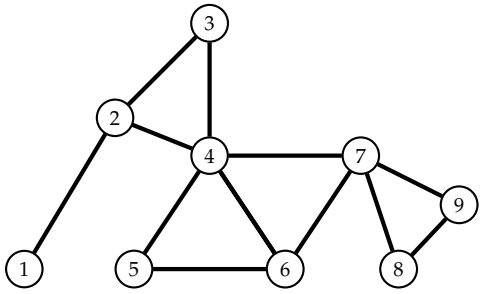



Figure 1.22 – Labeling of a graph $G$ of bandwidth at most three.

The graph $G$ in Figure 1.22 has bandwidth at most three, as there exists a labeling $\ell$ such that $|\ell(u) - \ell(v)| \leq 3$, for any pair of adjacent vertices $u, v$. Moreover, $bw(G) > 2$ since $\Delta(G) = 5$ (the vertex $u$ such that $\ell(u) = 4$ has degree five). Hence, we have $bw(G) = 3$. Moreover, there exists a relation between the treewidth, the pathwidth and the bandwidth of $G$, as pointed out by the following observation:

**Observation 1.25** ([BPTW10]). *Let $G = (V, E)$ be a graph. We have $tw(G) \leq pw(G) \leq bw(G)$.*

*Proof.* We have already explained the inequality $tw(G) \leq pw(G)$. It remains to prove that $pw(G) \leq bw(G)$. For the sake of simplicity, let us fix $k = bw(G)$, and $n = |V|$. Let $\ell : V \mapsto \{1, 2, \ldots, n\}$ be a labeling such that $|\ell(u) - \ell(v)| \leq k$ holds for any pair of adjacent vertices $u, v$. We define a path decomposition of $G$ of width at most $k$ as follows:

- the path decomposition has $n - k$ bags, let us say $X_1, X_2, \ldots, X_{n-k}$;
- $X_i = \{u \in V \mid i \leq \ell(u) \leq i + k\}$;
- two bags $X_i$ and $X_j$ are adjacent if and only if $|i - j| = 1$.

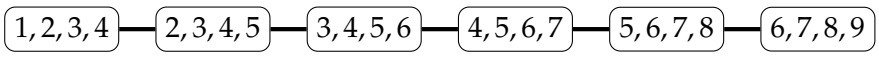



Figure 1.23 – This decomposition applied to the labeling of the graph of Figure 1.22.

It is clear that we have $|X_i| \leq k + 1$, for every $1 \leq i \leq n - k$. Hence, the width of this decomposition is at most $k$. And it is clear that this path decomposition induces a path. Moreover, we have the following properties:

- each vertex is contained in at least one bag. Indeed, let us consider the vertex $u$ of label $i$, i.e., the vertex such that $\ell(u) = i$. If $i \in [1, n - k]$, $u$ belongs to the bags $X_j$, for every $\max\{1, i - k\} \leq j \leq i$. If $i \in [n - k + 1, n]$, $u$ belongs to the bags $X_j$ for every $i - k \leq j \leq n - k$. Hence, one can observe that the bags containing $u$ occur consecutively, and thus induce a connected subgraph.
- for every edge $uv \in E$, there exists a bag which contains both $u$ and $v$. Let us assume without loss of generality that $\ell(u) < \ell(v)$, and recall that $\ell(v) - \ell(u) \leq k$. If $\ell(u) \leq n - k$, the bag $X_{\ell(u)}$ contains both $u$ and $v$ by definition. Otherwise, $u$ and $v$ both belong to $X_{n-k}$ since $X_{n-k}$ contains all the vertices labeled $n - k, n - k + 1, \ldots, n$.

If follows that this decomposition indeed is a path decomposition of $G$ of width $k$. Hence, $pw(G) \leq bw(G)$, and the conclusion follows. □

## 1.5 Combinatorial reconfiguration

This section is a general introduction on combinatorial reconfiguration. For a more complete overview of recent results on reconfiguration problems, the reader is referred to the surveys by Jan van den Heuvel [vdH13] and Naomi Nishimura [Nis18].

### 1.5.1 Illustration of the problem

Let us first illustrate what is a reconfiguration problem with the simple following example, taken from [HHS98]. Suppose that we have a group of people representing a community. These people wish to elect a committee of representatives as small as possible to lead their community. However, there are differences of opinion within the community on many points. Therefore, and in the interest of representativeness, it is important that each person from the community who has not been chosen to be part of the committee has (at least) one representative whose ideas are close to her or his own. How many people should be elected? And who? This question can be solved by modeling the situation as a graph theory problem: each person is a vertex, and two vertices are adjacent if the two corresponding persons share the same ideas. So asking the minimum number of people that we need in the committee is the same as the minimum size of a dominating set of the corresponding graph.

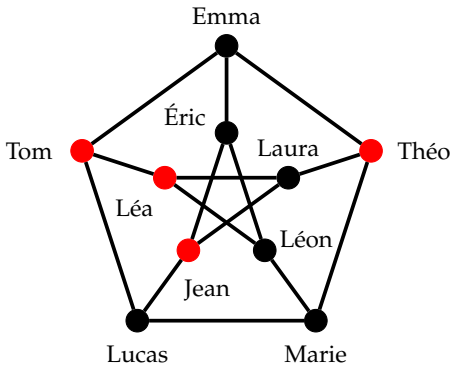



Figure 1.24 – Modeling of this problem: the dominating set is depicted by the red vertices.

In addition, it is important to regularly renew the members of the committee in order to advocate new ideas. However, we do not want to replace more than one member at a time so that change is not too much abrupt. Moreover, whenever a member of the committee is replaced by another, it is essential that the new committee also satisfies the requirement that all the people in the community are represented by someone. Thus, a legitimate question is to ask whether it is possible to renew all or part of a Committee $A$ into a Committee $B$. If so, how should we process?

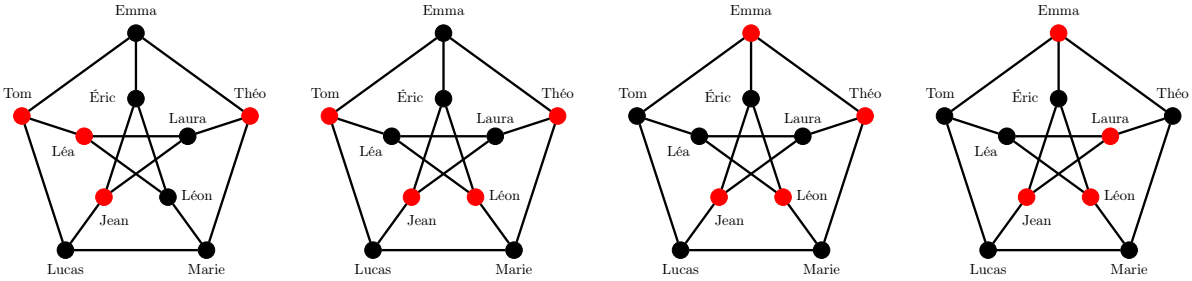



Figure 1.25 – Modification of the committee {Jean, Léa, Théo, Tom} into the committee {Emma, Léon, Jean, Laura}, where each intermediate committee is a valid one.

More generally, given an instance $\mathcal{I}$ of a combinatorial problem $\Pi$, is it possible to transform two feasible solutions of $\mathcal{I}$, let us say $S_1$ and $S_2$, while satisfying the desired property (e.g., being a dominating set of the graph) at any time during the transformation? We emphasize that $S_1$ and $S_2$ are not necessarily part of the instance $\mathcal{I}$. This is typically the kind of questions that are studied in the *reconfiguration framework*. Obviously, one needs a rule which allows us to transform a solution into another one. For instance, in the previous example, one can replace one member of the committee by another people. This corresponds to removing a vertex from the dominating set and adding another one in a single step. Before moving on to the formalization of the reconfiguration framework, let us consider another (slightly different) example of a reconfiguration problem.

Rush Hour™ is a one-player game that is played on a $n \times n$ grid. Some cars are placed on the cells of the grid, and they occupy two or three cells, depending on their size. We distinguish one special car (the *target car*) and a single exit is located on the perimeter of the grid, at a fixed position. The goal of this game is to get the target car out through the exit. For that purpose, one can move cars either horizontally or vertically, depending on their orientation in the initial configuration of the grid. However, this initial configuration often contains traffic jams, making it impossible to solve the problem in a single move of the target car. Hence, the player has to find a sequence of moves of the cars that makes all the cells located between the target car and the exit free.

Figure 1.26 – Rush Hour configuration: the target car is the police car. © Newgrounds Inc.

Figure 1.26 gives an example of a Rush Hour configuration on a $6 \times 6$ grid. Surprisingly, this game is more complex than it seems: it might take forty moves to exit the police car. Actually, Flake and Baum [FB02] showed that a generalization of this game where the grid has arbitrary height and width and where the exit can be located at any position on the perimeter of the grid is a PSPACE-complete problem.

### 1.5.2 Formalization

*Reconfiguration problems* model dynamic situations where we are given an instance $\mathcal{I}$ of a combinatorial search problem $\Pi$ and we seek to find a step-by-step transformation between two feasible solutions of $\mathcal{I}$ such that each intermediate solution satisfies the two following properties (i) it is also a feasible solution of $\mathcal{I}$; and (ii) it is obtained from the previous one by applying a specified (and unique) rule, called *reconfiguration rule*. The original problem $\Pi$ is called the *host* problem. The reconfiguration rule that allows us to modify a solution depends on the problem; we will discuss it in more detail in Section 1.5.5. Such transformation between two solutions $S_s$ and $S_t$ of $\mathcal{I}$ is called a *reconfiguration sequence*, and is denoted by $\langle S_0 = S_s, S_1, S_2, \ldots, S_\ell = S_t \rangle$. $S_s$ is called the *source* solution, and $S_t$ the *target* solution. Unfortunately, a reconfiguration sequence does not always exist and some solutions may even be *frozen*, meaning that they cannot be modified at all.

Reconfiguration problems can be expressed in terms of properties of the *reconfiguration graph* which is defined as follows:

**Definition 1.26** (Reconfiguration graph)**.** *The reconfiguration graph associated with the instance $\mathcal{I}$ of $\Pi$ is the graph $\mathcal{R}(\mathcal{I})$ defined as follows:*

- *the vertices of $\mathcal{R}(\mathcal{I})$ are the feasible solutions of $\mathcal{I}$; and*
- *two vertices $u, v$ of $\mathcal{R}(\mathcal{I})$ are adjacent if and only if one can transform the solution associated with vertex $u$ into the solution associated with $v$ by applying the specified reconfiguration rule.*

The reconfiguration framework can be applied to many combinatorial problems by defining an adjacency relation between two feasible solutions. Interest in the reconfiguration of graphs problems steadily increased during the last decade. Many works in this context deal with structural properties of the reconfiguration graph. One of the most studied questions is to find conditions that guarantee the reconfiguration graph to be connected. Conversely, finding necessary conditions for the connectivity of the reconfiguration graph has also been studied extensively. This can be done for instance by exhibiting infinite families of graphs for which some conditions must hold in order to guarantee this connectivity. One another well-studied property concerns the diameter of the reconfiguration graph, or of its connected components if it is not connected. In Chapter 3, we focus on the reconfiguration of dominating sets and in Section 3.1, we study the diameter of the reconfiguration graph $\mathcal{R}_k(G)$ for some specific values of $k$.

On the other hand, Ito et al. [IDH+08] initiated a systematic study of the computational complexity of reconfiguration problems. The three main problems that are mainly studied are the following:

- $\Pi$-Reachability: given an instance $\mathcal{I}$ of $\Pi$ and two solutions $S_s$ and $S_t$ of $\mathcal{I}$, does there exist a path in $\mathcal{R}(\mathcal{I})$ between $S_s$ and $S_t$?
- $\Pi$-Connectivity: given an instance $\mathcal{I}$ of $\Pi$, is the reconfiguration graph $\mathcal{R}(\mathcal{I})$ connected?
- $\Pi$-Shortest Path: given an instance $\mathcal{I}$ of $\Pi$, two solutions $S_s$ and $S_t$ of $\mathcal{I}$ and an integer $\ell \geq 1$, does there exist a path in $\mathcal{R}(\mathcal{I})$ between $S_s$ and $S_t$ of length at most $\ell$?

### 1.5.3 Complexity of reconfiguration problems

It turns out that many reconfiguration problems are in PSPACE, and are actually PSPACE-complete. Indeed, Ito et al. [IDH+08] observed that if a problem $\Pi$ is in NP, then its reconfiguration version is in NPSPACE. Let $\mathcal{I}$ be an instance of $\Pi$. Since $\Pi$ is in NP, we can enumerate all possible solutions of $\mathcal{I}$ with a polynomial amount of space. One can then check whether each solution indeed is a feasible solution of $\mathcal{I}$ or not. We then non deterministically traverse the solutions that are adjacent with the current solution (the adjacency can be checked in polynomial time by assumption). This nondeterministc algorithm can be converted into a deterministic one since NPSPACE = PSPACE by Savitch's theorem [Sav70].

One possible analogous of SAT to prove the PSPACE-hardness of (graph) reconfiguration problems is the *Nondeterministic Constraint Logic* (or NCL for short), introduced by Hearn and Demaine [HD05]. Indeed, a lot of PSPACE-hardness proofs in combinatorial reconfiguration are based on a polynomial-time reduction from NCL. In this problem, we are given a 3-regular

graph $G = (V, E)$, where a weight in $\{1, 2\}$ is assigned to each edge of $G$. One may represent the weights by colors: the edge $e \in E$ is colored red (respectively blue) if the weight of $e$ is 1 (resp. 2). Moreover, each vertex is incident to an even number of red edges. This graph is called a *constraint graph*. Since $G$ is 3-regular, one can distinguish two different types of vertices:

- the ones which are not incident to a red edge: they are called OR vertices; and
- the ones which are incident to exactly two red edges: they are called AND vertices.


(a) An OR vertex (b) An AND vertex

Figure 1.27 – The two types of vertices in an instance of NCL.

In NCL, a feasible solution is an orientation of the edges in such a way that for each node, the sum of the weights of its incoming edges is at least two. In particular, each OR vertex must have at least one incoming edge (and actually one is enough since it is only incident to edges of weight two) so it behaves like an OR gate in boolean logic. On the other hand, in an AND vertex, the only edge of weight two may be directed outwards only if the two edges of weight one are directed inwards so it behaves like an AND gate in a sense.

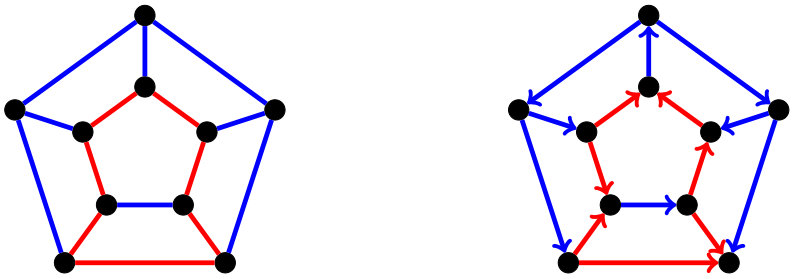


Figure 1.28 – A possible instance for the NCL problem and a feasible solution for this instance.

In the NCL-REACHABILITY, we are given two different orientations $\alpha$ and $\beta$ of the edges of a constraint graph and we want to find a step-by-step transformation from $\alpha$ to $\beta$ such that all intermediate solutions are also feasible. In this case, the reconfiguration rule that allows us to modify a solution into another is to reverse the orientation of a single edge. Hearn and Demaine [HD05] showed that this problem is PSPACE-complete, even if the input constraint graph is planar. They reduced from QUANTIFIED BOOLEAN FORMULA, the problem where each variable of the propositional formula is quantified with either an existential or a universal quantifier, which is PSPACE-complete [GJ79]. This result was later improved by van der Zanden [vdZ15] who showed that it remains PSPACE-complete even if the constraint graph is a 3-regular planar graph with bounded bandwidth.

As we said, if the host problem is NP-complete, then the reconfiguration version is often PSPACE-complete. However, this is not always the case. For instance, the 3-coloring problem which asks whether a given graph $G$ can be properly colored with at most three colors is NP-complete. However, finding a reconfiguration sequence between two given 3-colorings of $G$ is polynomial-time solvable [CVDHJ11]. However, the reachability variant of $k$-recoloring

becomes PSPACE-complete whenever $k$ is at least four [BC09]. On the other hand, a lot of reconfiguration problems whose host problem is in P are also in P. This includes MAXIMUM MATCHING or MINIMUM SPANNING TREE for instance [IDH+08]. In contrast, it is well-known that finding a shortest path between two given vertices $s$ and $t$ is polynomial-time solvable. For instance, if the input graph $G = (V, E)$ is undirected and the weights (on the edges) are positive real numbers, then it is solvable in time $O(|E| + |V| \cdot \log |V|)$ [FT87], or even in time $O(|E|)$ if the weights are positive integers and if we assume that multiplications can be done in constant time [Tho99]. However, the reconfiguration of $(s, t)$-shortest paths is PSPACE-complete in the general case, while it is in P if the input graph is $K_{1,3}$-free or $C_4$-free [Bon12]. This problem is not the first one to be PSPACE-complete even if the host problem is in P. Indeed, Bonsma et al. [BC09] showed in 2009 that $k$-recoloring is PSPACE-complete if $k = 4$ and the graph is a bipartite planar graph which is therefore 2-colorable.

### 1.5.4 Example of reconfiguration problems

In this section, we give a brief outline of problems that are studied in combinatorial reconfiguration. For a more complete overview, we refer the reader to the surveys of Jan van den Heuvel [vdH13] and Naomi Nishimura [Nis18], or to Amer Mouawad's PhD thesis [Mou15]. The reconfiguration framework can be applied to a lot of combinatorial problems, and not only to graph problems. For instance, we have already seen that the one-player game *Rush Hour* can be seen as a reconfiguration problem.

**Distance between triangulations.** Some problems that arise from discrete geometry for instance can be expressed as reconfiguration problems. Given a set $\mathcal{P}$ of $n$ points in the Euclidean plane, a *triangulation* of $\mathcal{P}$ is a maximal straight-line planar graph whose vertex set is $\mathcal{P}$. Given a triangulation of $\mathcal{P}$, a *flip* is an operation that replaces the diagonal of a convex quadrilateral by the other diagonal (the quadrilateral must be convex otherwise the flip is not allowed).


Figure 1.29 – Example of flip: the diagonal $e$ is replaced by the other diagonal $f$.

Given a set $\mathcal{P}$ of $n$ points of the Euclidean plane, one can define the flip graph $\mathcal{R}(\mathcal{P})$ where (i) each triangulation of $\mathcal{P}$ is a vertex; and (ii) there is an edge between two vertices (i.e., triangulations) if one can be obtained from the other by a flip. Aichholzer et al. [AHN04] showed that the lower bound on the number of vertices of $\mathcal{R}(\mathcal{P})$ is $\Omega(2.33^n)$ and Lawson [Law72] showed that it is connected and has diameter $O(n^2)$. Hurtado et al. [HNU99] showed that this bound is tight. Given two triangulations $T_s$ and $T_t$ of the same set $\mathcal{P}$ of $n$ points, we know that there always exists a flip sequence between $T_s$ and $T_t$ of length at most $O(n^2)$. The problem of determining the shortest reconfiguration sequence between two triangulations if called FLIP DISTANCE. Lubiw and Pathak [LP15] showed that finding a shortest sequence between two triangulations of a set of points in the Euclidean plane is NP-complete, and Pilz proved that it is APX-hard [Pil14]. However, it is known that it is fixed-parameter tractable with respect to the length of the solution [LFMW17]. More precisely, they provided an algorithm that determines whether the length of a reconfiguration sequence between two triangulations is at most $k$ or not in time $O^*(k \cdot 32^k)$; the $O^*$ notation means that we omit the polynomial factor on $n$, the number of points. Aichholzer et al. [AMP15] showed that FLIP DISTANCE is NP-complete

on triangulations of simple polygons. However, the complexity on triangulations of convex polygons is still open. Sleator et al. [STT86] showed that the shortest sequence between two triangulations of a convex polygon is at most $2n - 10$ for $n > 12$, and that it is precisely $2n - 10$ for all sufficiently large values of $n$. Sleator et al. [STT86] observed that there is a bijection between triangulations of convex polygons and binary trees, a data structure widely used in computer science. A rotation in a binary tree is an operation that changes the structure without changing the order of the elements. It is used for instance to balance the height of binary search trees and thus to speed-up queries on these data structures (see, e.g., [GMAV62]). More precisely, there is a one-to-one correspondence between a triangulation of an $n$-sided convex polygon and a binary tree on $n - 2$ nodes. And a flip between two triangulations is equivalent to a rotation in the two corresponding binary trees. Hence, the problem FLIP DISTANCE on convex polygons is equivalent to the rotation distance between two binary trees.

**Satisfiability reconfiguration.** Recall that a boolean formula $\phi$ is in CNF if it is a conjunction of disjunctions. The formula $\phi$ is satisfiable if there is an assignment $\nu$ of the variables of $\phi$ such that $\phi$ is true. However, this assignment may not be unique. Let us consider again the Example 1.10, with $\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \vee (\neg x_3 \vee \neg x_4)$. We have already seen that the assignment $\nu(x_1) = \top, \nu(x_2) = \top, \nu(x_3) = \top, \nu(x_4) = \bot$ satisfies $\phi$. However, $\nu(x_1) = \bot, \nu(x_2) = \bot, \nu(x_3) = \bot, \nu(x_4) = \bot$ also satisfies $\phi$. One can compute all the assignments satisfying $\phi$ by computing its truth table. If $\phi$ has $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses $C_1, C_2, \ldots, C_m$, then the number of assignments is bounded above by $2^n$. Moreover, one can check in time $O(m)$ whether a given assignment satisfies $\phi$ or not. So one can compute in time $O(2^n \cdot m)$ the set of feasible solutions of $\phi$. Let $\nu_s$ and $\nu_t$ be two assignments that both satisfy $\phi$. We want to determine whether $\nu_s$ can be transformed into $\nu_t$. Then, a natural operation that allows us to modify an assignment is to change the value of a single variable at a time.

One may represent an assignment $\nu$ by a binary word $w_\nu$ of length $n$ where the i-th bit of $w_\nu$ is 1 if and only if $\nu(x_i) = \top$. For instance, $\nu(x_1) = \top, \nu(x_2) = \top, \nu(x_3) = \top, \nu(x_4) = \bot$ is equivalent to the binary word 1110. So an assignment $\nu$ can be transformed in one step into another assignment $\nu'$ thanks to the aforementioned operation if and only if $w_\nu$ and $w_{\nu'}$ differ by exactly one bit, i.e., the Hamming distance of $w_\nu$ and $w_{\nu'}$ is equal to one. Let $\phi$ be a CNF formula on $n$ variables. The reconfiguration graph $\mathcal{R}(\phi)$ is the graph with vertex set $\{w_\nu \mid \nu \text{ satisfies } \phi\}$, and two vertices $w_\nu$, $w_{\nu'}$ are adjacent if and only they differ by exactly one bit. See Figure 1.30 for an example of the reconfiguration graph associated with the CNF formula of Example 1.10. Note that this reconfiguration graph is not connected.

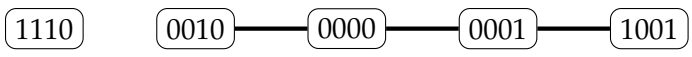



Figure 1.30 – $\mathcal{R}(\phi)$ of $\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee \neg x_4)$.

In SATISFIABILITY RECONFIGURATION, we are given a CNF formula $\phi$ and two assignments $\nu_s$ and $\nu_t$ satisfying $\phi$, and we want to determine if there exists a path in $\mathcal{R}(\phi)$ between $\nu_s$ and $\nu_t$. If $\phi$ is a 2-CNF formula (i.e., if each clause of $\phi$ contains at most two literals), then Gopalan et al. [GKMP09] showed that SATISFIABILITY RECONFIGURATION is polynomial-time solvable. This result is not surprising as discussed in Subsection 1.5.3 since it is well-known that the original problem 2-SAT is also in P [APT79]. Actually, Thomas Schaefer proved in 1978 a dichotomy theorem regarding the complexity of the SATISFIABILITY problem [Sch78]. More precisely, he even proved a more general result since he fully characterized the complexity of what he called GENERALIZED SATISFIABILITY. In this problem, we are given a finite set $S$ of relations over propositional variables. An instance of this problem is an *$S$-formula*, i.e., a conjunction of clauses of the form $R(x_{i_1}, \ldots, x_{i_k})$ with $R \in S$ and all the $x_{i_j}$'s are propositional variables.

Schaefer showed that GENERALIZED SATISFIABILITY is in P only for formulas built from some identified Boolean relations (called *Schaefer relations*), and is NP-complete otherwise. This result was extended to the reconfiguration framework by Gopalan et al. [GKMP09], later corrected by the work of Schwerdtfeger [Sch14]. More precisely, they proved that SATISFIABILITY RECONFIGURATION is in P for formulas built from *tight relations* (a superset of Schaefer relations), and is PSPACE-complete otherwise. Moreover, Gopalan et al. [GKMP09] proved a dichotomy result regarding the diameter of the connected components of the reconfiguration graph $\mathcal{R}(\phi)$: it can be exponential if the problem is PSPACE-complete, while it is linear otherwise.

Mouawad et al. [MNPR15] studied the shortest variant of SATISFIABILITY RECONFIGURATION. More precisely, they proved a trichotomy theorem based on tight relations and a subset of tight relations called *navigable relations*. Their result is that finding a shortest sequence between two assignments $\nu_s$ and $\nu_t$ is in P if the formula is built from navigable relations, NP-complete if it is built from tight relations which are not navigable, and PSPACE-complete otherwise. Their result is of special interest since the shortest reconfiguration sequence (that can be found in polynomial time) may be greater than the size of the symmetric difference of the values of the variables in $\nu_s$ and $\nu_t$. Indeed, it is the first result where a polynomial-time algorithm computing the shortest sequence does not decrease the size of the symmetric difference along the transformation.

**Reconfiguration of edge-subsets problems.** In a lot of graph problems, a solution is a subset of edges. For instance, one can cite the following problems: SHORTEST PATH, MINIMUM SPANNING TREE, MAXIMUM MATCHING and so on. The first ones that were studied under the reconfiguration framework in the seminal paper by Ito et.al [IDH+08] are the MINIMUM SPANNING TREE and MAXIMUM MATCHING problems. The reachability variant of these two problems is in P, which is not surprising since the host problems also are in P. In the MINIMUM SPANNING TREE RECONFIGURATION problem, we are given an edge-weighted graph $G$, an integer $k$ and two spanning trees $T_s$ and $T_t$ of $G$ both of weight at most $k$ and we seek to transform $T_s$ into $T_t$ via edge flips without ever getting into a tree with weight greater than $k$. Note that this positive result follows from a more general result on matroids regarding the exchange property. In Section 4.1, we focus on the reconfiguration of spanning trees with constraints on the number of leaves of each spanning tree.

For the MAXIMUM MATCHING RECONFIGURATION problem, the rule considered by Ito et al. in [IDH+08] is not the same since they can only either add or remove a single edge at each step. The problem is defined as follows: we are given an edge-weighted graph $G$, a positive integer $k$ and two matchings $M_s$ and $M_t$ (both of weight at least $k$) and we seek to transform $T_s$ into $T_t$ via the addition or deletion of a single edge at each step, and without ever getting into a matching with weight less than $k$. The proof consists in showing that the greedy algorithm indeed solves the problem. Moreover, for positive instances, it provides a bound in $O(n^2)$ on the length of the reconfiguration sequence.

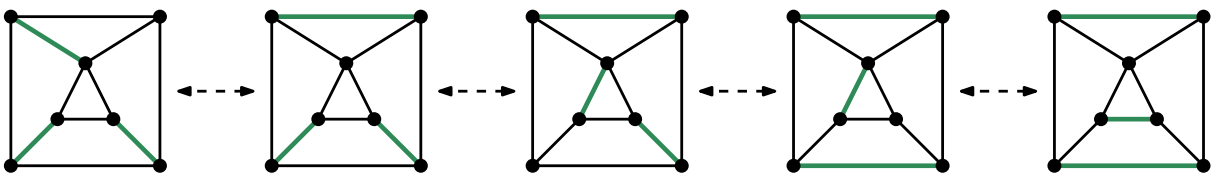


Figure 1.31 – Reconfiguration sequence of perfect matchings via edge flips.

Gupta et al. [GKM19] studied the shortest variant of maximum matchings reconfiguration, where two matchings are adjacent if and only they differ by exactly two edges. More precisely,

they showed that determining if the length of the shortest reconfiguration sequence between two maximum matchings $M_s$ and $M_t$ is at most a given integer $\ell$ is NP-hard, even if the input graph is an unweighted bipartite graph of maximum degree four. This NP-hardness result was independently found by Bousquet et al. in [BHIM19], where they also proved that the problem is fixed-parameter tractable on bipartite graphs when parameterized by $|M_s \triangle M_t|$. If the two matchings are maximum, then they showed that there is a $|M_s \triangle M_t|^\epsilon$-factor approximation algorithm for every $\epsilon > 0$. On the positive side, they proved that the problem becomes tractable if at least one of the two input matchings is not inclusion-wise maximal.

Bonamy et al. [BBH+19] studied the reconfiguration of perfect matchings. Obviously, since a perfect matching touches all the vertices, it is not possible to add or remove an edge without breaking the property. Hence, a natural rule is to exchange the edges around an alternating $C_4$.

Figure 1.32 – Example of flip: the green edges are edges of a perfect matching $M$ and the red ones are edges of another perfect matching $M'$.

They investigated the complexity of the reachability of PERFECT MATCHING RECONFIGURATION and showed that this problem is PSPACE-complete, even if the input graph is a split graph or is bipartite with bounded bandwidth and maximum degree five. On the other hand, they proved that it is polynomial-time solvable on strongly orderable graphs, cographs or outerplanar graphs. Moreover, a reconfiguration sequence (if it exists) of linear size can be computed in polynomial time for each of these three graph classes. Ito et al. [IKK+19] investigated the complexity of the shortest variant where they are allowed to exchange the edges of any alternating cycle at each step. They showed that this problem is NP-hard even if the input graph is bipartite or planar, while it is in P on cographs. They also observed that it is equivalent to the combinatorial shortest path problem in perfect matching polytopes.

Mizuta et al. [MIZ16] studied the reconfiguration of minimum Steiner trees, a generalization of the MINIMUM SPANNING TREE problem. In the MINIMUM STEINER TREE problem, we are given a (weighted) graph $G = (V, E)$, and a subset of vertices $S \subseteq V$ called *terminals* and the goal is to find a subset of edges of minimum weight that induces a tree and that covers all the terminals. Hence, if $S = V$, then it corresponds to the MINIMUM SPANNING TREE problem. In this paper, they showed that the reconfiguration of minimum Steiner trees via edge flips is PSPACE-complete, even if the input graph is a split graph, and all the edges have weight one. On the positive side, they proved that the problem is linear-time solvable if the input graph is an interval graph. Mizuta et al. [MHIZ19] later studied the same problem under two different reconfiguration rules, where it is allowed to exchange vertices that are used in the Steiner trees to connect the terminals. Note that if the set of terminals is just a pair $(s, t)$ of vertices, then the problem corresponds to $(s, t)$-shortest paths reconfiguration, which is PSPACE-complete as discussed above. They showed that the complexity may change depending on the reconfiguration rule. For instance, in one case, the problem is PSPACE-complete on split graphs (and thus on chordal graphs) while it is polynomial-time solvable on chordal graphs in the other case [MHIZ19].

**Independent set and vertex cover reconfiguration.** The reconfiguration of independent sets is probably the first graph problem that was considered under the reconfiguration framework by Hearn and Demaine [HD05]. It was proposed as a natural extension of token-sliding puzzles, for which Rush Hour (see Subsection 1.5.1) is a particular case and where the goal is to move (not necessarily horizontally or vertically) pieces (whose shape can be integral rectangle, L shape, etc.) so that a particular piece can reach a given target position. Hearn and Demaine [HD05] proved that if the pieces form an independent set, and a move can only be done to an adjacent position, then the problem is PSPACE-complete. As a corollary, their result implies that the reconfiguration of independent sets where one can exchange a single vertex with one of its neighbors at each step is PSPACE-complete. A few years later, Ito et al. [IDH+08] studied the reachability of independent sets via vertex addition or removal and showed that this problem is also PSPACE-complete if we require that each intermediate independent set must contain a certain number of vertices. Extensive work has since been done on this topic, and a large part of this research has focused on the complexity (in comparison with the complexity of the host problem) of this problem from a graph classes perspective, or depending on the operation that allows us to modify a solution. As discussed in Subsection 1.5.3, it is common that when the host problem is in P, so is the reconfiguration variant. And the INDEPENDENT SET RECONFIGURATION is no exception to this pattern. For instance, the INDEPENDENT SET PROBLEM is polynomial-time solvable on cographs [MS99], $K_{1,3}$-free graphs [Sbi80], or interval graphs [Gav74, RT75]. For both of these two operations (i) add or remove a single vertex; and (ii) exchange a vertex with one of its neighbors, INDEPENDENT SET RECONFIGURATION has been proven to be in P as well for each of these three classes (see Figure 1.33 for an illustration with the latter rule).

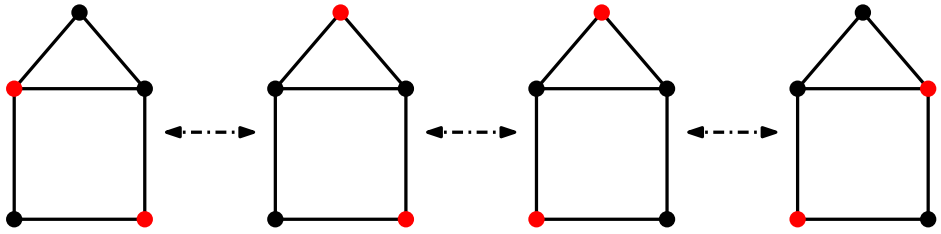


Figure 1.33 – Reconfiguration sequence of independent sets with rule (ii).

For the operation (ii), see [KMM12] for cographs, [BB17] for chordal graphs, and [BKW14] for $K_{1,3}$-free graphs. For the operation (i), see [BB14, Bon16] for cographs, [KMM12, MNR14, INZ15] for chordal graphs (actually they show the result on even-hole free graphs, a superclass of chordal graphs), and [BKW14] for $K_{1,3}$-free graphs. However, there are exceptions to this pattern. For instance, it is known that the INDEPENDENT SET problem is polynomial-time solvable on bounded treewidth graphs [AP89] but its reconfiguration version is PSPACE-complete for both operations (i) and (ii) [Wro18].

> The result of Wrochna is more general since it is shown that INDEPENDENT SET RECONFIGURATION is PSPACE-complete on bounded bandwidth graphs, a proper subclass of bounded treewidth graphs (see Observation 1.25).

Another very interesting result is the one by Lokshtanov and Mouawad [LM18] where it is showed that the complexity of INDEPENDENT SET RECONFIGURATION is not the same for operations (i) and (ii). More precisely, they proved that the problem is NP-complete if we consider operation (i), but PSPACE-complete for operation (ii). Recall that the host problem is in P for bipartite graphs. Moreover, we have seen that INDEPENDENT SET RECONFIGURATION is in

P for even-hole free graphs with rule (i) [KMM12, MNR14, INZ15]. However, the complexity remains open for both the host problem and rule (ii) on this graph class.

The INDEPENDENT SET problem is closely related to the MAXIMUM MATCHING problem. Indeed, a maximum independent set $I$ in the line graph $L(G)$ of a graph $G$ corresponds to a maximum matching in $G$. Recall that the line graph $L(G)$ can be obtained from $G$ by creating a vertex for each edge of $G$, and two vertices of $L(G)$ are adjacent if the two corresponding edges of $G$ are adjacent. Moreover, given a line graph $G$, one can construct in linear time a graph $H$ such that $G$ is the line graph of $H$ [Rou73]. It follows that the INDEPENDENT SET RECONFIGURATION problem in $H$ is equivalent to the MAXIMUM MATCHING RECONFIGURATION in $G$ (see Figure 1.34).

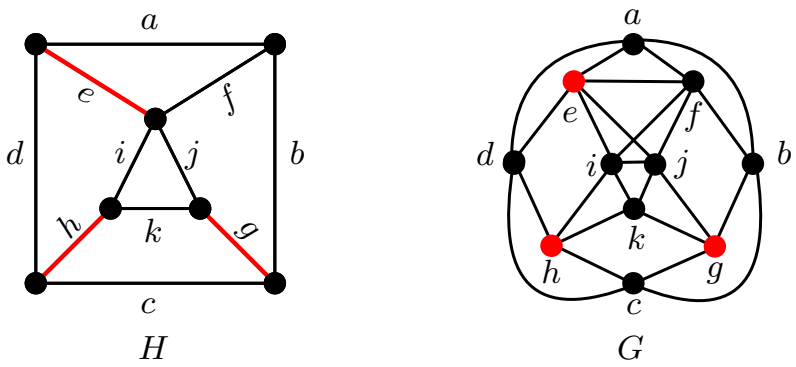



Figure 1.34 – Maximum matching in $H$ and the corresponding maximum independent set in its line graph $G$.

Another problem which is closely related to INDEPENDENT SET is the VERTEX COVER problem. Indeed, recall that given a graph $G = (V, E)$, a subset of vertices $I \subseteq V$ is an independent set of $G$ if and only if $V \setminus I$ is a vertex cover of $G$ (see Observation 1.21).

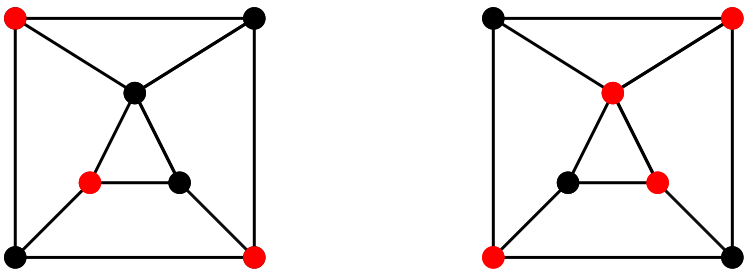


Figure 1.35 – A maximum independent set whose complement is a minimum vertex cover.

Therefore, INDEPENDENT SET RECONFIGURATION and VERTEX COVER RECONFIGURATION are equivalent with respect to classical complexity. Indeed, suppose that there is reconfiguration sequence $\langle I_0 = I_s, I_1, I_2, \ldots, I_\ell = I_t \rangle$ between two independent sets $I_s$ and $I_t$. Then, $\langle V \setminus I_0 = V \setminus I_s, V \setminus I_1, V \setminus I_2, \ldots, V \setminus I_\ell = V \setminus I_t \rangle$ is a reconfiguration sequence between $C_s = V \setminus I_s$ and $C_t = V \setminus I_t$. However, the parameterized complexity of these two problems is not the same since VERTEX COVER is FPT with respect to the solution size $k$ (see Example 1.13) while INDEPENDENT SET is W[1]-hard when parameterized by $k$ [DF99]. Hence, it is not surprising that the reconfiguration of independent sets is W[1]-hard parameterized by $k + \ell$ where $\ell$ is the length of the desired reconfiguration sequence, and thus W[1]-hard parameterized by $k$ [MNR+17]. This result contrasts with the one on VERTEX COVER RECONFIGURATION which has been shown to be FPT with respect to $k$ [MNR+17]. On the positive side, INDEPENDENT

SET RECONFIGURATION is FPT when parameterized by both the solution size $k$ and the maximum degree $\Delta$ on general graphs [IKO+20] and on planar graphs (actually they proved the result for any graph that excludes $K_{3,d}$ as a subgraph for any $d \geq 3$) with respect to $k$ [IKO14]. This result was latter generalized to graphs that exclude $K_{d,d}$ as a minor, for any $d \geq 3$ [BMP17]. The FPT algorithms for planar graphs and bounded maximum degree graphs are implied by a more general FPT algorithm on nowhere dense graphs [LMP+18]. In the same paper, Lokshtanov et al. also provided an FPT algorithm parameterized by $k$ for bounded degeneracy graphs. However, Bonamy and Bousquet [BB17] showed that if the operation that modifies a solution consists in exchanging a vertex against one of its neighbors, then the problem becomes W[2]-hard (also with respect to the solution size $k$) on split graphs. For VERTEX COVER RECONFIGURATION which is FPT with respect to $k$ on general graphs, it has also been shown that it is W[1]-hard when parameterized by $\ell$, the length of the desired reconfiguration sequence [MNR+17]. However, it is FPT when parameterized by both $\ell$ and the treewidth of the input graph [MNRW14]. Even if VERTEX COVER is NP-complete on planar graphs, or on bounded maximum degree $\Delta$ graphs (with $\Delta \geq 4$), FPT algorithms parameterized by $\ell$ have been designed for these two classes [MNRW14, MNR14]. On the other side, VERTEX COVER RECONFIGURATION is W[1]-hard when parameterized by $\ell$ on bipartite graphs. However, the host problem is in P by König-Egerváry's theorem which states that the maximum size of a matching is equal to the minimum size of a vertex cover in bipartite graphs and Edmonds' blossom algorithm to solve MAXIMUM MATCHING on general graphs in time $O(|E| \cdot |V|^2)$ [Edm65].

### 1.5.5 Defining an adjacency rule

Reconfiguration problems are closely related to the state space of the host problem as discussed in Section 1.5.2. Typical questions that arise in this framework are for instance: "Does there exist a path in the state space between two solutions?" or "What is the minimum length of such a path, if it exists?". All these questions can be studied through the lens of the reconfiguration graph (see Section 1.5.2): is this graph connected? What is its diameter? Indeed, recall that the vertices of the reconfiguration graphs $\mathcal{R}(\mathcal{I})$ associated with an instance $\mathcal{I}$ of a problem $\Pi$ correspond to the feasible solutions of $\mathcal{I}$. The adjacency between two vertices of $\mathcal{R}(\mathcal{I})$ is determined by the reconfiguration rule. Obviously, this rule deeply depends on the host problem.

Let us focus on the reconfiguration of graphs problems, and let $G = (V, E)$ be the input graph of the host problem. It is clear that if a solution of a graph problem $\Pi$ is a subset of edges $E' \subseteq E$, the reconfiguration rule is unlikely to be the same as if it is a subset of vertices $V' \subseteq V$ for instance. We have seen that if the host problem is the MATCHING problem, then a modification of a given solution often consists in adding new edges or removing some of its edges, sometimes in a single step. On the other hand, if we are interested in the INDEPENDENT SET problem, the reconfiguration rule may consist either in exchanging a vertex, or adding or removing a vertex. However, this is not always the case. For instance, Mizuta et al. [MHIZ19] considered vertex exchanges as a reconfiguration rule in their paper on MINIMUM STEINER TREES RECONFIGURATION for which a solution is a subset of edges connecting the terminals.

In some problems, there is only one natural choice for the elementary operation that defines the adjacency in $\mathcal{R}(\mathcal{I})$. For instance, in SATISFIABILITY RECONFIGURATION, the natural transformation is to change the value of a single variable at each step. However, for many problems, there may be more than one possible reconfiguration rule. For instance, for the PERFECT MATCHING RECONFIGURATION problem, the operation used in [IKK+19] is more general than the one used in [BBH+19]. Indeed, the rule in [BBH+19] consists in exchanging the edges of an alternating cycle of length four, i.e., two matching $M_1$ and $M_2$ are adjacent if and only if their symmetric difference is a $C_4$. With this rule, it is not always possible to transform a matching $M_s$ into a matching $M_t$ and Bonamy et al. [BBH+19] proved that the reachability problem is

PSPACE-complete. However, in [IKK+19], one can exchange the edges of any alternating cycle. This model is strictly more powerful than the one in [BBH+19], as it is always possible to transform a matching into another one. Indeed, the symmetric difference of two disjoint matchings $M_s$ and $M_t$ is a collection of even-length alternating cycles. Hence, Ito et al. [IKK+19] focused on the length of the shortest transformation between $M_s$ and $M_t$, which can be much smaller than the number of disjoint alternating cycles in the symmetric difference $M_s \triangle M_t$. Note that the rule introduced by Ito et al. in [IDH+08] for the MAXIMUM MATCHING problem and which allows the addition or removal of a single edge does not make any sense for the PERFECT MATCHING PROBLEM. Indeed, the addition of an edge to a perfect matching $M$ breaks the independence property of the edges of $M$, while the removal of an edge $e \in M$ yields a matching which is not perfect.

When the solution of $\Pi$ is a subset of vertices, one can represent the solution $S$ by a set of tokens, where exactly one token is placed on each vertex that belongs to $S$. Then, modifying a solution corresponds to shifting the tokens according to the reconfiguration rule. In the literature, three kinds of operations have been mainly studied:

- Token Addition and Removal (TAR): one can add or remove a token;
- Token Jumping (TJ): one can move a token to any vertex of the graph;
- Token Sliding (TS): one can slide a token along an edge, i.e., one moves a token to a neighbor of its current location.

One can observe that in the last two models, the size of each solution remains constant at any time, as opposed to what happens in the TAR model. Moreover, any reconfiguration sequence under the token sliding rule is also a valid sequence under the token jumping model. However, the converse is not always true, and actually TJ is strictly more powerful than TS as showed below for INDEPENDENT SET RECONFIGURATION.

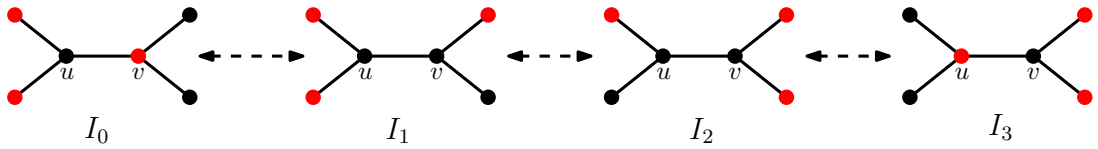



Figure 1.36 – Reconfiguration sequence under TJ between two independent sets $I_0$ and $I_3$.

In Figure 1.36, one can see a reconfiguration sequence between two independent sets $I_0$ and $I_3$ under token jumping. However, one can observe that $I_0$ cannot be transformed into $I_3$ by token sliding. Indeed, before moving a token from the left part to the right part, one first need to move the token from $v$ to one of its two degree-one neighbors. But then, no token cannot be put again on $v$. Since there is a unique path between a vertex in the left part and a vertex in the right part and that path goes through $v$, the conclusion follows.

Unlike what happens under token jumping and token sliding, token addition and removal modifies the size of the solutions. Hence, in order to make the problem more interesting, TAR comes with a threshold $k$ which gives a bound on the size of each solution. Then, if $\Pi$ is a minimization (respectively maximization) problem like VERTEX COVER (resp. INDEPENDENT SET), $k$ correspond to the maximum (resp. minimum) size of each intermediate solution. Indeed, if we do not put any restrictions on the size of the intermediate solutions, the following trivial algorithm yields a shortest reconfiguration sequence between two independent sets $I_s$ and $I_t$: (i) remove one-by-one each vertex in $I_s \setminus I_t$; and (ii) add one-by-one each vertex in $I_t \setminus I_s$. However, the TAR model and the TJ model are equivalent under some conditions, as observed by Kamiński et al. [KMM12]:

**Theorem 1.27** ([KMM12], Theorem 1). *There exists a reconfiguration sequence between two independent sets $I_s$ and $I_t$ of size $k$ under* TJ *if and only if there exists a reconfiguration sequence under* TAR *where each solution has size at least $k-1$. Moreover, there exists an algorithm that, given a reconfiguration sequence between $I_s$ and $I_t$ in one of these two models, outputs a reconfiguration sequence connecting the two sets in the other model in time polynomial in the length of the sequence. The algorithm maps every shortest* TAR *sequence to a shortest* TJ *sequence, and vice versa. The* TAR *sequence is twice as long as the* TJ *sequence.*

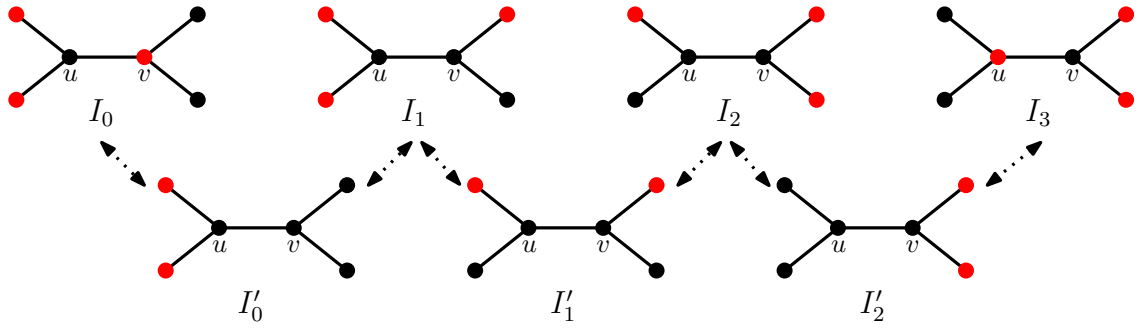



Figure 1.37 – Reconfiguration sequence under TAR between two independent sets $I_0$ and $I_3$.

In Figure 1.37, one can see the reconfiguration sequence $S'$ under TAR between $I_0$ and $I_3$ corresponding to the sequence $S'$ under TJ depicted in Figure 1.36. The length of $S'$ is twice the length of $S$. Moreover, each intermediate solution contains at least two tokens. For an example of a reconfiguration under TS, see Figure 1.33.

In the remaining of this thesis, we denote by TAR($k$) the reconfiguration rule TAR associated with the threshold $k$. Let $\Pi$ be a graph problem, let $\mathcal{I}$ be an instance of $\Pi$ and let $S_s$ and $S_t$ be two solutions of $\mathcal{I}$. We denote by TAR($k$)-sequence (or simply TAR-sequence) a reconfiguration sequence $\langle S_0 = S_s, S_1, \ldots, S_\ell = S_t \rangle$ under the TAR($k$) rule between $S_s$ and $S_t$, where:

(i) each intermediate solution $S_i$ is a feasible solution of $\mathcal{I}$ of size at least (or at most) $k$; and

(ii) $S_i$ can be obtain from $S_{i-1}$ by applying once the TAR rule.

We adopt the same notation for the TJ and TS rules, and denote by TJ-sequence (respectively TS-sequence) a reconfiguration sequence under token jumping (resp. token sliding). We also introduce the three following notation, where $D_s$ and $D_t$ are two dominating sets of $G$, both of size $k$.

- $D_s \overset{\text{TAR(k)}}{\rightsquigarrow} D_t$: there is a TAR($k$)-sequence that transforms $D_s$ into $D_t$;
- $D_s \overset{\text{TJ}}{\rightsquigarrow} D_t$: there is a TJ-sequence that transforms $D_s$ into $D_t$;
- $D_s \overset{\text{TS}}{\rightsquigarrow} D_t$: there is a TS-sequence that transforms $D_s$ into $D_t$;

A useful observation is that each reconfiguration sequence that we consider in this thesis is reversible: if $D_s \rightsquigarrow D_t$ holds, then $D_t \rightsquigarrow D_s$ holds too. We thus denote this relation by $D_s \leftrightsquigarrow D_t$ in the remaining of this thesis.

# 2 Domination in graphs

**Contents**



## 2.1 Introduction on domination

We start this chapter by a brief introduction on domination in graphs. For a more complete overview, the reader is referred to the books by Haynes, Hedetniemi and Slater [HHS97, HHS98].

### 2.1.1 History

Even if the study of domination problems in graphs began around 1960, the first domination-related problems appeared more than a hundred years earlier. Indeed, the problem that can be considered to be the origin of the study of dominating sets is related to chess. This game is played on a $8 \times 8$ chessboard. Among the game pieces, the queen moves horizontally, vertically or diagonally, as long as each square in its way is free of any other piece (see Figure 2.1).

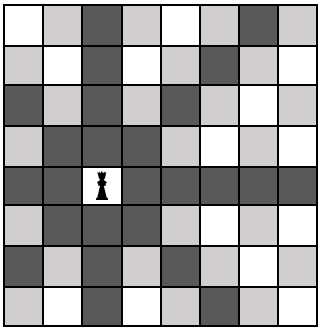


Figure 2.1 – The queen can move to any dark gray shaded square.

We say that the queen *dominates* all the squares located horizontally, vertically and diagonally from its current position. For example, all the dark gray shaded on Figure 2.1 are dominated by the queen drawn. Chess players wondered what is the minimum number of queens needed to dominate all the squares. They thought that the answer was five on a $8 \times 8$ grid, which turned out to be correct [BCM97].

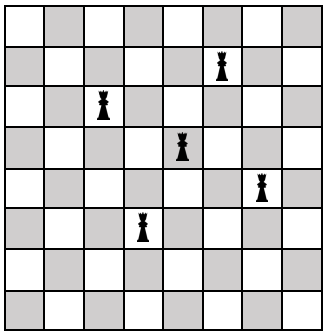


Figure 2.2 – A possible solution.

This problem can easily be modeled as a graph problem. Indeed, consider the graph $G$ defined as follows: associate a vertex with each square and let two vertices be adjacent if and only if the corresponding squares are vertically or horizontally aligned, or if they belong to the same diagonal. Then, finding the minimum number of queens that dominate all the free squares (i.e., squares that do not carry a queen) is the same as finding the smallest possible subset $D$ of vertices of $G$ such that each vertex not in $D$ has at least one neighbor in $D$.

The study of domination in graphs started in 1960s with the seminal book "Graph theory and its applications" by Claude Berge [Ber58]. In this book, Berge introduced the *external stability number* of a graph, nowadays known as the *domination number*. The names *dominating set* and *domination number* were actually introduced a few years later by Øystein Ore in his book "Graph theory" [Ore62]. Berge continued the study of dominating sets in his book "Graphs and hypergraphs" published in 1973 [Ber73]. However, the paper that stimulated work on this topic is probably the survey of Cockayne and Hedetniemi from 1977 [CH77]. This paper introduced the notations $\gamma(G)$ and $\Gamma(G)$ to denote respectively the minimum size of a dominating set, and the maximum size of a minimal dominating set.

### 2.1.2 Definitions and simple results

Let $G = (V, E)$ be a graph on $n$ vertices. A dominating set of $G$ is a subset of vertices $D \subseteq V$ such that $N[D] = V$. In other words, each vertex that does not belong to $D$ must have (at least) one neighbor in $D$. It is clear that every graph admits a dominating set since we can simply select all the vertices to be in the dominating set. Moreover, we have $\gamma(G) = 0$ if and only $G$ has no vertex, and $\gamma(G) = n$ if and only if $G$ is edgeless. However, for connected graphs on at least two vertices, we can have a much better upper bound:

**Proposition 2.1** ([Ore62]). *If $G$ is a connected graph on at least two vertices, then $\gamma(G) \leq \lfloor n/2 \rfloor$.*

*Proof.* Let $T$ be a spanning tree of $G$. Root $T$ on an arbitrary node $r$. Let $X$ (respectively $Y$) be the set of vertices which are at even (respectively odd) distance from $r$ in $T$. It is clear that we have $|X| \leq \lfloor n/2 \rfloor$, or $|Y| \leq \lfloor n/2 \rfloor$ since $X, Y$ is a partition of $V(T)$ and thus of $V(G)$. We assume without loss of generality that $|X| \leq |Y|$. $X$ is a dominating set of $T$ since each vertex in $Y$ has at least one neighbor in $X$. Note that $X$ is also a dominating set of $G$. Indeed, any vertex $u$ of $G$ dominated by a vertex $v \in N[u] \cap D$ in $T$ is also dominated by $v$ in $G$ since $V(G) = V(T)$ and every edge of $T$ belongs to $G$. Hence, $X$ is a dominating set of $G$ of size at most $\lfloor n/2 \rfloor$. □

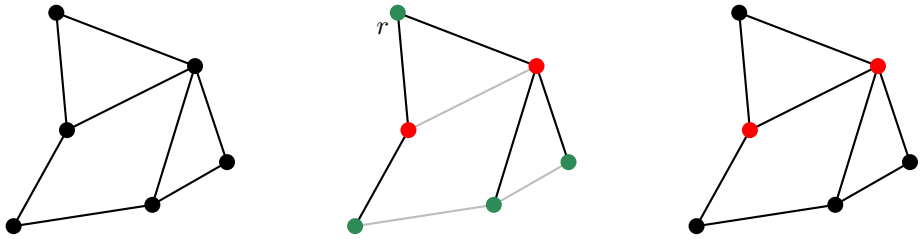



Figure 2.3 – Illustration for the proof of Proposition 2.1: the red vertices dominate the graph.

Actually, the result of Proposition 2.1 can be generalized to graphs without isolated vertices. Indeed, let $X_1, X_2, \ldots, X_k$ be the connected components of $G$. We denote by $n_i$ the number of vertices of $X_i$. Note that $n_i \geq 2$ holds for any $1 \leq i \leq k$ as $G$ does not contain isolated vertices. For any connected component $X_i$, we apply Proposition 2.1 and obtain a dominating set $D_i$ of size at most $\lfloor n_i/2 \rfloor$. Clearly, $\bigcup_{i=1}^{k} D_i$ is a dominating set of $G$ of size at most $\sum_{i=1}^{k} \lfloor n_i/2 \rfloor \leq \lfloor n/2 \rfloor$. So Proposition 2.1 can be replaced by the following theorem:

**Theorem 2.2** ([Ore62]). *If $G$ be a graph without isolated vertices, then $\gamma(G) \leq \lfloor n/2 \rfloor$.*

Let $G' = (V, E')$ be a partial graph of $G$, i.e., $V(G') = V(G)$ and $E' \subseteq E$. As discussed in the proof of Proposition 2.1, any dominating set $D$ of $G'$ also is a dominating set of $G$ since $N_{G'}[u] \subseteq N_G[u]$ holds for any vertex $u$ of $G$ (and thus $G'$). With a similar idea, one can prove the following observation:

**Observation 2.3.** *Let $G = (V, E)$ be a graph, let $D$ be a dominating set of $G$, and let $u \in D$. If there exists a vertex $v \in V \setminus D$ such that $N[u] \subseteq N[v]$, then $(D \setminus \{u\}) \cup \{v\}$ is also a dominating set of $G$.*

This observation states that one can replace a vertex $u$ of a dominating set $D$ by another vertex $v \in V \setminus D$ as long as $v$ dominates all the vertices dominated by $u$. Hence, if a dominating set $D$ has two vertices $u$ and $v$ such that $N[u] \subseteq N[v]$, then $D \setminus \{u\}$ also is a dominating set of $G$. This observation leads to the following characterization of minimal dominating sets:

**Proposition 2.4.** *A dominating set $D$ of a graph $G$ is inclusion-wise minimal if and only if there is no pair of vertices $u, v \in D$ such that $N[u] \subseteq N[v]$ or $N[v] \subseteq N[u]$.*

*Proof.* ($\Rightarrow$) Suppose that $D$ contains two vertices $u$ and $v$ such that $N[u] \subseteq N[v]$. Then, $D \setminus \{u\}$ also is a dominating set, a contradiction.
($\Leftarrow$) Suppose that $D$ does not contain a pair of vertices $u, v$ such that $N[u] \subseteq N[v]$ or $N[v] \subseteq N[u]$. So, for any vertex $u \in D$, there exists a vertex $x \in N[u]$ (note that we may have $x = u$) such that $x$ has no neighbor in $D \setminus \{u\}$. Hence, the removal of $u$ leaves $x$ undominated, and the conclusion follows. □

The vertex $x$ in the proof of Proposition 2.4 is called a *private neighbor of $u$ with respect to $D$*. Hence, Proposition 2.4 can be rephrased as follows:

**Proposition 2.5.** *A dominating set $D$ is minimal if and only if each vertex in $D$ has a private neighbor.*

As we said before, the minimum size of a minimal dominating set of $G$ is the domination number of $G$ and it is denoted by $\gamma(G)$. On the other hand, the maximum size of a minimal dominating set of $G$ is the *upper domination number*, and it is denoted by $\Gamma(G)$. A dominating set of size $\gamma(G)$ is a *minimum dominating set* of $G$, while a dominating set of size $\Gamma(G)$ is an *upper dominating set* of $G$. The complete graph $K_n$ is the only graph satisfying $\gamma(G) = \Gamma(G) = 1$. However, note that the difference between $\gamma(G)$ and $\Gamma(G)$ can be arbitrarily large, as pointed out by the following proposition:

**Proposition 2.6.** *The complete graph $K_{1,n}$ satisfies $\gamma(G) = 1$ and $\Gamma(G) = n$.*

*Proof.* Note that we may assume without loss of generality that $n > 2$ since $\gamma(K_{1,1,}) \simeq K_2$. Let $u$ be the universal vertex of $G$. Since $K_{1,n}$ has at least one edge and contains a universal vertex, we have $\gamma(G) = 1$. On the other hand, any minimal dominating set that does not contain $u$ must contain all the $n$ degree-one vertices since they induce an independent set. Since there is no minimal dominating set of size at least two containing $u$, $\Gamma(K_{1,n}) = n$. □

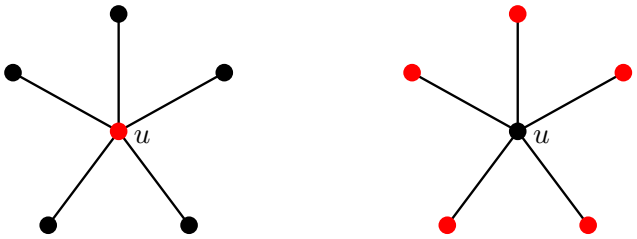



Figure 2.4 – A minimum dominating set and an upper dominating set of $K_{1,5}$.

### 2.1.3 Relation with other graph parameters

For any graph $G$, there is a very interesting inequality chain that bounds $\gamma(G)$ and $\Gamma(G)$ by other graphs parameters related with maximal independent sets and irredundant sets (they will be define later) of $G$. This chain was introduced by Cockayne, Hedetniemi and Miller in 1978 [CHM78]. Let us briefly discuss it here.

#### Maximal independent sets

Given a graph $G = (V, E)$, the dominating sets of $G$ and its maximal independent sets are related by the following well-known observation:

**Observation 2.7.** *Let $S \subseteq V$ be a maximal independent set of $G$. Then, $S$ also is a dominating set of $G$.*

*Proof.* Let $S$ be a maximal independent set of $G$, and suppose that $S$ is not dominating. Then, there exists a vertex $u \in V$ which is not dominated by $S$. This means that $N[u] \cap S = \emptyset$. But then, $S \cup \{v\}$ is an independent set that contradicts the maximality of $S$. □

In Observation 2.7, the fact that $S$ is inclusion-wise maximal is crucial since obviously any independent set does not necessarily dominate $G$. More precisely, an independent set $S$ is maximal if and only if the following condition holds:

$$\text{for every vertex } u \in V \setminus S, u \text{ has at least one neighbor in } S.$$

Note that this condition corresponds precisely to the definition of domination: any vertex not in the set must have at least one neighbor in the set. However, the converse of Observation 2.7 is not true: a dominating set $D$ does not necessarily induce a maximal independent set of $G$, even if $D$ is minimal (see Figure 2.5).

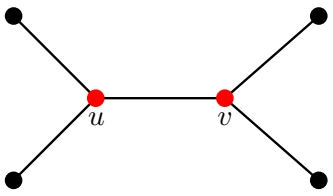



Figure 2.5 – The set $\{u, v\}$ is a minimal dominating set, but not a stable set.

So one can replace Observation 2.7 by the following, which was first observed by Berge:

**Proposition 2.8** ([Ber58]). *A set $S \subseteq V$ is a maximal independent set of $G$ if and only if $G[S]$ is edgeless and $S$ dominates $G$.*

*Proof.* We have already seen in Observation 2.7 that if $S$ is a maximal independent set, then it is also a dominating set. Let us show the converse direction. Suppose that $S$ is both independent (i.e., $G[S]$ is edgeless) and dominating. Assume that $S$ is not inclusion-wise maximal. Then, there exists $u \in V \setminus S$ such that $S \cup \{u\}$ is also an independent set. Since we can add $u$ to $S$ without breaking the independence property, this means that $u$ has no neighbor in $S$ and thus $N[u] \cap S = \emptyset$. But then $u$ is not dominated by $S$, a contradiction. □

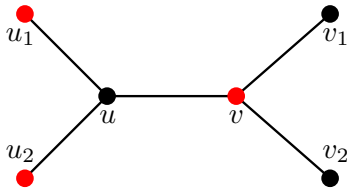



Figure 2.6 – The set $\{u_1, u_2, v\}$ is a maximal independent set.

Recall that $\alpha(G)$ is the independence number of $G$, that is the maximum size of an independent set of $G$. We denote by $i(G)$ the minimum size of a maximal independent set of $G$. This parameter is called the *independent domination number* of $G$ because it corresponds to the minimum size of a dominating set of $G$ that is also an independent set by Proposition 2.8. Hence, it immediately follows from Proposition 2.8 that the following inequalities hold for any graph $G$:

$$\gamma(G) \leq i(G) \leq \alpha(G) \leq \Gamma(G) \tag{2.1}$$

Note that for complete graphs, we have $\gamma(K_n) = i(K_n) = \alpha(K_n) = \Gamma(K_n) = 1$. However, the difference between any two of these parameters can be make arbitrarily large. For instance, let us consider the graph $S_{r,r}$ which is obtained by creating an edge $uv$ and then $r$ pendant vertices $u_1, u_2, \ldots, u_r$ (respectively $v_1, v_2, \ldots, v_r$) adjacent to $u$ (respectively $v$). Observe that the graph showed in Figure 2.6 is the graph $S_{2,2}$. Note that for any $r \geq 2$, the graph $S_{r,r}$ satisfies $\gamma(S_{r,r}) = 2$ (see Figure 2.5), $i(S_{r,r}) = r + 1$ (see Figure 2.6) and $\alpha(S_{r,r}) = \Gamma(S_{r,r}) = 2r$. Hence, a natural question is to determine if there exists a graph $G$ such that $\gamma(G) = s_1$, $i(G) = s_2$, $\alpha(G) = s_3$, $\Gamma(G) = s_4$ for given integers $s_1$, $s_2$, $s_3$ and $s_4$. If such a graph exists, then $(s_1, s_2, s_3, s_4)$ is called a *domination sequence*. These sequences were characterized by Cockayne and Mynhardt:

**Theorem 2.9** ([CM93]). *Let $s_1$, $s_2$, $s_3$ and $s_4$ be four integers. Then, $(s_1, s_2, s_3, s_4)$ is a domination sequence if and only if the three following conditions hold:*

*(i)* $1 \leq s_1 \leq s_2 \leq s_3 \leq s_4$;

*(ii)* $s_1 = 1 \Rightarrow s_2 = 1$; *and*

*(iii)* $s_3 = 1 \Rightarrow s_4 = 1$.

**Domination perfect graphs.** In the same spirit as the concept of *perfect graphs* which are graphs that satisfy $\chi(H) = \omega(H)$ for any induced subgraph $H$ of the graph $G$, Sumner and Moore [SM88] introduced the concept of domination perfect graphs. A graph $G$ is *domination perfect* if and only if $\gamma(H) = i(H)$ for any induced subgraph $H$ of $G$. Zvervich and Zverovich [ZZ95] gave a full characterization of domination perfect graphs in terms of forbidden induced

subgraphs. More precisely, they found seventeen graphs which are *minimally domination imperfects*, meaning that they are not domination perfects but all their proper induced subgraphs are. They then showed that any graph that does not contain any of these seventeen graphs as an induced subgraph indeed is domination perfect. For a more complete overview on domination perfect graphs, we refer the reader to the survey of Sumner [Sum90].

> In Section 2.2, we study a similar concept: given a graph $G$, we give necessary and sufficient conditions (in terms of forbidden induced subgraphs as well) to determine if $\gamma_c(H) \leq 2\gamma(H)$ (where $\gamma_c(H)$ is the minimum size of a connected dominating set of $H$) holds for any induced subgraph $H$ of $G$.

### Irredundant sets and the domination chain

Even if it is beyond the scope of this thesis, let us briefly discuss how we can extend the inequality chain 2.1 to bound below $\gamma(G)$ and to bound above $\Gamma(G)$. Recall that by Proposition 2.5, a dominating set $D$ is minimal if and only if each vertex in $D$ has a private neighbor with respect to $D$. In other words, for any $u \in S$, there exists $v \in N[u]$ such that $N[v] \cap (D \setminus \{u\}) = \emptyset$, i.e., the removal of $u$ from $D$ leaves $v$ undominated. Recall that we may have $u = v$. This idea leads to the notion of irredundant sets. Let $G = (V, E)$ be a graph, and let $S \subseteq V$ be a subset of vertices. Then, $S$ is *irredundant* if every vertex in $S$ has a private neighbor with respect to $S$. It follows that the minimality condition for a dominating set corresponds to the definition of an irredundant set.

An irredundant set $S$ is maximal if for any vertex $u \in V \setminus S$, the set $S \cup \{u\}$ is not irredundant, meaning that there exists a vertex $v \in S \cup \{u\}$ without any private neighbor with respect to $S \cup \{u\}$. Let us denote by $ir(G)$ and $IR(G)$ the minimum size and the maximum size of a maximal irredundant set of $G$, respectively. The following result was first observed by Bollobás and Cockayne:

**Proposition 2.10** ([BC79]). *Let $G$ be a graph, and let $D$ be a minimal dominating set of $G$. Then, $D$ is also a maximal irredundant set of $G$.*

We are know ready to define the *domination chain* which was first observed by Cockayne, Hedetniemi, and Miller in 1978 and was then extensively studied in domination theory:

**Theorem 2.11** ([CHM78]). *The following domination chain holds for any graph $G$:*

$$ir(G) \leq \gamma(G) \leq i(G) \leq \alpha(G) \leq \Gamma(G) \leq IR(G) \tag{2.2}$$

Cockayne and Mynhardt showed a stronger result than the one presented in Theorem 2.9 since they actually proved the following theorem:

**Theorem 2.12** ([CM93]). *Let $s_1$, $s_2$, $s_3$, $s_4$, $s_5$ and $s_6$ be six integers. Then, there exists a graph $G$ such that $ir(G) = s_1$, $\gamma(G) = s_2$, $i(G) = s_3$, $\alpha(G) = s_4$, $\Gamma(G) = s_5$ and $IR(G) = s_6$ if and only if the four conditions hold:*

*(i)* $1 \leq s_1 \leq s_2 \leq s_3 \leq s_4 \leq s_5 \leq s_6$;

*(ii)* $s_1 = 1 \Rightarrow s_3 = 1$;

*(iii)* $s_4 = 1 \Rightarrow s_6 = 1$; *and*

*(iv)* $s_2 \leq 2s_1 - 1$.

### 2.1.4 Computational complexity

In this section, we focus on the computational complexity of finding a dominating set of a given size in a graph. More precisely, we are interested in the decision version of the DOMINATING SET problem, which is defined as follows:

DOMINATING SET

**Instance:** A graph $G = (V, E)$, an integer $k$.
**Question:** Does $G$ have a dominating set of size *at most* $k$?

#### Hardness of the DOMINATING SET problem

There exist many polynomial-time reductions to show the NP-completeness of the DOMINATING SET problem. Let us present two different ones: the first is a reduction from 3-SAT, and the second from VERTEX COVER. First, observe that the problem belongs to NP, as it is sufficient to check if $N[u] \cap D \neq \emptyset$ holds for any vertex $u$ to determine whether a subset of vertices $D$ is a dominating set or not.

**Reduction from 3-SAT.** The reduction presented here is a very slight adaptation of the well-known reduction that we can find in [CHM78] for instance. Let $\phi$ be a 3-CNF formula with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses $C_1, C_2, \ldots, C_m$. We construct the corresponding graph $G_\phi$ as follows. For any variable $x_i$, we create the gadget $\mathcal{G}_i$ depicted in the figure below:

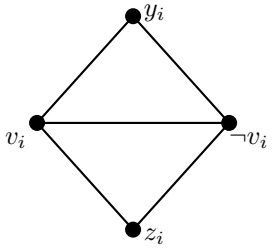



Figure 2.7 – The gadget $\mathcal{G}_i$ corresponding to the variable $x_i$ of $\phi$.

Now, for each clause $C_j$, we add a vertex $a_j$. If the clause $C_j$ contains the literal $x_i$, we add the edge $a_j v_i$. If it contains the literal $\neg x_i$, we add an edge between $a_j$ and $\neg v_i$. Note that each vertex $a_j$ has degree three since each clause of $\phi$ contains exactly three literals. Therefore, the graph $G_\phi$ has $4n + m$ vertices, and $5n + 3m$ edges. Thus $G_\phi$ can be constructed in linear time.

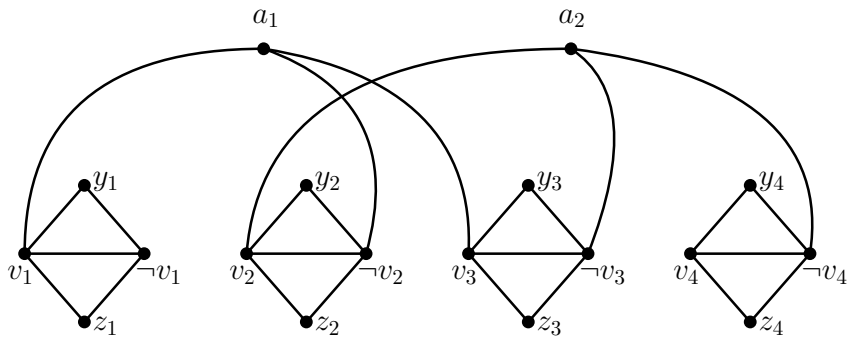



Figure 2.8 – The graph $G_\phi$ corresponding to the formula $\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$.

Note that each dominating set of $G_\phi$ must contain at least one vertex in each gadget $\mathcal{G}_i$ since $N(y_i) = N(z_i) \subseteq \mathcal{G}_i$. Let $(G_\phi, n)$ be the resulting instance of DOMINATING SET, and let $\nu$ be an assignment of the variables of $\phi$ that satisfies the formula $\phi$. One constructs a dominating set of $G_\phi$ of size $n$ as follows: if $\nu(x_i) = \top$, we add $v_i$ to $D$. Otherwise, we add $\neg v_i$ to $D$. Since both $v_i$ and $\neg v_i$ dominate $\mathcal{G}_i$, all the vertices in $\mathcal{G}_1 \cup \mathcal{G}_2 \cup \ldots \cup \mathcal{G}_m$ are dominated. Now, let $a_j$ be the vertex corresponding to the clause $C_j$ of $\phi$. Since $\nu$ satisfies $\phi$, at least one literal satisfies $C_j$. But then $D$ contains the corresponding vertex ($v_i$ or $\neg v_i$) and thus is dominated by $D$.

For the other direction, let $D$ be a dominating set of $G_\phi$ of size $n$ (recall that $G_\phi$ has no dominating set of size less than $n$). Thus $|D \cap \mathcal{G}_i| = 1$ for every $1 \leq i \leq n$. Since $y_i z_i \notin E(G_\phi)$, $D$ cannot contain $y_i$ or $z_i$. So it contains either $v_i$ or $\neg v_i$, but not both. Finally, $\{a_1, a_2, \ldots, a_m\} \cap D = \emptyset$. Hence, one can build an assignment $\nu$ of the variables of $\phi$ such that $\nu(x_i) = \top$ if $D$ contains $v_i$, and $\nu(x_i) = \bot$ if $D$ contains $\neg(v_i)$. Since each vertex corresponding to a clause is dominated by at least one of its neighbors, each clause of $\phi$ is satisfied by one of its literals and the conclusion follows.

**Reduction from Vertex Cover.** This reduction to show the NP-hardness of the DOMINATING SET problem is probably the most famous one. It has been found by Garey and Johnson in their seminal book "Computers and Intractability: a guide to the theory of NP-completeness" [GJ79]. The reduction works as follows. Let $G = (V, E)$ be a graph, and let $(G, k)$ be the instance of VERTEX COVER. We may assume without loss of generality that $G$ has no vertex of degree zero. We construct the corresponding graph $G'$ by first making a copy of $G$. Then, for each edge $uv \in E$, we add a new vertex $z_{uv}$ of degree two, which is adjacent to both $u$ and $v$. So $G'$ has $|V| + |E|$ vertices and $3|E|$ edges. Thus, it can be constructed in linear time.

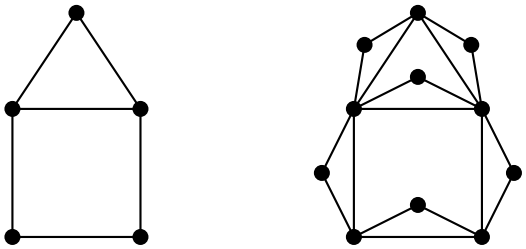


Figure 2.9 – Reduction from VERTEX COVER to DOMINATING SET.

We claim that $G$ has a vertex cover of size $k$ if and only if $G'$ has a dominating set of size $k$. Suppose first that $G$ has a vertex cover $S$ of size at most $k$. We show that $S$ also is a dominating set of $G'$. Since $S$ is a vertex cover, we have $S \cap \{u, v\} \neq \emptyset$ for every edge $uv \in E(G)$. Hence, the vertex $z_{uv}$ is dominated. Since $G$ has no isolated vertex, each vertex $u$ is adjacent to at least one vertex $v$. Since $S$ is a vertex cover, at least one vertex in $\{u, v\}$ belongs to $S$ and thus $u$ is dominated as well. Hence, $S$ is a dominating set of $G'$ of size at most $k$.

Suppose now that $G'$ has a dominating set $D$ of size at most $k$. Since $D$ is a dominating set, each vertex $z_{uv} \in V(G') \setminus V(G)$ that corresponds to the edge $uv \in E(G)$ is dominated, i.e., $N[z_{uv}] \cap D \neq \emptyset$. If $N[z_{uv}] \cap D = \{z_{uv}\}$, one can replace $z_{uv}$ by either $u$ or $v$ and still obtain a dominating set of size at most $k$ by Observation 2.3. Hence, we may assume that $D \subseteq V(G) \cap V(G')$ and thus $D$ is a vertex cover of $G$ of size at most $k$.

**NP-hardness for split and bipartite graphs.** We now show that the DOMINATING SET problem remains NP-complete, even when restricted to bipartite or split graphs. These two reductions were found independently by Chang and Nemhauser [CN84] and Bertossi [Ber84] in 1984. They consist in a polynomial-time reduction from DOMINATING SET in general graphs.

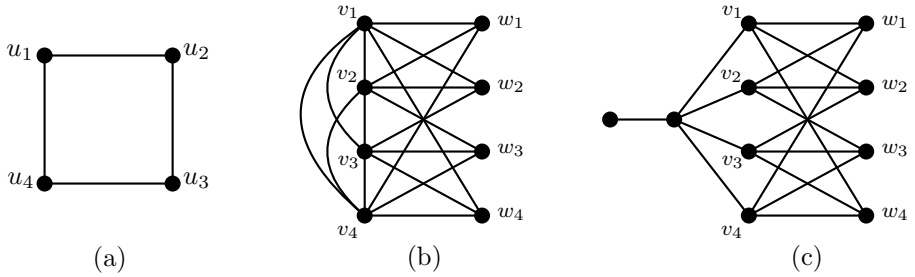



Figure 2.10 – (a) Original graph with (b) corresponding split graph and (c) bipartite graph.

We first explain the construction for split graphs. Let $G = (V, E)$ be a graph with $V = \{u_1, u_2, \ldots, u_n\}$. Then, the corresponding split graph $G'$ is the following:

- $V(G') = V_1 \cup V_2$ where $V_1 = \{v_1, v_2, \ldots, v_n\}$ and $V_2 = \{w_1, w_2, \ldots, w_n\}$, i.e., $V_1$ and $V_2$ are two copies of the vertex set of $G$; and
- $E(G') = \{uv \mid u, v \in V_1\} \cup \{v_i w_j \mid u_j \in N_G[u_i]\}$.

Note that $G'[V_1]$ is a clique, and $G'[V_2]$ is an independent set. Hence, $G'$ is a split graph and it can be constructed in polynomial time. Moreover, a vertex $v_i \in V_1$ dominates $V_1$ and all the vertices of $V_2$ which belong to the closed neighborhood of $u_i$ in $G$. Hence, any dominating set of $G$ can be directly translated into a dominating set of $G'$ of same size by adding the corresponding vertices of $V_1$. Now, let $D'$ be a dominating set of $G'$ of size $k$. By Observation 2.3, we may assume that $D' \subseteq V_1$ as any vertex $w_i \in V_2$ satisfies $N_{G'}[w_i] \subseteq N_{G'}[v_i]$. So each vertex in $V_2$ is dominated by a vertex in $V_1$. Hence, the set $D = \{u_i \mid v_i \in D\}$ is a dominating set of $G$ and the conclusion follows.

Let us now move on to bipartite graphs. The construction is very similar to the one for split graphs. More precisely, we start from the split graph and we first remove all the edges between two vertices in $V_1$ so that $G[V_1]$ becomes independent. Now, we add two new vertices $x$ and $y$. We make $x$ adjacent to all the vertices in $V_1$ and to $y$. So observe that $G'[V_1 \cup \{y\}]$ and $G'[V_2 \cup \{x\}]$ induce two independent sets. We can easily prove that $G$ has a dominating set of size $k$ if and only if $G'$ has a dominating set of size $k + 1$. Indeed, let $D$ be a dominating set of $G$. Then, observe that the set $D' = \{v_i \mid u_i \in D\} \cup \{x\}$ dominates $G'$ and has size $|D| + 1$. For the converse direction, let $D'$ be a dominating set of $G'$. Since $N_{G'}[y] \subseteq N_{G'}[x]$, we may assume that $D'$ does not contain $y$ but $x$. Even if now $N_{G'}[w_i] \subseteq N_{G'}[v_i]$ is not true anymore, observe that if $D'$ contains a vertex $w_i \in V_2$, then $(D' \setminus \{w_i\}) \cup \{v_i\}$ also dominates $G'$. Indeed, recall that $V_1 \subseteq N_{G'}(x)$ and $N_{G'}[w_i] \subseteq V_1 \cup \{w_i\}$. Therefore, since $v_i w_i \in E(G')$, one can replace $w_i$ by $v_i$ and still get a dominating set. So we may assume that $D' \subseteq V_1 \cup \{x\}$ and thus the set $D = \{u_i \mid v_i \in D'\}$ dominates $G$ and has size at most $k$.

**Polynomial-time algorithms for several graph classes**

In this section, we give two well-known algorithms to solve the DOMINATING SET problem in linear time on respectively interval graphs and trees. These algorithms are simple, and we will see in Section 3.2 how they can be generalized to solve the problem on a superclass of both trees and interval, namely the class of dually chordal graphs.

**The case of paths.** Recall that the path on $n$ vertices denoted by $P_n$ is the unique tree of diameter $n - 1$. It is well-known that $\gamma(P_n) = \lceil n/3 \rceil$. This value matches the following lower bound on the domination number of any graph:

**Proposition 2.13.** *Let $G$ be a graph on $n$ vertices with maximum degree $\Delta$. Then, $\gamma(G) \geq \lceil \frac{n}{\Delta(G)+1} \rceil$.*

*Proof.* Let $u$ be any vertex of $G$, and note that $\deg(u) \leq \Delta(G)$. Hence, $|N[u]| \leq \Delta(G) + 1$. Since all the vertices have to be dominated, we obtain the inequality. □

> We also have $\gamma(G) \leq n - \Delta(G)$. Indeed, let $u$ be a vertex such that $\deg(u) = \Delta(G)$. Then, adding $u$ to a dominating set leaves $n - \Delta(G) - 1$ vertices undominated. By adding each of them to the set, we obtain a dominating set of size $n - \Delta(G)$.

Let $v_1, v_2, \ldots, v_n$ be the ordering of the vertices of $P_n$ such that $v_i v_{i+1} \in E(P_n)$ holds for every $1 \leq i < n$. Let us give a dominating set $D$ of $P_n$ of size $\lceil n/3 \rceil$. If $n \equiv 0 \mod 3$ or $n \equiv 2 \mod 3$, we set $D = \{v_i \mid i \equiv 2 \mod 3\}$. Otherwise, if $n \equiv 1 \mod 3$, we set $D = \{v_i \mid i \equiv 2 \mod 3\} \cup \{v_n\}$. In both cases, $D$ can be seen as the "rightmost" dominating set of $P_n$, i.e., moving any vertex to the right leaves a vertex undominated (see Figure 2.11).

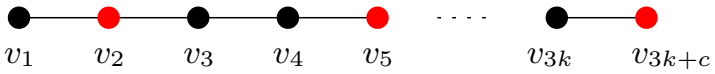



Figure 2.11 – Rightmost minimum dominating set of $P_n$ with $n \not\equiv 0 \mod 3$.

**Interval graphs.** Recall that an interval graph is the intersection graph of intervals on the real line. Let $G = (V, E)$ be an interval graph. By ordering the vertices of $G$ according to their right value, we obtain an ordering $v_1, v_2, \ldots, v_n$ such that for any $i < k$, if $v_i v_k \in E$, then $v_j v_k \in E$ for any $i < j < k$ by Observation 1.4. We define the *rightmost neighbor* of a vertex $v_i \in V$ as the neighbor of $v_i$ with the highest index, i.e., $v_k$ is the rightmost neighbor of $v_i$ if $k = \max\{j \mid v_j \in N[v_i]\}$. Thanks to this property, one can define the "rightmost" dominating set which is the one computed by the following algorithm:

**Algorithm 2** Interval graphs Domination

**Require:** An interval graph $G = (V, E)$.
**Ensure:** The rightmost dominating set of $G$.
1: $D = \emptyset$
2: **for** $i$ from 1 to $n$ **do**
3: **if** $v_i$ is not dominated by $D$ **then**
4: Let $v_k$ be the rightmost neighbor of $v_i$
5: Add $v_k$ to $D$
6: **return** $D$

This algorithm runs in time $O(|V| + |E|)$. Moreover, one can prove that it actually computes a minimum dominating set of $G$. Roughly speaking, the reason why this is true is because at step $i$, all the vertices with index less than $i$ are already dominated by vertices in $\{v_1, v_2, \ldots, v_{i-1}\}$. So it remains to dominate the vertices of the graph $G_i = G[\{v_i, v_{i+1}, \ldots, v_n\}]$. Let $v_j \in N[v_i]$ such that $v_j$ is not the rightmost neighbor of $v_i$ (note that we possibly have $v_i = v_j$ but $v_i$ and $v_j$ are different from $v_k$). Therefore, we have $v_i < v_j < v_k$ assuming that $v_i \neq v_j$. So for any neighbor $v_\ell$ of $v_j$ in $G_i$, $v_\ell v_k \in E$ by Observation 1.4 and thus $N_{G_i}[v_\ell] \subseteq N_{G_i}[v_k]$. Hence, $v_k$ can be seen as an "optimal" vertex to dominate $v_i$ (see Figure 2.12).

**Trees.** Another possible way to see the rightmost dominating set of the path $P_n$ is due to its "tree structure". More precisely, let us denote by $X_i$ the set $\{v_1, v_2, \ldots, v_i\}$. So at step $i$ of Algorithm 2, all the vertices in $X_{i-1}$ are dominated by vertices in $X_{i-1}$. And if $i \equiv 1 \mod 3$,

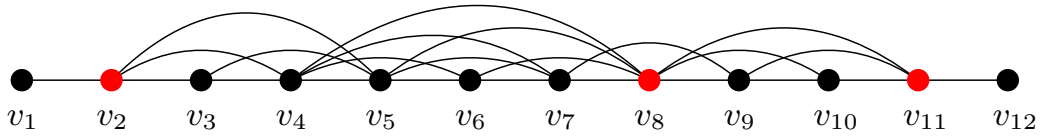



Figure 2.12 – Rightmost dominating set $\{v_2, v_8, v_{11}\}$ of the interval graph showed in Figure 1.12.

$v_i$ is the first vertex not yet dominated. But then, observe that $v_i$ is a leaf in the graph $G_i = G[V \setminus X_{i-1}]$. To dominate $v_i$, the algorithm adds its unique neighbor $v_{i+1}$ to the dominating set since $N_{G_i}[v_i] \subseteq N_{G_i}[v_{i+1}]$. This idea can be generalized to any tree $T$ to compute $\gamma(T)$. This algorithm was found by Cockayne, Goodman and Hedetniemi in 1975 [CGH75] and it works as follows. We first root $T$ on an arbitrary node $r$ and we orient all the edges of $T$ towards the root $r$. This orientation gives a unique *parent* to each node, which corresponds to its unique out-neighbor. Now, we label each node according to a post-order traversal of $T$. This ensures that for every $1 \leq i \leq n$, $v_i$ is a leaf in $T_i$. The algorithm is exactly the same as Algorithm 2 except that instead of adding the rightmost neighbor of $v_i$ to $D$, we add its parent (see Figure 2.13).

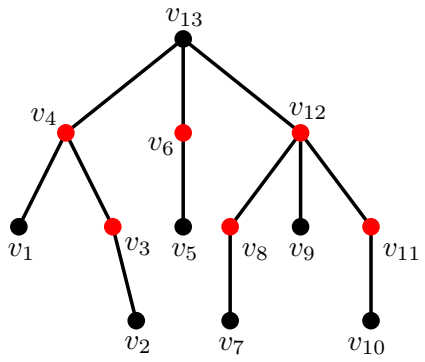



Figure 2.13 – Minimum dominating set $\{v_3, v_4, v_6, v_8, v_{11}, v_{12}\}$ of a tree $T$.

> Actually, there is a common point between the orderings described for interval graphs and trees. Indeed, we ensure that the vertex $v_k$ added to dominate $v_i$ (which is its rightmost neighbor for interval graphs, and its parent for trees) satisfies $N_{G_i}[v_j] \subseteq N_{G_i}[v_k]$ for any neighbor $v_j$ or $v_k$ in $G_i$. We will use this property in Section 3.2.3.

**Parameterized complexity of the problem**

Let us now discuss the parameterized complexity of DOMINATING SET. This part is inspired by the lecture notes by Cyril Gavoille [Gav20].

**Fixed-parameter tractability.** Here, we consider the parameterized complexity of DOMINATING SET with respect to its natural parameter $k$, the size of the desired solution. Downey and Fellows showed that this problem is W[2]-complete [DF95], meaning that there is no hope for an algorithm finding a dominating set of size at most $k$ in time $O(f(k) \cdot |G|^{O(1)})$ for any graph $G$, unless the W-hierarchy collapses. This suggests that it is harder to solve than VERTEX COVER (which is in FPT) or even INDEPENDENT SET which is W[1]-complete [DF95].

What could explain why DOMINATING SET is harder than INDEPENDENT SET is related to the definability of these two properties by first-order logic formulas. Indeed, given a graph $G = (V, E)$ and an integer $k$, the fact that $G$ has an independent set of size $k$ can be expressed by the following first-order logic formula:

$$(\exists v_1, \exists v_2, \ldots, \exists v_k) : \bigwedge_{1 \leq i < j \leq k} v_i v_j \notin E$$

By contrast, if one wants to express that $G$ admits a dominating set of size $k$, one needs to add a universal quantifier, creating two alternating blocks of quantifiers:

$$(\exists v_1, \exists v_2, \ldots, \exists v_k) : (\forall u) \bigvee_{1 \leq i \leq k} (u = v_i \vee u v_i \in E)$$

This universal quantifier over all the vertices of the graph makes the problem "global" because one needs to check that all the vertices are dominated. On the other hand, the first formula for defining an independent set is entirely "local" since only the vertices $v_1, v_2, \ldots, v_k$ are involved (see [DF13] for an extensive discussion on this subject).

**Positive results.** Since the problem is W[2]-complete with respect to its natural parameter, most work has been focused on graph classes for which FPT algorithms could exist. A good candidate is the class of planar graphs since any planar graph is 5-degenerate, meaning that any induced subgraph of a planar graph has a vertex of degree at most five. Indeed, any dominating set $D$ of a graph $G = (V, E)$ satisfies $N[u] \cap D \neq \emptyset$. Therefore, one could try to design an algorithm very similar to the one given in Example 1.13: select of vertex $u$ of degree at most five and then branch on all the different possibilities to dominate $u$. In other words, for any vertex $v \in N[u]$, we add $v$ to the dominating set and recurse on $G - v$ and with parameter $k - 1$. There are (at least) two different possible graphs $G - u$: (i) remove $u$ from $G$ (i.e., $u$ and all the edges incident to $u$); or (ii) remove $N[u]$. However, none of these two ideas seems to work.

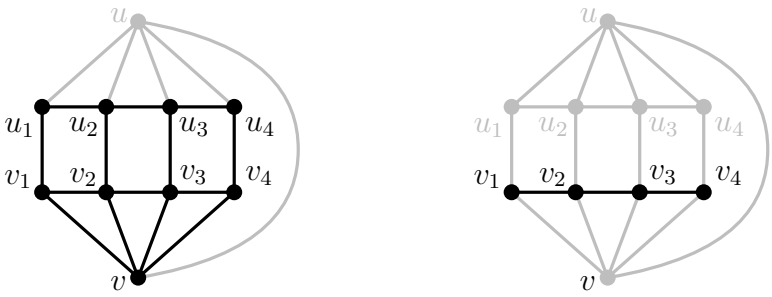



Figure 2.14 – Neither the graph (a) $G - u$ nor (b) $G - N[u]$ has a dominating set of size one.

Hence, one needs to remember that if $u$ is added to the dominating set, then all the vertices in $N[u]$ are dominated. This means that we should solve a slightly different problem, which is actually a generalization of DOMINATING SET:

ANNOTATED DOMINATING SET

**Instance:** A graph $G = (B \uplus W, E)$, an integer $k$.
**Question:** Does $G$ have a subset $D \subseteq B \uplus W$ such that $|D| \leq k$ and $W \subseteq N[D]$?

In other words, in the ANNOTATED DOMINATING SET, we are given a graph where each vertex is either *black* (in $B$) or *white* (in $W$), which is called a *black and white graph*. And the goal is to find a subset of at most $k$ vertices (which may be either black or white) that dominates the black vertices. Hence, if $W = \emptyset$, then ANNOTATED DOMINATING SET is equivalent

to DOMINATING SET. One can see the white vertices as the vertices which are already dominated. Alber et al. [AFF+05] showed that after applying some reduction rules, any black and white planar graph has a black vertex $u$ of degree at most seven. By branching on each vertex in $N[u]$, we obtain an algorithm with running time $O(8^k \cdot n)$ similar to the one for VERTEX COVER (see Example 1.13). Obviously, we have to whiten the vertices which are now dominated in each recursive call. Actually, one can compute a kernel of size $O(k)$ in time $O(n^3)$ (where $n$ is the number of vertices) [AFN04]. By computing this kernel as a preprocessing, we obtain an algorithm that runs in time $O(8^k \cdot k + n^3)$ for ANNOTATED DOMINATING SET in planar graphs. This result was later improved by Fomin and Thilikos [FT06] by using a dynamic programming algorithm to a specific decomposition of the edges of the kernel, yielding an $O(2^{15.13\sqrt{k}} + n^3)$ time algorithm. FPT algorithms have been designed for various graph classes of graph: bounded genus graphs [EFF04], $H$-minor-free graphs [DFHT05], bounded expansion graphs [NDM08] and finally for nowhere dense classes of graph [DK09]. On the other hand, Alon and Gutner [AG08] provided a $k^{O(dk)}n$ time algorithm for finding a dominating set of size at most $k$ on $d$-degenerate graphs with $n$ vertices, showing that the problem is in FPT for $d$-degenerate graphs. However, nowhere dense classes and $d$-degenerate classes of graphs are uncomparable. Telle and Villanger [TV12] generalized the results of [DK09] and [AG08] by showing that DOMINATING SET (and some of its variants) is FPT when parameterized by $k + t$ on graphs that do not contain $K_{t,t}$ as a subgraph.

Drange et al. [DDF+16] also studied the parameterized complexity of DOMINATING SET. They showed that this problem has a linear kernel for every graph class $\mathcal{G}$ of bounded expansion. In the more general case where $\mathcal{G}$ is only assumed to be nowhere dense, then the problem has an "almost" linear kernel. More precisely, they proved the following theorem:

**Theorem 2.14** ([DDF+16]). *For every hereditary graph class $\mathcal{G}$ with bounded expansion,* DOMINATING SET *admits a kernel of size $O(k)$ on graphs from $\mathcal{G}$. For every hereditary and nowhere dense graph class $\mathcal{G}$ and every $\epsilon > 0$,* DOMINATING SET *admits a kernel of size $O(k^{1+\epsilon})$ on graphs from $\mathcal{G}$.*

To prove Theorem 2.14, Drange et al. [DDF+16] introduced the concept of domination core: for a graph $G = (V, E)$, a *domination core* is a subset of vertices $C \subseteq V$ such that any vertex subset $D \subseteq V$ is a dominating set of $G$ if and only if $C \subseteq N[D]$. In other words, it is sufficient to dominate the vertices in $C$ to dominate the whole graph $G$. Note that any $d$-degenerate graph $G$ that admits a dominating set of size at most $k$ has a domination core of size at most $dk^d$, and it can be computed in FPT time with respect to $k + d$ [LMP+18]; we will use it in Section 3.3.3.

## 2.2 Price of Connectivity for domination

### 2.2.1 Introduction

#### Connected dominating sets

A *connected dominating set* of a graph $G = (V, E)$ is a subset of vertices $D \subseteq V$ that is both dominating and connected, i.e., $N[D] = V$ and $G[D]$ is connected. Moreover, it is clear that only connected graphs admit a connected dominating set. We denote by $\gamma_c(G)$ the minimum size of a connected dominating set of $G$.

> The CONNECTED DOMINATING SET problem which asks if the connected domination number of a graph is at most $k$ is NP-complete. To see this, observe that we can use

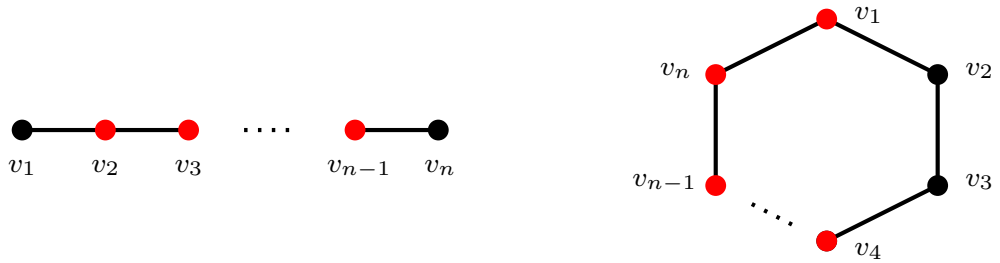
> exactly the same polynomial-time reductions as the ones described in Section 2.1.4 for split and bipartite graphs.

Since a connected dominating set is a dominating set, we obtain $\gamma(G) \leq \gamma_c(G)$. Moreover, we have the following upper bound:

**Proposition 2.15.** *Let $G$ be a graph on $n$ vertices. Then, $\gamma_c(G) \leq n - 2$.*

*Proof.* Let $T$ be a spanning tree of $G$, and let $U$ be the set of leaves of $T$. Then, $V \setminus U$ is a connected dominating set of size at most $n - 2$, as $|U| \geq 2$ holds for any tree. □

Surprisingly, this trivial upper bound turned out to be tight since $\gamma_c(P_n) = \gamma_c(C_n) = n - 2$. Indeed, let $v_1 v_2 \dots v_n$ be an ordering of the vertices of $P_n$ and $C_n$ such that $v_i$ and $v_{i+1}$ are adjacent for any $1 \leq i < n$ (the cycle $C_n$ additionally has the edge $v_1 v_n$). Since $v_1$ and $v_n$ are the two leaves of $P_n$, we can select $v_2$ and $v_{n-1}$ to dominate them. But we are forced to take each intermediate vertex, yielding a set of size $n - 2$. If $G$ is the cycle on $n$ vertices, observe that all the vertices but at most two have to be in a connected dominating set otherwise it is not connected (if the vertices not in the set are not consecutive in the ordering) or there is a vertex which is not dominated (if there is three consecutive vertices which are not in the set).




(a) (b)

Figure 2.15 – (a) Minimum connected dominating set of $P_n$ and (b) of $C_n$.

As we have seen in the proof of Proposition 2.15, connected dominating sets of a graph $G = (V, E)$ are related to the spanning trees of $G$. More precisely, from a connected dominating set $D$ of $G$, one can construct a spanning tree $T$ of $G$ with $|V| - |D|$ leaves. We first take a spanning tree of $G[D]$ and then we attach each vertex in $V \setminus D$ to an arbitrary vertex in $D$ that dominates it. Conversely, if $T$ is a spanning tree of a graph $G$ with at least two vertices, then the internal nodes of $T$ correspond to a connected dominating set of $G$. In particular, we obtain the following:

**Proposition 2.16** (Folklore (see, e.g., [Dou92])**.** *Let $G$ be a graph on $n > 2$ vertices. Let $\epsilon_T$ denote the maximum number of leaves over all spanning trees of $G$. Then, $\gamma_c(G) = n - \epsilon_T$.*

Note that given a spanning tree $T$ of $G$, the set of internal nodes of $T$ is a dominating set of $T$, and thus of $G$ by extension. Hence, $\gamma_c(G) \leq \gamma(T)$. This was first observed by Sampathkumar and Walikar in 1979:

**Proposition 2.17** ([SW79])**.** *Let $G$ be a connected graph, and let $H$ be a connected spanning subgraph of $G$. Then, $\gamma_c(G) \leq \gamma_c(H)$.*

Recall that $\gamma_c(G) \geq \gamma(G)$. However, the ratio cannot be arbitrarily large:

**Proposition 2.18** ([DM82])**.** *For any connected graph $G$, $\gamma_c(G) \leq 3\gamma(G) - 2$.*

*Proof.* Let $D$ be a minimum dominating set of $G$. Let $C_1, C_2, \ldots, C_k$ be the connected components of $G[D]$, and note that $k \leq \gamma(G)$. Observe that for any set $T \subseteq \cup_i\{C_i\}$, the distance between $T$ and $\cup_i\{C_i\} \setminus T$ is at most three. Indeed, let $C_i$ and $C_j$ be two connected component minimizing the distance between $T$ and $\cup_i\{C_i\} \setminus T$. Suppose that the distance between $C_i$ and $C_j$ is at least four, and let $P$ be a shortest path between $C_i$ and $C_j$. Then, observe that the middle vertex of $P$ is not dominated by $D$, a contradiction. So the distance between $C_i$ and $C_j$ is at most three. Hence, $G[D \cup P]$ has at most $k-1$ connected components and $|D \cup P| \leq |D| + 2$. In other words, adding (at most) two vertices is enough to decrease the number of connected components by at least one. Therefore, one can iteratively apply this argument and obtain a connected dominating set $D'$ such that $D \subseteq D'$ and $|D'| \leq |D| + 2k - 2 \leq 3\gamma(G) - 2$. □

### Price of Connectivity for domination

Since for any graph $G$ we have $\gamma(G) \leq \gamma_c(G) < 3\gamma(G)$, a natural question is the following: how can we bound the ratio?

**Definition 2.19** (*PoC*). *Let $G$ be a graph. $PoC(G) = \gamma_c(G)/\gamma(G)$.*

The price of connectivity has been originally introduced by Cardinal and Levy [Lev09, CL10] for the VERTEX COVER problem. It has then been studied for the FEEDBACK VERTEX SET problem by Belmonte et al. [Bv'HKP17]. Finally, the price of connectivity for domination was formally introduced in a work by Camby and Schaudt [CS14] but research on this direction has been done before, as discussed below.

As a corollary of Proposition 2.18, it follows that $PoC(G)$ lies in the interval $[1, 3)$ for any graph $G$. Moreover, Camby and Schaudt [CS14] proved that this upper bound is asymptotically tight. Indeed, recall that $\gamma(P_n) = \gamma(C_n) = \lceil n/3 \rceil$ and $\gamma_c(P_n) = \gamma_c(C_n) = n - 2$. Hence, $\lim_{n\to\infty} \gamma_c(P_n)/\gamma(P_n) = \lim_{n\to\infty} \gamma_c(C_n)/\gamma(C_n) = 3$. More surprisingly, it is also asymptotically sharp in the class of $(P_9, C_9)$-free graphs. Indeed, for any $k$, there exists a graph $G_k$ such (i) $G_k$ is $(P_9, C_9)$-free, (ii) $\gamma(G_k) = k + 1$, and (iii) $\gamma_c(G) = 3k$. The construction of $G_k$ is very simple: we just attach to each vertex of $K_k$ a path on three vertices (see Figure 2.16 below).

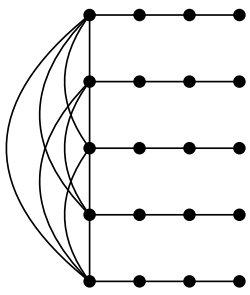


Figure 2.16 – The graph $G_5$.

Recall the following result by Zvervich and Zverovich on domination perfect graphs:

**Theorem 2.20** ([ZZ95]). *There exist seventeen graphs $G_1, G_2, \ldots, G_{17}$ such that the following assertions are equivalent for every graph $G$:*

*(i) $\gamma(H) = i(H)$ for any induced subgraph $H$ of $G$;*

*(ii) $G$ is $(G_1, G_2, \ldots, G_{17})$-free.*

A few years later, Zverovich obtained a similar result regarding connected dominating sets. More precisely, he proved the following:

**Theorem 2.21** ([Zve03]). *For every graph $G$, the following assertions are equivalent:*

*(i)* $\gamma(H) = \gamma_c(H)$ *for any connected induced subgraph $H$ of $G$;*

*(ii)* *$G$ is $(P_5, C_5)$-free.*

A $(P_5, C_5)$-free graph is called a *PoC*-Perfect graph. This result is the starting point of the work initiated by Camby and Schaudt [CS14]. They proved that a graph $G$ is $(P_6, C_6)$-free if and only if $\gamma_c(H) \leq \gamma(H) + 1$ holds for every connected induced subgraph $H$ of $G$. They also showed that this bound is sharp for an infinite family of $(P_6, C_6)$-free graphs with arbitrarily large domination number. This result motivated the study of *PoC*-Near-Perfect graphs:

**Definition 2.22** (*PoC*-Near-Perfect graph). *A graph $G$ is PoC-Near-Perfect with threshold $t$ if and only if $\gamma_c(H) \leq t \times \gamma(H)$ for any connected induced subgraph $H$ of $G$.*

Hence, Camby and Schaudt [CS14] proved that a graph $G$ is *PoC*-Near-Perfect with threshold $3/2$ if and only $G$ is $(P_6, C_6)$-free. They also tried to characterize *PoC*-Near-Perfect graphs with threshold two and proved the following:

**Theorem 2.23** ([CS14]). *Every $(P_8, C_8)$-free graph $G$ satisfies $PoC(G) \leq 2$.*

However, this result is not a characterization of *PoC*-Near-Perfect graphs with threshold two since $\gamma_c(P_8) = \gamma_c(C_8) = 6$ and $\gamma(P_8) = \gamma(C_8) = 3$. So these two graphs do not violate the inequality $\gamma_c(G) \leq 2\gamma(G)$. Actually, the smallest paths and cycles with price of connectivity larger than two are $P_9$ and $C_9$ since their domination number is three but their connected domination number is seven. While intensively searching for minimal connected graphs $G$ with $PoC(G) > 2$, they found the following graph $H$ in addition to $P_9$ and $C_9$:

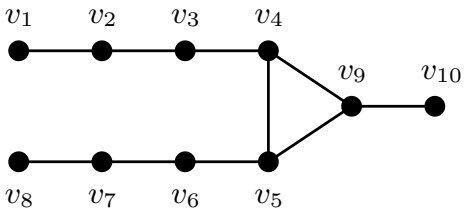



Figure 2.17 – The graph $H$.

Camby and Schaudt [CS14] got "the strong impression that $P_9$, $C_9$ and $H$ might be the only minimal graphs with *PoC* larger than two". They established the following conjecture:

**Conjecture 2.24.** *A graph $G$ is PoC-Near-Perfect with threshold two if and only if $G$ is $(P_9, C_9, H)$-free.*

### 2.2.2 *PoC*-Near-Perfect graphs with threshold two

In the remaining of this section, we present a result from an ongoing joint work with Marthe Bonamy, Nicolas Bousquet and Tereza Klimošová. More precisely, we prove that the Conjecture 2.24 by Camby and Schaudt is true, i.e., we show the following result:

**Theorem 2.25.** *Let $G$ be a graph. Then $PoC(G) \leq 2$ if and only if $G$ is $(P_9, C_9, H)$-free.*

Camby and Schaudt [CS14] observed that for any graph $G \in \{P_9, C_9, H\}$, we have $\gamma_c(G) > 2\gamma(G)$. So we have to exclude these graphs as induced subgraphs to have price of connectivity at most two. However, the main question was to determine whether these three graphs are the only obstructions to get price of connectivity at most two. In the remaining, we prove that the answer is positive, i.e., if a graph $G$ is $(P_9, C_9, H)$-free, then $PoC(G) \leq 2$. We give a proof by contradiction. Note that we have the following useful remark:

**Remark 2.26.** *Let $H$ be the graph depicted in Figure 2.17. The addition of one of the following set of edges to $H$ creates an induced $P_9$ or $C_9$:*

- $v_1v_8$;
- $v_1v_{10}$ *or* $v_8v_{10}$;
- *both* $v_1v_{10}$ *and* $v_8v_{10}$.

*Proof.* If $v_1v_8 \in E$, then $v_{10}v_9v_5v_6v_7v_8v_1v_2v_3$ is an induced $P_9$. If $v_1v_{10} \in E$ (respectively $v_8v_{10} \in E$) then $v_{10}v_1v_2v_3v_4v_5v_6v_7v_8$ (resp. $v_{10}v_8v_7v_6v_5v_4v_3v_2v_1$) in an induced $P_9$. Finally, if we have both $v_1v_{10} \in E$ and $v_8v_{10} \in E$, then $v_1v_2v_3v_4v_5v_6v_7v_8v_{10}v_1$ is an induced $C_9$. □

Let $G = (V, E)$ be a connected $(P_9, C_9, H)$-free graph. We assume for contradiction that $\gamma_c \geq 2\gamma + 1$. Let $D^0$ be a minimum dominating set of $G$ with the fewest number of connected components. Note that $|D^0| = \gamma$ and $N[D^0] = V$. Let us denote by $\mathcal{CC}(D^0)$ the set of connected components of $G[D^0]$ and by $D_1^0, \ldots, D_g^0$ the connected components in $\mathcal{CC}(D^0)$. Therefore, $|\mathcal{CC}(D^0)| = g$. Since $\gamma_c \geq 2\gamma + 1$, we have $g \geq 2$. We emphasize that $G$ does not have a minimum dominating set with strictly less than $g$ connected components. Moreover, as a consequence of the proof of Proposition 2.18, we obtain the following:

**Observation 2.27.** *For any set $T \subseteq \cup_i\{D_i\}$, the distance between $T$ and $\cup_i\{D_i\} \setminus T$ is at most three.*

Let $\ell$ be the largest integer such that there exists a dominating set $D^\ell$ of $G$ that satisfies the following properties:

- $G[D^\ell]$ has $g - \ell$ connected components;
- $|D^\ell| \leq \gamma + \ell$;
- there exists an increasing function $f$ that maps every connected component of $D^\ell$ to a non-empty set of connected components of $D^0$.

More formally, we want a function $f$ satisfying the following properties:

- Codomain: for every $D_j^\ell \in \mathcal{CC}(D^\ell)$, $f(D_j^\ell) \subseteq 2^{\mathcal{CC}(D^0)} \setminus \{\emptyset\}$;
- Injective property: for every $D_i^0 \in \mathcal{CC}(D^0)$, there exists a unique connected component $D_j^\ell$ of $\mathcal{CC}(D^\ell)$ such that $f(D_j^\ell)$ contains $D_i^0$;
- Size constraint: for every $D_j^\ell \in \mathcal{CC}(D^\ell)$, $|D_j^\ell| \leq (\sum_{D_i^0 \in f(D_j^\ell)} |D_i^0|) + |f(D_j^\ell)| - 1$;
- Increasing property: for every $D_j^\ell \in \mathcal{CC}(D^\ell)$, $\bigcup_{D_i^0 \in f(D_j^\ell)} N[D_i^0] \subseteq N[D_j^\ell]$.

Let us now give a less formal definition of the function $f$ for the sake of intuition. For a more algorithmic point of view, we can think about $f$ as follows: if there exist $q \geq 2$ connected components of $D^0$, without loss of generality $D_1^0, \ldots D_q^0$, such that there exists $X \subseteq V$ satisfying the following properties: (i) $G[X]$ is connected, (ii) $|X| \leq (\sum_{i=1}^q |D_i^0|) + (q-1)$, and (iii) $\bigcup_{i=1}^q N[D_i^0] \subseteq N[X]$, then we replace $D_1^0, \ldots D_q^0$ by $X$. We set $f(X) = \bigcup_{i=1}^q D_i^0$. Therefore, to get $D^\ell$, we apply as many times as possible the previous procedure. Note that we decreased by at least one the number of connected components.

> Let $D_i^\ell$ be a connected component of $D^\ell$, and let $D_j^0 \in f(D_i^\ell)$. Note that we do not have necessarily $D_j^0 \subseteq D_i^\ell$. However, $N[w_k] \subseteq N[D_i^\ell]$ must hold, for every $w_k \in D_j^0$.

Among all the dominating sets satisfying the four properties above, we pick one of smallest size, and write for short $D = D^\ell$. Note that we possibly have $\ell = 0$. Again, since $\gamma_c \geq 2\gamma + 1$, we have $\ell \leq g - 2$. Let us denote by $D_1, \ldots, D_{g-\ell}$ the connected components of $G[D]$.

We define the following two useful notions:

**Definition 2.28** (Relevant set). *A set $S \subseteq V \setminus D$ is relevant if $G[D \cup S]$ has at most $g - \ell - |S|$ connected components.*

By maximality of $\ell$, there is no relevant set. In particular, the following holds:

**Observation 2.29.** *For every $1 \leq i \neq j \leq g - \ell$, $N(D_i) \cap N(D_j) = \emptyset$.*

**Definition 2.30** (Semi-relevant set). *A set $S \subseteq V \setminus D$ is semi-relevant if $G[D \cup S]$ has at most $g - \ell - |S| + 1$ connected components.*

By Observations 2.27 and 2.29, every $D_i$ is incident to a semi-relevant set of size two.

**Proposition 2.31.** *Let $S \subseteq V$ be a semi-relevant set. Then, $G[S]$ is connected.*

*Proof.* Assume for the sake of contradiction that $G[S]$ is not connected, and let $S_1, \ldots, S_k$ be the connected components of $G[S]$. Since $G$ does not contain any relevant set, each $S_i$ is (in the best case) a semi-relevant set and thus $G[D \cup S_i]$ has at most $g - \ell - |S_i| + 1$ connected components. Therefore, the total number of connected components of $G[D \cup S] = G[D \cup S_1 \cup \ldots \cup S_k]$ is at most $g - \ell - |S_1| - \ldots - |S_k| + k = g - \ell - |S| + k$ which is larger than $g - \ell - |S| + 1$ since $k$ is at least two, a contradiction. □

**Definition 2.32** (Action of a set). *Given a semi-relevant set $S$, we can define its action $A(S)$ as the set of connected components of $G[D]$ that it is adjacent to.*

Then, we have the following observation:

**Observation 2.33.** *Given a semi-relevant set $S$, every vertex in $S$ is adjacent to a unique connected component in $A(S)$, and conversely.*

*Proof.* First, observe that no vertex in $S$ is adjacent to two (or more) connected component of $A(S)$ otherwise we get a contradiction with the maximality of $\ell$. On the other hand, each vertex in $S$ is adjacent to at least one connected component of $A(S)$. Indeed, if $S$ contains a vertex $v$ without any neighbor in $A(S)$, then $S$ *must* contain a vertex with at least two neighbors in $A(S)$ since adding $S$ to $D$ decreases by at least $|S| - 1$ the number of connected components, which is not possible.

On the other hand, it is clear that every component of $A(S)$ is adjacent to at least one vertex in $S$. Let $w_i$ be a vertex of a connected component $D_i \in A(S)$, and let $v_i$ be a neighbor of $w_i$ in $S$. We claim that $v_i$ is the only neighbor of $w_i$ in $S$, i.e., $N(w_i) \cap S = \{v_i\}$. Suppose that there exists another vertex $v_j \in S$ (with $j \neq i$) such that $w_i v_j \in E$. Then, if $v_j$ is adjacent to another connected component $D_j \in A(S)$, we get a contradiction with the choice of $\ell$. However, since $v_j$ is only adjacent to $D_i$ which is already spanned by $v_i$, we get a contradiction with the fact

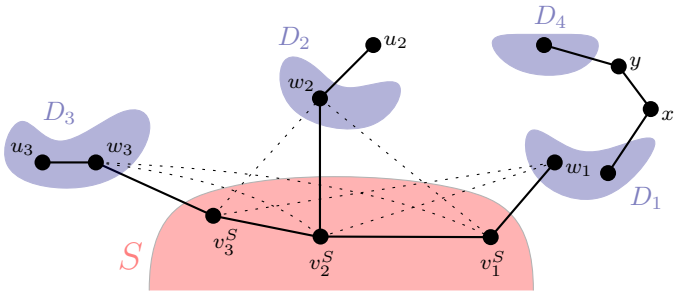



Figure 2.18 – The vertex $x$ is 4-linking for $S$.

that $G[D \cup S]$ has at most $g - \ell + 1$ connected components. As a consequence, every connected component of $A(S)$ is adjacent to exactly one vertex in $S$. The conclusion follows. □

It follows from Observation 2.33 that there exists a bijection between $S$ and $A(S)$. In particular, $|S| = |A(S)|$. As a consequence, we get the following:

**Proposition 2.34.** *For every semi-relevant set $S$, there exists a connected component $D_i$ such that $D_i \notin A(S)$.*

*Proof.* By contradiction. Suppose that there exists a semi-relevant set $S$ with $A(S) = \bigcup_{i=1}^{g-\ell} D_i$. We consider the set $D \cup S$ which is a connected dominating set of $G$ since we assumed that $S$ spans all the connected components of $G[D]$. Besides, $|D \cup S| = |D| + |S| = |D| + |A(S)| \leq \gamma + g - \ell \leq 2\gamma$, which contradicts the fact that $\gamma_c \geq 2\gamma + 1$. □

All along the proof, for every semi-relevant set $S$ and for every $D_i \in A(S)$, we refer to the vertex of $S$ adjacent to $D_i$ as $v_i^S$. We drop the exponent when there is no ambiguity.

**Useful definitions.** A semi-relevant set $S$ is *p-maximal* if $D_p \notin A(S)$ and there is no semi-relevant set $S'$ such that $|S'| \leq |S| + 1$ and $A(S) \cup D_p \subseteq A(S')$.

**Definition 2.35** (Maximal semi-relevant set)**.** *A semi-relevant set $S$ is maximal if its action is inclusion-wise maximal among all actions of semi-relevant sets with one more element. More formally, a semi-relevant set $S$ is maximal if, for every semi-relevant set $S'$ such that $A(S) \subsetneq A(S')$, we have $|S'| \geq |S| + 2$.*

For every semi-relevant set $S$ and for every $D_j \notin A(S)$, a vertex $x \notin S \cup D$ is *j-linking for $S$* if there is $D_i \in A(S)$ such that $x$ has a neighbor in $D_i$ and a neighbor in the neighborhood of $D_j$, i.e. $x$ is at distance two from $D_j$ (see Figure 2.18). For every semi-relevant set $S$, we say that a vertex $x \notin S \cup D$ is *linking for $S$* if there is some $j$ such that $x$ is $j$-linking for $S$. In other words, the vertex $x$ is linking for $S$ if it has a neighbor $y$ such that $\{x, y\}$ is a semi-relevant set with action $A(\{x, y\}) = \{D_i, D_j\}$. We can now rephrase Observation 2.27:

**Observation 2.36.** *Every maximal semi-relevant set $S$ admits a linking vertex for $S$.*

*Proof.* Let $S$ be a maximal semi-relevant set. By Proposition 2.34, $S$ does not span all the connected components of $G[D]$. Among all connected components that do not belong to $A(S)$, at least one of them is at distance exactly three from some $D_i \in A(S)$ by Observation 2.27. Hence, the vertex in this path incident to $D_i$ is a linking vertex for $S$. □

**Lemma 2.37.** *For every semi-relevant set $S$, for every $D_i \in A(S)$, there exists a vertex $u_i \in N[D_i]$ such that $u_i \notin N[S]$ and there exists a vertex $w_i \in D_i \cap N(S)$ incident to $u_i$.*

*Proof.* By contradiction. If $S$ dominates $N[D_i]$ then we consider $D' = (D \cup S) \setminus D_i$. It is clear that $D'$ is a dominating set of $G$. Besides, $|D'| \leq |D| + |S| - 1$ since $|D_i| \geq 1$ and $G[D']$ has at most $g - \ell - |S| + 1$ connected components. Therefore, we get a contradiction with the maximality of $\ell$. □

Note that $w_i$ can only be adjacent to a single vertex $v_i$ of $S$ by Observation 2.33, which is the one incident to $D_i$. All along the proof $w_i$ and $u_i$ are vertices such that $v_i, w_i, u_i$ is an induced $P_3$ that satisfies Lemma 2.37. Similarly, we obtain the following observation:

**Observation 2.38.** *For any $1 \leq i \neq j \leq g - \ell$ and for any two adjacent vertices $x, y$ such that $x \in N(D_i)$ and $y \in N(D_j)$, there is a vertex $u_j \in N[D_j]$ that does not belong to $N(\{x, y\})$ but has a common neighbor $w_j \in D_j$ with $y$.*

*Proof.* This is a direct consequence of Lemma 2.37 since $\{x, y\}$ is a semi-relevant set. □

**Lemma 2.39.** *Let $S$ be a $j$-maximal semi-relevant set and let $x$ be a vertex that is $j$-linking for $S$. Let $p$ be the index such that $D_p \in A(S)$ and $x \in N(D_p)$. We have that $xv_p^S \in E$. Moreover, there is an index $q$ such that $v_q^S$ lies in a different connected component of $G[S \cup \{x\} \setminus \{v_p^S\}]$ than $x$ and some vertex $u_q$ in $N[D_q]$ that does not belong to $N(S \cup \{x, y\})$ (where $y$ is a neighbor of $x$ incident to $D_j$) but has a common neighbor $w_q \in D_q$ with $v_q^S$.*

*Proof.* We first prove that there is an index $q$ such that $v_q^S$ is in a different connected component of $G[(S \cup \{x\}) \setminus \{v_p^S\}]$ than $x$, and that there exists a vertex $u_q$ in $N[D_q]$ at distance two from $v_q^S$ that does not belong to $N(S \cup \{x, y\})$. By contradiction, assume that for every index $q$, either $v_q^S \in (S \cup \{x\}) \setminus \{v_p^S\}$ is in the connected component of $x$, or $N[D_q] \subseteq N(S \cup \{x, y\})$. Let $C$ be the connected component of $x$ in $G[(S \setminus \{v_p^S\}) \cup \{x\}]$. Let $U = C \setminus \{x\}$ and $T = S \setminus U$. For every index $q$ such that $v_q^S \in T$, we apply Lemma 2.37 and obtain a vertex $u_q$. We then note that $S' = (S \setminus T \setminus \{v_p^S\}) \cup \{x, y\} \cup \{u_q \mid v_q^S \in T\}$ is a semi-relevant set that contradicts the $j$-maximality of $S$. Therefore, an index $q$ and a vertex $u_q$ as desired exist.

We then apply Observation 2.38, and get a vertex $u_j$ at distance three from $x$. Note that due to $j$-maximality of $S$, $u_j$ has no neighbor in $S$. It follows that $u_q w_q v_q v_p w_p x y w_j u_j$ is an induced $P_9$ or $C_9$, depending on the presence of the edge $u_q u_j$. □

By Proposition 2.34, $G$ does not contain a semi-relevant set whose action covers all the connected components of $G[D]$. Nevertheless, we show in what follows that starting from any maximal semi-relevant set $S$, there exists a semi-relevant set $S'$ that shares exactly one connected component with $S$ and such that $A(S) \cup A(S') = \bigcup_{i=1}^{g-\ell} D_i = D$. Note that the connectivity of $G[D \cup S \cup S']$ is ensured since $A(S) \cap A(S') \neq \emptyset$.

In the remaining of the proof, given a semi-relevant set $S$ and a connected component $D_i \in A(S)$, we refer to the vertex of $D_i$ adjacent to $v_i^S$ as $w_i$. Furthermore, we apply Lemma 2.37 for every $D_i \in A(S)$ and denote by $u_i$ the vertex in $N[D_i]$ adjacent to $w_i$ and without any neighbor in $S$. Let $D_j$ be a connected component such that $D_j \notin A(S)$. Observe that for any $1 \leq i \leq g - \ell$, the vertex $u_i$ is not adjacent to any vertex $u_j \in N(D_j)$. Indeed otherwise, when $u_i u_j$ is an edge, $u_i$ is $j$-linking for $S$ and thus $u_i v_i^S \in E$ by Lemma 2.39, a contradiction.

**Lemma 2.40.** *Let $S$ be a $j$-maximal semi-relevant set and let $x$ be a vertex that is $j$-linking for $S$. Let $p$ be the index such that $D_p \in A(S)$ and $x \in N(D_p)$. Then, there exists an induced path on four vertices $P$ starting from $v_p^S$ such that $P \cap \{x\} = \emptyset$ and the only edge between a vertex in $P$ and $x$ is $xv_p^S$. Moreover, we have $P \subseteq S \cup A(S)$ or the three first vertices of $P$ are in $S \cup A(S)$ and the last one is a vertex $u_i$.*

*Proof.* First, observe that $xv_p^S \in E$ by Lemma 2.39 since $x$ is $j$-linking for $S$. Assume by contradiction that $P$ does not exist. So for any neighbor $v_i^S \in S$ of $v_p^S$, we have $xv_i^S \in E$ or $xu_i \in E$ (or both) otherwise $v_p^S v_i^S w_i u_i$ would be an induced $P_4$ as desired. Let $y$ be a neighbor of $x$ such that $y \in N(D_r)$. If $N_S(v_p^S) \subseteq N(x)$, then $S' = (S \setminus \{v_p^S\}) \cup \{x, y\}$ is a semi-relevant set that contradicts the $r$-maximality of $S$. We denote by $N(v_p^S, \bar{x})$ the neighbors of $v_p^S$ in $S$ that are not adjacent to $x$, i.e., $N(v_p^S, \bar{x}) = \{v_i^S \mid v_i^S \in N_S(v_p^S) \text{ and } xv_i^S \notin E\}$. Observe that $N(v_p^S, \bar{x}) \neq \emptyset$. We let $U = \{u_i \mid v_i^S \in N(v_p^S, \bar{x})\}$ and note that $U \subseteq N(x)$. Let $S^* = (S \setminus (N(v_p^S, \bar{x}) \cup \{v_p^S\})) \cup U \cup \{x, y\}$ and note that $|S^*| = |S| + 1$ since $|N(v_p^S, \bar{x})| = |U|$. If $G[S^*]$ is connected, we are done because $S^*$ is a semi-relevant set that contradicts the $r$-maximality of $S$. Hence, one can assume that $G[S^*]$ is not connected. Let $C$ be the connected component of $x$ in $G[S^*]$ and let $v$ be a vertex in $S^* \setminus C$. Recall that $S^* \setminus S = U \cup \{x, y\}$ and $U \cup \{y\} \subseteq N(x)$. It follows that $v \notin U \cup \{x, y\}$ and thus $v \in S$. Therefore, we refer to $v$ as $v_j^S$, with $j \neq i$. Besides, observe that each vertex in $N_S(v_p^S) \cap S^*$ belongs to $C$. Since $G[S]$ is connected, let us consider a shortest path $P'$ between $v_p^S$ and $v_j^S$ in $G[S]$ and note that the length of $P'$ is at least two since $v_p^S v_j^S \notin E$. Since $v_j^S \notin C$, no vertex of $P' \setminus \{v_p^S\}$ is adjacent to $x$. If $|P' \cap S| \geq 4$, we set $P$ to be the subpath of $P'$ of length three starting in $v_p^S$, and observe that $P \subseteq S$, as desired. Otherwise, $|P' \cap S| = 3$ and we set $P$ to be the concatenation of $P'$ with $w_j$. In that case, $P \subseteq S \cup A(S)$, and the conclusion follows. □

We are now ready to prove the following lemma:

**Lemma 2.41.** *For every maximal semi-relevant set $S$, there is a semi-relevant set $S'$ that satisfies the following two properties:*

*(i)* $A(S) \cup A(S') = \bigcup_{i=1}^{g-\ell} D_i$;

*(ii)* $|A(S) \cap A(S')| = 1$.

*Proof.* Let $S$ be a maximal semi-relevant set. We consider a semi-relevant set $S'$ of maximal size under the condition that $|A(S) \cap A(S')| = 1$. Note that $|A(S')| \geq 2$ since $S$ admits a linking vertex by Observation 2.36. Moreover, since $S'$ is a semi-relevant set, we have $|S'| = |A(S')|$ by Observation 2.33 and thus $|S'| \geq 2$. If Property (i) is satisfied, the conclusion holds. We can therefore assume that $A(S) \cup A(S') \subsetneq \bigcup_{i=1}^{g-\ell} D_i$. We assume without loss of generality that $A(S) \cap A(S') = \{D_1\}$, and note that $v_1^S v_1^{S'} \in E$ by Lemma 2.39. We emphasize that the size of $S'$ is maximal under $|A(S) \cap A(S')| = 1$, and not $A(S) \cap A(S') = \{D_1\}$. In other words, there is no semi-relevant set $S''$ such that $|S''| > |S'|$ and $|A(S) \cap A(S'')| = 1$.

By Observation 2.27, there is an index $p$ with $D_p \in A(S) \cup A(S')$ such that $S$ or $S'$ (or both) has a linking vertex $x$ adjacent to $D_p$. The vertex $x$ has a neighbor $y$ adjacent to $D_r \notin A(S) \cup A(S')$. Observe that $\{x, y\}$ is a semi-relevant set. Hence, $xyw_r u_r$ is an induced $P_4$ by Lemma 2.37. Note that due to the maximality of $\ell$, $S$ is $r$-maximal. Moreover, we claim that $S'$ is $r$-maximal. Let $S''$ be a semi-relevant set such that $|S''| \leq |S'| + 1$ and $A(S') \cup \{D_r\} \subseteq A(S'')$. So we have $|S'| + 1 \geq |S''| = |A(S'')| \geq |A(S')| + 1$. Since there is no relevant set, we actually have $|A(S'')| = |A(S')| + 1$ and thus $A(S'') = A(S') \cup \{D_r\}$. So $|A(S'') \cap A(S)| = |(A(S') \cup \{D_r\}) \cap A(S)| = |A(S') \cap A(S)| = 1$ since $A(S')$ et $\{D_r\}$ are disjoint. So $S''$ is a semi-relevant

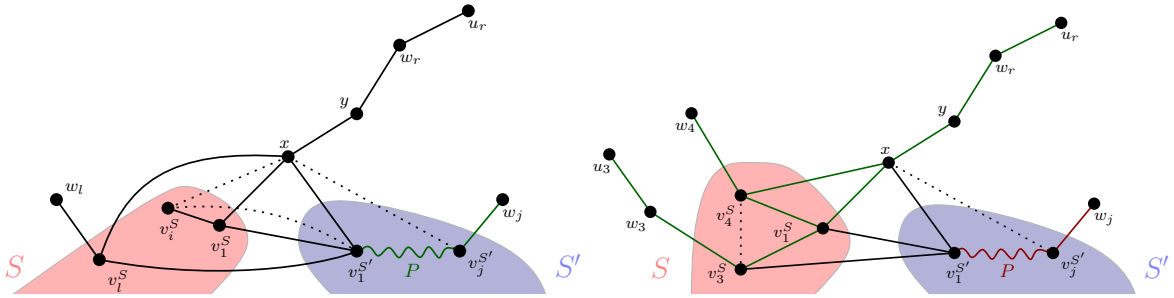



Figure 2.19 – Illustration for the proof of Case 1 of Lemma 2.41

set that satisfies $|A(S) \cap A(S'')| = 1$ and that contradicts the maximality of $S'$. Hence, $S'$ is $r$-maximal. We consider the following three different cases depending on whether $x$ is linking for $S$, $S'$ or both.

**Case 1.** If $p = 1$, i.e., if $x$ is $r$-linking for both $S$ and $S'$ (this case is illustrated on Figure 2.19a).

By Lemma 2.39, we have $xv_1^S \in E$ since $S$ is $r$-maximal and $x$ is $r$-linking. Similarly, $xv_1^{S'} \in E$. Moreover, since $S$ (respectively $S'$) is $r$-maximal, $w_r$, $u_r$ and $y$ have no neighbor in $S$ (resp. $S'$). By Lemma 2.40, there exists an induced path $P$ on four vertices, starting from $v_1^{S'}$ such that $v_1^{S'}$ is the only vertex of $P$ adjacent to $x$. Moreover, this path is in $G[S' \cup A(S')]$. Let us denote $P = v_1^{S'} v_j^{S'} w_j u_j$ in the remaining of the proof of Case 1.

Suppose first that $v_1^S$ has a neighbor $v_i^S \in S$ which is neither adjacent to $x$ nor to $v_1^{S'}$. Then $u_r w_r y x v_1^{S'} v_j^{S'} w_j u_j$ is an induced $H$ with antenna $v_1^S v_i^S$. On the other hand, if $v_1^S$ has a neighbor in $S$, let us say $v_l^S$, which is adjacent to both $x$ and $v_1^{S'}$, we also get an induced $H$ but with antenna $v_l^S w_l$. See Figure 2.19a for an illustration of these two cases where $P \subseteq S' \cup A(S')$ since $P$ ends with $w_j \in D_j$. Therefore, the neighborhood of $v_1^S$ in $S$ can be partitioned into two subsets:

- $N_x$, the neighbors of $v_1^S$ (in $S$) adjacent only to $x$ but not $v_1^{S'}$, i.e., $N_x = \{v_i^S \mid v_i^S \in N_S(v_1^S) \cap N_S(x) \text{ and } v_i^S v_1^{S'} \notin E\}$;
- $N_1$, the neighbors of $v_1^S$ (in $S$) adjacent only to $v_1^{S'}$ but not $x$, i.e., $N_1 = \{v_i^S \mid v_i^S \in N_S(v_1^S) \cap N_S(v_1^{S'}) \text{ and } xv_i^S \notin E\}$.

Observe that $N_x$ (resp. $N_1$) is not empty otherwise $(S \setminus \{v_1^S\}) \cup \{x, y\}$ (resp. $(S \setminus \{v_1^S\}) \cup \{v_1^{S'}, v_i^{S'}\}$) would be a semi-relevant set that contradicts the maximality of $S$. Besides, we claim that $N_x$ and $N_1$ are complete to each other, meaning that for any pair $u, v$ of vertices with $u \in N_x$ and $v \in N_1$, we have $uv \in E$. Assume by contradiction that there exist two vertices $v_3^S \in N_1$ and $v_4^S \in N_x$ such that $v_3^S v_4^S$ is not an edge. By Lemma 2.39, there exists an index $q$ such that $v_q^S$ is not in the connected component that contains $x$ in $G[(S \setminus \{v_1^S\}) \cup \{x\}]$ and a vertex $u_q$ at distance two from $v_q^S$ with no neighbor in $S \cup \{x, y\}$. We claim that $q = 3$, i.e., $v_q^S$ is a neighbor of $v_1^S$. Indeed, consider a shortest path $P$ (thus of length at least two) in $G[S]$ between $v_q^S$ and $v_1^S$. Since $v_q^S$ is not in the connected component of $x$ in $G[(S \setminus \{v_1^S\}) \cup \{x, y\}]$ and $xv_4^S \in E$, no vertex of $P$ is adjacent to $v_4^S$ or $x$. Hence, $w_q P x y w_r u_r$ is an induced $H$ with antenna $v_4^S w_4$, a contradiction.

So $v_q^S = v_3^S$. But then $u_3 w_3 v_3^S v_1^S x y w_r u_r$ is an induced $H$ with antenna $v_4^S w_4$, a contradiction (see Figure 2.19b).

As a result, since $N_x$ is not empty, the set $S^* = (S \setminus \{v_1^S\}) \cup \{x, y\}$ is a semi-relevant set that contradicts the maximality of $S$.

**Case 2.** If $x$ is $r$-linking but only for $S'$ and not $S$.

First, recall that $D_p \in A(S') \setminus A(S)$ and $x \in N(D_p)$. Since $S$ is maximal, there is no edge between a vertex in $S$ and $x$ (otherwise the set $S \cup \{x\}$ would contradict the $p$-maximality of $S$). Let us consider a shortest path $P$ in $G[S' \cup \{x, v_1^S\}]$ between $x$ and $v_1^S$. Note that this path exists since we have $xv_p^{S'} \in E$, $G[S']$ is connected and $v_1^S v_1^{S'} \in E$ by Lemma 2.39. Note that the length of $P$ is at least two as $xv_1^S \notin E$. Moreover, note that since $P$ is a (shortest) path of $G[S' \cup \{x, v_1^S\}]$ that ends in $v_1^S$, $v_1^S$ is the only vertex of $S$ that belongs to $P$. Moreover, the only possible edge between $v_1^S$ and a vertex in $S'$ is $v_1^S v_1^{S'}$ by maximality of $S$. So the neighbor of $v_1^S$ in $P$ is $v_1^{S'}$. By Lemma 2.40, there exists an induced path $P'$ on four vertices, starting from $v_1^S$ and such that $v_1^S$ is the only vertex of $P'$ adjacent to a vertex in $S'$, which is $v_1^{S'}$. Moreover, we have $|P' \cap S| \in \{3, 4\}$. If $P' \subseteq S$, we consider $P'Pyw_r u_r$ which is an induced $P_9$. Otherwise, $w_i P' Pyw_r u_r$ is an induced $P_9$ (see Figure 2.20).

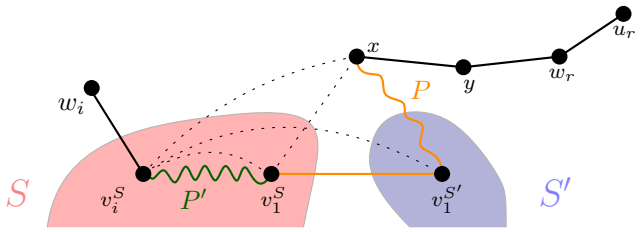



Figure 2.20 – Illustration for the proof of Case 2.

**Case 3.** If $x$ is $r$-linking but only for $S$ and not $S'$ (and thus $p \neq 1$).

We claim that there exists a path $P$ on four vertices starting from $v_1^{S'}$ such that $v_1^{S'}$ is the only vertex of $P$ adjacent to $x$ (if $xv_1^{S'} \in E$). Since $x$ is not $r$-linking for $S'$ in this case, we cannot directly apply Lemma 2.40. However, note that the proof is very similar. Suppose that $P$ does not exist. As in the proof of Lemma 2.40, we denote by $N(v_1^{S'}, \bar{x})$ the neighbors (in $S'$) of $v_1^{S'}$ which are not adjacent to $x$, i.e., $N(v_1^{S'}, \bar{x}) = N_{S'}(v_1^{S'}) \setminus N(x)$. First, note that $N(v_1^{S'}, \bar{x})$ is not empty, otherwise $S'' = (S' \setminus \{v_1^{S'}\}) \cup \{x, y\}$ would be a semi-relevant set that satisfies Property (ii) since $A(S'') = (A(S') \setminus \{D_1\}) \cup \{D_p, D_r\}$ and that contradicts the fact that the size of $S'$ is maximal under $|A(S') \cap A(S)| = 1$. Let $U = \{u_i \mid v_i^{S'} \in N(v_1^{S'}, \bar{x})\}$ and note that $xu_i \in E$. Indeed, otherwise $v_1^{S'} v_i^{S'} w_i u_i$ would be an induced $P_4$ since $v_i^{S'}$ is a neighbor of $v_1^{S'}$ not adjacent to $x$ by definition of $U$. We then consider $S^* = (S' \setminus (N(v_1^{S'}, \bar{x}) \cup \{v_1^{S'}\})) \cup U \cup \{x, y\}$ and note that $|S^*| = |S'| + 1$. If $G[S^*]$ is connected, $S^*$ is a semi-relevant set that satisfies Property (ii) since $A(S^*) = (A(S') \setminus D_1) \cup \{D_p, D_r\}$ that contradicts the fact that the size of $S'$ is maximal under $|A(S) \cap A(S^*)| = 1$. Therefore, $G[S^*]$ is not connected. So there exists a vertex $v_j^{S'}$ which is not in the connected component of $x$ in $G[S^*]$, and we obtain the desired induced $P_4$ similarly as in the proof of Lemma 2.40. Note that every vertex in $P$ belongs to $S' \cup D_j$ with $D_j \in A(S')$. Therefore, the only vertex of $P$ which can be adjacent to a vertex in $S$ is $v_1^{S'}$. We assume moreover that $P$ ends with a vertex $w_j \in D_j$ with $D_j \in A(S')$. Let $v_2^{S'}$ be the neighbor of $v_1^{S'}$ in $P$. Hence, $v_1^{S'}$ is 2-linking for $S$ and $P = v_1^{S'} v_2^{S'} v_j^{S'} w_j$.

*Claim* 1. $xv_1^S$ is not an edge.

*Proof.* Assume by contradiction that $xv_1^S \in E$. If $xv_1^{S'}$ is not an edge, then $u_r w_r yx v_1^S v_1^{S'} v_2^{S'} v_j^{S'} w_j$ is an induced $P_9$ or $C_9$. Hence, one can assume that $xv_1^{S'} \in E$. It follows that $v_1^S x v_1^{S'}$ is a triangle. Let $v_i^S \in N_S(v_1^S)$. If $v_i^S$ is not adjacent to $x$ and $v_1^{S'}$, then $u_r w_r yx v_1^{S'} v_2^{S'} w_2 u_2$ is an induced $H$ with antenna $v_1^S v_i^S$. On the other hand, if $v_i^S$ is adjacent to both $v_1^{S'}$ and $x$, then $u_r w_r yx v_1^{S'} v_2^{S'} v_j^{S'} w_j$ is

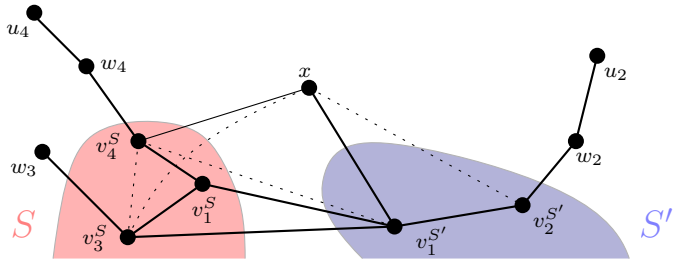


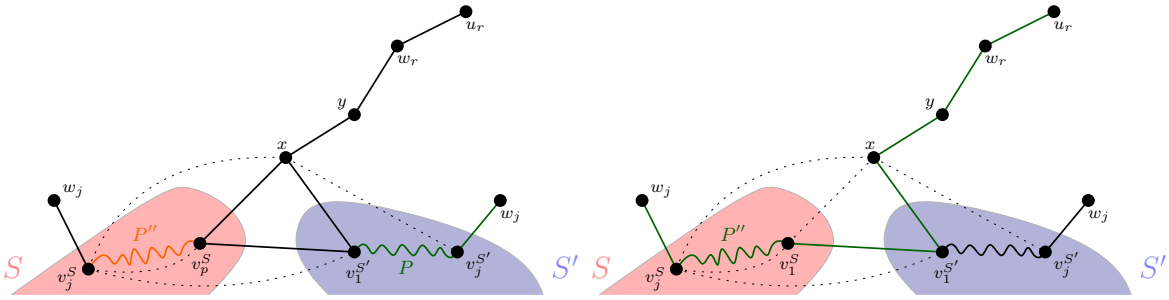

Figure 2.21 – Case 3 of Lemma 2.41: $q = 4$.

an induced $H$ with antenna $v_i^S w_i$. So as in the proof of Case 1, we partition the neighborhood of $v_1^S$ in $S$ into two subsets. We use the same notation and denote by $N_x$ (resp. $N_1$) the neighbors of $v_1^S$ in $S$ which are only adjacent to $x$ (resp. $v_1^{S'}$) and not $v_1^{S'}$ (resp. $x$). We claim that $N_x$ and $N_1$ are complete to each other. Indeed, suppose that there exist $v_3^S \in N_1$ and $v_4^S \in N_x$ such that $v_3^S v_4^S \notin E$. By Lemma 2.39, there exists an index $q$ such that $v_q^S$ lies in a different connected component of $G[(S \setminus \{v_1^S\}) \cup \{v_1^{S'}\}]$ than $v_1^{S'}$ and a vertex $u_q$ at distance two from $v_q^S$ with no neighbor in $S \cup \{v_1^{S'}, v_2^{S'}\}$. Note that $q \neq 3$ since $v_3^S v_1^{S'} \in E$. If $q = 4$, then we are done since $u_4 w_4 v_4^S v_1^S v_1^{S'} v_2^{S'} w_2 u_2$ is an induced $H$ with antenna $v_3^S w_3$ (see Figure 2.21). So one can assume that $q \neq 4$. We claim that $v_q^S$ is at distance at least two from $v_1^S$. Indeed, suppose that $v_1^S v_q^S \in E$. Since $v_q^S$ is not in the connected component of $v_1^{S'}$ (and thus $v_3^S$), we have $v_q^S v_3^S \in E$ and thus this case is the same as $q = 4$. So $v_1^S v_q^S \geq 2$. Let us consider a shortest path $P'$ between $v_q^S$ and $v_1^S$ in $G[S]$. Since $v_q^S$ is not in the connected component of $v_1^{S'}$, there is no edge between $v_1^{S'}$ and a vertex in $P \setminus \{v_1^S\}$. Due to the maximality of $S$, no vertex of $P$ is adjacent to $v_2^{S'}$ or $v_j^{S'}$. So $u_q w_q P v_1^{S'} v_2^{S'} v_1^{S'} w_j$ is an induced $P_9$ ($u_q w_j \notin E$ by maximality of $\ell$), a contradiction. ◊

First assume that $x v_1^{S'}$ is not an edge. Then, consider a shortest path $P$ between $x$ and $v_1^{S'}$ in $G[S \cup \{x, v_1^{S'}\}]$. Hence, the only vertices in $P$ but not in $S$ are precisely $x$ and $v_1^{S'}$. Note that $P$ exists since $x$ is adjacent to $v_p^S \in S$, $v_1^S v_1^{S'} \in E$ and $G[S]$ is connected. Moreover, the length of $P$ is at least two since $x v_1^{S'} \notin E$. So $u_r w_r y P v_2^{S'} v_1^{S'} w_j$ is an induced $P_9$, a contradiction. Hence, in the remaining of this proof, we assume that $x v_1^{S'} \in E$.

We claim that there exists a path $P'$ on four vertices, starting from a vertex $v_i^S \in \{v_1^S, v_p^S\}$ and such that $v_i^S$ is the only vertex of $P'$ adjacent to $\{x, v_1^{S'}\}$. Here again, the proof is very similar to the one of Lemma 2.40. However, let us briefly explain it. Assume by contradiction that the statement does not hold. Observe that this implies that for each vertex $v_j^S \in N_S(v_1^S) \cup N_S(v_p^S)$, either $v_j^S$ has a neighbor in $\{x, v_1^{S'}\}$ or $u_j$ does. Let $T = \{v_j^S \mid v_j^S \in N_S(v_1^S) \cup N_S(v_p^S)$ and $x v_j^S \notin E\}$, and $U = \{u_j \mid v_j^S \in T\}$. We then consider the set $S^* = (S \setminus (\{v_p^S, v_1^S\} \cup T)) \cup \{x, y, v_1^{S'}\} \cup U$. If $G[S^*]$ is connected, we get a contradiction with the maximality of $S$ since $|S^*| = |S| + 1$ and $A(S) \subsetneq A(S^*)$. So $G[S^*]$ is not connected. Note that since $x v_1^{S'} \in E$, $x$ and $v_1^{S'}$ are in the same connected component of $G[S^*]$, let us say $C$. Let $v$ be a vertex of $S^*$ which is not in $C$. Recall that $S^* \setminus S = U \cup \{x, y, v_1^{S'}\}$ and $U \subseteq N(x) \cup N(v_1^{S'})$. Therefore, we have $v \in S$ and we denote $v$ by $v_j^S$. Moreover, $(N_S(v_p^S) \cup N_S(v_1^S)) \cap S^* \subseteq N(x) \cup N(v_1^{S'})$ so $v_j^S$ is at distance at least two from $v_1^S$ and $v_p^S$. Any path $P''$ from $v_j^S$ to $v_p^S$ and $v_1^S$ avoids both $x$ and $v_1^{S'}$. And there is no edge between a vertex of $P''$ and a vertex in $\{x, v_1^{S'}\}$ since $v_j^S$ is not in $C$. Hence, the concatenation of $w_j$ with $P''$ gives us $P'$, as desired. We consider different cases depending on the value of $i$:

- if $v_i^S = v_p^S$, i.e., if the starting point of $P'$ is $v_p^S$. If $v_p^S v_1^{S'} \in E$, $P'P$ is an induced $H$ with antenna $xy$. Otherwise, $P'xP$ is an induced $P_9$. See Figure 2.22a.
- if $v_i^S = v_1^S$, i.e., the starting point of $P'$ is $v_1^S$. Recall that by Claim 1, $xv_1^S$ is not an edge. Therefore, $u_r w_r y x v_1^S P'$ is an induced $P_9$. See Figure 2.22b.




(a) (b)

Figure 2.22 – Illustration for the proof of Case 3 of Lemma 2.41.

So in all the cases we obtain a contradiction. This concludes the proof of Lemma 2.41. □

Let $S$ be a maximal semi-relevant set. By Lemma 2.41, there exists a semi-relevant set $S'$ such that $A(S) \cup A(S') = \cup_i \{D_i\} = D$ and $|A(S) \cap A(S')| = 1$. We then consider $D' = D \cup S \cup S'$ which is a connected dominating set of $G$ since it contains $D$ and each connected component of $G[D]$ has a neighbor in $S \cup S'$. Moreover, we have:

$$\begin{aligned} |D'| &= |D| + |A(S)| + |A(S')| \\ &= |D| + |A(S) \cup A(S')| + 1 && \text{By Property (ii)} \\ &= |D| + g - \ell + 1 && \text{By Property (i)} \\ &\leq \gamma + \ell + g - \ell + 1 \\ &= \gamma + g + 1 \end{aligned}$$

Therefore, it follows that $g = \gamma$ otherwise we get that $\gamma_c \leq 2\gamma$, a contradiction. In other words, $|D_i^0| = 1$ for every $1 \leq i \leq g$. For every $i$, we denote by $x_i$ the unique vertex in $D_i^0$. In the remaining of the proof, we assume without loss of generality that $A(S) \cap A(S') = \{D_1\}$.

**Lemma 2.42.** *We have* $|D_1| = 1$.

*Proof.* Suppose that $|D_1| > 1$ and let $k_1 = |f(D_1)|$. Since $1 < |D_1| \leq 2k_1 - 1$, we get that $k_1 \geq 2$. Besides, it follows from the definition of $D_1$ that $|D_1| = 2k_1 - 1$ otherwise $D \cup S \cup S'$ would be a connected dominating set of size at most $2\gamma$. Indeed, suppose that $|D_1| < 2k_1 - 1$ and let $k_i = |f(D_i)|$, for every $2 \leq i \leq g - \ell$. Note that $\sum_{i=1}^{g-\ell} k_i = \gamma$. Then, $|D| = \sum_{i=1}^{g-\ell} |D_i| < 2\gamma - (g - \ell)$ and thus $|D \cup S \cup S'| < 2\gamma - g + \ell + g - \ell + 1 \leq 2\gamma$, a contradiction.

In the remaining of this proof, we denote $k = k_1$ for short and we consider $f(D_1) = \{x_1, \ldots, x_k\}$. Let $X = (D \setminus D_1) \cup S \cup S'$. We first recall that:

- $D^0$ is a minimum dominating set;
- $D^0$ induces an independent set;

- $N[x_i] \subseteq N[D_1]$ for every $x_i \in f(D_1)$ due to the increasing property of $f$.

Recall also that $|S'| \geq 2$. Therefore, there exists $v_2^{S'} \in N_{S'}(v_1^{S'})$ such that $v_1^S v_2^{S'} \notin E$ since $S$ is a maximal semi-relevant set. Moreover, $v_1^S v_1^{S'} \in E$ by Lemma 2.39 since $v_1^{S'}$ is 2-linking for $S$. By Lemma 2.40, there exists a path $P$ on four vertices starting from $v_1^S$ such that $v_1^S$ is the only vertex of $P$ adjacent to $v_1^{S'}$. Moreover, we have either $P \subseteq S \cup A(S)$, or the first three vertices of $P$ are in $S \cup A(S)$ and the last one is a vertex $u_i$, with $D_i \in A(S)$. In the remaining of this proof, we assume that $P$ ends with $u_3 \in N[D_3]$, with $D_3 \in A(S)$. We claim that the concatenation of $P$ with $v_1^{S'} v_2^{S'} w_2 u_2$ is an induced $P_8$. By maximality of $S$, $v_2^{S'}$ has no neighbor in $P \cap S$. Moreover, $v_2^{S'}$ has no neighbor in $D_3$ by maximality of $\ell$, and neither $v_2^{S'} u_3 \in E$ nor $u_2 v_3^{S'} \in E$ (otherwise $u_3$ would be 2-linking for $S$ and we would have $u_3 v_3^S \in E$ by Lemma 2.39, a contradiction). By maximality of $S$ and $\ell$, $w_2$ (resp. $u_2$) has no neighbor in $S \cup N[A(S)]$ (resp. $S \cup A(S)$).

*Claim* 1. For every $1 \leq i \leq k$, there exist $t_i \in N[X]$ and $r_i \in N[t_i]$ such that $N[x_i] \subseteq N[X \cup \{t_i, r_i\}]$.

*Proof.* We consider several cases depending on the distance from $x_i$ to $X$. First, observe that if $x_i \in X$, one can pick $t_i = r_i = x_i$. If $\text{dist}(x_i, X) = 1$, we choose $t_i$ to be a neighbor of $x_i$ in $X$ and $r_i$ to be $x_i$. If $\text{dist}(x_i, X) = 2$, we let $r_i = x_i$, and $t_i$ to be a neighbor of $x_i$ adjacent to a vertex in $X$. Since in all cases $\{r_i, t_i\}$ contains $x_i$, we have $N[x_i] \subseteq N[X \cup \{r_i, t_i\}]$ and the conclusion follows. So one can now assume that $\text{dist}(x_i, X) \geq 3$.

Recall that since $x_i \in f(D_1)$, $x_i \in N[D_1]$. Recall also that both $v_1^S$ and $v_1^{S'}$ have a neighbor in $D_1$ (which might be the same). Since $G[D_1]$ is connected, let us consider a path $P'$ in $G[D_1 \cup \{x_i, v_1^S, v_1^{S'}\}]$ between $x_i$ and the vertex in $\{v_1^S, v_1^{S'}\}$ that minimizes the distance. In other words, if $\text{dist}(x_i, v_1^S) \leq \text{dist}(x_i, v_1^{S'})$, then $P'$ is between $x_i$ and $v_1^S$. Otherwise, $P'$ is between $x_i$ and $v_1^{S'}$. Note that $P'$ must be a shortest path, and the length of $P'$ is at least three. Let $P' = a_1 a_2 \cdots a_{k-1} a_k$ (with $k \geq 4$) where $a_1 = x_i$, $\{a_2, a_3, \ldots a_{k-1}\} \subseteq D_1$, and $a_k \in \{v_1^S, v_1^{S'}\}$. Then, we obtain the following:

- neither $v_1^S$ nor $v_1^{S'}$ has a neighbor in $\{a_1, a_2, \ldots, a_{k-2}\}$ since $P'$ minimizes the distance between $x_i$ and $\{v_1^S, v_1^{S'}\}$;
- since $\text{dist}(a_1, X) \geq 3$, $a_1$ has no neighbor in $\{u_3, w_3, v_3^S, v_1^S, v_1^{S'}, v_2^{S'}, w_2, u_2\}$;
- since $w_2 \in D_2$, $w_3 \in D_3$ and $\{a_2, a_3, \ldots, a_{k-1}\} \subseteq D_1$, neither $w_2$ nor $w_3$ has a neighbor in the set $\{a_2, a_3, \ldots, a_{k-1}\}$;
- since $\{a_2, a_3, \ldots, a_{k-1}\} \subseteq D_1$, $P \setminus \{w_3\} \subseteq S \cup \{u_3\}$ and $\ell$ is maximal, there is no edge between a vertex in $(P \setminus \{w_3\}) \cup \{v_2^{S'}, u_2\}$ and a vertex in $\{a_2, a_3 \ldots, a_{k-1}\}$.

Suppose first that the length of $P'$ is at least four. We then consider two cases depending on whether $a_{k-1}$ is adjacent to both $v_1^S$ and $v_1^{S'}$, or only one of them. We first consider the case where $a_{k-1}$ is adjacent to both $v_1^S$ and $v_1^{S'}$. Then, $P v_1^{S'} v_2^{S'} w_2 u_2$ is an induced $H$ with antenna $a_{k-1} a_{k-2}$ (this case is illustrated in Figure 2.23a). So one can assume that $a_{k-1}$ has exactly one neighbor in $\{v_1^S, v_1^{S'}\}$. If $a_{k-1} v_1^S \in E$ (respectively $a_{k-1} v_1^{S'} \in E$), then the concatenation of $P'$ with $v_1^{S'} v_2^{S'} w_2 u_2$ (resp. $P$) in an induced $P_9$. The case $a_5 = v_1^S$ is illustrated in Figure 2.23b). In both cases, we get a contradiction.

Hence, we assume that the length of $P'$ is three (recall that it cannot be less than three as $\text{dist}(x_i, X) \geq 3$). Let $t_i$ be the vertex of $P'$ at distance one from $\{v_1^S, v_1^{S'}\}$, and let $r_i$ be the neighbor of $x_i$ in $P'$. Recall that $\{t_i, r_i\} \subseteq D_1$. If $t_i$ is adjacent to both $v_1^S$ and $v_1^{S'}$, $P v_1^{S'} v_2^{S'} w_2 u_2$ is an induced $H$ with antenna $t_i r_i$. So one can assume $|N(t_i) \cap \{v_1^S, v_1^{S'}\}| = 1$. First, observe that if $X \cup \{t_i, r_i\}$ dominates $N[x_i]$, the conclusion directly follows. So one can assume that there exists a vertex

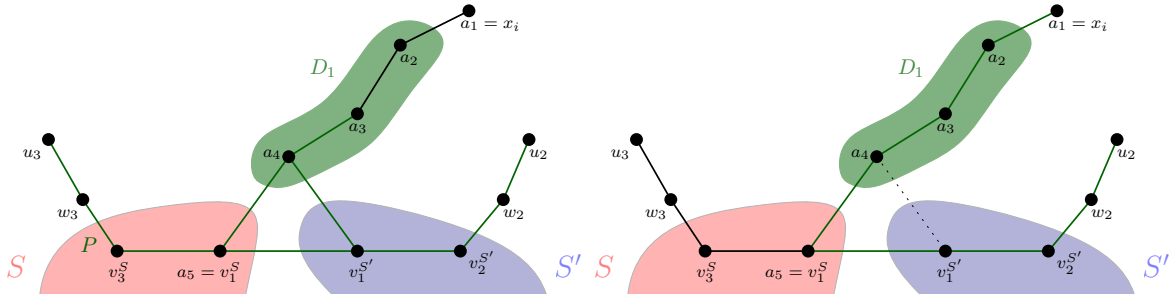



Figure 2.23 – Illustration for the proof of Claim 1 where $|P| \geq 5$.

$u_i \in N[x_i]$ with no neighbor in $X \cup \{t_i, r_i\}$. Since $x_i r_i \in E$, $u_i \neq x_i$. Moreover, $P \cup \{v_1^{S'}, v_2^{S'}, w_2\} \subseteq X$. But then observe that if $t_i v_1^S \in E$ (respectively $t_i v_1^{S'} \in E$) then $u_i x_i r_i t_i v_1^S v_1^{S'} v_2^{S'} w_2 u_2$ (resp. $u_i x_i r_i t_i v_1^{S'} P$) is an induced $P_9$ or $C_9$ if $u_i u_2 \in E$ (resp. $u_i u_3 \in E$). So $N[x_i]$ is dominated by $X \cup \{t_i, r_i\}$, as desired. This concludes the proof of Claim 1. ◊

In particular, Claim 1 implies that for any vertex $x_i \in f(D_1)$, there exist two vertices $t_i$ and $r_i$ such that $N[x_i] \subseteq N[X \cup \{t_i, r_i\}]$ and $G[X \cup \{t_i, r_i\}]$ is connected. This claim is of special interest due to the following:

*Claim* 2. The set $D' = (D \setminus D_1) \cup \{x_1, x_2, \ldots, x_k\}$ is a dominating set of $G$.

*Proof.* Suppose by contradiction that $D'$ is not a dominating set of $G$ and let $u$ be an undominated vertex. Since $D \setminus D_1 \subseteq D'$ and $D$ is dominating, we have $u \in N[D_1]$. Let $v$ be a vertex of $D^0$ that dominates $u$. Since $u$ is not dominated by $D'$, $v \notin f(D_1) = \{x_1, x_2, \ldots, x_k\}$. Due to the injective property of $f$, there exists a connected component $D_j$ with $j \neq 1$ such that $v \in f(D_j)$. Due to the increasing property of $f$, $N[v] \subseteq N[D_j]$. In particular, we have that $u \in N[D_j]$. But then observe that $u \in N[D_1] \cap N[D_j]$, which contradicts the maximality of $\ell$ (since $j \neq 1$). ◊

Recall that $X = (D \setminus D_1) \cup S \cup S'$ and that $D \cup S \cup S'$ is a connected dominating set of size $2\gamma + 1$. By Claims 1 and 2, for every $1 \leq i \leq k$, there exist $r_i, t_i \in V$ such that $(D' \setminus \{x_i\}) \cup S \cup S' \cup \{t_i, r_i\}$ also is a dominating set of $G$. So Claims 1 and 2 together imply that for any set $Y \subseteq \{x_1, x_2, \ldots x_k\}$, there exists a set $S(Y)$ such that (i) $|S(Y)| \leq 2 \cdot |Y|$; and (ii) $X \cup (\{x_1, x_2, \ldots, x_k\} \setminus Y) \cup S(Y)$ is a connected dominating set of $G$. Moreover, observe that if $Y = \{x_1, x_2, \ldots, x_k\}$, $X \cup S(Y)$ is a connected dominating set of size at most $2\gamma + 2$ (since $|D_1| = 2k - 1$ and $|S(Y)| \leq 2k$).

Suppose first that there exists $x_i \in f(D_1)$ such that $x_i \in X$ and let $Y = \{x_1, x_2, \ldots, x_k\} \setminus \{x_i\}$. Then $X \cup S(Y)$ is a connected dominated set of size at most $2\gamma(G)$ since $|S(Y)| \leq 2k - 2 < |D_1|$. So we assume that $\{x_1, x_2, \ldots, x_k\} \cap X = \emptyset$.

So there exist $x_i, x_j \in \{x_1, x_2, \ldots, x_k\}$ such that $x_i v_1^S \in E$ and $x_j v_1^{S'} \in E$. Indeed, $\{v_1^S, v_1^{S'}\} \subseteq N(D_1)$. Hence, similarly as in the proof of Claim 2 and due to the increasing property of the function $f$, both $v_1^S$ and $v_1^{S'}$ must be dominated in $D^0$ by a vertex in $f(D_1)$. Suppose first that $i \neq j$, i.e., $x_i$ and $x_j$ are two distinct vertices of $f(D_1)$ at distance one from $S \cup S'$. In that case, take $Y = \{x_1, x_2, \ldots, x_k\} \setminus \{x_i, x_j\}$. But then $X \cup S(Y) \cup \{x_i, x_j\}$ is a connected dominating of size at most $2\gamma$ since $|S(Y) \cup \{x_i, x_j\}| \leq 2k - 2 < |D_1|$, a contradiction. So one can assume that $x_i = x_j$ and thus $x_i v_1^S v_1^{S'}$ is a triangle. If $N[x_i] \subseteq N[X]$, then $X \cup S(\{x_1, x_2, \ldots, x_k\} \setminus \{x_i\})$ is a connected dominating set of size at most $2\gamma$, a contradiction. So there exists $u_i \in N(x_i)$ such that $u_i$ has no neighbor in $X$, and thus $P v_1^{S'} v_2^{S'} w_2 u_2$ is an $H$ with antenna $x_i u_i$. The only possible missing edges are $u_i u_2$ or $u_i u_3$. But in that case, Remark 2.26 allows us to conclude (see Figure 2.24). □

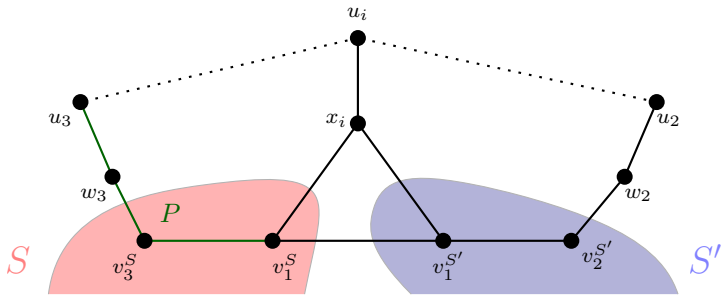



Figure 2.24 – Illustration for the proof of Lemma 2.42: the only possible missing edges are the dotted ones.

As a result, let us denote by $w_1$ the unique vertex of $D_1 = D_1^\ell$. To conclude the proof of Theorem 2.25, we derive the same contradiction as in the proof of Lemma 2.42. More precisely, either $D' = D \setminus \{w_1\} \cup S \cup S'$ is a connected dominating set, in which case the conclusion follows since $|D'| \leq 2\gamma$, or it is not. If $D'$ is not a connected dominating set, then we claim that we can find a copy of a forbidden subgraph. Let $u_1$ be a neighbor of $w_1$ with no neighbor in $D'$. First, recall that $w_1 v_1^S v_1^{S'}$ is a triangle by Lemmas 2.39 and 2.42. Since $|S'| \geq 2$, let $v_2^{S'}$ be a neighbor of $v_1^{S'}$ and recall that $v_2^{S'}$ has no neighbor in $S$ due to maximality of $S$ ($v_2^{S'}$ is incident to $D_2^\ell \notin A(S)$ and then the addition of $v_2^{S'}$ to $S$ provides a contradiction).

By Lemma 2.40, there exists an induced path $P$ on four vertices such that only $v_1^S$ has a neighbor in $S'$ (which is $v_1^{S'}$). So the concatenation of $P$ with $v_1^{S'} v_2^{S'} w_2 u_2$ is an induced $P_8$. Indeed, the only possible missing edge is $u_2 u_3$, if $P$ ends with a vertex $u_3 \in N[D_3]$. But in that case, $u_3$ would be 2-linking for $S'$ and we would have $u_3 v_3^S \in E$ by Lemma 2.39, a contradiction. Hence, $P v_1^{S'} v_2^{S'} w_2 u_2$ is an induced $H$ with antenna $w_1 u_1$ unless $u_1 u_2 \in E$ or $u_1 u_3 \in E$. In all cases, we can find an induced copy of a forbidden subgraph by Remark 2.26. This concludes the proof of Theorem 2.25.

### 2.2.3 Concluding remarks

In this section, we characterized the class of *PoC*-Near-Perfect graphs with threshold two. One can observe that for any $k \in \{10, 11\}$, $PoC(P_k) < 7/3 = PoC(P_9)$. Similarly, $PoC(C_k) < 7/3 = PoC(C_9)$. However, $PoC(P_{12}) = PoC(C_{12}) = 5/2$. Hence, if one wants to characterize *PoC*-Near-Perfect graphs with threshold $7/3$ in terms of forbidden induced subgraphs, one must exclude $P_{12}$ and $C_{12}$.

More generally, one can observe that $P_9$ can be obtained from $P_3$ (with $V(P_3) = \{v_1, v_2, v_3\}$) by (i) first subdividing each edge twice, (ii) adding two new leaves adjacent to $v_1$ and $v_3$, respectively. Let $P'$ be the resulting path. The graph $C_9$ is obtained from $P'$ by adding an edge between the leaves of $P'$. And the graph $H$ is obtained from $P'$ by adding an edge between the neighbors of $v_2$, and then add a new leaf adjacent to $v_2$. These constructions can be generalized in the following way (see Figure 2.25):

**Definition 2.43.** *For any $k \geq 3$, we define the class $\mathcal{T}_k$ consisting of graphs created as follows:*

*(i) start with any tree $T$ with vertex set $\{v_1, v_2, \ldots, v_k\}$;*

*(ii) subdivide each edge of $T$ twice;*

*(iii) to each original leaf $u$ of $T$, add a new leaf adjacent to $u$;*

*(iv) we may add edges between the neighbors of $v_i$ if $v_i$ is not adjacent to a leaf. If at some point the subgraph induced by $N(v_i)$ becomes connected by the addition of these edges, we attach a new leaf to $v_i$ and we do not add any more edges between vertices in $N(v_i)$;*

*(v) we may add a matching between the leaves of the graph obtained at step (iii).*

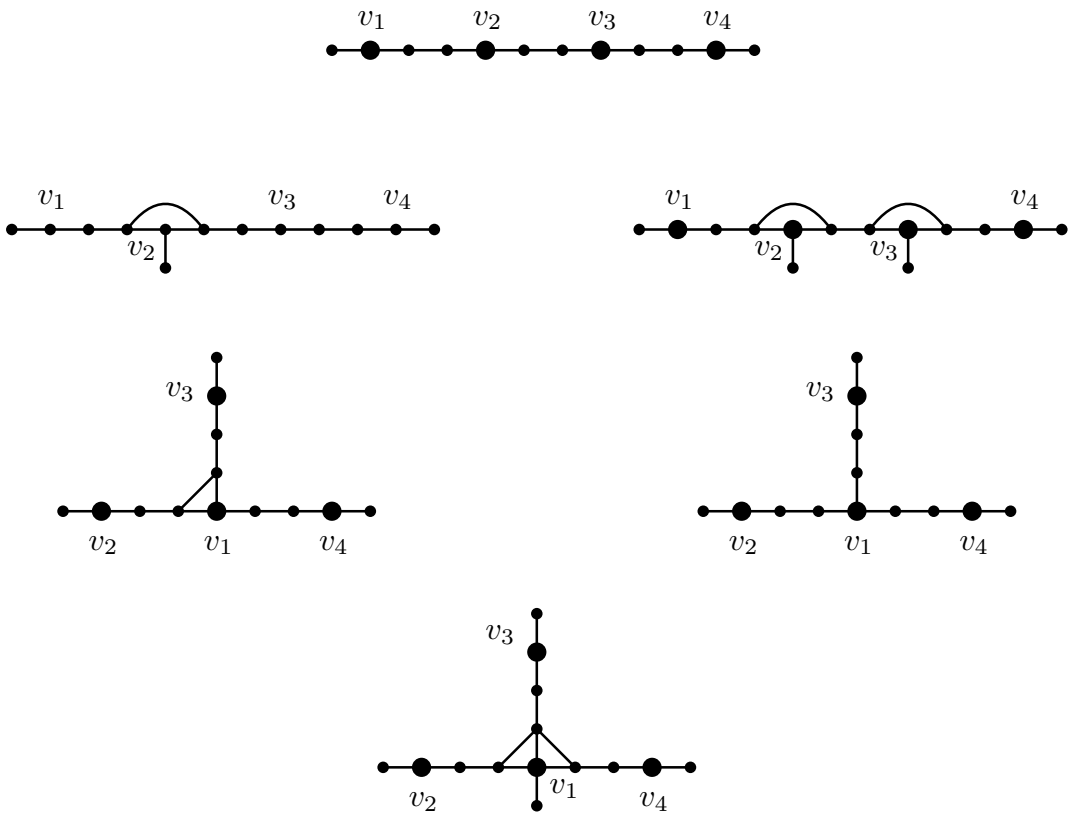



Figure 2.25 – Any graph from $\mathcal{T}_4$ is obtained from one of these graphs by adding a (non necessarily maximal) matching between the leaves.

One can prove that any graph $G$ in $\mathcal{T}_k$ has domination number $k$ and connected domination number $3k-2$. Hence $PoC(G) = 3 - \frac{2}{k}$. We thought that the graphs in $\mathcal{T}_k$ were the only forbidden induced subgraphs to be *PoC*-Near-Perfect with threshold $3 - \frac{2}{k-1}$. More precisely, we conjectured the following:

**Conjecture 2.44.** *Let $G$ be a graph. Any connected induced subgraph $H$ of $G$ satisfies $\frac{\gamma_c(H)}{\gamma(H)} \leq 3 - \frac{2}{k-1}$ if and only if $G$ does not contain any graph in $\mathcal{T}_k$ as an induced subgraph.*

However, very recently, Klimošová and Hulcová [HK20] disproved Conjecture 2.44. Indeed, our conjecture states that $G$ is $\mathcal{T}_4$-free if and only if $G$ is *PoC*-Near-Perfect with threshold $t = 7/3$. However, they found the following graph which is $\mathcal{T}_4$-free but with price of connectivity $12/5 > 7/3$.

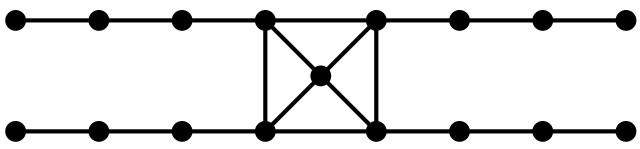


Figure 2.26 – Counterexample for Conjecture 2.44.

They generalized this conjecture for any value of $k$ in the following way. For any $k \geq 5$, there exists a graph $G_{k+1}$ with $PoC(G_{k+1}) = 3 - \frac{3}{k+1}$ and $G_{k+1}$ is critical (i.e., $PoC(H) < PoC(G_{k+1})$ for any induced subgraph $H$ of $G_{k+1}$). And this graph is $\mathcal{T}_k$-free. Indeed, any graph from $\mathcal{T}_k$ has price of connectivity $3 - \frac{2}{k}$, which is larger than $PoC(G_{k+1}) = 3 - \frac{3}{k+1}$ and thus $G_{k+1}$ is critical.

# 3 Reconfiguration of dominating sets

**Contents**



In this chapter, we study the reconfiguration of dominating sets, mainly under token addition and removal (TAR) and token sliding (TS). The results presented are from joint work with Nicolas Bousquet and Marthe Bonamy [BDO21], Nicolas Bousquet and Alice Joffard [BJO20], and Alexandre Blanché, Haruka Mizuta and Akira Suzuki [BMOS20]. In Section 3.1, we consider questions regarding the connectivity of the reconfiguration graph under TAR. We then study in Section 3.2 the computational complexity of the reachability of dominating sets reconfiguration under TS from a graph classes viewpoint. Finally, in Section Section 3.3, we mainly investigate the parameterized complexity of an optimization variant recently introduced by Ito et al. [IMNS19] for INDEPENDENT SET RECONFIGURATION.

## 3.1 Connectivity of the reconfiguration graph under TAR

### 3.1.1 Introduction

As we said in Section 1.5, reconfiguration problems can be studied through the lens of the reconfiguration graph. Recall that the vertices of the reconfiguration graph are the feasible solutions of an instance $\mathcal{I}$ of a problem $\Pi$, and two vertices (i.e., two solutions of $\mathcal{I}$) are adjacent if

and only if one solution can be obtained from the other by applying the specified reconfiguration rule. Hence, this graph deeply depends on the choice of this rule since it defines the vertex set as well as the adjacency between the vertices. Indeed, the reconfiguration graph under TAR contains solutions of various sizes, while the size of each solution must remain constant under TJ or TS. Hence, the reconfiguration graphs associated with the TAR rule contains more vertices than the ones associated with TJ and TS.

**Reconfiguration graph of minimum dominating sets.** The first papers dealing with the reconfiguration of dominating sets focused on the reconfiguration graph under TJ. Sridharan and Subramanaian [SS08] introduced the concept of $\gamma$-graph which corresponds to the reconfiguration graph $\mathcal{R}_\gamma(G)$ where the vertices are the minimum dominating sets of $G$, and the adjacency is defined by the token jumping rule (see Figure 3.1).

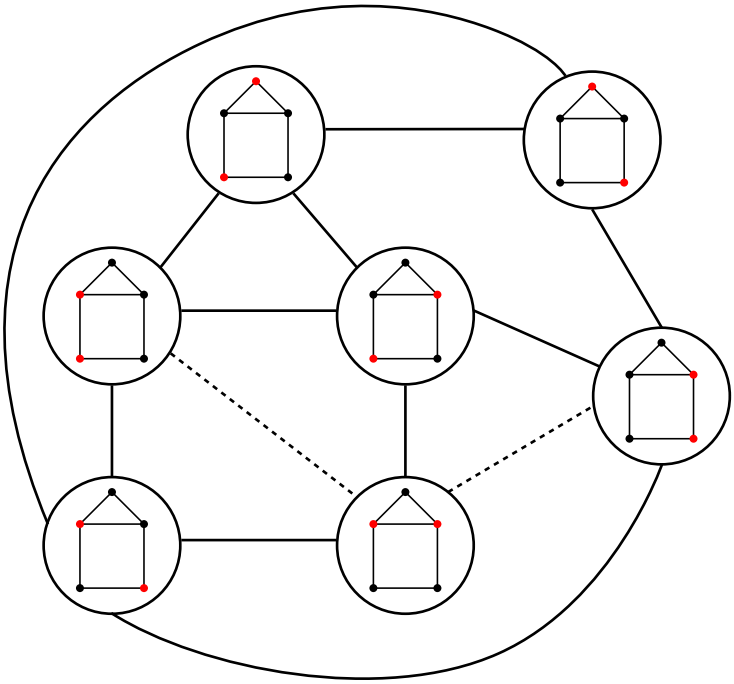


Figure 3.1 – $\gamma$-graph of the House graph under TJ (where we take into account the dashed edges). If we remove the dashed edges, it is the $\gamma$-graph under TS.

In particular, they determined the $\gamma$-graph of cycles and paths, and proved that the $\gamma$-graph of any tree is connected. The same authors showed that if $G$ has at most one cycle (i.e., $G$ is a tree or a unicyclic graph), then $G$ is the $\gamma$-graph of some graph. On the other hand, they proved that any graph that contains the graph $K_{2,3}$ where we add an edge between the two vertices of degree three as an induced subgraph cannot be the $\gamma$-graph of any graph. Following this idea, Lakshmanan and Vijayakumar [LVA10] showed that if a graph $H$ is a $\gamma$-graph, then $H$ is $(K_{2,3}, K_2 + P_3, (K_1 \cup K_2) + 2K_1)$-free. They also showed that the $\gamma$-graph of any cograph $G$ has diameter at most two. And the diameter is one if and only if $G$ has a universal vertex. Very recently, DeVos et al. [DDJS20] extended the concept of $\gamma$-graphs by studying the reconfiguration graph of minimum $d$-dominating sets (i.e., a vertex dominates itself and all the vertices at distance at most $d$).

In 2011, Fricke et al. [FHHH11] studied $\gamma$-graphs where the adjacency is determined by the token sliding rule (see Figure 3.1 where we do not take into account the dashed edges). In particular, they proved that the $\gamma$-graph of any triangle-free graph is also triangle-free. Moreover, they showed that the $\gamma$-graph of a tree is connected and bipartite. Conversely, they also

showed that any tree is the $\gamma$-graph of some graph. The latter was then extended by Connelly et al. [CHHH11] since they proved that any graph is the $\gamma$-graph of infinitely many graphs. They also showed that the $\gamma$-graph of graphs with at most five vertices is connected and characterized graphs on six vertices whose $\gamma$-graph is not connected. Edwards et al. [EMN18] investigated the order, maximum degree and diameter of $\gamma$-graphs of trees. Very recently, Lemańska and Żyliński [LŻ20] studied the diameter of $\gamma$-graphs defining by both TS and TJ. More precisely, they showed that for any tree on $n$ vertices ($n \geq 3$), the diameter of its $\gamma$-graph associated with TJ is at most $n/2$, while it is at most $2(n-1)/3$ if we consider the TS rule.

**$k$-reconfiguration graph.** Haas and Seyffarth [HS14] were the first to study properties of the reconfiguration graph where the adjacency is defined by the token addition and removal rule with threshold $k$. They called this graph the $k$-dominating graph; we call it $k$-reconfiguration graph in this manuscript and denote it by $\mathcal{R}_k(G)$, where $G$ is the input graph. Recall that the vertices of $\mathcal{R}_k(G)$ correspond to the dominating sets of $G$ of size at most $k$. And two vertices of $\mathcal{R}_k(G)$ are adjacent if and only if the size of the symmetric difference of the two corresponding dominating sets is equal to one. Haas and Seyffarth [HS14] obtained a similar result than the one by Kamiński et al. [KMM12] regarding the relation between the rules TAR and TJ. More precisely, they showed that there exists a reconfiguration sequence between two dominating sets $D_s$ and $D_t$, both of size $k$, under the TJ rule if and only if there exists a TAR($k+1$)-sequence between $D_s$ and $D_t$. Let us give a shorter proof of this result:

**Lemma 3.1.** *Let $G$ be a graph and $D_s$ and $D_t$ be two dominating sets of $G$ of size $k$. Then, there exists a* TAR*($k+1$)-sequence between $D_s$ and $D_t$ if and only if $D_s \overset{\mathsf{TJ}}{\leftrightsquigarrow} D_t$ holds.*

*Proof.* The proof is an adaptation of the Theorem 1 of Kamiński et al. [KMM12].

($\Leftarrow$) Suppose first $D_s \overset{\mathsf{TJ}}{\leftrightsquigarrow} D_t$, and let $S$ be a TJ-sequence that reconfigures $D_s$ into $D_t$. This sequence corresponds to a sequence of moves $u \overset{\mathsf{TJ}}{\rightsquigarrow} v$. We construct a TAR-sequence by replacing each atomic move $u \overset{\mathsf{TJ}}{\rightsquigarrow} v$ with two moves of the TAR model: we first add $v$ and then delete $u$. By first adding $v$, we preserve the domination property. Besides, since we immediately delete $u$ after the addition of $v$, each intermediate solution is of size at most $k+1$, as desired.

($\Rightarrow$) For the other direction, let $S'$ be a TAR($k+1$)-sequence that reconfigures $D_s$ into $D_t$. Note that since $|D_s| = |D_t| = k$, $S'$ is of even length. Moreover, by hypothesis, $S'$ does not contain a configuration of size more than $k+1$. If all the configurations of $S'$ are of size $k$ or $k+1$, this means that $S'$ corresponds to an alternation of an addition of a token on some vertex $v$ immediately followed by the deletion of a token on a vertex $u$. Therefore, to get a TJ-sequence, we simply replace each of these subsequences by a move $u \overset{\mathsf{TJ}}{\rightsquigarrow} v$. Suppose now that $S'$ contains some configuration of size less than $k$ and consider a configuration, let us say $D_i$, of smallest size. Since $D_i$ is a configuration of smallest size, this means that it has been obtained from $D_{i-1}$ by the deletion of some vertex $x$. We also know that the configuration $D_{i+1}$ is obtained from $D_i$ by the addition of some vertex $y$. If $x = y$, then these two steps are redundant and can simply be ignored. Otherwise, observe that if we first add $y$ and then delete $x$, the new sequence is still valid. If all the configurations are of size $k$ or $k+1$, we immediately obtain a TJ-sequence. Otherwise, we can repeat this process until this is the case. □

In their seminal paper, Haas and Seyffarth [HS14] showed that $\mathcal{R}_2(K_{1,k}) \simeq K_{1,k}$ and asked whether there are other graphs $G$ for which $\mathcal{R}_k(G) \simeq G$. Alikhani et al. [AFK17] answered negatively as long as $G$ has no isolated vertex. More precisely, if $G$ is a graph on $n$ vertices with minimum degree at least one such that $\mathcal{R}_k(G) \simeq G$ (with $\gamma(G) \leq k \leq n$), then $k = 2$ and $G \simeq K_{1,n-1}$, for some $n \geq 4$. They also showed that for any $d \geq 1$, there exists a finite number

of connected $d$-regular graphs which are reconfiguration graphs of connected graphs. For the case $d = 2$, they proved that $C_6$ and $C_8$ are the only connected 2-regular graphs (i.e., cycles) which are reconfiguration graphs of connected graphs. Moreover, $P_1$ and $P_3$ are the only paths which are reconfiguration graphs of connected graphs.

The case of complete bipartite graphs $K_{1,n}$ (which are called star graphs) is of special interest due to the following observation by Haas and Seyffarth:

**Observation 3.2** ([HS14]). *For any $n \geq 3$, the graph $\mathcal{R}_k(K_{1,n})$ is connected for any $1 \leq k \leq n-1$. However, if $k = n$, then $\mathcal{R}_k(K_{1,n})$ is disconnected.*

*Proof.* First, observe that any dominating set of $K_{1,n}$ of size at most $n-1$ must contain the universal vertex. So let $D_s$ and $D_t$ be two dominating sets of size at most $n-1$. One can transform $D_s$ into $D_t$ by (i) removing one-by-one each vertex in $D_s \setminus D_t$; and (ii) adding one-by-one all the vertices in $D_t \setminus D_s$. However, note that the dominating set $D$ containing the $n$ degree-one vertices is a minimal dominating set of $K_{1,n}$. Since it is minimal, one cannot remove any vertex from $D$. Since $|D| = n$ (the size of the threshold we cannot exceed), we cannot add any vertex to $D$ neither. Therefore, $D$ is frozen, i.e., it is an isolated vertex in $\mathcal{R}_n(K_{1,n})$. But the graph $R_n(K_{1,n})$ also contains the dominating sets of size at most $n-1$ which are in the same connected component of $R_n(K_{1,n})$ (see Figure 3.2). □

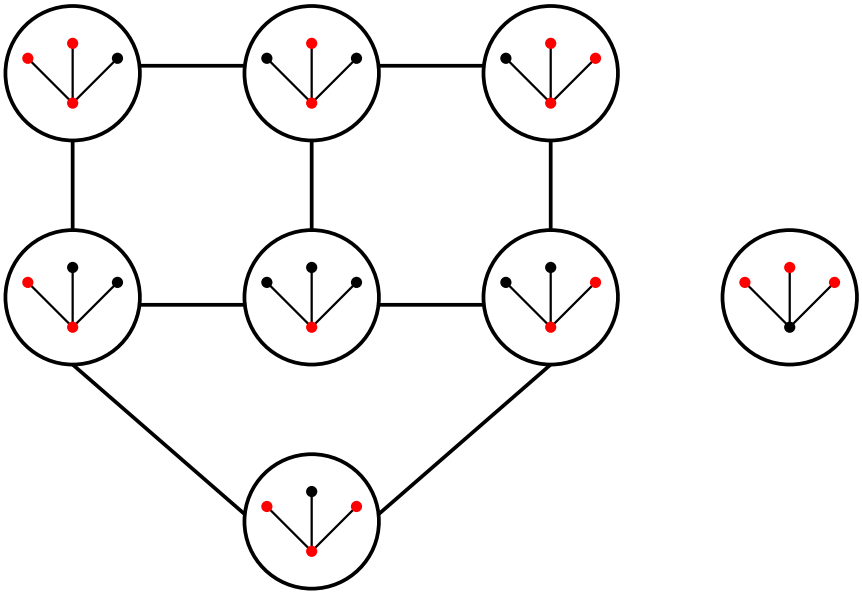


Figure 3.2 – The reconfiguration graph $\mathcal{R}_3(K_{1,3})$ is disconnected.

Actually, they observed that any upper dominating set is an isolated vertex in the reconfiguration graph $\mathcal{R}_\Gamma(G)$. Hence, $\mathcal{R}_\Gamma(K_{1,n})$ is disconnected whenever $G$ has at least one edge. Moreover, Observation 3.2 states that being reconfigurable is not a monotone property, i.e., $\mathcal{R}_k(G)$ being connected does not necessarily imply that $\mathcal{R}_{k+1}(G)$ also is connected. In that context, Haas and Seyffarth [HS14] introduced a new parameter $d_0(G)$ which is the smallest integer such that for any $k \geq d_0(G)$, the reconfiguration graph $\mathcal{R}_k(G)$ is connected. Note that this parameter is well defined as $\mathcal{R}_n(G)$ is always connected for any graph $G$ with $n$ vertices. Indeed, given two dominating sets $D_s$ and $D_t$, one can first add all the vertices of $D_t \setminus D_s$ and then remove each vertex in $D_s \setminus D_t$ to transform $D_s$ into $D_t$. Note that this corresponds to a shortest transformation since at each step, we decrease by one the size of the symmetric difference $|D_s \triangle D_t|$, which is the best we can hope for within the TAR model. On the positive side, Haas and Seyffarth proved the following:

**Lemma 3.3** ([HS14]). *Let $G$ be a graph. If $k > \Gamma(G)$ and $\mathcal{R}_k(G)$ is connected, then $\mathcal{R}_{k+1}(G)$ is connected.*

Moreover, they proved that if $G$ has at least two independent edges, then $d_0 \leq \min\{n-1, \Gamma(G)+\gamma(G)\}$. They also showed that this value can be lowered to $\Gamma(G)+1$ if $G$ is bipartite or a chordal graph. This result is tight since $K_{1,n}$ is bipartite and chordal and $\mathcal{R}_n(K_{1,n})$ is not connected. They then asked if this result can be generalized to any graph. Suzuki et al. [SMN14] answered negatively this question by constructing an infinite family of graphs for which $\mathcal{R}_{\Gamma(G)+1}(G)$ is not connected. Mynhardt et al. [MTR19] improved this result by constructing two infinite families of graphs:

- the first construction provides graphs with arbitrary $\Gamma \geq 3$, arbitrary domination number in the range $2 \leq \gamma \leq \Gamma$ such that $d_0 = \Gamma + \gamma - 1$;
- the second one gives graphs with arbitrary $\Gamma \geq 3$, arbitrary domination number in the range $1 \leq \gamma \leq \Gamma - 1$ for which $d_0 = \Gamma + \gamma$. For $\gamma \geq 2$, this is the first construction of graphs with $d_0 = \Gamma + \gamma$.

Note that this lower bound is somehow the best we can hope for in the general case since $d_0 \leq \min\{n-1, \Gamma(G)+\gamma(G)\}$ holds for any graph $G$ with at least two independent edges.

Suzuki et al. [SMN14] generalized the upper bound of $n-1$ on $d_0$ for graphs with at least two edges by relating $d_0$ to the size of a maximum matching of $G$. More precisely, they proved that if a graph $G$ on $n$ vertices has a matching of size (at least) $\mu+1$, then $\mathcal{R}_{n-\mu}$ is connected. This result is best possible since the path $P_{2k}$ has matching number $\mu(P_{2k}) = k = \Gamma(P_{2k})$ and recall that $\mathcal{R}_\Gamma(P_{2k})$ is disconnected. Moreover, it follows from the result by Suzuki et al. [SMN14] that the diameter of $\mathcal{R}_{n-\mu}(G)$ is linear. However, they constructed an infinite family of graphs such that for each graph $G_n$ in the family, $\mathcal{R}_{\gamma+1}(G_n)$ has diameter $\Omega(2^n)$.

Very recently, Rautenbach and Redl [RR20] studied how many additional vertices should be allowed in order to transform a given dominating set into another one. They showed that there is a positive constant $c$ such that for every positive integer $n$, there is a 4-regular graph $G$ of order at least $n$ that can be embedded on the torus, and there are two dominating sets $D_s$ and $D_t$ of $G$, both of size $n/5$, such that there is a TAR($k$)-sequence that transforms $D_s$ into $D_t$ only if $k \geq c\sqrt{n}$. On the other hand, they proved that for an hereditary class of graph $\mathcal{G}$ that has balanced separators of order $n \mapsto n^\alpha$ (for some $\alpha < 1$), there is a constant $C$ such that for any graph $G \in \mathcal{G}$ on $n$ vertices, and for any two dominating sets $D_s$ and $D_t$ of $G$ there is a TAR($k$)-sequence that transforms $D_s$ into $D_t$ if $k = \max\{|D_s|, |D_t|\} + \lfloor Cn^\alpha \rfloor$.

Finally, Haas and Seyffarth [HS17] related the parameter $d_0$ to the independence number of a graph. More precisely, they proved that if $k = \Gamma(G) + \alpha(G) - 1$, then the reconfiguration graph $\mathcal{R}_k(G)$ is connected. To obtain this result, they proved that all the independent dominating sets of $G$ are in the same connected component of $\mathcal{R}_{\Gamma(G)+1}(G)$.

**Our contribution.** In this section, we complete some of the results mentioned above by providing upper bounds on $d_0$ depending on several graph parameters. This is joint work with Nicolas Bousquet and Alice Joffard [BJO20]. Note that all the proofs lead to linear transformations between any pair of dominating sets. Each transformation can be computed in polynomial time, or in FPT time for the upper bound depending on the treewidth.

### 3.1.2 Upper bound related to the independence number

In this section, we relate the upper bound on $d_0$ to the upper domination number and the independence number of a graph $G$. More precisely, we improve the following result by Haas and Seyffarth:

**Lemma 3.4.** *[HS17] Let $G$ be a graph. If $k = \Gamma(G) + \alpha(G) - 1$, then $\mathcal{R}_k(G)$ is connected.*

Indeed, their result states that the reconfiguration graph is connected, but their induction-based proof does not give any bound on its diameter. Here, we give a new (simpler) proof of the same result that moreover implies that the diameter of $\mathcal{R}_k(G)$ is linear. The proof is constructive and provides an algorithm that computes a TAR($k$)-sequence between two dominating sets of size at most $k$ of $G$.

As discussed in Chapter 1, computing a maximum independent set of a given graph $G$ is a classical NP-complete problem [Kar72]. However, computing a maximal one can be done trivially in linear time by a greedy algorithm. Moreover, given an independent set $S'$ which is not maximal, one can greedily complete $S'$ into a maximal independent set $S$ such that $S' \subseteq S$. In particular, if there exist two vertices $u$ and $v$ such that $uv \notin E$, then there exists a maximal independent set of $G$ which contains both $u$ and $v$. Obviously, this is also true when $S'$ is reduced to a single vertex. We will use this fact in the proof of Theorem 3.6. Recall also that any maximal independent set of $G$ also is a minimal dominating set of $G$ (see Proposition 2.8). In particular, any maximal independent set $S$ of $G$ satisfies $|S| \leq \alpha(G) \leq \Gamma(G)$. We also need the following observation:

**Observation 3.5.** *Let $D$ be a minimal dominating set of $G$, and let $S$ be a maximal independent set of $G$ such that $D \cap S \neq \emptyset$. If $k = \Gamma(G) + \alpha(G) - 1$, then there exists a* TAR*($k$)-reconfiguration sequence between $D$ and $S$ of length at most $|D| + \alpha(G) - 2$.*

*Proof.* Recall that since $S$ is a maximal independent set, $|S| \leq \alpha(G) \leq \Gamma(G)$. We first add to $D$ each vertex in $S \setminus D$ one by one. Note that there are at most $\alpha(G) - 1$ such vertices. We thus obtain the set $D' = D \cup S$. We then remove one by one each vertex in $D \setminus S$. There are at most $|D| - 1$ such vertices since $S \cap D \neq \emptyset$. Each intermediate solution is indeed a dominating set since it either contains $D$ or $S$ which are both dominating sets. Moreover, each solution is of size at most $|D'| \leq |D| + |S| - 1 \leq k$. □

We are now ready to prove the main result of this subsection regarding the diameter of the reconfiguration graph $\mathcal{R}_k(G)$, with $k = \Gamma(G) + \alpha(G) - 1$:

**Theorem 3.6.** *Let $G = (V, E)$ be a graph on $n$ vertices. If $k = \Gamma(G) + \alpha(G) - 1$ then $\mathcal{R}_k(G)$ has diameter at most $10n$.*

*Proof.* Let $D_1$ and $D_2$ be two dominating sets, both of size at most $k$. Free to remove at most $2 \cdot (\Gamma(G) + \alpha(G) - 2)$ vertices in total, one can assume without loss of generality that $D_1$ and $D_2$ are both inclusion-wise minimal dominating sets of $G$. Hence $|D_1| \leq \Gamma(G)$ and $|D_2| \leq \Gamma(G)$. We outline a path between $D_1$ and $D_2$ in $\mathcal{R}_k(G)$. The next claim deals with the case where $D_1$ and $D_2$ have a non-empty intersection.

*Claim* 1. If $D_1 \cap D_2 \neq \emptyset$, then there exists a reconfiguration sequence from $D_1$ to $D_2$ of length at most $2 \cdot (\alpha(G) + \Gamma(G) - 2)$.

*Proof.* Let $x$ be a vertex that belongs to both $D_1$ and $D_2$. One first constructs greedily (and thus in polynomial-time) a maximal independent set $S$ of $G$ which contains $x$ (which is then of size at most $\alpha(G)$). By Observation 3.5, one can transform $D_1$ into $S$ under the TAR($k$) rule. And the length of the reconfiguration sequence is at most $\Gamma(G) + \alpha(G) - 2$. Similarly, there exists a

reconfiguration sequence of length at most $\Gamma(G) + \alpha(G) - 2$ from $D_2$ to $S$. By combining these two transformations, we obtain a reconfiguration sequence between $D_1$ and $D_2$ of length at most $2 \cdot (\alpha(G) + \Gamma(G) - 2)$, as desired. ◊

In the remainder of this proof, we assume that $D_1 \cap D_2 = \emptyset$ otherwise we can directly conclude the proof by Claim 1. If there exist $u_i \in D_1$ and $v_j \in D_2$ such that the set $D' = (D_1 \setminus \{u_i\}) \cup \{v_j\}$ is a dominating set of $G$, then we can conclude the proof by Claim 1 since $D' \cap D_2 \neq \emptyset$ and $D'$ can be obtained from $D_1$ in two steps. Suppose now that $D' = (D_1 \setminus \{u_i\}) \cup \{v_j\}$ is not a dominating set of $G$. This means that $u_i$ is adjacent to a vertex $x$ with no neighbors in $(D_1 \setminus \{u_i\}) \cup \{v_j\}$. Hence, there exists a maximal independent set $S_1$ of $G$ which contains both $x$ and a vertex $u_k \in D_1 \setminus \{u_i\}$. Similarly, there exists a maximal independent set $S_2$ which contains both $x$ and $v_j$. By Observation 3.5, there exists a reconfiguration sequence of length at most $\Gamma(G) + \alpha(G) - 2$ between $S_1$ (respectively $S_2$) and $D_1$ (respectively $D_2$) under the TAR($k$) rule. Finally, since $S_1$ and $S_2$ intersect, we can again use Observation 3.5 that ensures that there exists a transformation from $S_1$ to $S_2$ of length at most $2\alpha(G) - 2$.

Hence, we obtain a TAR($k$)-reconfiguration sequence from $D_1$ to $D_2$ of length at most $4 \cdot (\Gamma(G) + \alpha(G) - 2) + 2 \cdot (\alpha(G) - 1) < 10n$. □

### 3.1.3 $H$-minor free graphs

In this section, we prove some better bounds on $k$ for minor-free graphs. We say that a graph is *$d$-minor sparse* if all its bipartite minors have average degree less than $d$. Note that it is equivalent to say that the ratio between the number of edges and the number of vertices of any bipartite minor of $G$ is strictly less than $\frac{d}{2}$. Let us first prove the following lemma:

> **Lemma 3.7.** *Let $G$ be a $d$-minor sparse graph. Let $A$ and $B$ be two dominating sets of $G$ such that $|A| = |B|$ and $|B \setminus A| \geq d$. Then, there exists a vertex $a \in A \setminus B$ and a set $S \subset B \setminus A$ with $|S| = d - 1$ such that $(A \cup S) \setminus \{a\}$ is a dominating set of $G$.*

*Proof.* We prove it by contradiction. For every $a_i \in A \setminus B$, let $S_{i,1}$ be a subset of $B \setminus A$ of size $d - 1$. Let $x_{i,1}$ be a vertex that is only dominated by $a_i$ in $A$ and not dominated by $S_{i,1}$ in $B$ (such a vertex must exist otherwise the conclusion follows). Note that this vertex can be a vertex of $A$, a vertex of $B$, a vertex of both or a vertex of neither. Let $b_{i,1}$ be a vertex of $(B \setminus A) \setminus S_{i,1}$ that dominates $x_{i,1}$. This vertex exists since $B$ is a dominating set and $x_{i,1}$ is only dominated by $a_i$ in $A$. Now, for every $2 \leq j \leq d$, we define recursively $S_{i,j}$, $b_{i,j}$ and $x_{i,j}$ as follows. The set $S_{i,j}$ is a subset of size $d - 1$ of $B \setminus A$ containing $\{b_{i,1}, \ldots, b_{i,j-1}\}$. We let $x_{i,j}$ be a vertex only dominated by $a_i$ in $A$ that is not dominated by $S_{i,j}$ in $B$, and $b_{i,j}$ be a vertex of $(B \setminus A) \setminus S_{i,j}$ that dominates $x_{i,j}$. Note that, for every $j$, since $x_{i,j}$ is incident to $b_{i,j}$ and not to $S_{i,j}$, $b_{i,j} \notin \{b_{i,1}, \ldots, b_{i,j-1}\}$. In particular, $B_i = \{b_{i,1}, \ldots, b_{i,d}\}$ has size exactly $d$. Note that $B_i \subseteq B \setminus A$. The construction of the set $B_i$ is illustrated in Figure 3.3.

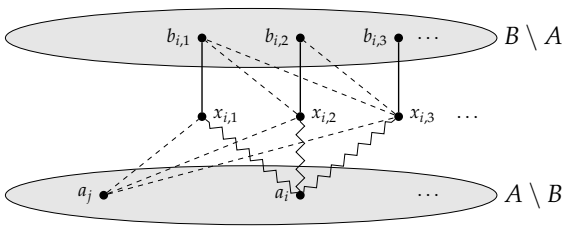



Figure 3.3 – The set $B_i$. The dotted lines represent the non-edges, and the zigzags represent the edges that are contracted in $G'$.

Let us construct a minor $G'$ of $G$ of density at least $d$. In this minor, every vertex $a_i$ in $A \setminus B$ will be adjacent to every vertex of $B_i$. To that end, for every $a_i \in A \setminus B$, we contract the edges $a_i x_{i,j}$ for any $j$ such that $x_{i,j} \notin B \setminus A$ and $x_{i,j} \notin A \setminus B$. If $x_{i,j} \in A \setminus B$, then $x_{i,j} = a_i$ and $a_i$ is already adjacent to $b_{i,j}$, so no contraction is needed. If $x_{i,j} \in B \setminus A$, then by construction $x_{i,j} = b_{i,j}$ and no contraction is needed. By abuse of notations, we still denote by $a_i$ the vertex resulting from the contractions involving $a_i$. Note that the vertices $x_{i,j}$ are pairwise disjoint. If $x_{i,j} = x_{i',j'}$ then, since $x_{i,j}$ is only dominated by $a_i$ and $x_{i',j'}$ by $a'_i$, we must have $a_i = a'_i$. And by construction in the previous paragraph, $x_{i,j} \neq x_{i,j'}$ if $j \neq j'$. So the contractions above are well defined. Moreover, the size of $A \setminus B$ is left unchanged. Similarly the size of $B \setminus A$ is not modified. We finally remove from the graph any vertex which is not in $(A \setminus B) \cup (B \setminus A)$, and any edge internal to $A \setminus B$ or to $B \setminus A$. The resulting graph $G'$ is a minor of $G$ and is bipartite.

For every $i$ and every vertex $v$ in $B_i$, there exists a $j$ such that $v$ is adjacent to $x_{i,j}$ or $a_i$ in $G$. Thus, $a_i$ is adjacent to every vertex of $B_i$ in $G'$. Therefore, for any $a_i \in A \setminus B$, $a_i$ has degree at least $d$ in $G'$. Thus, there are at least $d \cdot |A \setminus B|$ edges in $G'$. Since $G'$ has $|A \setminus B| + |B \setminus A| = 2|A \setminus B|$ vertices, it contradicts the fact that $G$ is a $d$-minor sparse graph. □

**Lemma 3.8.** *Let $G$ be a $d$-minor sparse graph. If $k = \Gamma(G) + d - 1$, then $\mathcal{R}_k(G)$ is connected and the diameter of $\mathcal{R}_k(G)$ is at most $2\Gamma(G) \cdot (d-1) + 2 \cdot \max\{\Gamma(G) - 1, d - 1\}$.*

*Proof.* Let $D_s$ and $D_t$ be two dominating sets of $G$ of size at most $k$. Since $\Gamma(G)$ is the maximum size of a dominating set minimal by inclusion, we can add or remove vertices from $D_s$ and $D_t$ so that $D_s$ and $D_t$ both have size exactly $\Gamma(G)$, while still remaining dominating sets. To do so, we need to remove or add at most $2 \cdot \max\{\Gamma(G) - 1, d - 1\}$ vertices in total. So from now on, we assume that $|D_s| = |D_t| = \Gamma(G)$. Let us show that there is a path from $D_s$ to $D_t$ in $\mathcal{R}_k(G)$ of length at most $2|D_t \setminus D_s| \cdot (d-1)$. Since $|D_t \setminus D_s| \leq \Gamma(G)$, and by taking into account the at most $2 \cdot \max\{\Gamma(G) - 1, d - 1\}$ vertices initially added or removed, this will give the desired result. We proceed by induction on $|D_t \setminus D_s|$.

If $|D_t \setminus D_s| \leq d - 1$ then, since $|D_s| = \Gamma(G)$, we have $|D_s \cup D_t| \leq \Gamma(G) + d - 1$. Thus, we can simply add all the vertices of $D_t \setminus D_s$ to $D_s$ and then remove the vertices of $D_s \setminus D_t$. We thus obtain a path from $D_s$ to $D_t$ in $\mathcal{R}_k(G)$ of length at most $2d - 2 \leq 2|D_t \setminus D_s| \cdot (d-1)$.

Assume now that $|D_t \setminus D_s| \geq d$. By Lemma 3.7, there exists a vertex $v \in D_s \setminus D_t$ and a set $S \subset D_t \setminus D_s$ with $|S| = d - 1$ such that $D'_s = (D_s \cup S) \setminus \{v\}$ is a dominating set of $G$. Let $D''_s$ be any dominating set of size exactly $\Gamma(G)$ obtained by removing vertices of $D'_s$, i.e., such that $D''_s \subseteq D'_s$. Since $|S| = d - 1$ and $|D_s| = \Gamma(G)$, the transformation that consists in adding every vertex of $S$ to $D_s$ and then removing $v$ and every vertex of $D'_s \setminus D''_s$ is a path from $D_s$ to $D''_s$ in $\mathcal{R}_k(G)$. Moreover, $|D'_s| = \Gamma(G) + d - 2$. Thus, this path has length $2d - 2$.

We have $D'_s = (D_s \cup S) \setminus \{v\}$ where $v \in D_s \setminus D_t$ and $S \subset D_t \setminus D_s$ with $|S| = d - 1$. Thus, $|D_t \setminus D'_s| = |D_t \setminus D_s| - d + 1$. Since $D''_s \subseteq D'_s$ and $|D'_s \setminus D''_s| \leq d - 2$, it gives $|D_t \setminus D''_s| \leq |D_t \setminus D_s| - 1$. By the induction hypothesis, there exists a path from $D''_s$ to $D_t$ in $\mathcal{R}_k(G)$ of length at most $|D_t \setminus D''_s| \cdot (2d - 2)$. The concatenation of the two paths gives a path from $D_s$ to $D_t$ in $\mathcal{R}_k(G)$ of length at most $2|D_t \setminus D_s| \cdot (d-1)$. This concludes the proof. □

As an immediate corollary of Lemma 3.8, we obtain the two following results about planar graphs and $K_\ell$-minor free graphs:

**Corollary 3.9.** *Let $G$ be a graph. Then, we have the following:*

- *if $G$ is planar, then $\mathcal{R}_k(G)$ is connected and has linear diameter for every $k \geq \Gamma(G) + 3$.*
- *if $G$ is $K_\ell$-minor free, then there exists a constant $C$ such that $\mathcal{R}_k(G)$ is connected and has linear diameter for every $k \geq \Gamma(G) + C\ell\sqrt{\log_2 \ell}$.*

*Proof.* Every minor of a planar graph is planar. Moreover every bipartite planar graph has at most $2n-4$ edges. Thus every planar graph is a 4-minor sparse graph and the first point follows from Lemma 3.8.

A result of Thomason [Tho84] (improving a result of Mader [Mad68]) ensures that the average degree of a $K_\ell$-minor free graph is at most $0.265 \cdot \ell\sqrt{\log_2 \ell}(1+o(1))$. In particular, there exists a constant $C$ such that, for every $\ell$ and every $K_\ell$-minor free graph $G$, the average degree of $G$ is at most $C\ell\sqrt{\log_2 \ell}$. Thus $G$ is $C\ell\sqrt{\log_2 \ell}$-minor sparse and the second point follows from Lemma 3.8. □

### 3.1.4 Bounded treewidth graphs

Let us now move on to the last result of this section which provides a new upper bound on $d_0$ depending on the upper domination number and the treewidth of $G$. More precisely, our result is the following:

**Theorem 3.10.** *Let $G=(V,E)$ be a graph. If $k = \Gamma(G) + tw(G) + 1$, then $\mathcal{R}_k(G)$ is connected. Moreover, the diameter of $\mathcal{R}_k(G)$ is at most $4(n+1)\cdot(tw(G)+1)$.*

*Proof.* Let $(X,T)$ be a tree decomposition of $G$ such that the maximum size of a bag of $X$ is $tw(G)+1$. Let $b = |X|$. We root the tree $T$ in an arbitrary bag, then set $X = \{X_1,\dots,X_b\}$, where for any $X_i, X_j$ such that $X_i$ is a child of $X_j$, we have $i<j$. In other words, $X_1,\dots,X_b$ is an elimination ordering of the (rooted) tree $T$ where at each step we remove a leaf of the remaining tree. We say that a bag $X_i$ is a *descendant* of $X_j$ if $X_j$ is on the path from the root to $X_i$ (in other words, $X_i$ belongs to the subtree rooted in $X_j$ in $T$). Note that, free to contract edges if a bag is included in another, we can assume $b \leq n$. We denote by $V_i$ the set of vertices that do not appear in the set of bags $\cup_{j=i+1}^{b} X_j$. We set $V_0 = \emptyset$.

Let $D_s$ and $D_t$ be two dominating sets. Free to first remove vertices from $D_s$ and $D_t$ if possible (which can be done in at most $2(tw(G)+1)$ operations in total), we can assume that $D_s$ and $D_t$ have size at most $\Gamma(G)$. Let $D$ be a minimum dominating set of $G$. Instead of proving directly that there exists a reconfiguration sequence from $D_s$ to $D_t$, we will prove that that there exists a reconfiguration sequence from $D_s$ to $D$ and from $D_t$ to $D$ of length at most $2n\cdot(tw(G)+1)$ each. Since the reverse of a reconfiguration sequence also is reconfiguration sequence, that will give a reconfiguration sequence of the desired length. So the rest of the proof is devoted to prove the following:

**Lemma 3.11.** *Let $G=(V,E)$ be a graph and let $D_s$ be a dominating set of $G$ of size at most $\Gamma(G)$ and $D$ be a minimum dominating set of $G$. If $k = \Gamma(G) + tw(G) + 1$, then there is a reconfiguration sequence from $D_s$ to $D$. Moreover, the length of this reconfiguration sequence is at most $2n\cdot(tw(G)+1)$.*

In order to prove Lemma 3.11, we prove that there exists a sequence $\langle D_1 = D_s, D_2, \dots, D_b\rangle$ of dominating sets such that, for every $j$, $D_j$ satisfies the following property $\mathcal{P}$:

(i) $D_j$ is a dominating set of $G$ of size at most $\Gamma(G)$,

(ii) For every $j>1$, there exists a transformation sequence of length at most $2(tw(G)+1)$ from $D_{j-1}$ to $D_j$ in $\mathcal{R}_k(G)$,

(iii) $D_j \cap V_{j-1} \subseteq D$. Recall that $V_{j-1}$ is the set of vertices that do not appear in the set of bags $\cup_{q=j}^{b} X_q$ so in other words, the vertices of $D_j$ that only belong to bags in $X_1 \cup \dots \cup X_{j-1}$ are also in $D$.

So that will provide a reconfiguration sequence in $\mathcal{R}_k(G)$ from $D_s$ to a dominating set $D_b$ sufficiently close to $D$ to ensure the existence of a transformation from $D_b$ to $D$ of length at most $2n \cdot (tw(G)+1)$. To prove the existence of the sequence, we use induction on $j$.

First note that since $D_s$ is a dominating set of $G$ of size at most $\Gamma(G)$ and $V_0$ is empty, $D_s$ satisfies property $\mathcal{P}$. Let us now show that if $D_j$ satisfies property $\mathcal{P}$, then there exists a set $D_{j+1}$ that satisfies property $\mathcal{P}$. A vertex $v$ is a *left vertex* (for $X_j$) if $v$ only appears in bags that are descendant of $X_j$. Note that by definition, $X_j$ is a descendant of itself. Otherwise, we say that $v$ is a *right vertex*. When no confusion is possible, we will omit the mention of $X_j$.

*Claim* 1. If a left vertex $u$ (for $X_j$) is adjacent to a right vertex $v$ (for $X_j$), then $v \in X_j$.

*Proof.* Since $u$ and $v$ are adjacent in $G$, there exists a bag $X_i$ which contains both $u$ and $v$. Note that since $u$ is a left vertex, $X_i$ is a descendant of $X_j$. Besides, since $v$ is a right vertex, there exists a bag $X_{i'}$ that contains $v$ and which is not a descendant of $X_i$. Since the set of bags that contain $v$ induces a connected tree, $v$ must belong to each bag on the unique path from $X_i$ to $X_{i'}$. In particular, $v \in X_j$. ◊

In order to construct $D_{j+1}$, we define the following subsets of vertices (see Figure 3.4 below):

- $A$ is the set of left vertices of $X_j \cap (D_j \setminus D)$. In other words, $A$ is the set of left vertices of $X_j$ that are in $D_j$ but not in $D$.
- $B$ is the set of right vertices of $X_j$. In other words, $B$ is the set of vertices of $X_j$ that also appear in a bag $X_{j'}$ with $j' > j$.
- $C$ is the set of left vertices of $D \setminus D_j$. In other words, $C$ is the set of vertices of $D$ at the left of $X_j$ that are missing in $D_j$.

We partition again $B$ into three parts:

- $B_1$ is the set of vertices of $B \setminus D$ that are dominated by $C$
- $B_2 = B \cap D_j$
- $B_3 = B \setminus (B_1 \cup B_2)$.

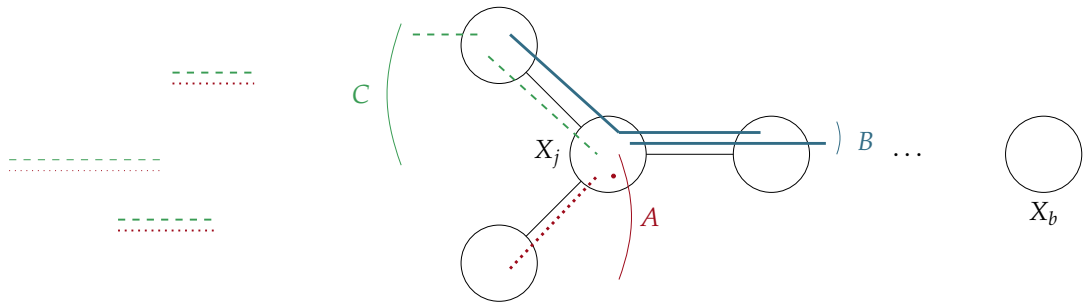



Figure 3.4 – The tree decomposition of $G$, and the sets $A$, $B$ and $C$. The circles represent the bags of the tree decomposition. The vertices are represented by lines, or dots, that go throughout the bags they belong to. The thick full lines represent the vertices of $B$, the dashed lines represent the vertices of $D$, and the dotted lines represent the vertices of $D_j$. By the induction hypothesis, the left vertices of $D_j$ that do not belong to $X_j$ belong to $D$.

We set $D'_j = (D_j \setminus A) \cup C \cup B_3$. Let us first prove that $D'_j$ is a dominating set of $G$.

*Claim* 2. The set $D'_j$ is a dominating set of $G$.

*Proof.* Since $D_j$ is a dominating set of $G$ and $D_j \setminus A \subseteq D'_j$, the only vertices that can be undominated in $D'_j$ are the ones dominated only by vertices of $A$ in $D_j$. Let $N_r(A)$ (resp. $N_l(A)$) be the right vertices (resp. left vertices) that are only dominated by $A$ in $D_j$. Note that $N_l(A)$ might contain vertices of $A$, while $N_r(A)$ does not, since by definition the vertices of $A$ are left vertices. Let us show that all the vertices in $N_r(A) \cup N_l(A)$ are dominated by $D'_j$.

We start with $N_r(A)$. Since the vertices of $A$ are left vertices and the vertices of $N_r(A)$ are right vertices, by Claim 1, we have $N_r(A) \subseteq X_j$. Since the vertices in $N_r(A)$ are right vertices, we have $N_r(A) \subseteq B$. Moreover, since every vertex of $N_r(A)$ is only dominated by $A$ in $D_j$ but does not belong to $A$, it is not in $D_j$ and thus not in $B_2$. Thus, the vertices of $N_r(A)$ either belong to $B_1$ (and are by definition dominated by $C$), or they belong to $B_3$. Therefore, $N_r(A)$ is dominated by $C \cup B_3$ and thus by $D'_j$.

Let us now focus on $N_l(A)$. In $D$, $N_l(A)$ is dominated by vertices that we partition into two sets: the right vertices $Y$ and the left vertices $Z$. We show that both $Y$ and $Z$ are included in $D'_j$, which implies that $D'_j$ dominates $N_l(A)$. Since the vertices of $N_l(A)$ are left vertices and the vertices of $Y$ are right vertices, Claim 1 gives $Y \subseteq X_j$. Thus, by definition, $Y \subseteq B$. Moreover, the vertices of $Y$ that belong to $D_j$ do not belong to $A$ as they are right vertices, and thus belong to $D_j \setminus A$, and the vertices of $Y$ that do not belong to $D_j$ belong by definition to $B \cap (D \setminus D_j) \subseteq B_3$. Thus, $Y \subseteq (D_j \setminus A) \cup B_3 \subseteq D'_j$. Finally, the vertices of $Z$ either belong to $D_j$ and thus by definition to $D_j \cap D \subseteq D_j \setminus A$, or they do not belong to $D_j$ and by definition they thus belong to $C$. Therefore, $Z \subseteq (D_j \setminus A) \cup C \subseteq D'_j$. Therefore, $N_l(A)$ is dominated by $D'_j$, which concludes the proof of this claim. ◊

Let us now prove the following:

*Claim* 3. $|D_j \cup C \cup B_3| \leq \Gamma(G) + tw(G) + 1$.

*Proof.* Let us first show that the set $D' = (D \setminus C) \cup A \cup B_1 \cup B_2$ is a dominating set of $G$. We will then explain how to exploit this property to prove that $|D_j \cup C \cup B_3| \leq \Gamma(G) + tw(G) + 1$.

Since $D$ is a dominating set, the only vertices that can be undominated in $(D \setminus C) \cup A \cup B_1 \cup B_2$ are vertices that are only dominated by $C$ in $D$. Let $N_r(C)$ (resp. $N_l(C)$) be the subset of right (resp. left) vertices that are only dominated by $C$ in $D$. Note that $N_l(C)$ might contain vertices of $C$ and $N_r(C)$ does not, since the vertices of $C$ are left vertices. We prove that $N_r(C)$ and $N_l(C)$ are dominated by $D'$.

We first prove that the vertices of $N_r(C)$ are dominated in $D'$. Since $C$ only contains left vertices and $N_r(C)$ only contains right vertices, Claim 1 ensures that $N_r(C) \subseteq X_j$. Thus, by definition of $B$, $N_r(C) \subseteq B$. Since the vertices of $N_r(C)$ are only dominated by $C$ in $D$, $N_r(C) \subseteq B_1$. Therefore $(D \setminus C) \cup A \cup B_1 \cup B_2$ dominates $N_r(C)$.

Let us now prove that $N_l(C)$ is dominated in $D'$. Every vertex $v \in N_l(C)$ is dominated in $D_j$ by either a right vertex or a left vertex. Assume that $v$ is dominated in $D_j$ by a right vertex $w$. Since $v$ is a left vertex and $w$ a right vertex, Claim 1 ensures that $w \in X_j$ and thus $w \in B$. Since $w \in D_j$, $w \in B_2 \subseteq D'$. Assume now that $v$ is dominated in $D_j$ by a left vertex $u$. If $u$ belongs to $D$, it is in $D \cap D_j \subseteq D \setminus C \subseteq D'$. So we can assume that $u \notin D$. By the induction hypothesis, $D_j$ satisfies (iii) and since $u \notin D$, the vertex $u$ necessarily belongs to $X_j$. So we finally have $u \in A$. Thus, $u \in (D \setminus C) \cup A \subseteq D'$. So $N_l(C)$ is dominated in $D'$. And then $D'$ is a dominates $G$.

We can now show that $|D_j \cup C \cup B_3| \leq \Gamma(G) + tw(G) + 1$. Since $D$ is a minimum dominating set of $G$ and $D' = (D \setminus C) \cup (A \cup B_1 \cup B_2)$ also is a dominating set of $G$, we have $|C| \leq |A \cup B_1 \cup B_2|$. Thus, $|C \cup B_3| \leq |A| + |B_1 \cup B_2| + |B_3|$. But $A$, $B_1 \cup B_2$ and $B_3$ are pairwise disjoint subsets of $X_j$. Thus, $|A| + |B_1 \cup B_2| + |B_3| \leq |X_j| \leq tw(G) + 1$, and $|C \cup B_3| \leq tw(G) + 1$. Since, by the induction hypothesis, $D_j$ has size at most $\Gamma(G)$, this gives $|D_j \cup C \cup B_3| \leq \Gamma(G) + tw(G) + 1$. ◊

We now have a reconfiguration sequence of length at most $tw(G)+1$ from $D_j$ to $D'_j$ by simply adding all the vertices of $C \cup B_3$ and then removing all the vertices of $A$. All along the sequence, the corresponding set is dominating. Indeed, it contains $D_j$ during the first part and $D'_j$ during the second one. One is dominating by assumption and the other is dominating by Claim 2. By Claim 3, this reconfiguration sequence exists in $\mathcal{R}_{\Gamma(G)+tw(G)+1}(G)$.

The dominating set $D_{j+1}$ will be any dominating set of size at most $\Gamma(G)$ obtained from $D'_j$ by removing vertices, i.e., any dominating set $D_{j+1}$ satisfying $D_{j+1} \subseteq D'_j$ and $|D_{j+1}| = \Gamma(G)$, which necessarily exists by definition of $\Gamma(G)$. This can be done in at most $tw(G)+1$ deletions. Thus, there exist a sequence in $\mathcal{R}_{\Gamma(G)+tw(G)+1}(G)$ from $D_j$ to $D_{j+1}$ of length at most $2(tw(G)+1)$, and $D_{j+1}$ thus satisfies (i) and (ii). Let us now justify why $D_{j+1}$ satisfies (iii).

Since $D_{j+1}$ is a subset of $D'_j$, if (iii) holds for $D'_j$ it holds for $D_{j+1}$. We have $D'_j = (D_j \setminus A) \cup C \cup B_3$. Since $C \subseteq D$, if a left vertex $v$ (for $X_j$) appears in $D'_j$ but not in $D$, it is either in $D_j \setminus A$ or in $B_3$. Since $B_3$ only contains right vertices, it must be in $D_j \setminus A$. Since $A$ contains the left vertices of $X_j \cap (D_j \setminus D)$, it means that $v$ should be in $V_{j-1}$. But, by the induction hypothesis, the vertices of $D_j$ that belong to $V_{j-1}$ belong to $D$. So $v$ does not exist and $D'_j$ satisfies (iii). Thus, $D_{j+1}$ satisfies property $\mathcal{P}$, and by induction, there exists a set $D_b$ that satisfies property $\mathcal{P}$. Moreover, since for any $i$ such that $2 \leq i \leq b$, there is a path of length at most $2(tw(G)+1)$ from $D_{i-1}$ to $D_i$ in $\mathcal{R}_k(G)$, there is a transformation of length at most $2(b-1)\cdot(tw(G)+1)$ from $D_s$ to $D_b$ in $\mathcal{R}_k(G)$.

To complete the construction of a path from $D_s$ to $D$ in $\mathcal{R}_k(G)$, we show that there exists a transformation from $D_b$ to $D$ in $\mathcal{R}_k(G)$ of length at most $2(tw(G)+1)$. Let $A' = D_b \setminus D$, and $C' = D \setminus D_b$. We have $D = (D_b \cup C') \setminus A'$. Let $S'_1$ be the reconfiguration sequence from $D_b$ to $D_b \cup C'$ which consists in adding one by one every vertex of $C'$. Since each of the sets of $S'_1$ contains $D_b$, they are all dominating sets of $G$. Note that $S'_1$ has length $|C'|$. Let $S'_2$ be the reconfiguration sequence from $D_b \cup C'$ to $D$ which consists in removing one by one each vertex of $A'$. Since each of the sets of $S'_2$ contains $D$, they all are dominating sets. Note that $S'_2$ has length $|A'|$. Thus, applying $S'_1$ then $S'_2$ gives a reconfiguration sequence from $D_b$ to $D$ of length $|C'|+|A'|$. Moreover, the maximum size of a dominating set reached in this sequence is $|D_b \cup C'|$. Let us show that $|D_b \cup C'| \leq \Gamma(G)+tw(G)+1$. We have $D_b = (D \setminus C') \cup A'$. Thus, since $D$ is a minimum dominating set, $|C'| \leq |A'|$. Since $D_b$ satisfies (iii), every vertex of $D_b$ that does not belong to $X_b$ also belongs to $D$. Thus, $A' \subseteq X_b$, and $|A'| \leq tw(G)+1$, which gives $|C'| \leq tw(G)+1$, as well as $|C'|+|A'| \leq 2(tw(G)+1)$. Since $D_b$ is a minimal dominating set of $G$, we have therefore $|D_b \cup C'| \leq \Gamma(G)+tw(G)+1$. Thus, there is a path of length at most $2(tw(G)+1)$ from $D_b$ to $D$ in $\mathcal{R}_k(G)$ which completes the transformation of length at most $2b\cdot(tw(G)+1)$ from $D_s$ to $D$ in $\mathcal{R}_k(G)$. Since $b \leq n$, the conclusion follows. □

### 3.1.5 Concluding remarks

In this section, we focused on the connectivity of the reconfiguration graph of dominating sets under the token and addition rule. More precisely, we first improved a result by Haas and Seyffarth [HS17] by showing that $\mathcal{R}_k(G)$ is connected and has linear diameter whenever $k \geq \Gamma(G)+\alpha(G)-1$. Moreover, the proof is constructive and outputs a transformation between two dominating sets in polynomial time.

We then studied the class of $d$-minor sparse graphs which contains in particular planar graphs. We showed that if $G$ is a $d$-minor sparse graph, then $\mathcal{R}_k(G)$ is connected and has linear diameter whenever $k \geq \Gamma(G)+d-1$. Since planar graphs are 4-minor sparse graphs, it follows that $\mathcal{R}_k(G)$ is connected and has linear diameter for any $k \geq \Gamma(G)+3$. This result is almost tight. Indeed, Suzuki et al. [SMN14] found a planar graph $G$ for which $\mathcal{R}_k(G)$ is disconnected if $k = \Gamma(G)+1$ (see Figure 3.5).

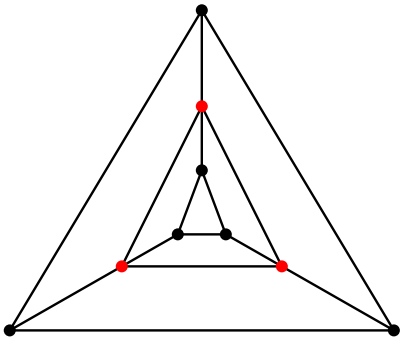


Figure 3.5 – The planar graph $G$ such that $\mathcal{R}_{\Gamma+1}(G)$ is not connected.

It is easily seen that $\Gamma(G) = 3$. Moreover, if we consider the dominating set depicted by the red vertices, in order to remove a vertex, we must add the two black vertices it is adjacent to, thus reaching a dominating set of size $\Gamma(G) + 2$. However, we were not able to find a planar graph $G$ for which $\mathcal{R}_{\Gamma+2}(G)$ is not connected and thus we conjecture the following:

**Conjecture 3.12.** *Let $G$ be a planar graph. If $k \geq \Gamma(G) + 2$, then $\mathcal{R}_k(G)$ is connected.*

Finally, we considered an upper bound on $d_0$ depending on the treewidth of a graph. More precisely, we showed that if $k \geq \Gamma(G) + tw(G) + 1$, then $\mathcal{R}_k(G)$ is connected and has linear diameter. We claim that this bound is tight up to an additive constant factor. Mynhardt et al. [MTR19] constructed an infinite family of graphs $G_{\ell,r}$ (with $\ell \geq 3$ and $1 \leq r \leq \ell - 1$) for which $2\Gamma(G) - 1$ tokens are necessary to guarantee the connectivity of the reconfiguration graph. Let us describe their construction when $r = \ell - 1$. The graph $G_{\ell,\ell-1}$ contains $\ell - 1$ cliques $C_1, C_2, \ldots, C_{\ell-1}$ called *inner cliques*, each of size $\ell$. We denote by $c_i^j$ the $j$-th vertex of the clique $C_i$. We then add a new clique $C_0$ of size $\ell$, called the *outer clique* and we add a new vertex $u_0$ adjacent to all the vertices of $C_0$ (hence, $C_0$ can be seen as a clique of size $\ell + 1$). For every $1 \leq i \leq \ell - 1$ and for every $1 \leq j \leq \ell$, we add an edge between $c_i^j$ and $c_0^j$. This completes the construction of $G_{\ell,\ell-1}$ (see Figure 3.6 for an example).

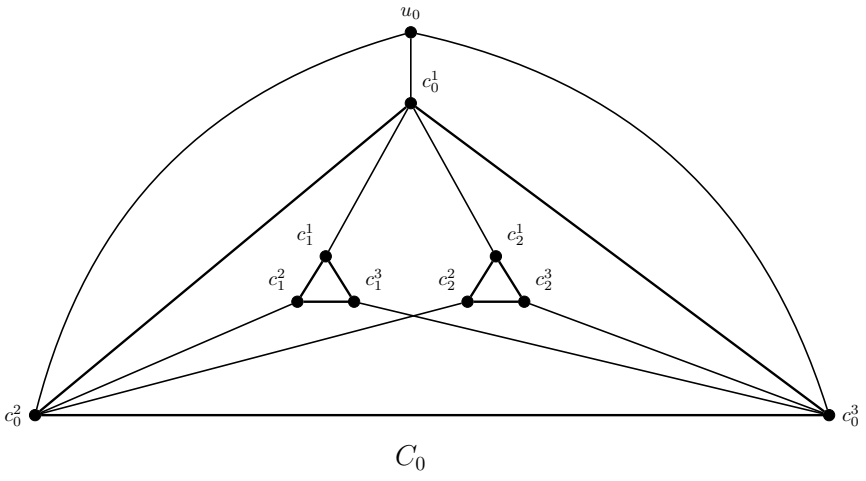



Figure 3.6 – The graph $G_{3,2}$

Mynhardt et al. [MTR19] showed that $G_{\ell,\ell-1}$ satisfies $\Gamma(G_{\ell,\ell-1}) = \ell$. They moreover proved that $\mathcal{R}_{2\ell-2}(G_{\ell,\ell-1})$ is not connected. Let us show that $G_{\ell,\ell-1}$ has treewidth $\ell$:

**Proposition 3.13.** *The graph $G_{\ell,\ell-1}$ has treewidth $\ell$.*

*Proof.* First, observe that $tw(G_{\ell,\ell-1}) \geq \ell$ since $G[C_0 \cup \{u_0\}]$ is a clique of size $\ell+1$. Let us now give a tree decomposition of $G_{\ell,\ell-1}$ of width $\ell$. We first create a "central" bag $B_0$ containing all the vertices of $C_0$ and the vertex $u_0$. For each inner clique $C_i$ with $1 \leq i \leq \ell-1$, we attach to $B_0$ a path $B_i^1 B_i^2 \cdots B_i^\ell$ where $B_i^j$ contains the vertices $(C_0 \setminus \bigcup_{k=0}^{j-1} c_0^k) \cup \bigcup_{k=1}^{j} c_i^k$ (see Figure 3.7 for an example). Observe that for any $1 \leq i \leq \ell-1$, the bag $B_i^\ell$ contains all the vertices of $C_i$. And the bag $B_i^j$ contains both $c_0^j$ and $c_i^j$. Hence, each edge is contained in at least one bag. For every $1 \leq j \leq \ell$, the vertex $c_0^j$ is contained in the bags $B_0 \cup \bigcup_{i=1}^{\ell-1} \bigcup_{k=1}^{j} B_i^k$. And for every $1 \leq i \leq \ell-1$ and every $1 \leq j \leq \ell$, the vertex $c_i^j$ is contained in $B_i^1, B_i^2, \ldots, B_i^j$. It follows that for every vertex $u \in V(G_{\ell,\ell-1})$ the set of bags containing $u$ induces a connected subtree. Finally, one can easily check that each bag contains exactly $\ell+1$ vertices. Hence, this decomposition indeed is a tree decomposition of $G_{\ell,\ell-1}$ of width $\ell$ and the conclusions follows. ◊

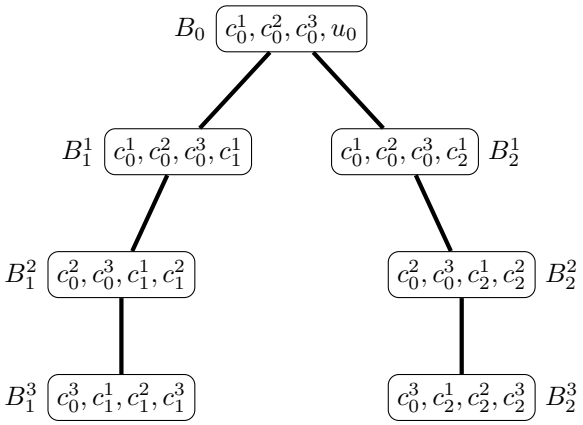



Figure 3.7 – Tree decomposition of $G_{3,2}$ of width $tw(G_{3,2})$.

Hence, the reconfiguration graph $\mathcal{R}_{\Gamma(G)+tw(G)-2}$ is not necessarily connected and thus our function of the treewidth is tight up to an additive constant factor. Moreover, we have the following about the pathwidth of $G_{\ell,\ell-1}$:

**Proposition 3.14.** *The pathwidth of $G_{\ell,\ell-1}$ is at most $2\ell-1$.*

*Proof.* We give a path decomposition of width at most $2\ell-1$ of $G_{\ell,\ell-1}$. We first create a bag $B_0$ which contains $C_0 \cup \{u_0\}$. For every $1 \leq i \leq \ell-1$, we create a bag $B_i = C_0 \cup C_i$ such that $B_1 B_2 \ldots B_{\ell-1}$ induces a path. One can easily check that it is a path decomposition of width $2\ell-1$ of $G_{\ell,\ell-1}$. ◊

However, it is not clear if and how we can obtain a better upper bound for bounded pathwidth graphs. To sum up, $\mathcal{R}_k(G)$ is not necessarily connected if $k < \Gamma(G) + pw(G)/2 + O(1)$ and is connected if $k > \Gamma(G) + pw(G) + 1$. We were not able to close this gap and left it as an open problem as well.

## 3.2 Complexity under Token Sliding

### 3.2.1 Introduction

In this section, we study the reachability of dominating sets under token sliding from a graph classes perspective. This is joint work with Marthe Bonamy and Paul Dorbec [BDO21]. The complexity of the reconfiguration of dominating sets was initiated by the work of Haddadan

et al. [HIM+16] who studied the reachability of dominating set under the TAR rule. More precisely, they considered the complexity of the following problem that we denote by $\mathsf{DSR}_{\mathsf{TAR}}$ (for DOMINATING SET RECONFIGURATION under TAR):

$\mathsf{DSR}_{\mathsf{TAR}}$

**Instance:** A graph $G$, two dominating sets $D_s, D_t$ of $G$, an integer $k \geq \max\{|D_s|, |D_t|\}$.
**Question:** Is there a TAR($k$)-sequence that transforms $D_s$ into $D_t$?

First, recall that if there is a reconfiguration sequence $\langle D_0 = D_s, D_1, \ldots, D_{\ell-1}, D_\ell = D_t \rangle$ from $D_s$ to $D_t$ (no matter what the reconfiguration rule is), then $\langle D_\ell = D_t, D_{\ell-1}, \ldots, D_1, D_0 = D_s \rangle$ is a reconfiguration sequence from $D_t$ to $D_s$. We denote this by $D_s \leftrightsquigarrow D_t$.

Let $(G, D_s, D_t, k)$ be an instance of $\mathsf{DSR}_{\mathsf{TAR}}$. Haddadan et al. [HIM+16] showed that the problem $\mathsf{DSR}_{\mathsf{TAR}}$ is PSPACE-complete, even if $G$ is bipartite, a split graph, a planar graph with maximum degree six, or has bounded pathwidth. On the other hand, they proved that the problem is linear-time solvable if $G$ is a tree, a cograph or an interval graph. Interestingly, all these positive results were obtained by applying the same strategy. Indeed, they introduced the concept of canonical dominating set:

**Definition 3.15** (Canonical dominating set). *A dominating set $C$ of a graph $G$ is canonical if it is both minimum and reachable from any dominating set $D$ of $G$ via a* TAR($|D|+1$)*-sequence.*

They showed that each of these three graph classes admits a canonical dominating set. This implies that if $G$ is an interval graph, a cograph or a tree, then $(G, D_s, D_t, k)$ is a yes-instance whenever $k \geq \max\{|D_s|, |D_t|\} + 1$. Indeed, we can transform both $D_s$ and $D_t$ into the canonical dominating set $C$, i.e., $D_s \leftrightsquigarrow C \leftrightsquigarrow D_t$ is a reconfiguration sequence between $D_s$ and $D_t$. Finally, if $k = \max\{|D_s|, |D_t|\}$, then $(G, D_s, D_t, k)$ is a yes-instance if and only if $D_s$ and $D_t$ are not minimal dominating set of $G$. Since one can check in linear time whether a dominating set is minimal or not, $\mathsf{DSR}_{\mathsf{TAR}}$ is linear-time solvable on each graph class that admits a canonical dominating set.

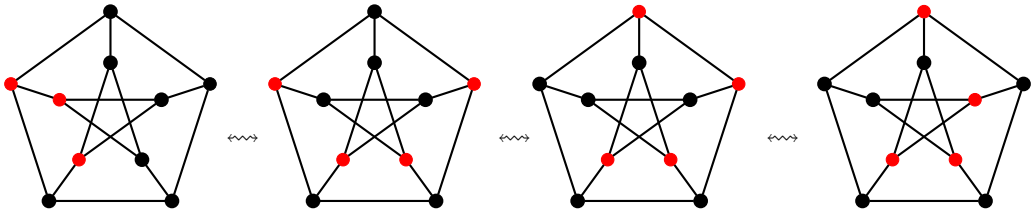
Mouawad et al. [MNR+17] studied the parameterized complexity of the reconfiguration of dominating sets under token addition and removal. They proved that this problem is W[2]-hard when parameterized by $k + \ell$, where $k$ is the threshold and $\ell$ the length of the reconfiguration sequence. As a positive result, Lokshtanov et al. [LMP+18] gave a fixed-parameter algorithm with respect to $k$ for graphs excluding $K_{d,d}$ as a subgraph, for any constant $d$. More recently, Lokshtanov et al. [LMPS19] studied the parameterized complexity of the reconfiguration of connected dominating sets. They showed that this problem is W[2]-hard when parameterized by $k + \ell$, even if the input graph is 5-degenerate.

As we said, we focus on the complexity of the reachability question of DOMINATING SET RECONFIGURATION under token sliding. This reconfiguration rule has already been studied for various reconfiguration problems but not for dominating sets, to the best of our knowledge. In this model, a natural question is whether we should authorize more than one token to be placed on a vertex during the reconfiguration sequence. Here is an example where it makes a difference: consider the star graph $K_{1,n}$ on $n+1$ vertices and two dominating sets $D_1$ and $D_2$ of $K_{1,n}$ of size $k$, with $k \in [2, n-1]$. Any dominating set of that size necessarily contains the central vertex. To reconfigure $D_1$ into $D_2$, we are forced to move a token from one leaf to another, which can only be done by going through the central vertex which already contains a token. Given such artificially negative examples, we choose to allow the superposition of tokens on a vertex. Note that this question did not arise in previous works considering the

token sliding model, to the best of our knowledge. Indeed, for problems like independent set, there can be no question of superposing two tokens, as two tokens cannot be adjacent in the first place. In the aforementioned results considering token sliding for dominating sets (see Section 3.1.1), they exclusively consider that model in the case of minimum dominating sets: if superposition was an option, there would be a smaller dominating set, which is impossible.

Let $G$ be a graph, $D_s$ and $D_t$ be two dominating sets of $G$ of same size $k$. We say that $D_s$ is reconfigurable into $D_t$ by token sliding if there exists a TS-sequence $\langle D_0 = D_s, D_1, \ldots, D_{\ell-1}, D_\ell = D_t \rangle$ that satisfies the two following properties (see Figure 3.8):

- each $D_i$ is a multiset of size $k$ that is a dominating set of $G$;
- there exists an edge $uv$ such that $D_{i+1} = (D_i \setminus \{u\}) \cup \{v\}$, i.e., we slide the token placed on the vertex $u$ along the edge $uv$. We denote this move by $u \overset{\mathsf{TS}}{\rightsquigarrow} v$.



(a) $D_0 = D_s$ (b) $D_1$ (c) $D_2$ (d) $D_3 = D_t$

Figure 3.8 – Example of TS-sequence from $D_s$ to $D_t$.

We are now ready to define more formally the DOMINATING SET RECONFIGURATION problem under token sliding, denoted by $\mathrm{DSR}_{\mathsf{TS}}$:

**$\mathrm{DSR}_{\mathsf{TS}}$**

**Instance:** A graph $G = (V, E)$ and two dominating sets $D_s$ and $D_t$ of cardinality $k$ of $G$.
**Question:** Is there a TS-sequence between $D_s$ and $D_t$, i.e., does $D_s \overset{\mathsf{TS}}{\leftrightsquigarrow} D_t$ hold?

We first obtain a similar result as the one obtained by Haas and Seyffarth [HS14] regarding the connectivity of the reconfiguration graph under TAR. More precisely, we show that being reconfigurable is not a monotone property also for the token sliding model.

**Theorem 3.16.** *For every $\ell \geq 3$, there exists a graph $G_\ell$ where, for every $k < \ell$, every dominating set of size $k$ can be reconfigured into any other, while there are two dominating sets of size $\ell$ that cannot be reconfigured one into the other.*

*Proof.* We first prove the statement for $k = 2$. For every integer $\ell > 2$, we define the graph $G_\ell$ such that $G_\ell$ contains exactly one dominating set of size $\gamma(G) = 2$ but for which the dominating sets of size $\ell$ are not reconfigurable. To construct $G_\ell$, we first create $\ell$ pairs of triangles $\{(G_1^i, G_2^i), \ldots, (G_1^\ell, G_2^\ell)\}$ such that $G_1^i$ and $G_2^i$ share exactly one vertex $w_i$. Moreover, let all the $G_1^i$'s share a vertex $u$ and all the $G_2^i$'s share a vertex $v$ (see Figure 3.9 for $G_3$ as an example). Note that we have $\gamma(G_\ell) = 2$ since $\{u, v\}$ is a dominating set and $G_\ell$ does not contain a universal vertex (i.e., a vertex adjacent to all the other vertices).

Consider the dominating set $D_s = \{w_1, \ldots, w_\ell\}$. It is a dominating set of $G_\ell$ of size $\ell$. By token sliding, $D_s$ cannot be reconfigured into any other dominating set of size $\ell$. Indeed, in $D_s$ we cannot move any $w_i$ in a triangle because it would leave the other triangle of the pair

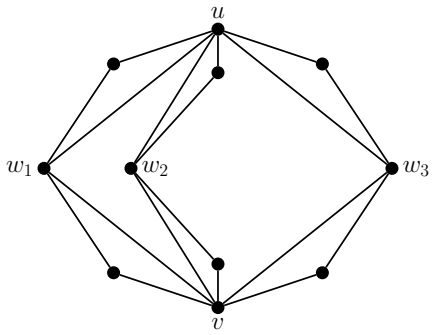



Figure 3.9 – The graph $G_3$ from Theorem 3.16.

$(G_1^i, G_2^i)$ not dominated. Note that any set of $\ell$ vertices containing $u$ and $v$ is a dominating set of $G_\ell$, hence the existence of two dominating sets of size $\ell$ as desired.

Consider now $k < \ell$. Any dominating set of $G_\ell$ on fewer than $\ell$ vertices contains both $u$ and $v$. Indeed, if for instance $u$ is not in the dominating set, then $\ell$ extra vertices are necessary to dominate the triangles $G_1^i$. Therefore, any dominating set $D$ of $G_\ell$ on $k$ vertices contains both $u$ and $v$. The other vertices are therefore not necessary for domination purposes, and we can slide them around as desired, superposing them with $u$ and $v$ arbitrarily. There are many dominating sets on $k$ vertices, but they all contain $u$ and $v$ and can be trivially reconfigured one into another. □

### 3.2.2 PSPACE-completeness results

In this section, we study the complexity of $\mathrm{DSR}_{\mathsf{TS}}$ in the general case. We show that this problem is PSPACE-complete, even when restricted to split graphs, bipartite graphs or bounded treewidth graphs. Let us first recall the following result from Haddadan et al. [HIM+16], stating the complexity of the reconfiguration problem for the TAR model.

**Theorem 3.17** ([HIM+16]). *Let $G$ be a graph and $D_\mathsf{s}, D_\mathsf{t}$ be two dominating sets of $G$ of size $k$. Deciding whether $D_\mathsf{s} \overset{\mathsf{TAR}}{\leftrightsquigarrow} D_\mathsf{t}$ is PSPACE-complete.*

Note that the problem remains PSPACE-complete, even if the input graph is a planar graph with maximum degree six, has bounded bandwidth, is bipartite or is a split graph as discussed previously. Recall also the following result by Haas and Seyffarth [HS14]:

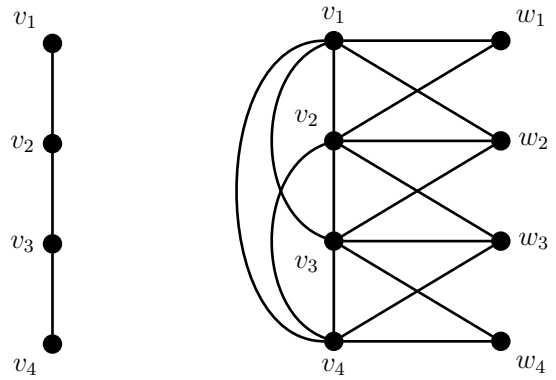
**Theorem 3.18** ([HS14]). *Let $G$ be a graph, and let $D_\mathsf{s}$ and $D_\mathsf{t}$ be two dominating sets of $G$ both of size $k$. Then, $D_\mathsf{s} \overset{\mathsf{TJ}}{\leftrightsquigarrow} D_\mathsf{t}$ holds if and only if $D_\mathsf{s} \overset{\mathsf{TAR}}{\leftrightsquigarrow}$ holds with threshold $k+1$.*

As a corollary of Theorem 3.17 and Theorem 3.18, we obtain that deciding whether two dominating sets of size $k$ of a graph $G$ can be reconfigured under the token jumping model is a PSPACE-complete problem. We are now ready to prove Theorem 3.19.

**Theorem 3.19.** *$\mathrm{DSR}_{\mathsf{TS}}$ is PSPACE-complete on split graphs.*

*Proof.* First, note that the problem is in PSPACE [IDH+08]. We give a polynomial-time reduction from $\mathrm{DSR}_{\mathsf{TJ}}$, which is PSPACE-complete as discussed above. Let $G = (V, E)$ be a graph with $V(G) = \{v_1, \ldots, v_n\}$. We construct the corresponding split graph $G'$ as follows; the construction is the one given in [CN84, Ber84] (see Section 2.1.4).

- $V(G') = V_1 \cup V_2$ where $V_1 = \{v_1, \ldots, v_n\}$ and $V_2 = \{w_1, \ldots, w_n\}$;
- $E(G') = \{uv \mid u, v \in V_1\} \cup \{v_i w_j \mid v_j \in N_G[v_i]\}$, i.e., we add all possible edges in $V_1$ so that $V_1$ forms a clique. We also add an edge between a vertex $v_i \in V_1$ and a vertex $w_j \in V_2$ if and only if the corresponding vertex $v_j$ in the original graph $G$ belongs to the closed neighborhood of $v_i$ in $G$.




(a) $G$ (b) $G'$

Figure 3.10 – Example for the reduction of Theorem 3.19.

Observe that $G'$ is a split graph since $V_1$ forms a clique and $V_2$ an independent set (see Figure 3.10 for an example). To a set of vertices of $G$, we associate the corresponding vertices of $V_1$ in $G'$. By definition of $G'$, any dominating set $D$ of $G$ is also a dominating set for $G'$: indeed, a vertex $v_i \in V_1$ dominates all the vertices in $V_1$ (since it is a clique) and all the vertices in $V_2$ that correspond to vertices in its closed neighborhood in $G$. That $D$ dominates $G$ allows us to conclude that the corresponding set also dominates $V_2$. Hence, $D$ is also a dominating set of $G'$.

Let $(G, D_s, D_t)$ be an instance of $\mathrm{DSR}_{\mathsf{TJ}}$; we reduce this instance to the instance of $\mathrm{DSR}_{\mathsf{TS}}$ $(G', D_s, D_t)$. This reduction can be done in quadratic time. It remains to prove that $D_s \overset{\mathsf{TS}}{\leftrightsquigarrow} D_t$ holds in $G'$ if and only $D_s \overset{\mathsf{TJ}}{\leftrightsquigarrow} D_t$ holds in $G$.

($\Leftarrow$) Consider a TJ-sequence in $G$, and translate it to $G'$. All intermediate sets still are dominating sets, and since all pairs of vertices are joined by an edge in $V_1$, this sequence is a valid TS-sequence in $G'$.

($\Rightarrow$) We now prove the other direction. Let $\langle D_0 = D_s, \ldots, D_p = D_t \rangle$ be a TS-sequence in $G'$. First, observe that any dominating $D'$ of $G'$ such that $D' \subseteq V_1$ corresponds to a dominating set of $G$. Indeed, any vertex $w_j \in V_2$ is dominated by a vertex $v_i \in V_1$ and by construction of $G'$, $v_i v_j \in E(G)$. Hence, $v_j$ is dominated by $v_i$ and thus $D'$ is also a dominating set of $G$. Hence, if the sequence does not use vertices in $V_2$, we immediately obtain a TJ-sequence in $G$ from $D_s$ to $D_t$, as the token jumping model does not require adjacency. Suppose on the other hand that the sequence goes through some vertices in $V_2$. Since all vertices are initially in $V_1$, there is a subsequence that contains a move $v_i \overset{\mathsf{TS}}{\rightsquigarrow} w_j$. Since $w_j \notin V_1$, there exists a later step where the token on $w_j$ is moved to an adjacent vertex $v_k$ in $V_1$ (since $V_2$ is independent). However, $w_j$ does not dominate any vertex in $V_2$ (since $V_2$ is a stable set) and thus $N[w_j] \subseteq N[v_k]$. Therefore, we simply replace these two moves by a single move $v_i \overset{\mathsf{TS}}{\rightsquigarrow} v_k$. We can thus assume that the reconfiguration sequence only uses vertices in $V_1$. The conclusion follows. □

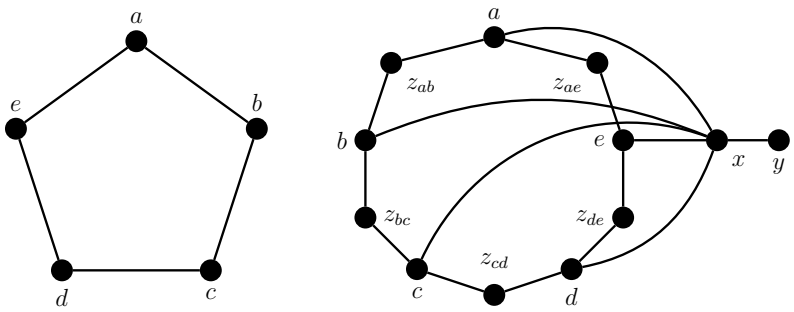
Next, we prove that $\mathrm{DSR}_{\mathsf{TS}}$ is PSPACE-complete on bipartite graphs. We use a reduction from Vertex Cover Reconfiguration under token sliding (or $\mathrm{VCR}_{\mathsf{TS}}$ for short). Recall that

a vertex cover is a set of vertices such that every edge has an endpoint in the set. The complement of a vertex cover is an independent set whose reconfiguration is known to be PSPACE-complete on planar graphs of maximum degree three [HD05, BC09] or on bounded bandwidth graphs [Wro18]. Hence, $\text{VCR}_{\text{TS}}$ is PSPACE-complete, even when restricted to these two classes.

**Theorem 3.20.** *$\text{DSR}_{\text{TS}}$ is PSPACE-complete on bipartite graphs.*

*Proof.* We give a polynomial-time reduction from $\text{VCR}_{\text{TS}}$. This is an adaptation of the well-known reduction from VERTEX COVER to DOMINATING SET [GJ79]. Let $G = (V, E)$ be a graph. We construct the corresponding bipartite graph $G' = (V_1 \uplus V_2, E')$ as follows: for each edge $uv \in E$, add $u$ and $v$ to $V_1$ and a new vertex $z_{uv}$ of degree two to $V_2$ that is adjacent to exactly $u$ and $v$. Note that $E'$ does not contain the edge $uv$ so that $V_1$ induces an independent set. Finally, add to $V_2$ a vertex $x$ adjacent to all the vertices in $V_1$ and attach to $x$ a degree-one vertex $y$ which is added to $V_1$ (see Figure 3.11 for an example). Formally, the graph $G'$ is the following:

- $V(G') = V_1 \cup V_2$ where $V_1 = V(G) \cup \{y\}$ and $V_2 = \{z_{uv} \mid uv \in E\} \cup \{x\}$;
- $E' = \{uz_{uv} \text{ and } z_{uv}v \mid u, v \in V_1 \text{ and } z_{uv} \in V_2\} \cup \{xv \mid v \in V_1\} \cup \{xy\}$.




(a) $G$ (b) $G'$

Figure 3.11 – Example for the reduction of Theorem 3.20.

Observe that $G'$ is bipartite and the reduction can be done in polynomial time. We now prove that the vertex covers of $G$ of size $k$ are reconfigurable if and only if the dominating sets of $G'$ of size $k+1$ are. Let $(G, C_s, C_t)$ be an instance for the $\text{VCR}_{\text{TS}}$ problem. We define the corresponding instance for the $\text{DSR}_{\text{TS}}$ problem as $(G', C_s \cup \{x\}, C_t \cup \{x\})$. Since $C_s$ is a vertex cover of $G$, for every edge $uv \in E(G)$ we have $\{u, v\} \cap C_s \neq \emptyset$ and thus the vertices $u, v, z_{uv}$ are dominated by $C_s$ in $G'$. Now $x$ dominates both $x$ and $y$, so $D_s = C_s \cup \{x\}$ is a dominating set of $G'$, and by the same argument, so is $D_t = C_t \cup \{x\}$. Since $\text{VCR}_{\text{TS}}$ and $\text{DSR}_{\text{TS}}$ both employ the same reconfiguration rule, we simply denote by $u \rightsquigarrow v$ a move of a reconfiguration sequence between $C_s$ and $C_t$ (respectively $D_s$ and $D_t$).

($\Rightarrow$) We start with the only if direction. First, it immediately follows from the definition of $D_s$ and $D_t$ that $D_s \setminus \{x\} = C_s$ and $D_t \setminus \{x\} = C_t$. Let us assume that $(G, C_s, C_t)$ is a yes-instance for the $\text{VCR}_{\text{TS}}$ problem. Then, there exists a reconfiguration sequence $S$ using the token sliding model between $C_s$ and $C_t$. One can construct a sequence $S'$ for $G'$ by replacing a move $u \rightsquigarrow v$ (where $uv \in E(G)$) of $S$ into two moves: $u \rightsquigarrow z_{uv}$ followed by $z_{uv} \rightsquigarrow v$. We need to prove that the domination property is preserved at every step. First, observe that each intermediate solution contains $x$, so each move of the form $z_{uv} \rightsquigarrow v$ is safe because $u$ is still dominated by $x$ and $z_{uv}$ by $v$. Therefore, the only risk is to leave some vertex $z_{wu}$ non dominated after a move $u \rightsquigarrow z_{uv}$. In that case, this implies that $w$ does not belong to the solution, which in turn means that the edges $wu$ and $uv$ are covered only by $u$. Therefore, the move $u \rightsquigarrow v$ of the sequence $S$

is not valid (because the edge $wu$ is no longer covered), a contradiction. Therefore, $(G', D_s, D_t)$ is a yes-instance for the $\mathrm{DSR}_{\mathsf{TS}}$ problem.

($\Leftarrow$) It remains to prove the if direction. Suppose that $(G', D_s, D_t)$ is a yes-instance for the $\mathrm{DSR}_{\mathsf{TS}}$ problem. Then, there exists a reconfiguration sequence $S' = \langle D_s, \ldots, D_t \rangle$ in $G'$. First, observe that at each step, $y$ needs to be dominated and thus either $x$ or $y$ belongs to each solution. Moreover, initially, $D_s$ does not contain $y$. We can ignore all moves of the form $x \rightsquigarrow y$ (each such move will be eventually followed by a move $y \rightsquigarrow x$), and assume that $x$ contains at least one token in each solution. Therefore, the only vertices whose domination is not immediate by the existence of a token on $x$ are the vertices of the form $z_{uv}$, i.e., the vertices that correspond to the edges of $G$. Recall that $D_s = C_s \cup \{x\}$, so every vertex in $D_s \setminus \{x\}$ belongs to $V(G') \cap V(G)$. We consider in turn the two other possible moves $u \rightsquigarrow v$, where $u \in V(G) \cap V(G')$ (i.e., $u$ corresponds to a vertex of the original graph $G$), and $v$ either belongs to $V(G') \setminus V(G)$ or $v = x$. We focus on the next operation (which may not be consecutive) that touches the vertex $v$. Suppose first that $v \in V(G') \setminus V(G)$, i.e., $v$ corresponds to a vertex $z_{uu'}$ for some vertex $u' \in N_G[u]$. If the next move that touches $z_{uu'}$ is $z_{uu'} \rightsquigarrow u$, these two operations can be ignored. Otherwise, since $z_{uu'}$ has degree two, the next operation that touches $z_{uu'}$ is $z_{uu'} \rightsquigarrow u'$. Moreover, we claim that we can assume that $z_{uu'} \rightsquigarrow v$ is the operation that immediately follows the move $u \rightsquigarrow z_{uu'}$. Indeed, $N_{G'}[z_{uu'}] \subseteq N_{G'}[u]$ so if a dominating set $D$ contains $z_{uu'}$, $D' = (D \setminus \{z_{uu'}\}) \cup \{u'\}$ is also a dominating set of $G'$. So one can assume that in $S'$, if we have a move $u \rightsquigarrow z_{uu'}$, it will be immediately followed by a move $z_{uu'} \rightsquigarrow u'$. In that case, one can replace these two moves by $u \rightsquigarrow u'$ in a reconfiguration sequence from $C_s$ to $C_t$. Let us now consider the other possible move: $u \rightsquigarrow x$. If the next move that touches $x$ is $x \rightsquigarrow u$, we again simply ignore these two steps. Let $D_i$ be the dominating set of $S'$ to which the move $u \rightsquigarrow x$ is applied. Recall that when a token is moved from a vertex $a$ to a vertex $z_{ab}$ (for some neighbor $b$ of $a$), it is followed by $z_{ab} \rightsquigarrow b$. Therefore, we know that $D_i$ does not contain any vertex of the form $z_{uv}$. So for every edge $uu'$ incident to $u$, $D_{i+1}$ must contain $u$ (this is possible if $u$ has at least two tokens in $D_i$) or $u'$ in order to dominate $z_{uu'}$. Hence, $C_{i+1} = D_{i+1} \setminus \{x\}$ is a vertex cover of $G$. If the next move that touches $x$ is $x \rightsquigarrow u'$, one can safely replace these two moves $u \rightsquigarrow x$ and $x \rightsquigarrow u'$ by $d$ moves where $d$ is the distance between $u$ and $u'$ in $G$. Therefore, one can obtain from $S'$ a TS-sequence that reconfigures $C_s$ into $C_t$ and thus $(G, C_s, C_t)$ is a yes-instance for $\mathrm{VCR}_{\mathsf{TS}}$, as desired. This concludes the proof of Theorem 3.20. □

Finally, we prove that $\mathrm{DSR}_{\mathsf{TS}}$ is PSPACE-complete on planar graphs of maximum degree six and bounded bandwidth graphs. Recall that a graph has *bandwidth* at most $k$ if there exists a numbering $\ell$ of the vertices with distinct integers between 1 and $n$ (where $n$ is the number of vertices of the graph) such that adjacent vertices must have labels at distance at most $k$ (i.e., for every edge $uv \in E$, $|\ell(u) - \ell(v)| \leq k$).

**Theorem 3.21.** *$\mathrm{DSR}_{\mathsf{TS}}$ is PSPACE-complete on planar graphs of maximum degree six and bounded bandwidth graphs.*

*Proof.* First, recall that $\mathrm{VCR}_{\mathsf{TS}}$ is PSPACE-complete on planar graphs of maximum degree three (see, e.g., [HD05, BC09]) and on bounded bandwidth graphs [Wro18]. The proof for dominating sets reconfiguration under TAR on planar graphs from [HIM+16] works also here since $\mathrm{VCR}_{\mathsf{TS}}$ is PSPACE-complete on planar graphs. We use the well-known reduction mentioned in Theorem 3.20, which is the following: start with a copy of the original graph $G$ and for each edge $uv$, add a vertex $z_{uv}$ of degree two adjacent to $u$ and $v$. Let $G'$ be the resulting graph, and note that the planarity property of $G'$ is preserved.

Let $G$ be a graph whose bandwidth is bounded by some constant $k$. Since a vertex can have at most $k$ neighbors of lower label and $k$ neighbors of higher label, this implies that the

maximum degree of $G$ is bounded by $2k$. We use this observation to prove that the graph $G'$ obtained from the reduction has its bandwidth bounded by $k \cdot (k+1)$. We explain how to obtain a labeling $\ell'$ of $G'$ from $\ell$ that satisfies the bandwidth property. The underlying idea is to leave $k$ free values between any two vertices labeled consecutively in the original labeling (i.e., vertices $u$ and $v$ such that $\ell(v) = \ell(u) + 1$) in order to label the vertices in $V(G') \setminus V(G)$.

More precisely, for all $i > 1$, we relabel the vertex labeled $i$ in $\ell$ with the label $1 + (i-1) \cdot (k+1)$ in $\ell'$. Let $u$ and $v$ be two adjacent vertices of $G$ with $\ell(u) < \ell(v)$, then $\ell(v) - \ell(u) \leq k$ and thus $\ell'(v) - \ell'(u) \leq k \cdot (k+1)$. Moreover, we label the new vertex $z_{uv}$ with label $\ell'(u) + (\ell(v) - \ell(u))$, which lies between $\ell'(u) + 1$ and $\ell'(u) + k$ by the bandwidth hypothesis. We also have $\ell'(v) - \ell'(z_{uv}) < k \cdot (k+1)$, and the bandwidth condition is satisfied. So the difference between the labels of any two adjacent vertices in $G'$ is at most $k \cdot (k+1)$.

Observe however that not all vertices have $k$ neighbors of higher label in $G$, and thus the labeling $\ell'$ does not use consecutive values. To fix this, we just relabel the graph with values between 1 and $|V(G')|$, maintaining the ordering of $\ell'$. The new labeling $\ell''$ obtained does not increase the distance from $\ell'$, and thus satisfies the bandwidth condition, as required. □

Recall that the pathwidth and thus the treewidth of a graph are bounded by its bandwidth (see Section 1.4.2). Therefore, we immediately get from Theorem 3.21 that $\mathrm{DSR}_{\mathsf{TS}}$ is PSPACE-complete on bounded pathwidth and bounded treewidth graphs.

### 3.2.3 Polynomial-time algorithms

In this section, we focus on graph classes for which $\mathrm{DSR}_{\mathsf{TS}}$ can be solved in polynomial time. As discussed before, a natural way to solve this problem is to distinguish a special dominating set, called *canonical* [HIM+16]. However, note that our definition of *canonical dominating set* is different from the one in [HIM+16]. Indeed, we only define a canonical as a minimum dominating set. We will then show that this special dominating is reachable from any dominating set under token sliding.

The canonical dominating set is not part of the original instance, so it is crucial to be able to compute it in polynomial time if we aim to compute the reconfiguration sequence in polynomial time as well. However, this is not an issue if we are only interested in the decision problem. We moreover emphasize the fact that this canonical dominating set must be uniquely defined, i.e., the set of vertices that hold a token as well as the number of tokens on each of these vertices must be fixed.

#### Join and cographs

Recall that the domination number of a join $G_1 + G_2$ is always at most two, since taking a vertex from each operand of the join dominates the whole graph.

**Theorem 3.22.** *Let $G_1$ and $G_2$ be two graphs, and $D_s$ and $D_t$ be two dominating sets of $G_1 + G_2$ of the same size. The dominating set $D_s$ can be reconfigured into $D_t$ by token sliding if and only if one of the three following conditions holds:*

*(i)* $|D_s| = |D_t| \geq 3$;

*(ii) the domination number of $G_1$ or of $G_2$ is at most two;*

*(iii) both $G_1$ and $G_2$ are connected.*

*Proof.* We first show that if none of these conditions hold, then $(G_1 + G_2, D_s, D_t)$ is a no-instance. Let $G_1$ and $G_2$ be two graphs with $\gamma(G_1) > 2$ and $\gamma(G_2) > 2$, and assume without loss of generality that $G_1$ is not connected, say with two components $C_1$ and $C_2$. Note that $\gamma(G_1 + G_2) = 2$ since neither $G_1$ nor $G_2$ has a universal vertex.

Let $D_s = \{u, v\}$ and $D_t = \{w, v\}$ be two minimum dominating sets of $G$ with $u \in C_1$, $w \in C_2$ and $v \in V(G_2)$. We prove that $D_s$ can not be reconfigured into $D_t$. Since $G_1$ is not connected, there is no path between $u$ and $w$ in $G[V_1]$. Therefore, the only way to reach $w$ from $u$ is to go through $V(G_2)$. But since $\gamma(G_2) > 2$ no pair of vertices in $G_2$ can dominate $G_2$, and thus no move from $V(G_1)$ to $V(G_2)$ is possible.

We now prove the sufficiency, i.e., we prove that each of the above conditions is sufficient for the dominating sets to be reconfigured.

*Condition (i).* Suppose $|D_s| = |D_t| \geq 3$. Recall that picking a vertex of $G_1$ and one of $G_2$ always forms a dominating set of $G_1 + G_2$. We infer that it is always possible to make one move from $D_s$ to reach a configuration with tokens in both $G_1$ and $G_2$, then from such position tokens can be slid freely in their part, until reaching $D_t$ with a last move.

We assume now that $|D_s| = |D_t| \leq 2$.

*Condition (ii).* For the case where $G_1$ or $G_2$ has domination number at most two, we consider different cases depending on whether a graph has domination number one or not.

**Case 1.** If $\gamma(G_1) = 1$ or $\gamma(G_2) = 1$: then $G_1 + G_2$ contains a universal vertex. Then, from $D_s$, one can place a token on this vertex, reconfigure other possible tokens freely, then move that token to reach $D_t$.

**Case 2.** If $\gamma(G_1) = 2$ and $\gamma(G_2) = 2$. Assume without loss of generality that $\gamma(G_1) = 2$. Note that in this case, $\gamma(G_1 + G_2) = 2$, let $D_s = \{v_1, v_2\}$. We define an arbitrary canonical dominating set $C$ by taking a vertex (e.g., of smallest index) in each of $G_1$ and $G_2$; we denote these vertices $u_1 \in V(G_1)$ and $u_2 \in V(G_2)$. Recall that each reconfiguration sequence is reversible. Hence, it is sufficient to prove that both $D_s$ and $D_t$ can been transformed into $C$. We only show this statement for $D_s$; the proof for $D_t$ follows by symmetry.

Suppose first that $v_1$ and $v_2$ belong to the same original graph, say $v_1, v_2 \in V(G_1)$. We show how to reconfigure $D_s$ into $C$ in at most two steps. First, observe that since $C$ is a dominating set of $G$, $u_1 \in N[\{v_1, v_2\}]$, say $u_1$ belongs to $N[v_1]$. Our first step is to slide the token from $v_2$ to $u_2$, along the corresponding edge of the join. Then, by our observation that $u_1 \in N[v_1]$, we can slide if necessary the token from $v_1$ to $u_1$.

Suppose now that $v_1$ and $v_2$ belong respectively to $V(G_1)$ and $V(G_2)$. Since $\gamma(G_1) = 2$, let $\{w_1, w_2\}$ be a dominating set of $G_1$ and thus of $G_1 + G_2$ (it can be computed naively in cubic time). It dominates $v_1$ so assume without loss of generality that $v_1w_1$ is an edge. First moving the token from $v_1$ to $w_1$ (if $v_1 \neq w_1$) and then from $v_2$ to $w_2$, at most two steps permit us to reconfigure $D_s$ into $\{w_1, w_2\}$, which we can then reconfigure into $C$ by the above argument.

*Condition (iii).* Suppose finally that $\gamma(G_1) \geq 3$ and $\gamma(G_2) \geq 3$ but both $G_1$ and $G_2$ are connected. Then $\gamma(G_1 + G_2) = 2$ and the minimum dominating sets of $G_1 + G_2$ are exactly the sets containing a vertex in $G_1$ and a vertex in $G_2$. Let $D_s = \{v_1, v_2\}$ and $D_t = \{w_1, w_2\}$ with $v_1, w_1 \in V(G_1)$ and $v_2, w_2 \in V(G_2)$. Since $G_1$ is connected, there exists a path from $v_1$ to $w_1$ in $G[V(G_1)]$. Moving the token along this path, we always keep a dominating set by the above observation. Doing similarly along a path from $v_2$ to $w_2$, we have a reconfiguration sequence from $D_s$ to $D_t$. This concludes the proof of Theorem 3.22. □

We now consider the special case of cographs. Recall that the family of cographs can be defined as the family of $P_4$-free graphs, or equivalently by the following recursive definition:

- $K_1$ is a cograph;
- for $G_1$ and $G_2$ any two cographs, the disjoint union $G_1 \cup G_2$ is a cograph;
- for $G_1$ and $G_2$ any two cographs, the join $G_1 + G_2$ is a cograph.

Brandelt and Mulder gave in [BM86] an alternative characterization of cographs: $G$ is a cograph if and only if $G$ is the disjoint union of distance-hereditary graphs with diameter at most two. Note that computing a minimum dominating set in distance-hereditary graphs is linear-time solvable [NS01]. Hence, we can compute the domination number of a cograph in linear time as well.

By the previous theorem, we infer that if a cograph is constructed as a join of two cographs, the case is polynomial-time decidable. The case when $G = K_1$, is straightforward. If $G = G_1 \cup G_2$ is the disjoint union of two cographs, then for two dominating sets $D_s$ and $D_t$, deciding whether $D_s \overset{\mathsf{TS}}{\leftrightsquigarrow} D_t$ is equivalent to deciding whether $D_s \cap V(G_1) \overset{\mathsf{TS}}{\leftrightsquigarrow} D_t \cap V(G_1)$ in $G_1$, and $D_s \cap V(G_2) \overset{\mathsf{TS}}{\leftrightsquigarrow} D_t \cap V(G_2)$ in $G_2$, which can be done inductively by induction. As a consequence, we obtain the following:

**Theorem 3.23.** *There is a polynomial-time algorithm deciding $DSR_{\mathsf{TS}}$ in cographs.*

### Dually chordal graphs

Let $G = (V, E)$ be a graph with $V = \{v_1, v_2, \ldots, v_n\}$. We denote by $G_i$ the graph induced by $\{v_i, v_{i+1}, \ldots, v_n\}$. A *maximum neighbor* of a vertex $u$ is a vertex $v \in N[u]$ such that we have $N[w] \subseteq N[v]$ for every vertex $w \in N[u]$. In other words, $v$ contains in its closed neighborhood every vertex at distance at most two from $u$. A *maximum neighborhood ordering* (or *mno* for short) is an ordering of the vertices in such a way that $v_i$ has a maximum neighbor in the graph $G_i$. A graph is *dually chordal* if it has a maximum neighborhood ordering. This ordering can be computed in linear time [BCD98]. Moreover, the *mno* computed by this algorithm is such that for every vertex $v_i$ (with $i < n$), $v_i's$ maximum neighbor is different from $v_i$ (for connected graphs). An alternative proof of a similar statement for not necessarily connected graphs can be found in [DKR15]. In the following, we always assume that an *mno* is associated with a function $mn : V \longrightarrow V$ that associates with each vertex a maximum neighbor.

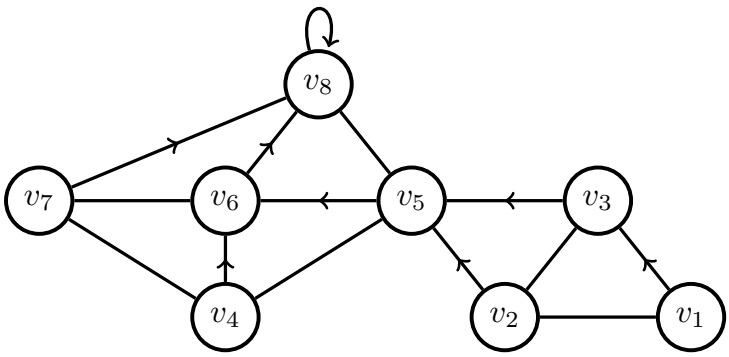



Figure 3.12 – A dually chordal graph.

Note that a dually chordal graph is not necessarily chordal. Figure 3.12 gives an example of a graph which is dually chordal but not chordal, since it contains an induced cycle on four vertices. The label inside each vertex corresponds to its rank in the ordering, and its maximum neighbor is the endpoint of its single outgoing edge (note that $v_8$'s maximum neighbor is itself).

Moreover, observe that any tree $T$ is a dually chordal graph: root the tree in some vertex and orient all edges toward the root; any numbering keeping all $G_i$ connected is an *mno* where arcs point towards the vertex maximum neighbor.

**Link with interval graphs.** Recall that an interval graph is the intersection graph of a family of intervals on the real line. In other words, let $\{I_1, I_2, \ldots, I_n\}$ be a set of intervals. Each interval $I$ can be represented by its extremities $\ell(I), r(I)$ with $\ell(I) \leq r(I) \in \mathbb{R}$. We call these values respectively the $\ell$-value and $r$-value (for left and right). The corresponding interval graph $G = (V, E)$ is the following:

- $V = \{I_1, I_2, \ldots, I_n\}$;
- $I_i I_j \in E \Leftrightarrow I_i \cap I_j \neq \emptyset$, i.e., $\ell(I_j) \leq r(I_i)$ and $\ell(I_i) \leq r(I_j)$.

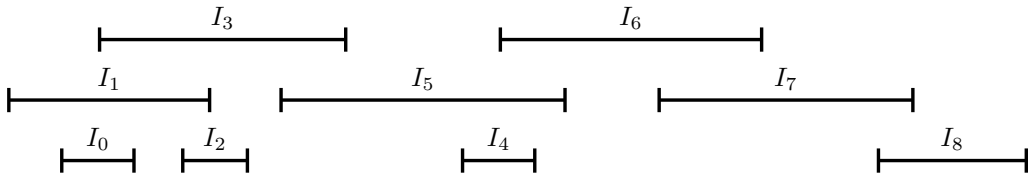



Figure 1

(a) Set of intervals

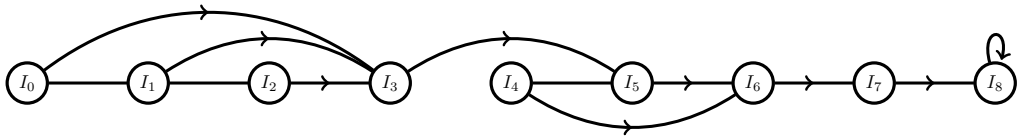



Figure 1

(b) Corresponding interval graph

Figure 3.13 – Interval graph and its maximum neighborhood ordering.

Let $G = (V, E)$ be an interval graph. For convenience, we denote by $v_i$ the vertex related to the interval $I_i$. We now order the vertices of $G$ with respect to their $r$-values, i.e., $v_i < v_j$ if and only if $r(I_i) < r(I_j)$ (or $r(I_i) = r(I_j)$ and $\ell(I_i) < \ell(I_j)$). Then, recall that we have the following property (see Section 1.2.2 for a proof):

**Observation 3.24.** *Let $v_i$ and $v_j$ be two vertices of $G$ such that $v_i < v_j$. If $v_i v_j \in E$, then for any $v_k$ such that $v_i < v_k < v_j$, we have $v_k v_j \in E$.*

We can now prove the following:

**Observation 3.25.** *Interval graphs are dually chordal graphs.*

*Proof.* To see this observation, we prove that the ordering described above is an *mno*. For every vertex $v_i$, we show that its neighbor of maximum index $v_j$ in the ordering is a maximum neighbor. Indeed, consider any neighbor $v_k$ of $v_i$ in $G_i$. By definition of $v_j$, we have $v_i < v_k \leq v_j$, and $v_k$ is adjacent to $v_j$. Moreover, any other neighbor $v_\ell > v_i$ of $v_k$ either satisfies $v_i < v_\ell < v_j$ or $v_k < v_j < v_\ell$. In both cases, Observation 1.4 concludes the proof. □

An example of the construction used above on an interval graph is given in Figure 3.13 where the maximum neighbor of $I_i$ is its only out-neighbor, i.e., the endpoint of the only directed edge incident to $I_i$.

**Computing the canonical dominating set.** Let $G$ be a dually chordal graph, whose vertices are ordered by an *mno*. Let $C = \{c_1, c_2, \ldots, c_k\}$ be a dominating set of $G$ and $T = \{t_1, t_2, \ldots, t_k\}$ a set of vertices, both sets in increasing order according to the *mno*. We say that $C$ is a *triggered dominating set* with triggering vertices $T$ if and only if the two following properties are satisfied:

(i) $c_i = mn(t_i)$ for all $1 \leq i \leq k$,

(ii) following the *mno*, $t_i$ is the least vertex not in $N[c_1, \ldots, c_{i-1}]$, for all $1 \leq i \leq k$.

It is known that the MINIMUM DOMINATING SET problem is linear-time solvable on dually chordal graphs [BCD98]. In our case, we give another algorithm to compute a triggered dominating set, that will serve as a canonical dominating set.

Observe that an *mno* is associated with exactly one triggered dominating set. The following algorithm, called MDS, is strongly inspired by the classical algorithm for computing minimum dominating sets in trees [MCH79]. It takes as input a dually chordal graph $G = (V, E)$ with an *mno* and computes a triggered dominating set $C$ of size $\gamma(G)$ and its corresponding set of triggering vertices $T$ in running time $O(|V| + |E|)$.

**Algorithm 3** MDS

**Require:** A dually chordal graph $G$ with an *mno*.
**Ensure:** A minimum triggered dominating set $C$ and its set of triggering vertices $T$.

```
1:  Mark all vertices BOUNDED
2:  C ← ∅
3:  for all i from 1 to n do
4:      if v_i is labeled BOUNDED then
5:          Label mn(v_i) with REQUIRED
6:          Add v_i to the set of triggering vertices T
7:          for all u ∈ N[mn(v_i)] do
8:              if u is not labeled REQUIRED then
9:                  Label u with FREE
10:     if v_i is labeled REQUIRED then
11:         C ← C ∪ {v_i}
12: return C and T
```

**Lemma 3.26.** *Given a dually chordal graph $G = (V, E)$, the algorithm* MDS *computes a triggered dominating set of $G$ of order $\gamma(G)$ in time $O(|V| + |E|)$.*

*Proof.* The fact that $C$ is a triggered dominating set with triggering vertices $T$ is a direct consequence of the construction of the algorithm. Statement (i) comes from line 5 of the algorithm, while statement (ii) simply comes from the fact that we deal with the vertices in increasing order in the loop of line 3. Still we need to prove that this dominating set is of size $\gamma(G)$.

A *labeled graph* is a graph whose vertices are labeled FREE, REQUIRED or BOUNDED, such that a vertex is labeled FREE if and only if it is adjacent to a vertex labeled REQUIRED and it is not labeled REQUIRED. Observe that the algorithm MDS maintains all along a labeled graph. In a labeled graph, we define a *labeled dominating set* a set of vertices containing all the vertices labeled REQUIRED and dominating all vertices labeled BOUNDED. The *labeled domination number* is the minimum size of a labeled dominating set. We show that the algorithm MDS keeps the labeled domination number of the graph invariant. At the beginning, when all the vertices are labeled BOUNDED, the labeled domination number of the graph is exactly its domination number. This will allow us to conclude that the set of vertices marked REQUIRED at the end forms a minimum dominating set of $G$, of order $\gamma(G)$.

Let $S$ be a minimum labeled dominating set of a labeled graph $G$, and let $v_i$ be the minimum vertex in the *mno* that is labeled BOUNDED. Let $w$ be the maximum neighbor of $v_i$ in $G_i$. If $w \in S$, then $S$ is also a minimum labeled dominating set of the graph $G$ where $w$ is labeled REQUIRED and all its neighbors previously labeled BOUNDED are labeled FREE, so the algorithm does not change the labeled domination number of $G$. Otherwise, say $v_j$ is the vertex that dominates $v_i$ in $S$. Since $v_i$ is marked BOUNDED, $v_j$ is not marked REQUIRED. If $j \geq i$, then by the maximum neighbor property, $w$ is adjacent to all the neighbors of $v_j$ that are in $G_i$, so $w$ dominates all neighbors of $v_j$ that are still marked BOUNDED. Thus we can replace $v_j$ by $w$ in $S$ and keep a minimum labeled dominating set of $G$. This concludes the case $j \geq i$. Suppose now $j < i$. Consider $v_k$ the maximum neighbor of $v_j$ in $G_j$. Observe that since no vertex less than $v_i$ is BOUNDED, by the maximum neighbor definition, $v_k$ dominates any BOUNDED vertex adjacent to $v_j$. Thus, we can again replace $v_j$ by $v_k$ in $S$ and keep a minimum dominating set. We can iterate until the vertex dominating $v_i$ in $S$ is no less than $v_i$, and then refer to the above argument, used when $j \geq i$. This concludes the proof that the algorithm MDS keeps the labeled domination number of the graph invariant, and thus that it produces a minimum dominating set of $G$.

For the time complexity of the algorithm, observe that the algorithm visits every vertex at most once in the main loop, and it visits the neighborhoods of each vertex at most once when it possibly labels it REQUIRED. So the complexity is upper bounded by $\sum_{v \in V} O(1 + |N(v)|) = O(|V| + |E|)$. □

**The reconfiguration algorithm.** We now show how to make use the canonical triggered dominating set $C$ computed by the MDS algorithm in order to reconfigure two dominating sets of a dually chordal graph. To do that, we provide an algorithm called DUALLY-CHORDAL-RECONF that modifies any dominating set $D$ of a dually chordal graph in such a way that $C \subseteq D$. The idea of this algorithm is to pick one vertex in $D$ that dominates the triggering vertex $t_i$ (from the output of algorithm MDS) and to replace it by the corresponding vertex $c_i$ of $C$. Recall that the notation $u \overset{\mathsf{TS}}{\rightsquigarrow} v$ is used for sliding the token along the edge $uv$.

**Algorithm 4** DUALLY-CHORDAL-RECONF

**Require:** A dually chordal graph $G = (V, E)$, a minimum dominating set $D$ of $G$
1: Compute an *mno* for $G$
2: $(C, T) \leftarrow$ MDS(G).
3: **for** $i$ from 1 to $\gamma(G)$ **do**
4: Let $x_i$ be the least vertex of $D \cap N[t_i]$
5: **if** $x_i c_i \in E$ **then**
6: $x_i \overset{\mathsf{TS}}{\rightsquigarrow} c_i$
7: **else**
8: $y_i \leftarrow mn(x_i)$
9: $x_i \overset{\mathsf{TS}}{\rightsquigarrow} y_i$
10: $y_i \overset{\mathsf{TS}}{\rightsquigarrow} c_i$

**Lemma 3.27.** *Given a dually chordal graph $G = (V, E)$ and a dominating set $D$,* DUALLY-CHORDAL-RECONF *modifies $D$ with respect to the token sliding model in such a way that $C \subseteq D$ in $O(|V|)$ time, where $C$ is the* canonical triggered dominating set *computed by the MDS algorithm.*

*Proof.* Let $T = (t_1, t_2, \ldots, t_\gamma)$ and $C = \{c_1, c_2, \ldots, c_\gamma\}$ be the output of algorithm MDS, with $c_i = mn(t_i)$. We denote by $C_i = \{c_1, \ldots, c_i\}$ the set of $i$ first vertices of $C$ according to the *mno*.

In order to prove the correctness of the algorithm, we need to prove that the two following constraints are satisfied:

(i) each move is valid with respect to the token sliding model;

(ii) every intermediate set is a dominating set of $G$. (Note that this ensures the existence of the $x_i$ of line 4.)

We prove these two properties by induction on the index $i$, with $0 < i \leq \gamma$. For some $i > 0$, assume that the algorithm reconfigured properly $D$ into $D_{i-1} = (D \setminus \{x_1, \ldots, x_{i-1}\}) \cup \{c_1, \ldots, c_{i-1}\}$ . We explain how to extend this up to rank $i$. By definition, $t_i$ is the least vertex which is not dominated by $N[C_{i-1}] = N[\{c_1, \ldots, c_{i-1}\}]$. Let $x_i$ be the least vertex dominating $t_i$ in $D$. Observe that $x_i \notin \{c_1, \ldots, c_{i-1}\}$ since $t_i$ is the triggering vertex of $c_i$. To simplify notation, we denote by $G'$ the subgraph of $G$ induced by vertices larger than $t_i$ in the *mno* (i.e., the subgraph $G_j$ where $j$ is the index of $t_i$ in the *mno*). Note that since $C_{i-1} \subset D_{i-1}$, all vertices in $G \setminus G'$ are dominated. We consider two cases:

**Case 1.** *$x_i$ is adjacent to $c_i$.* Observe first that this case occurs whenever $x_i \geq t_i$ in the *mno* (where $x_i \in N_{G'}[t_i] \subseteq N_{G'}[c_i]$). In that case, the algorithm executes line 6 and the token sliding constraint (i) is satisfied. Now, since $c_i = mn(t_i)$ and $x_i$ is adjacent to $t_i$, $N_{G'}[x_i] \subseteq N_{G'}[c_i]$, and constraint (ii) is also satisfied. The conclusion follows from the fact that all vertices in $G \setminus G'$ are dominated.

**Case 2.** *$x_i$ is not adjacent to $c_i$.* This is possibly the case when $x_i < t_i$ in the *mno*. The algorithm then first reconfigures $x_i$ into its maximum neighbor $y_i$, which is adjacent to $x_i$ and dominates all neighbors of $x_i$ that might not be dominated yet by $\{c_1, \ldots, c_{i-1}\}$, satisfying constraint (ii). Moreover, $x_i$ is adjacent to $t_i$ and $x_i < t_i$ in the *mno*, so $y_i$, as the maximum neighbor of $x_i$, must be adjacent to $t_i$ and all its neighbors in $G'$, among which there is $c_i$. So the next move to $c_i$ satisfies the token sliding constraint (i). Also, since $y_i$ is adjacent to $t_i$ and $c_i$ is the maximum neighbor of $t_i$, $N_{G'}(y_i) \subseteq N_{G'}[c_i]$, and constraint (ii) is also satisfied. □

**Theorem 3.28.** *$DSR_{\mathsf{TS}}$ can be solved in linear time on dually chordal graphs, and we can obtain a reconfiguration sequence between two dominating sets in quadratic time.*

*Proof.* Let $G = (V, E)$ be a dually chordal graph and $D_s, D_t$ be two dominating sets of $G$ of size $k$. We may assume that $G$ is connected, otherwise we proceed independently for each connected component by checking first that the number of tokens on each component fit. More precisely, $(G, D_s, D_t)$ is a yes-instance if and only if for each connected component $G_i$ of $G$, we have $|D_s \cap G_i| = |D_t \cap G_i|$. This can be done in linear time. We explain how to reconfigure $D_s$ into $D_t$ in at most quadratic time.

First, we compute in linear time the canonical dominating set $C$ of $G$ with the algorithm MDS. By Lemma 3.27, one can transform $D_s$ and $D_t$ in such a way that both contain $C$. This can be done in linear time (with respect to the order of $G$) since we move at most $\gamma(G)$ tokens and each move requires at most two steps. If $k = \gamma(G)$, we are done. Otherwise, choose a vertex $v$ of minimum eccentricity and move all the remaining tokens by a shortest path to $v$. Therefore, the total time complexity is $O(|V| + |E|) + O(|V|) + O((k - \gamma(G)) \cdot \epsilon(v))$, which is at most quadratic (when $k = \Omega(n)$). □

Observe that when $k$ is close to $\gamma$, the algorithm is linear. However, when the number of extra tokens is large (i.e., is linear in $n$), the quadratic overhead may be necessary. Indeed, consider a path on $n$ vertices $P_n$. The minimum eccentricity of $P_n$ is that of the middle vertex $v$ which is $\lfloor n/2 \rfloor$. Therefore, if all the extra tokens are on an extremity of the path, the time needed

to move all of them to $v$ is quadratic. Since a path is a dually chordal graph, the conclusion follows.

### 3.2.4 Concluding remarks

In this section, we initiated the study of the computational complexity of $\mathrm{DSR}_{\mathsf{TS}}$, the reachability question of DOMINATING SET RECONFIGURATION under token sliding. We proved that the problem is PSPACE-complete in several graph classes including split or bipartite graphs while it is polynomial-time solvable on cographs and dually chordal graphs (see Figure 3.14).

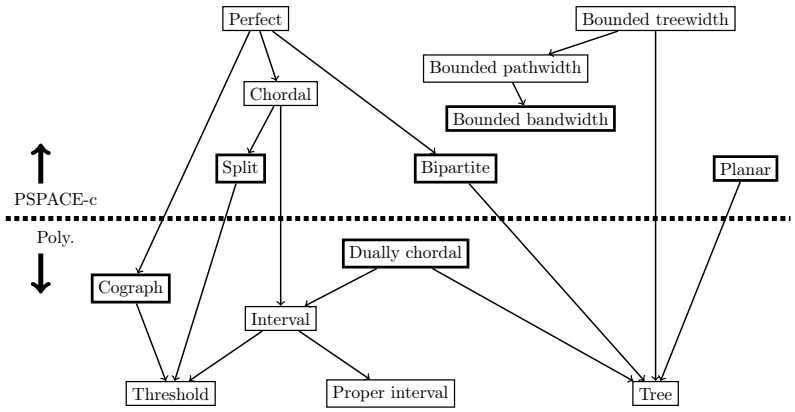



Figure 3.14 – Our results: the frontier between PSPACE-completeness and tractability.

However, in all of our polynomial results presented in Section 3.2.3, computing a minimum dominating set can be done in polynomial or even linear time on the graph classes considered. Therefore, a challenging question is the following: does there exist a graph class for which computing a minimum dominating set is NP-complete but $\mathrm{DSR}_{\mathsf{TS}}$ can be solved in polynomial time? As a result of Theorem 3.28, we get that $\mathrm{DSR}_{\mathsf{TS}}$ is polynomial-time solvable on interval graphs. Recall that a circle graph is the intersection graph of a set of chords of a circle. The DOMINATING SET PROBLEM has been shown to be NP-complete on this graph class [Kei93]. Hence, we ask the following question; note that if the problem is tractable, it would generalize our result on cographs:

**Question 3.29.** *What is the complexity of $\mathrm{DSR}_{\mathsf{TS}}$ on circle graphs?*

A circular-arc graph is the intersection graph of a set of arcs on the circle. Even if computing a minimum dominating set can be computed in linear time on circular-arc graphs [Cha98], we are interested in the complexity of the reconfiguration version under token sliding. More precisely, we ask for the following:

**Question 3.30.** *Is $\mathrm{DSR}_{\mathsf{TS}}$ polynomial-time solvable on circular-arc graphs?*

Recently, Nicolas Bousquet and Alice Joffard studied the complexity of $\mathrm{DSR}_{\mathsf{TS}}$ [Jof20]. They proved that the problem is PSPACE-complete on planar bipartite graph, unit disk graphs, line graphs and circle graphs (answering Question 3.29). Hence, it remains open to determine if there exists a graph class for which DOMINATING SET is NP-complete but $\mathrm{DSR}_{\mathsf{TS}}$ is in P. On the positive side, they answered positively to Question 3.30 (see Figure 3.15).

Besides, we found polynomial-time algorithms for cographs and dually chordal graphs but the underlying reconfiguration sequence is most likely not optimal. Indeed, it may be possible that the shortest path in $\mathcal{R}_k(G)$ between the two given solutions does not go through the canonical dominating set. Therefore, can we bound the diameter of the reconfiguration graph?

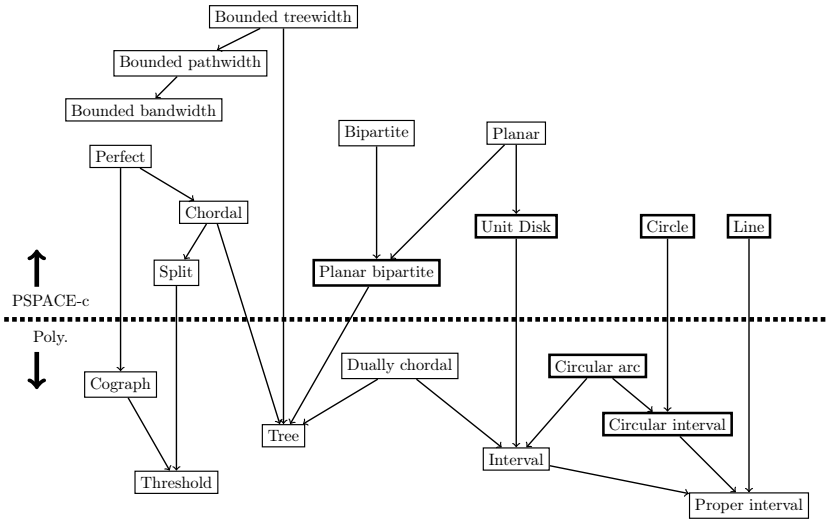



Figure 3.15 – Complexity of $DSR_{TS}$ with the results by Bousquet and Joffard [Jof20].

In other words, what is the maximum length of a shortest reconfiguration sequence between any pair of dominating sets? Moreover, what is the complexity of finding the optimal solution, i.e., the shortest reconfiguration sequence between two dominating sets on cographs or dually chordal graphs? Is it polynomial-time solvable, as the reachability variant studied here? Or does it become harder?

## 3.3 Optimization variants

In this section, we focus on an optimization variant of reconfiguration recently introduced by Ito et al. [IMNS19] for INDEPENDENT SET RECONFIGURATION. We apply it to dominating sets and we study the tractability and fixed-parameter tractability according to graph classes. The results presented in Sections 3.3.2 and 3.3.3 are from a joint work with Alexandre Blanché, Haruka Mizuta and Akira Suzuki [BMOS20], while the result of Section 3.3.4 is an ongoing work with Alexandre Blanché and Haruka Mizuta.

### 3.3.1 Introduction

Combinatorial reconfiguration models dynamic transformations of systems, where we wish to transform the current configuration of a system into a more desirable one by a step-by-step transformation. In the current framework of combinatorial reconfiguration, we need to have in advance a target, i.e., a more desirable configuration. However, it is sometimes hard to decide a target configuration, because there may exist exponentially many of them. Based on this situation, Ito et al. [IMNS19] introduced the new framework of reconfiguration problems, called *optimization variant*. In this variant, we are given a single solution as a current configuration, and asked for a more "desirable" solution reachable from the given one. They might be several definitions of what is a "more desirable" solution. In [IMNS19], it corresponds to a solution that maximizes its size, since INDEPENDENT SET is a maximization problem. More precisely, then studied the following problem called OPT-ISR:

OPT-ISR

**Instance:** A graph $G$, two integers $k, s \geq 0$, an independent set $I$ of $G$ such that $|I| \geq k$.
**Output:** An independent set $I_t$ of $G$ such that $|I_t| \geq s$ and $I_t$ is reachable from $I$ under the TAR($k$) rule if it exists, no-instance otherwise.

They proved that this problem is PSPACE-hard on bounded pathwidth while it is linear-time solvable on chordal graphs. They also studied the parameterized complexity of OPT-ISR according to three different parameters: the degeneracy $d$ of the input graph, the threshold $k$ and the solution size $s$. They showed that the problem is W[1]-hard when only one of $d$, $k$, and $s$ is taken as a parameter. However, OPT-ISR admits an XP algorithm with respect to the latter since they provided an $O(s^3 n^{2s})$ time algorithm. Moreover, they also gave a fixed-parameter algorithm with respect to $s + d$. This result implies that OPT-ISR is fixed-parameter tractable when parameterized by $s$ on planar graphs, and on bounded treewidth graphs.

As we said before, we apply this optimization framework to the reconfiguration of dominating sets. We consider the TAR rule as well and our goal is to minimize the size of the target dominating set. More precisely, the optimization variant of DOMINATING SET RECONFIGURATION (or OPT-DSR for short) is defined as follows:

OPT-DSR

**Instance:** A graph $G$, two integers $k, s \geq 0$, a dominating set $D$ of $G$ such that $|D| \leq k$.
**Output:** A dominating set $D_t$ of $G$ such that $|D_t| \leq s$ and $D_t$ is reachable from $D$ under the TAR($k$) rule if it exists, no-instance otherwise.

We denote by a 4-tuple $(G, k, s, D)$ an instance of OPT-DSR; the Figure 3.16 illustrates a yes-instance with threshold $k = 4$ and solution size $s = 2$.

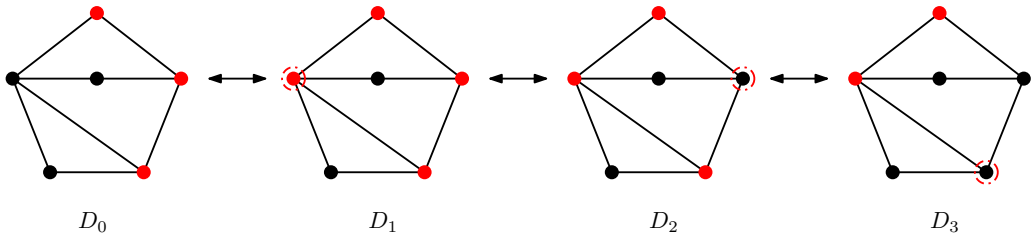



Figure 3.16 – Reconfiguration sequence between $D_0$ and $D_3$ via dominating sets $D_1, D_2$ with upper bound $k = 4$, where added or removed vertices are surrounded by dotted circles.

Let us now give some useful observations regarding OPT-DSR.

**Observation 3.31.** *Let $(G, k, s, D)$ be an instance of OPT-DSR. If $k, s$ and $|D|$ violate the inequality $s < |D| \leq k$, then $D$ is a solution of the instance.*

*Proof.* By the definition of $D$, we know $|D| \leq k$. Therefore if the inequality is violated, we have $|D| \leq s \leq k$ or $|D| \leq k \leq s$. In both cases, $|D| \leq s$ holds, and hence $D$ is a solution. □

It is observed that the condition in Observation 3.31 can be checked in linear time. Therefore, we sometimes assume without loss of generality that $s < |D| \leq k$ holds. Then, another observation follows.

**Observation 3.32.** *Let $(G, k, s, D)$ be an instance of OPT-DSR such that $s < |D|$ holds. If $D$ is minimal and $|D| = k$ holds, then the instance has no solution.*

*Proof.* Since $|D| = k$, we cannot add any vertex to $D$ without exceeding the threshold $k$. Besides, since $D$ is minimal, we cannot remove any vertex while maintaining the domination property. As a result, there is no dominating set $D_t$ of size at most $s$ which is reachable from $D$ under the TAR($k$) rule. □

Again, the conditions in Observation 3.32 can be checked in linear time, and hence we can assume without loss of generality that $D$ is not minimal or $|D| < k$ holds. Suppose that $D$ is not minimal. Then, we can always obtain a dominating set of size less than $k$ by removing some vertex without private neighbor from $D$, that is, we have a dominating set $D'$ reachable from $D$ under the TAR($k$) rule such that $|D'| < k$. Note that $(G, k, s, D)$ has a solution if and only if $(G, k, s, D')$ does. Therefore, it suffices to consider the case where $|D| < k$ holds. Combining it with Observation 3.31, we sometimes assume without loss of generality that $s < |D| < k$ holds.

Finally, we have the following observation which states that OPT-DSR is a generalization of the DOMINATING SET Problem; a similar result holds for OPT-ISR:

**Observation 3.33.** *Let $G = (V, E)$ be a graph and $s$ be an integer. The instance $(G, |V|, s, V)$ of OPT-DSR is equivalent to finding a dominating set of $G$ of size at most $s$.*

*Proof.* Suppose that $G$ admits a dominating set $D_t$ of size at most $s$. Since we started from a dominating set containing all the vertices of $G$, it is sufficient to remove one by one each vertex in $V \setminus D_t$ to reach $D_t$ and thus $(G, |V|, s, V)$ is a yes-instance. The converse direction is trivial: if $(G, |V|, s, V)$ is a yes-instance, then $G$ has a dominating set of size at most $s$. □

Observation 3.33 implies that the hardness results for the original DOMINATING SET problem extend to OPT-DSR. In particular, we obtain as a corollary that OPT-DSR is NP-hard even for the case where the input graph has maximum degree three, or is planar with maximum degree four [GJ79]. However, we will show in Section 3.3.2 that this problem is actually PSPACE-complete.

### 3.3.2 Polynomial-time (in)tractability

In this section, we give some results regarding the tractability of OPT-DSR with respect to graph classes. We first show that the problem is PSPACE-complete for bounded pathwidth graphs, for split graphs, and for bipartite graphs. Additionally, we explain how to extend some known results on $\mathrm{DSR}_{\mathsf{TAR}}$ to get polynomial-time algorithms for cographs, trees or interval graphs for instance.

**PSPACE-completeness for several graph classes** We first show that OPT-DSR is PSPACE-complete. More precisely, we prove the following:

**Theorem 3.34.** OPT-DSR *is PSPACE-complete even when restricted to bounded pathwidth graphs, for split graphs, and for bipartite graphs.*

First, observe that OPT-DSR is in PSPACE. Indeed, when we are given a dominating set $D_t$ as a solution for some instance of OPT-DSR, we can check in polynomial time whether it has size at most $s$ or not. Furthermore, since $\mathrm{DSR}_{\mathsf{TAR}}$ is in PSPACE, we can check in polynomial space whether it is reachable from the original dominating set $D$. Therefore, we can conclude that OPT-DSR is in PSPACE.

We now give three reductions to show the PSPACE-hardness for split graphs, bipartite graphs and bounded pathwidth graphs, respectively. These reductions are slight adaptations to the ones of PSPACE-hardness for $\mathrm{DSR}_{\mathsf{TAR}}$ developed in [HIM+16] To this end, we use a polynomial-time reduction from the optimization variant of VERTEX COVER RECONFIGURATION, denoted by OPT-VCR.

Recall that given a graph $G = (V, E)$, a *vertex cover* is a subset of vertices that contains at least one endpoint of each edge in $E$. We now give the formal definition of OPT-VCR. Suppose that we are given a graph $G$, two integers $k, s \geq 0$, and a vertex cover $C$ of $G$ whose

cardinality is at most $k$. Then OPT-VCR asks for a vertex cover $C_t$ of size at most $s$ reachable from $C$ under the TAR($k$) rule. This problem is PSPACE-complete even for bounded pathwidth graphs. Indeed, Ito et al. [IMNS19] showed that OPT-ISR is PSPACE-complete on bounded bandwidth graphs. Since any vertex cover of a graph is the complement of an independent set, the PSPACE-completeness of OPT-VCR follows.

**Lemma 3.35.** OPT-DSR *is PSPACE-hard even for split graphs.*

*Proof.* As we said, we give a polynomial-time reduction from OPT-VCR. More precisely, we extend the idea developed in [Ber84, CN84] to prove the NP-hardness of DOMINATING SET problem on split graphs. Let $(G', k', s', C)$ be an instance of OPT-VCR with $V(G') = \{v_1, v_2, \ldots, v_n\}$ and $E(G') = \{e_1, e_2, \ldots, e_m\}$. We construct the corresponding split graph $G$ as follows (see also Figure 3.17). Let $V(G) = A \cup B$, where $A = V(G')$ and $B = \{w_1, w_2, \ldots, w_m\}$; the vertex $w_i \in B$ corresponds to the edge $e_i \in E(G')$. We join all pairs of vertices in $A$ so that $A$ forms a clique in $G$. In addition, for each edge $e_i = v_p v_q$ in $E(G')$, we join $w_i \in B$ with each of $v_p$ and $v_q$. Let $G$ be the resulting graph, and let $(G, k = k', s = s', D = C)$ be the corresponding instance of OPT-DSR (we will prove later that $D$ is a dominating set of $G$). Clearly, this instance can be constructed in polynomial time.


(a) Original graph $G'$. (b) Corresponding split graph $G$.

Figure 3.17 – Reduction for Lemma 3.35. Note that $\{v_2, v_4\}$ forms a vertex cover of $G'$, and a dominating set of $G$.

It remains to prove that $(G', k', s', C)$ is a yes-instance for OPT-VCR if and only if $(G, k, s, D)$ is a yes-instance for OPT-DSR.

($\Rightarrow$) We start by the only-if direction. Suppose that $(G', k', s', C)$ is a yes-instance. Then, there exists a vertex cover $C_t$ of size at most $s'$ reachable from $C$ under the TAR($k'$) rule. Since $k' = k$, $s = s'$ and both problems employ the same reconfiguration rule, it suffices to prove that any vertex cover of $G'$ is a dominating set of $G$. Since $C \subseteq V(G') = A$ and $A$ is a clique, all vertices in $A \setminus C$ are dominated by the vertices in $C$. Thus, consider a vertex $w_i \in B$, which corresponds to the edge $e_i = v_p v_q$ in $E(G')$. Then, since $C$ is a vertex cover of $G'$, at least one of $v_p$ and $v_q$ must be contained in $C$. This means that $w_i$ is dominated by the endpoint $v_p$ or $v_q$ in $G$. Therefore, each vertex cover in the reconfiguration sequence between $C$ and $C_t$ is a dominating set of $G$ (including $D = C$ and $D_t = C_t$) and thus, $(G, k, s, D)$ is a yes-instance.

($\Leftarrow$) We now focus on the if direction. Suppose that $(G, k, s, D)$ is a yes-instance. Then, there exists a dominating set $D_t$ of $G$ of size at most $s$ reachable under the TAR($k$) rule by a sequence $\mathcal{R} = \langle D_0, D_1, \ldots, D_t \rangle$, with $D = D_0$. Recall that $D = C$ and thus $D$ is a vertex cover of $G'$. We want to produce a sequence of dominating sets that are subsets of $A$. To this end, we proceed by eliminating the vertices of $B$ that appears in $\mathcal{R}$ one by one from the sequence. Let $i$ be the smallest index such that $D_i \in \mathcal{R}$ contains a vertex $w_j \in B$ associated to the edge $v_k v_l \in E(G)$.

Let $j \geq i$ be the largest index such that every dominating $D_k \in \mathcal{R}$ ($i \leq k \leq j$) contains $w_j$. Then, for every $D_k \in \mathcal{R}$ ($i \leq k \leq j$) we instead consider the set $D'_k = (D_k \setminus w_j) \cup \{v_k\}$, where $v_k \in N_{G'}(w_j)$. Observe that each $D'_k$ is a dominating set since $N_{G'}[w_j] \subseteq N_{G'}[v_k]$. If $v_k \in D_{i-1}$, observe that $D_{i-1} = D'_i$. Otherwise, $D'_i$ is obtainable from $D_{i-1}$ in one step since we just replace the addition of $w_j$ by the one of $v_k$. Moreover, due to the choice of $j$, $D_{j+1} = D_j \setminus \{w_j\}$. Hence, $D_{j+1}$ contains a vertex in $A$ adjacent to $w_j$. If this vertex is $v_k$, $D'_j = D_{j+1}$. Otherwise, $D_{j+1} = D'_j \setminus \{v_k\}$, which corresponds to a valid TAR move. Finally, since we ensure that each dominating set $D'_k$ with $i \leq k \leq j$ contains $v_k$, we can ignore each move in the subsequence of $\mathcal{R}$ that touches $v_k$. Hence, $D'_k \leftrightarrow D'_{k+1}$ holds, for every $i \leq k < j$. The resulting subsequence does not touch $w_j$. Hence by repeating this process for each subsequence containing $w_j$ we get a new sequence that does not touch $w_j$ at all. We then repeat this process for every vertex of $B$ that appears in $\mathcal{R}$ and we obtain a sequence $\mathcal{R}'$ where each dominating set is a subset of $A$. Finally, observe that any dominating set $D$ of $G$ such that $D \subseteq A = V(G')$ forms a vertex cover of $G'$, because each vertex $w_i \in B$ is dominated by at least one vertex in $D \subseteq V(G')$. Therefore, $(G', s', k', C)$ is a yes-instance. □

Finally, the two following lemmas complete the proof of Theorem 3.34.

**Lemma 3.36.** OPT-DSR *is PSPACE-hard even for bipartite graphs.*

*Proof.* We give a polynomial-time reduction from OPT-DSR on split graphs to the same problem restricted to bipartite graphs. The same idea is used in the NP-hardness proof of DOMINATING SET problem on bipartite graphs [Ber84, CN84].

Let $(G', k', s', D')$ be an instance of OPT-DSR, where $G'$ is a split graph. Then $V(G')$ can be partitioned into two subsets $A$ and $B$ which form a clique and an independent set in $G'$, respectively. Furthermore, by the reduction given in the proof of Lemma 3.35, the problem on split graph remains PSPACE-complete even if the given dominating set $D'$ consists of vertices only in $A$. We thus assume that $D' \subseteq A$ holds.

We now construct the corresponding bipartite graph $G$, as follows. First, we delete any edge joining two vertices in $A$ so that $A$ forms an independent set. Then, we add a new edge consisting of two new vertices $x$ and $y$, and join $y$ with each vertex in $A$. The resulting graph $G$ is bipartite (see Figure 3.18 for an example).


(a) Original split graph $G'$. (b) Corresponding bipartite graph $G$.

Figure 3.18 – Reduction for Lemma 3.36. Note that $\{v_2, v_4\}$ forms a dominating set of $G'$, and $\{y, v_2, v_4\}$ a dominating set of $G$.

Let $D = D' \cup \{y\}$, $k = k' + 1$ and $s = s' + 1$. Then we obtain the corresponding OPT-DSR instance $(G, k, s, D)$ where $G$ is bipartite (here again, we will prove later that $D$ is dominating set of $G$). Clearly, this instance can be constructed in polynomial time. We then prove that $(G, k', s', D')$ is a yes-instance if and only if $(G, k, s, D)$ is a yes-instance.

($\Rightarrow$) We first prove the only-if direction. Suppose that there exists a dominating set $D'_t$ of $G'$ such that $D' \overset{k'}{\leftrightsquigarrow} D'_t$ and $|D'_t| \leq s'$. Consider any dominating set $D''$ of $G'$. Then, $B \subseteq N_G[D'']$ holds because $B \subseteq N_{G'}[D'']$ and we have deleted only the edges which have both endpoints in $A$. Since $N_G[y] = A \cup \{x\}$, we can conclude that $D'' \cup \{y\}$ is a dominating set of $G$. Furthermore, $|D'_t \cup \{y\}| \leq s' + 1 = s$. Thus there exists a dominating set $D_s$ of $G$ such that $D \overset{k}{\leftrightsquigarrow} D_t$ and $|D_t| \leq s$, as desired.

($\Leftarrow$) We then prove the if direction. Suppose that there exists a dominating set $D_t$ of $G$, of size at most $s$ and reachable from $D$ by a TAR($k$) sequence $\mathcal{R} = \langle D_0, D_1, \ldots, D_t \rangle$, with $D = D_0$. Recall that $D = D' \cup \{y\}$, and notice that any dominating set of $G$ contains at least one of $x$ and $y$. Since $N_G[x] \subset N_G[y]$, we can assume that $D_s$ contains $y$. Therefore, we can also assume that $y$ is contained in every dominating set of the reconfiguration sequence. Recall that the assumption $D' \subseteq A$ holds. As in the proof of Lemma 3.35, we can produce an equivalent sequence $\mathcal{R}'$ that does not touch any vertex of $B$. Again, if a dominating set $D_i$ touches a vertex $w_j$ associated to the edge $v_k, v_l$, we replace $D_i$ by $D'_i = (D_i \setminus w_j) \cup v_k$. We repeat the operation for all $w_j$ and obtain the wanted sequence. Consider any dominating set $D$ of $G$ in such a reconfiguration sequence. Since $y \in D$, we have $|D \cap V(G')| \leq k - 1 = k'$. Furthermore, since $D \cap V(G') \subseteq A$ and $A$ forms a clique in $G'$, we have $A \subseteq N_{G'}[D \cap V(G')]$. Since there is no edge joining $y$ and a vertex in $B$, each vertex in $B$ is dominated by some vertex in $D \cap V(G')$. Therefore, $D \cap V(G')$ is a dominating set of $G'$ with cardinality at most $k'$, and hence there exists a dominating set $D'_t$ of $G'$ such that $D' \overset{k'}{\leftrightsquigarrow} D'_t$ and $|D'_t| \leq s'$. □

**Lemma 3.37.** OPT-DSR *is PSPACE-hard even for bounded pathwidth graphs.*

*Proof.* Our reduction follows from the original reduction from VERTEX COVER to DOMINATING SET [GJ79]. Let $(G', k', s', C)$ be an instance of OPT-VCR, where the pathwidth of $G'$ is bounded; note again that OPT-VCR is PSPACE-complete even for bounded pathwidth graphs. Let $G$ be the graph constructed from $G'$ as follows: for each edge $u, w$, we add a new vertex $v_{uw}$ and join it with both of $u$ and $w$ by edges (see Figure 3.19).


Figure 3.19 – Reduction for Lemma 3.37. (a) Original edge $uv$ in $G'$ and (b) gadget in $G$ for $uv$.

We now claim that the pathwidth of $G$ is bounded. Let $\mathcal{P}'$ be the path decomposition of $G'$ of width $pw(G')$. Then we construct a path decomposition $\mathcal{P}$ of $G$ from $\mathcal{P}'$ by adding each new vertex $v_{uw}$ to any vertex subset $X'_i$ in $\mathcal{P}'$ in which both $u$ and $w$ are contained; from the definition of a path decomposition, such a vertex subset $X'_i$ always exists. For each vertex subset in $\mathcal{P}'$, the number of pairs of two vertices is $O(pw(G')^2)$, and hence the resulting path decomposition $\mathcal{P}$ has width $O(pw(G')^2)$; it is bounded by some constant since $pw(G')$ is. Let $(G, k = k', s = s', D = C)$ be the corresponding instance of OPT-DSR. This construction can clearly be done in polynomial time. Hence, it remains to prove that $(G', k', s', C)$ is a yes-instance for OPT-VCR if and only if $(G, k, s, D)$ is a yes-instance for OPT-DSR.

($\Rightarrow$) Suppose first that $(G', k', s', C)$ is a yes-instance and let $C_t$ be a vertex cover of size at most $s'$ reachable from $C$ under the TAR($k'$) rule, by a sequence $\mathcal{R}'$. Since any vertex cover of

$G'$ is a dominating set of $G$ and $k = k', s = s'$, then the sequence $\mathcal{R}'$ yields a reconfiguration sequence from $D = C$ to $D_t = C_t$. Thus, $(G, k, s, D)$ is a yes-instance.

($\Leftarrow$) We now prove the other direction. Suppose that $(G, k, s, D)$ is a yes-instance and let $\mathcal{R} = \langle D_0, D_1, \ldots, D_t \rangle$ be a TAR($k$) sequence of dominating sets of $G$ starting at $D = D_0$ and reaching a dominating set $D_t$ that satisfies $|D_t| \leq s$. Recall that $D$ does not contain any newly added vertex in $V(G) \setminus V(G')$. We want a sequence $\mathcal{R}'$ that does not touch any newly added vertex $v_{uw}$. To this end, in the same spirit as in the proofs of Lemmas 3.35 and 3.36, we eliminate the vertices of $V(G) \setminus V(G')$ one by one. If a $D_i$ contains a vertex $v_{uw}$, then we replace $D_i$ by $D'_i = (D_i \setminus v_{uw}) \cup \{u\}$, which is also a dominating set and is reachable in one step from $D_{i-1}$. Thus, the resulting sequence does not touch $v_{uw}$, and by repeating the operation to all vertices of $V(G) \setminus V(G')$, we obtain the wanted TAR($k$) sequence $\mathcal{R}'$ of subsets of $V(G')$. In this way, we can obtain a reconfiguration sequence of vertex covers in $G'$ between $C$ and $C_t = D_t$ as needed.

Since OPT-VCR is PSPACE-complete for bounded pathwidth graphs, the reduction above implies PSPACE-hardness on bounded pathwidth graphs. □

**Linear-time algorithms** We now explain how OPT-DSR can be solved in linear time for several graph classes including cographs, trees and interval graphs. To this end, we deal with the concept of a canonical dominating set seen in Section 3.2.1. Recall that a dominating set $C$ is *canonical* if $C$ is a minimum dominating set which is reachable from any dominating set $D$ under the TAR($|D| + 1$) rule. Then we have the following theorem.

**Theorem 3.38.** *Let $\mathcal{G}$ be a class of graphs such that any graph $G \in \mathcal{G}$ has a canonical dominating set that we can compute it in linear time. Then OPT-DSR can be solvable in linear time on $\mathcal{G}$.*

*Proof.* Let $(G, k, s, D)$ be an instance of OPT-DSR, where $G \in \mathcal{G}$. Recall that we can assume without loss of generality that $s < |D| < k$; we can check in linear time whether the inequality is satisfied or not, and if it is violated, then we know from Observations 3.31 and 3.32 that it is a trivial instance. Since $G \in \mathcal{G}$, $G$ admits a canonical dominating set and we can compute in linear time an actual one. Let $C$ be such a canonical dominating set. Then it follows from the definition that $C$ is reachable from $D$ under the TAR($k$) rule since $k \geq |D| + 1$. Since $C$ is a minimum dominating set, we can output it if $|C| \leq s$ holds, and no-instance otherwise. All processes can be done in linear time, and hence the theorem follows. □

Recall that Haddadan et al. showed in [HIM+16] that every cograph, tree (actually forest), and interval graph admit a canonical dominating set. Their proofs are constructive, and hence we can find an actual canonical dominating set. It is observed that the constructions on cographs and trees can be done in linear time. The construction on interval graphs can also be done in linear time with a nontrivial adaptation by using an appropriate data structure. Therefore, we have obtain the following result regarding the tractability of OPT-DSR.

**Corollary 3.39.** *OPT-DSR can be solved in linear time on cographs, trees, and interval graphs.*

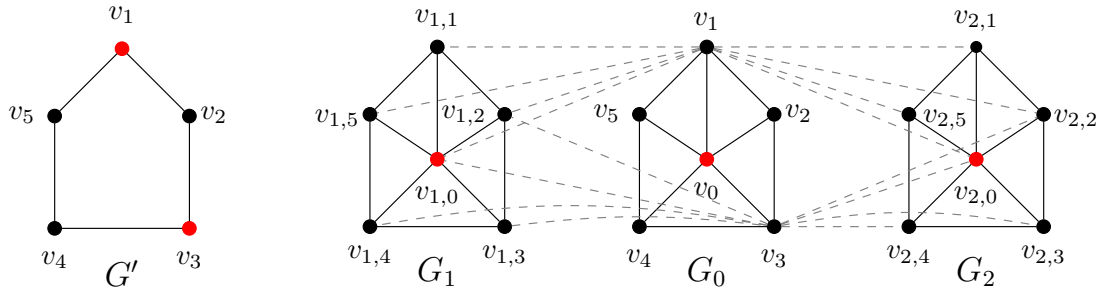
### 3.3.3 Parameterized complexity of OPT-DSR

We now study the fixed-parameter complexity of OPT-DSR with respect to several parameters: the threshold $k$, the solution size $s$, the minimum size of a vertex cover $\tau$ and the degeneracy $d$. We first show that OPT-DSR is W[2]-hard when parameterized by the upper bound $k$. We use the idea of the reduction constructed by Mouawad et al. [MNR+17] to show the W[2]-hardness of $\mathrm{DSR}_{\mathsf{TAR}}$.

**Theorem 3.40.** *OPT-DSR is W[2]-hard when parameterized by the threshold $k$.*

*Proof.* We give an FPT-reduction from the (original) DOMINATING SET problem that is W[2]-hard when parameterized by its natural parameter $k$ [DF99].

Let $(G', k')$ be an instance of the DOMINATING SET problem, where $|V(G')| = n'$ and $V(G') = \{v_1, v_2, \ldots, v_{n'}\}$. Then, we construct the corresponding instance $(G, k, s, D)$ of OPT-DSR, as follows. We first describe the construction of $G$. Let $G_0$ be the graph obtained by adding a universal vertex $v_0$ to $G'$, and $G_1, G_2, \ldots, G_{k'}$ be $k'$ copies of $G_0$. The vertex set of $G$ consists of $\bigcup_{j\in\{0,1,\ldots,k'\}} V(G_j)$. For any $j \in \{1, 2, \ldots, k'\}$ and $i \in \{0, 1, \ldots, n'\}$, we use $v_{j,i}$ to denote the vertex in $G_j$ corresponding to $v_i$ in $G_0$. Then, for each vertex $v_i$ in $G_0$ except for $v_0$, we connect $v_i$ by new edges to all vertices in $N_{G_j}[v_{j,i}]$ in each $j \in \{1, 2, \ldots, k'\}$; formally, the edge set of $G$ consists of $\bigcup_{j\in\{0,1,\ldots,k'\}} E(G_j) \cup \bigcup_{i\in\{1,2,\ldots,n'\}} \bigcup_{j\in\{1,2,\ldots,k'\}} \{v_i w \mid w \in N_{G_j}[v_{j,i}]\}$. This completes the construction of $G$; see Figure 3.20 for an example of this reduction.




(a) $(G', k' = 2)$. (b) $(G, k = 5, s = 2, D)$.

Figure 3.20 – Reduction for Theorem 3.40 with $D' = \{v_1, v_3\}$ and $D = \{v_0, v_{1,0}, v_{2,0}\}$.

Note that for readability purposes, we do not draw all the edges between the vertices in $G'$ and those of $G_j$, for $j \in \{1, 2\}$. The only such drawn edges are the dotted ones (in gray) that are incident to the vertices $v_1$ and $v_3$. We set $k = 2k' + 1$, $s = k'$, and $D = \{v_{j,0} \mid j \in \{0, 1, \ldots, k'\}\}$; notice that $D$ has $k' + 1$ vertices. In this way, we constructed the corresponding instance $(G, k, s, D)$. It remains to prove that $(G', k')$ is a yes-instance if and only if $(G, k, s, D)$ is a yes-instance.

($\Rightarrow$) We first prove the only-if direction. Suppose that $(G', k')$ is a yes-instance, hence there exists a dominating set $D'$ of $G'$ of size at most $k'$. Then by the construction of $G$, we know that $D'$ is also a dominating set of $G$ (if we identify the vertices of $G'$ with those of $G_0$). Thus it suffices to show that $D \overset{k}{\leftrightsquigarrow} D'$, since $D'$ has at most $s = k'$ vertices. We first add vertices in $D'$ to $D$ one by one; this transformation can be done under TAR($k$) since $|D \cup D'| \leq (k' + 1) + k' = k$. We then remove vertices in $D$ one by one. In this way, we can transform $D$ into $D'$ under TAR($k$), and hence $(G, k, s, D)$ has a solution $D'$.

($\Leftarrow$) We then prove the if direction. Suppose that $(G, k, s, D)$ has a solution $D'$. We know that $D'$ has at most $s = k'$ vertices. Then, since $G$ has $k' + 1$ copies $G_0, G_1, \ldots, G_{k'}$, there exists a copy $G_j \in \{G_0, G_1, \ldots, G_{k'}\}$ such that $V(G_j) \cap D' = \emptyset$. We know that $j \neq 0$ because all neighbors of $v_0$ are in $V(G_0)$, hence $D'$ contains at least one vertex in $V(G_0)$. For any $p \in \{1, 2, \ldots, k'\} \setminus \{j\}$, there is no edge joining a vertex in $V(G_j)$ and a vertex in $V(G_p)$. Therefore, for any vertex $v_{j,i}$ in $G_j$, a vertex $u \in D'$ which dominates $v_{j,i}$ is contained in $(V(G_0) \cap D') \setminus \{v_0\}$. Then, by the construction of $G$, $u$ also dominates the corresponding vertex $v_i$ in $G_0$. Thus, we know that $D'' = (V(G_0) \cap D') \setminus \{v_0\}$ is a dominating set of $G'$. Since $|D''| \leq |D'| \leq s = k'$ holds, $D''$ is a desired dominating set of $G'$. □

We now give two FPT algorithms with respect to the combination of the solution size $s$ and the degeneracy $d$, and then with parameter $\tau$, the minimum size of a vertex cover.

**Theorem 3.41.** *OPT-DSR is fixed-parameter tractable when parameterized by $d + s$, where $d$ is the degeneracy and $s$ the solution size.*

To prove the theorem, we give an FPT algorithm with respect to $d + s$. Note that our algorithm uses the idea of an FPT algorithm solving the reachability variant of DOMINATING SET RECONFIGURATION, developed by Lokshtanov et al. [LMP+18]. Their algorithm uses the concept of domination core; for a graph $G$, a *domination core* of $G$ is a vertex subset $C \subseteq V(G)$ such that any vertex subset $D \subseteq V(G)$ is a dominating set of $G$ if and only if $C \subseteq N_G[D]$ [DDF+16].

Suppose that we are given an instance $(G, k, s, D)$ of OPT-DSR where $G$ is a $d$-degenerate graph. By Observation 3.32, we can assume without loss of generality that $|D| < k$. We first check whether $G$ has a dominating set of size at most $s$: this can be done in FPT$(d + s)$ time for $d$-degenerate graphs [AG08]. If $G$ does not have it, then we can instantly conclude that $(G, k, s, D)$ is a no-instance.

In the remainder of this subsection, we assume that $G$ has a dominating set of size at most $s$. In this case, we kernelize the instance: we shrink $G$ by removing some vertices while keeping the existence of a solution until the size of the graph only depends on $d$ and $s$. To this end, we use the concept of domination core.

**Lemma 3.42** ([LMP+18])**.** *If $G$ is a $d$-degenerate graph and $G$ has a dominating set of size at most $s$, then $G$ has a domination core of size at most $ds^d$ and we can find it in FPT$(d + s)$ time.*

Therefore, one can compute a domination core of $G$ of size at most $ds^d$ in FPT$(d + s)$ time by Lemma 3.42. In order to shrink $G$, we use the following reduction rule:

**R1:** if there is a domination core $C$ and two vertices $v_r, v_l \in V(G) \setminus C$ such that $N_G(v_r) \cap C \subseteq N_G(v_l) \cap C$, we remove $v_r$.

We need to prove that **R1** is "safe", that is, we can remove $v_r$ from $G$ without changing the existence of a solution. However, if the input dominating set $D$ contains $v_r$, we cannot do it immediately. Therefore, we first remove $v_r$ from $D$.

**Lemma 3.43.** *Let $D$ be a dominating set such that both $|D| < k$ and $v_r \in D$ hold. Then there exists $D'$ such that $v_r \notin D'$ and $D \overset{k}{\leftrightsquigarrow} D'$, and $D'$ can be computed in linear time.*

*Proof.* We first consider the case where $v_l \in D$. In this case, we simply remove $v_r$ from $D$; let $D'$ be the resulting vertex subset. It is clear that $D \overset{k}{\leftrightsquigarrow} D'$, and hence it suffices to show that $D'$ is a dominating set of $G$. We know that $C \subseteq N_G[D]$ holds by the definition of a domination core. Then since $N_G(v_r) \cap C \subseteq N_G(v_l) \cap C$ and $v_l \in D$ hold, we have $C \subseteq N_G[D \setminus \{v_r\}] = N_G[D']$. Thus $D'$ is a dominating set of $G$.

We then consider the remaining case where $v_l \notin D$. In this case, we can add $v_l$ to $D$ since $|D| < k$. Then the resulting dominating set contains $v_l$, and we can remove $v_r$ as discussed above. □

We can now redefine $D$ as a dominating set which does not contain $v_r$. We then consider removing $v_r$ from $G$. Let $G' = G[V(G) \setminus \{v_r\}]$. The following lemma ensures that removing $v_r$ keeps the existence of a solution.

**Lemma 3.44.** *Let $(G, k, s, D)$ be an instance where $v_r \notin D$. Then, $(G, k, s, D)$ has a solution if and only if $(G', k, s, D)$ has a solution.*

*Proof.* ($\Leftarrow$) We first prove the if direction. Suppose that $(G', k, s, D)$ has a solution $D'_s$. Then there exists a reconfiguration sequence $\mathcal{D}' = \langle D = D'_0, D'_1, \ldots, D'_{\ell'} = D'_t \rangle$ of dominating sets of $G'$. It suffices to show that any dominating set $D'_i$ of $G'$ in $\mathcal{D}'$ is also a dominating set of $G$. Since $D'_i$ is a dominating set of $G'$ and $v_r \notin C$, we have $C \subseteq V(G') \subseteq N_{G'}[D'_i]$. By the definition of domination core, we know that $D'_i$ is also a dominating set of $G$.

($\Rightarrow$) We then prove the only-if direction. Suppose that $(G, k, s, D)$ has a solution $D_s$. Then there exists a reconfiguration sequence $\mathcal{D} = \langle D = D_0, D_1, \ldots, D_\ell = D_t \rangle$ of dominating sets of $G$. Based on $\mathcal{D}$, we construct another sequence $\mathcal{D}' = \langle D = D'_0, D'_1, \ldots, D'_\ell = D'_t \rangle$ of vertex subsets of $G'$, where

$$D'_i = \begin{cases} (D_i \setminus \{v_r\}) \cup \{v_l\} & \text{if } v_r \in D_i \\ D_i & \text{otherwise} \end{cases}$$

for each $i \in \{0, 1, \ldots, \ell\}$. Notice that any vertex subset in $\mathcal{D}'$ does not contain $v_r$. Our claim is that $D'_s$ is a solution of $(G', k, s, D)$. To prove it, we show the following two statements:

(i) for each $i \in \{0, 1, \ldots, \ell\}$, $D'_i$ is a dominating set of $G$ (and hence of $G'$); and

(ii) for each $i \in \{0, 1, \ldots, \ell - 1\}$, $|D'_i \triangle D'_{i+1}| \leq 1$ holds, i.e., we have $D'_i \leftrightarrow D'_{i+1}$.

Then the sequence obtained by removing redundant ones from $\mathcal{D}'$ is a reconfiguration sequence from $D$ to $D'_s$. We first show Statement (i). Let $D_i$ be any dominating set in $\mathcal{D}$. If $v_r \notin D_i$, then the statement clearly holds. Thus we consider the other case where $v_r \in D_i$. Since $D_i$ is a dominating set of $G$, we know $C \subseteq V(G) \subseteq N_G[D_i]$. Furthermore, since $N_G(v_r) \cap C \subseteq N_G(v_l) \cap C$, we have $C \subseteq N_G[(D_i \setminus \{v_r\}) \cup \{v_l\}] \subseteq N_G[D'_i]$. By the definition of domination core, $D'_i$ is a dominating set of $G$, and hence Statement (i) follows.

We then show Statement (ii). Let $D_i$ and $D_{i+1}$ be any two consecutive dominating sets in $\mathcal{D}$. Then, we know $|D_i \triangle D_{i+1}| = 1$. We assume without loss of generality that $D_i \subseteq D_{i+1}$; otherwise the proof is symmetric. We prove the statement in the following three cases:

- **Case 1:** both $v_r \notin D_i$ and $v_r \notin D_{i+1}$ hold;
- **Case 2:** either $v_r \in D_i$ or $v_r \in D_{i+1}$ holds (but not both); and
- **Case 3:** both $v_r \in D_i$ and $v_r \in D_{i+1}$ hold.

In **Case 1**, we know that $|D'_i \triangle D'_{i+1}| = |D_i \triangle D_{i+1}| = 1$, and hence the statement clearly holds. We then consider **Case 2**. In this case, since $D_i \subseteq D_{i+1}$, we observe that $v_r \notin D_i$ and $v_r \in D_{i+1}$, and hence $\{v_r\} = D_{i+1} \setminus D_i$ Therefore, $D'_i \triangle D'_{i+1} = D_i \triangle ((D_{i+1} \setminus \{v_r\}) \cup \{v_l\}) \subseteq \{v_l\}$. Thus we can conclude that $|D'_i \triangle D'_{i+1}| \leq 1$, and hence the statement follows. We finally deal with **Case 3**. In this case, we have $D'_i \triangle D'_{i+1} = ((D_i \setminus \{v_r\}) \cup \{v_l\}) \triangle ((D_{i+1} \setminus \{v_r\}) \cup \{v_l\}) \subseteq D_i \triangle D_{i+1}$. Therefore, $|D'_i \triangle D'_{i+1}| \leq |D_i \triangle D_{i+1}| = 1$ holds, and hence the statement follows. In this way, we can conclude that $D'_s$ is a solution of $(G', k, s, D)$. This concludes the proof. □

We exhaustively apply the reduction rule **R1** to shrink $G$. Let $G_k$ and $D_k$ be the resulting graph and dominating set, respectively. Then, any two vertices $u, v \in V(G_k) \setminus C$ satisfy $N_{G_k}(u) \cap C \neq N_{G_k}(v) \cap C$ (more precisely, $N_{G_k}(u) \cap C \not\subseteq N_{G_k}(v) \cap C$). Then the following lemma completes the proof of Theorem 3.41.

**Lemma 3.45.** *$(G_k, k, s, D_k)$ can be solved in FPT$(d + s)$ time.*

*Proof.* We first show that the size of the vertex set of $G_k$ is at most $f(d,s) = ds^d + 2^{ds^d}$. Since $|C| \leq ds^d$, it suffices to show that $|V(G_k) \setminus C| \leq 2^{ds^d}$ holds. Recall that any two vertices $u, v \in V(G_k) \setminus C$ satisfy $N_{G_k}(u) \cap C \neq N_{G_k}(v) \cap C$. Then since the number of combination of vertices in $C$ is at most $2^{|C|} \leq 2^{ds^d}$, we have the desired upper bound $|V(G_k) \setminus C| \leq 2^{ds^d}$.

We now prove that $(G_k, k, s, D_k)$ can be solved in FPT$(d+s)$ time. To this end, we construct an *auxiliary graph* $G_A$, where the vertex set of $G_A$ is the set of all dominating sets of $G_k$, and any two nodes (that correspond to dominating sets of $G_k$) $D$ and $D'$ in $G_A$ are adjacent if and only if $|D \triangle D'| = 1$ holds. Let $n = |V(G_k)|$ and $m = |E(G_k)|$. Then the number of candidate nodes in $G_A$ (vertex subsets of $G_k$) is bounded by $O(2^n)$. For each candidate, we can check in $O(n+m)$ time if it forms a dominating set. Thus we can construct the vertex set of $G_A$ in $O(2^n(n+m))$ time. We then construct the edge set of $G_A$. There are at most $O(|V(G_A)|^2) = O(4^n)$ pairs of nodes in $G_A$. For each pair of nodes, we can check in $O(n)$ time if their corresponding dominating sets differ in exactly one vertex. Therefore we can construct the edge set of $G_A$ in $O(4^n n)$ time, and hence the total time to construct $G_A$ is $O(4^n n + 2^n(n+m))$ time. We finally search a solution by running a breadth-first search algorithm from $D_k$ on $G_A$ in $O(|V(G_A)| + |E(G_A)|) = O(4^n)$ time.

We can conclude that our algorithm runs in time $O(4^n n + 2^n(n+m))$ in total. Since $n \leq f(d,s)$ and $m \leq n^2 \leq (f(d,s))^2$, this is an FPT time algorithm with parameter $d+s$. □

We end this section with the following theorem:

**Theorem 3.46.** *OPT-DSR is fixed-parameter tractable when parameterized by $\tau$.*

Let $(G, k, s, D)$ be an instance of OPT-DSR. As in the previous section, we may first assume by Observation 3.32 that $|D| < k$. We first establish the following fact:

**Observation 3.47.** *If $G$ is $d$-degenerate, then $d \leq \tau$.*

*Proof.* Let $G$ be a graph, $X$ a minimum vertex cover of $G$ and $H$ be any subgraph of $G$. Recall that $G[V \setminus X]$ is an independent set. If $H$ contains a vertex $v$ outside $X$, then $v$ has a degree at most $\tau$ in $G$ and therefore in $H$. Otherwise, $H$ is a subgraph of $G[X]$ and thus has at most $\tau$ vertices. Hence all vertices of $H$ have degree at most $\tau$ in $H$. Therefore, since any subgraph $H$ of $G$ contains a vertex of degree at most $\tau$, $G$ is $\tau$-degenerate. □

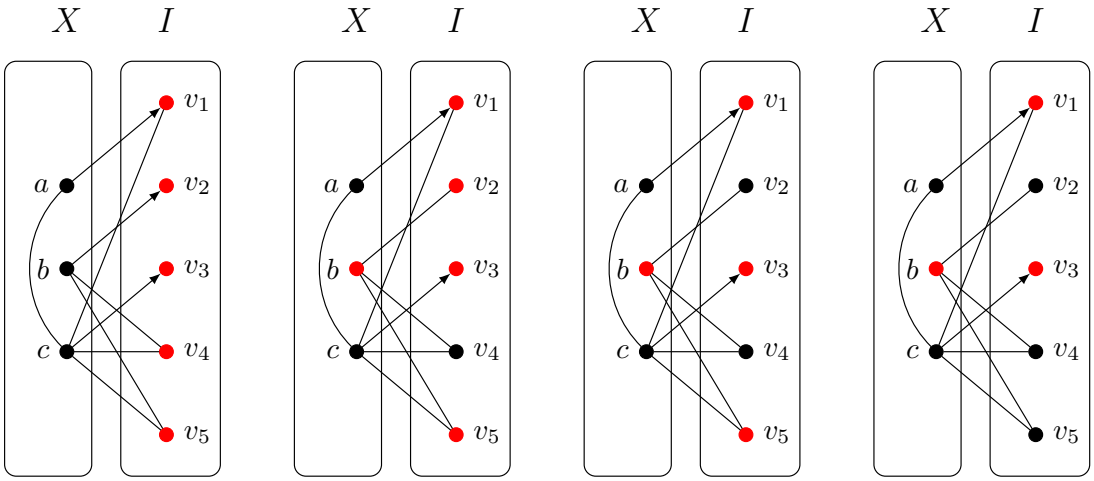
We are now able to get down to the proof of Theorem 3.46, by providing an algorithm that solves OPT-DSR and runs in time FPT$(\tau)$. We first compute a minimum vertex cover $X \subseteq V(G)$ of $G$ in time FPT$(\tau)$ [CKX10]. We partition the vertices of $G$ into two components, the vertex cover $X$ and the remaining vertices $I$. By definition of vertex cover, no edge can have both endpoints outside $X$, therefore $I$ is an independent set. Note that if $s \leq \tau$, then by Observation 3.47 we have $d + s \leq 2\tau$, where $d$ is the degeneracy of $G$. In this case we are able to use the algorithm of the last section, that runs in time FPT$(d+s)$.

We may therefore assume $\tau < s$. In that case, we have the following lemma:

**Lemma 3.48.** *If $\tau < s$, then $(G, k, s, D)$ is a* yes*-instance.*

*Proof.* In the remainder of the proof, we assume that the graph $G$ has no isolated vertex since an isolated vertex *must* belong to any dominating set of $G$. We now prove that $(G, k, s, D)$ is always a yes-instance, i.e., there exists a dominating set of size at most $\tau$ that is reachable from $D$ under the TAR($k$) rule.

We associate with every vertex $v \in X \setminus D$ a *special neighbor* among its neighbors that dominate it (which can be either in $X$ or $I$), i.e., we pick arbitrarily a vertex in $N_G[v] \cap D$. We denote




(a) $D_0 = D$ (b) $D_1$ (c) $D_2$ (d) $D_3 = D_t$

Figure 3.21 – Reconfiguration sequence from the original dominating set $D = I$ to the target one $D_t = \{b, v_1, v_3\}$. $D_1$ is obtained from $D_0$ by applying Rule (ii) and $D_2$ (resp. $D_3$) obtained from $D_1$ (resp. $D_2$) by applying Rule (i). The special neighbor of a vertex $v \in X \setminus D$ is the one pointed by its outgoing edge.

this special neighbor $t(v)$. Let $T$ be the set of special neighbors, i.e., $T = \{t(v) \mid v \in X \setminus D\}$. This corresponds to the set of vertices that are used to dominate the vertices in $X$ that do not belong to $D$. Note that $|T| \leq \tau$.

We are now able to describe the algorithm we use to output $D_t$, the target dominating set. It consists in exhaustively applying the two following rules on the vertices of $I$ that belong to the current dominating set:

(i) if there is a vertex $v$ in $I$ but not in $T$ that is already dominated by another vertex, then we remove $v$ from the dominating set; and

(ii) if there is a vertex $v$ in $I$ but not in $T$ that is dominated only by itself, then we add any one of its neighbors $u \in X$ to the dominating set, and then remove $v$. The vertex $u$ does not need a special neighbor anymore, since it now belongs to the dominating set. We thus update the set $T$ by only keeping the special neighbors $t(w)$ of vertices $w$ that are still in $X \setminus D$.

We first prove that these two rules are safe, i.e., we do not break the domination property at any step. Since Rule (i) removes a vertex $v$ that is not required to dominate itself or another vertex $u \in X$ (because it has not been chosen in $T$), we can safely remove it. In Rule (ii), after adding a neighbor of $v$ to the dominating set, $v$ is not required to dominate itself anymore. Since $v$ is not in $T$, we can now apply Rule (i) which is safe.

Recall that $|D| < k$. Then, each dominating set obtained after applying one of these rules is of size at most $k$ since Rule (i) only removes vertices and Rule (ii) consists in an addition immediately followed by a removal.

Now, let $D_t$ be the dominating set obtained once we cannot apply Rule (i) and Rule (ii) anymore (see Figure 3.21 for an example). All remaining vertices in $I \cap D_t$ now belong to $T$. By definition of $T$, each vertex in $X \setminus D_t$ has (exactly) one neighbor in $T$ (but they are not necessarily distinct). Therefore, $|I \cap D_t| \leq |X \setminus D_t|$. As a result, $|D_t| = |X \cap D_t| + |I \cap D_t| \leq |X \cap D_t| + |X \setminus D_t| = |X| = \tau$. Since $\tau < s$, the size of $D_t$ is at most $s$, as desired. □

It remains to discuss the complexity of this algorithm. As we already said, we first compute a minimum vertex cover $X$ of $G$ in time FPT($\tau$). If $s \leq \tau$, we run the FPT algorithm of Theorem 3.41. Otherwise, we first compute the set $T$ and then run the subroutine which are both described in the proof of Lemma 3.48. The two rules used in this subroutine only apply to vertices that belong to the set $I$ and whenever one is applied, exactly one vertex in $I$ is removed (and none is added). Hence, they are applied at most $|I \cap D|$ times. Therefore, the subroutine runs in polynomial time and produces the desired dominating set $D_t$. As a result, this algorithm is FPT with respect to $\tau$. This concludes the proof.

### 3.3.4 Changing the target dominating set

As discussed in Section 3.3.1, there might be several definitions of what is a "more desirable" solution. Here, we would like to output an independent dominating set if it exists, and no-instance otherwise. The result we present in this section is based on an ongoing joint work with Alexandre Blanché and Haruka Mizuta. For a discussion on independent dominating sets, we refer the reader to Section 2.1.3. More precisely, we focus on the following problem that we call OPT-IDSR:

OPT-IDSR

**Instance:** A graph $G$, two integers $k, s \geq 0$, a dominating set $D$ of $G$ such that $|D| \leq k$.
**Output:** An independent dominating set $D'$ of $G$ such that $|D'| \leq s$ and $D'$ is reachable from $D$ under the TAR($k$) rule if it exists, no-instance otherwise.

We denote by 4-tuple $(G, k, s, D)$ an instance of OPT-IDSR. First, observe that a similar result than Observation 3.33 holds for OPT-IDSR:

**Observation 3.49.** *Let $G = (V, E)$ be a graph and $s$ be an integer. The instance $(G, |V|, s, V)$ of OPT-IDSR is equivalent to finding an independent dominating set of $G$ of size at most $s$.*

*Proof.* Suppose that $G$ admits an independent dominating set $D_t$ of size at most $s$. Since we started from a dominating set containing all the vertices of $G$, it is sufficient to remove one by one each vertex in $V \setminus D_t$ to reach $D_t$ and thus $(G, |V|, s, V)$ is a yes-instance. The converse direction is trivial: if $(G, |V|, s, V)$ is a yes-instance, then $G$ has an independent dominating set of size at most $s$. □

Garey and Johnson proved that Independent Dominating Set is NP-complete [GJ79]. It was then shown to be NP-complete even when restricted to bipartite or comparability graphs by Corneil and Perl [CP84], or when restricted to line graphs by Yannakakis and Gavril [YG80]. Hence, OPT-IDSR is NP-hard on all these graph classes as a corollary of Observation 3.49.

In the remaining of this section, we actually prove that OPT-IDSR is PSPACE-complete even when restricted to bipartite graphs. First, observe that OPT-IDSR is in PSPACE. Indeed, when we are given a dominating set $D$ as a solution for some instance of OPT-IDSR, we can check in polynomial time whether it is actually an independent dominating set of size at most $s$ or not. Furthermore, since $\text{DSR}_{\text{TAR}}$ is in PSPACE, we can check in polynomial space whether it is reachable from the original dominating set $D$. Therefore, we can conclude that OPT-IDSR is in PSPACE. It remains to prove that it is PSPACE-hard.

Recall that Haddadan et al. [HIM+16] proved that $\text{DSR}_{\text{TAR}}$ is PSPACE-complete. Actually, one of their hardness proof is based on a reduction from Vertex Cover Reconfiguration, which is known to be PSPACE-complete even if the input graph has bounded bandwidth and the two vertex covers are minimum [Wro18]. More precisely, their reduction follows from the classical reduction from Vertex Cover to Dominating Set [GJ79] and it ensures that if the

vertex covers are minimum, so are the corresponding dominating sets. Let $G$ be a graph, and $D_s, D_t$ be two dominating sets of size $k$ of $G$. Recall that there exists a TJ-sequence between $D_s$ and $D_t$ if and only if there is a TAR$(k+1)$-sequence. It follows that the reconfiguration of minimum dominating sets under TJ (denoted by MDSR) is PSPACE-complete. Finally, recall that for any graph $G$ on $n$ vertices with no isolated vertex, $\gamma(G) \leq \lfloor n/2 \rfloor$ (see Section 2.1).

**Theorem 3.50.** *OPT-IDSR is PSPACE-hard, even if the input graph is a bipartite.*

To prove Theorem 3.50, we give a polynomial-time reduction from MDSR. We first explain the construction. Let $(G, D_s, D_t)$ be an instance of MDSR, where $D_s$ and $D_t$ are two minimum dominating sets of a connected graph $G$ on at least three vertices. Note that $|D_s| = |D_t| \leq \lfloor V(G)/2 \rfloor$. For convenience, let $k = |D_s| = |D_t|$. We construct the corresponding bipartite graph $G'$ as follows. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$. For each vertex $v_i \in V(G)$, we create a set $A_i = \{a_{i,1}, a_{i,2} \ldots, a_{i,k+2}\}$ containing $k+2$ copies of $v_i$. We also make another copy $b_i$ of $v_i$. Let $A = \bigcup_{i=1}^{n} A_i$ and $B = \{b_1, b_2, \ldots, b_n\}$. We add an edge between $a_{i,j}$ and $b_k$ in $G'$ if and only if $v_k \in N_G[v_i]$. In other words, every vertex $b_k \in B$ satisfies $N(b_k) = \{\cup A_i \mid v_i \in N_G[v_k]\}$. Note that any pair $(b_{i,j}, b_{i,j'})$ of vertices of $A_i$ satisfies $N(b_{i,j}) = N(b_{i,j'})$, for every $1 \leq i \leq n$. We then add a vertex $x_s$ adjacent to all the vertices in $B$, and a vertex $x_t$ adjacent to all the vertices in $B \setminus \{b_i \mid v_i \in D_t\}$. Finally, we add two vertices $x_m$ and $y_1$, both adjacent to $x_s$ and $x_t$, and we attach a leaf $y_2$ to $y_1$. Observe that $A \cup \{x_s, x_t, y_2\}$ and $B \cup \{x_m, y_1\}$ induce two independent sets. Hence, $G'$ is bipartite and the construction can be done in polynomial time. See Figure 3.22 for an example.

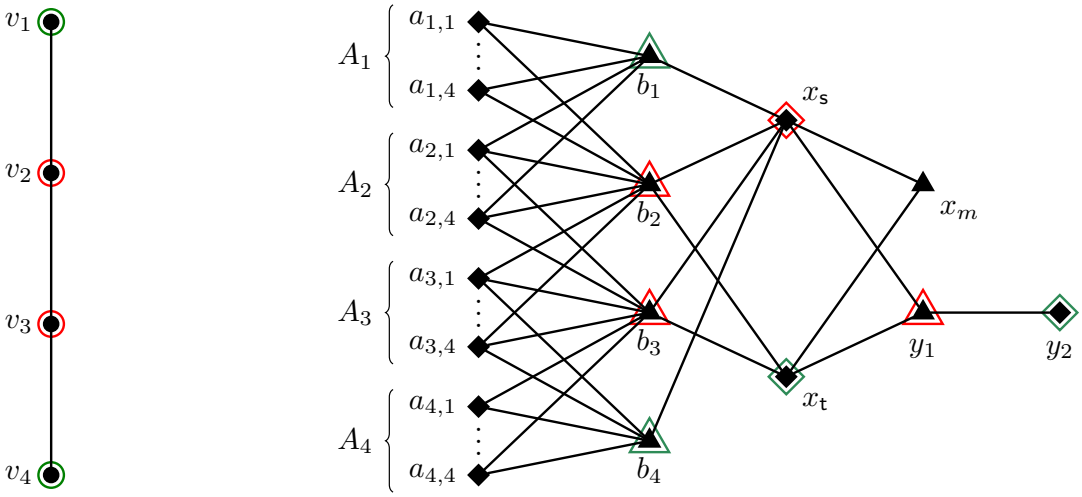



Figure 3.22 – Example for the reduction of Theorem 3.50: the original instance is $(G, D_s = \{v_2, v_3\}, D_t = \{v_1, v_4\})$ and the corresponding instance $(G', 5, 4, D'_s = \{b_2, b_3, x_s, y_1\})$ of OPT-IDSR, where $G'$ is a bipartite graph. The only independent dominating set of size $k+2 = 4$ of $G'$ is $D_t = \{b_1, b_4, x_t, y_2\}$.

**Proposition 3.51.** *The graph $G'$ has no dominating set of size at most $k+1$.*

*Proof.* First, observe that at least two vertices in $V(G') \setminus (A \cup B)$ are required in any dominating set to dominate $x_m$ and $y_2$. But at least $k = \gamma(G)$ vertices are needed to dominate the vertices in $A$ by construction of $G'$. □

**Proposition 3.52.** *The set $D'_s = \{b_i \in B \mid v_i \in D_s\} \cup \{x_s, y_1\}$ is a dominating set of size $k+2$ of $G'$.*

*Proof.* The fact that $|D'_s| = k+2$ is straightforward since $|D_s| = k$ and we add in $D'_s$ exactly one vertex for each vertex in $D_s$, plus $x_s$ and $y_1$. It remains to prove that $D'_s$ is a dominating set of $G'$.

First, observe that $\{x_s, y_1\}$ dominates $B \cup \{x_m, x_s, x_t, y_1, y_2\}$. Finally, consider a vertex $a_{i,j} \in A$. Suppose that the corresponding vertex $v_i \in V(G)$ is dominated by some vertex $v_k \in V(G)$ in $D_s$. By construction of $G'$, $a_{i,j}b_k \in E(G')$ and note that $b_k \in D'_s$. Hence, $a_{i,j}$ is dominated, and thus $D'_s$ is a dominating set of $G'$. □

**Lemma 3.53.** *There is no dominating set of size $k+2$ of $G'$ containing a vertex in $A$.*

*Proof.* Let $D'$ be a dominating set of size $k+2$ of $G'$. Note that since $N[x_m] = \{x_m, x_s, x_t\}$ and $N[y_2] = \{y_1, y_2\}$, $D'$ must contain at least two vertices in $V(G') \setminus (A \cup B)$ in order to dominate $x_m$ and $y_2$. It follows that $D' \cap (A \cup B) \leq k$. Note also that $|A_i| = k+2$ and $G'[A_i]$ is an independent set, for every $1 \leq i \leq n$. Hence, for every set $A_i$ (with $1 \leq i \leq n$), $D'$ must contain a vertex $b_j \in B$ such that $A_i \subseteq N(b_j)$. Any subset $X \subseteq B$ such that each $A_i$ is at distance one from $X$ is of size at least $k = \gamma(G)$, since it corresponds to a dominating set of $G$. Hence, $D'$ contains exactly $k$ vertices in $B$ and no vertex in $A$. □

**Lemma 3.54.** *$G'$ has exactly one independent dominating set of size $k+2$.*

*Proof.* We first prove that $G'$ admits an independent dominating set of size $k+2$. Let $X = \{b_i \in B \mid v_i \in D_t\}$. We consider the set $D'_t = X \cup \{x_t, y_2\}$. Recall that $x_t$ is by construction of $G'$ adjacent to all the vertices in $B \setminus X$. Hence, $G'[D'_t]$ is an independent set of size $k+2$. By definition, $X$ dominates $A \cup X \cup \{x_s\}$. The remaining vertices in $B$ (i.e., the set $B \setminus X$) are dominated by $x_t$, as well as $x_m$ and $y_1$. Finally, $y_2$ dominates itself. It follows that $D'_t$ is an independent dominating set of size $k+2$ of $G'$. It remains to prove that it is the only one. Let $D'$ be an independent dominating set of $G'$ of size $k+2$.

*Claim* 1. $D' \cap \{x_s, x_t\} = \{x_t\}$.

*Proof.* By Lemma 3.53, $D' \cap A = \emptyset$. Hence, in order to dominate the set $A$, $D'$ must contain at least one vertex of $B$. Besides, note that at least $k$ vertices of $B$ are required to dominate $A$ since $k$ corresponds to the size of a minimum dominating set of $G$. On the other hand, at least two vertices of $V(G') \setminus (A \cup B)$ are required to dominate $\{x_m, y_2\}$. It follows that $|D' \cap B| = k$. However, recall that $|B| = |V(G)|$ and $k \leq \lfloor V(G)/2 \rfloor$. Hence, $|B \setminus D'| \geq 1$. This implies that $D'$ contains at least one vertex in $\{x_s, x_t\}$ to dominate the vertices in $B \setminus D'$. Since $x_s$ is adjacent to all the vertices in $B$ and $D'$ contains at least one vertex of $B$, we must have $D' \cap \{x_s, x_t\} = \{x_t\}$ since $G[D']$ is independent. ◊

Note that it follows from Claim 1 that $D' \cap B = X$ since $x_t$ is adjacent to all the vertices in $B \setminus X$ but has no neighbor in $X$, and $X$ dominates $A$. Moreover, since $x_s$ is adjacent to all the vertices in $B$, $x_s$ is dominated by $X$. Finally, it remains to dominate $y_2$. Since $D'$ contains $x_t$ and $y_1x_t \in E(G')$, $D' \cap \{y_1, y_2\} = \{y_2\}$. As a result, $D' = X \cup \{x_t, y_2\} = D'_t$. This concludes the proof of Lemma 3.54. □

**Lemma 3.55.** *There is no independent dominating set of $G'$ of size $k+3$.*

*Proof.* Assume by contradiction that $G'$ has an independent dominating set $D$ of size $k+3$. We first show that $D$ does not contain any vertex in $A$. Indeed, suppose that $D$ contains a vertex $a_{i,j} \in A$ and recall that $N(a_{i,j'}) = N(a_{i,j})$ for any $1 \leq j' \leq k+2$. So all the vertices in $A_i \setminus \{a_{i,}\}$ must be dominated by themselves; note that this is possible so far since $G'[A_i]$ is independent. Hence, $A_i \subseteq D$ and thus $|D| \geq k+2$. However, at least two extra vertices are needed to dominate $x_m$ and $y_2$, yielding a dominating set of size at least $k+4$. Hence, $D \subseteq B \cup \{x, m, x_s, x_t, y_1, y_2\}$.

Recall that at least $k$ vertices of $B$ are needed to dominate $A$. We now claim that $|D \cap B| < |B|$. Indeed, if $B \subseteq D$ and since we need at least two extra vertices to dominate $x_m$ and $y_2$, we obtain $|D| \geq |B| + 2 = |V(G)| + 2 > \lfloor |V(G)|/2 \rfloor \geq k + 3 = \gamma(G) + 3$ whenever $|V(G)| \geq 3$, a contradiction. Since $x_s$ is adjacent to all the vertices in $B$ and $|D \cap B| \geq 1$, $x_s \notin B$ and $x_s$ is dominated. Hence, we must have $x_t \in D$ to dominate the vertices in $B \setminus D$. This implies that $D \cap B = X$, with $X = B \setminus N(x_t) = \{b_i \mid v_i \in D_t\}$. In particular, $|D \cap B| = k$ and $X \cup \{x_t\} \subseteq D$. However, $X \cup \{x_t\}$ dominates $V(G') \setminus \{y_2\}$. Hence, we cannot add two new vertices so that $|D| = k + 3$ without breaking the independence property. □

Recall that each dominating set of $G'$ has size at least $k + 2$ by Proposition 3.51. In particular, $G'$ has no independent dominating set of size less than $k + 2$. By Lemma 3.55, $G'$ has no independent dominating set of size at least $k + 3$. And there is a unique one, let us say $D'$, of size $k + 2$ by Lemma 3.54. Let $(G', k + 3, k + 2, D'_s)$ be the resulting instance of OPT-IDSR. The following lemma concludes the proof of Theorem 3.20:

**Lemma 3.56.** *$(G, D_s, D_t)$ is a* yes*-instance of $DSR_{\mathsf{TAR}}$ if and only if $(G', k + 3, k + 2, D'_s)$ is a* yes*-instance of* OPT-IDSR.

*Proof.* ($\Rightarrow$) Suppose first that $(G, D_s, D_t)$ is a yes-instance, and let $S$ be a TJ-sequence of length $\ell$ between $D_s$ and $D_t$. Recall that given a dominating set $D_i \in S$ of $G$, the set $D'_i = \{b_j \in B \mid v_j \in D_i\} \cup \{x_s, y_1\}$ is dominating set of $G'$. For any two consecutive dominating sets $D_i$ and $D_{i+1}$ of $S$ (i.e., $D_i = (D_{i+1} \setminus \{v_j\}) \cup \{v_i\}$), we first add $b_j$ and then remove $b_i$. Hence, we obtain a TAR($k + 3$)-sequence of length $2\ell$ between $D'_s$ and $D'_t$. Finally, we can reach the independent dominating set $D'$ from $D'_t$ in four more steps by first moving the token from $x_s$ to $x_t$, and then moving the token from $y_1$ to $y_2$. This yields a reconfiguration sequence of length $2(\ell + 2)$ from $D'_s$ to the unique independent dominating set $D'$ of $G'$.

($\Leftarrow$) Suppose now that $(G', k + 3, k + 2, D'_s)$ is a yes-instance. By Lemma 3.54, the target independent dominating set must be $D'$. Recall that $|D'| = |D'_s| = k + 2$. Hence, consider a TAR($k + 3$)-sequence $S'$ between $D'_s$ and $D'$. By Proposition 3.51, each dominating set in $S'$ has size $k + 2$ or $k + 3$. As discussed in Section 3.1.1, $S'$ can be replaced by an equivalent TJ-sequence, where each dominating set in $S'$ has size $k + 2$. By Lemma 3.53, $D'_i \cap A = \emptyset$ holds for every dominating set $D'_i \in S'$. Besides, it also follows from the proof that $D'_i \cap B = k$ and that the set $X_i = \{v_j \in V(G) \mid b_j \in B \cap D'_i\}$ is a dominating set of $G$. In particular, the restriction of $D'$ to $B$ corresponds to $D_t$, hence the existence of a TJ-sequence between $D_s$ and $D_t$. This concludes the proof of Lemma 3.56. □

### 3.3.5 Concluding remarks

In this section, we studied a new framework of combinatorial reconfiguration recently introduced by Ito et al. [IMNS19] for Independent Set Reconfiguration. This new framework is called *optimization variant* as it asks, given a solution, whether it is possible to reach a more desirable one under a fixed reconfiguration rule. We applied this new framework to the reconfiguration of dominating sets under token addition and removal. We were interested in two different possibilities of what could be a "more desirable" solution. We first considered the problem of finding a smallest dominating set that is reachable from the given one under TAR, that we called OPT-DSR. We observed that OPT-DSR generalizes Dominating Set; hence it is NP-hard. However, we actually showed that it is PSPACE-complete, even when restricted to bipartite graphs, split graphs or bounded bandwidth graphs. We know that OPT-DSR is NP-hard on planar graphs because the host problem is NP-complete on this class. We believe that OPT-DSR is PSPACE-complete on planar graphs but we were not able to prove it and we leave

it as an open problem. On the other hand, we observed that the problem becomes tractable on every graph class that admits a canonical dominating set (see Section 3.2.1).

We then studied the parameterized complexity of OPT-DSR. We showed that the problem is W[2]-hard when parameterized by the threshold $k$. However, it would be interesting to determine whether there exists an XP algorithm with parameter $k$, i.e., an algorithm that runs in time $O(|G|^{f(k)})$ for some computable function $f$. On the other hand, we proved that OPT-DSR is FPT with respect to $d + s$, where $d$ is the degeneracy of the input graph and $s$ the size of the target dominating set. Therefore, the problem is FPT with respect to $s$ for planar graphs or bounded bandwidth graphs. We also showed that it is fixed-parameter tractable with parameter $\tau$, the minimum size of a vertex cover of the input graph.

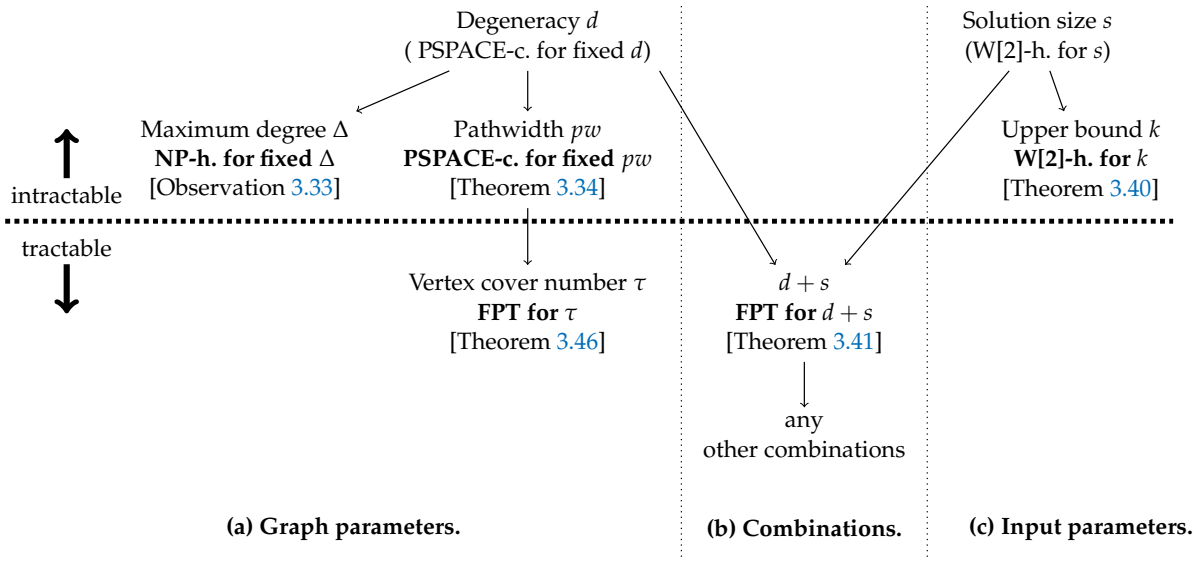



Figure 3.23 – Our results for fixed-parameter tractability, where A $\rightarrow$ B means that the parameter A is bounded by some function of B.

We finally considered a second optimization variant where the goal is to find the smallest independent dominating set that is reachable from the given one under the TAR($k$) rule. This problem generalizes INDEPENDENT DOMINATING SET and is therefore NP-hard on bipartite graphs, comparability graphs, or line graphs. We proved that it is actually PSPACE-complete on bipartite graphs. We did not manage to prove that it is also PSPACE-complete on split graphs (and thus on chordal graphs); we leave it as an open question. However, we believe that one can solve the problem in polynomial time on interval graphs with a dynamic programming algorithm based on a path decomposition of the interval graph.

# 4 Other reconfiguration problems

**Contents**



In this chapter, we change the host problem and we study the reconfiguration of two different problems. The first one is related to spanning trees with some additional constraints on the number of leaves of each spanning tree. This is joint work with Nicolas Bousquet, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Akira Suzuki and Kunihiro Wasa [BIK+20]. We then focus on the problem of recoloring in the so-called LOCAL model in Distributed Computing, based on a joint work with Marthe Bonamy, Mikaël Rabie, Jukka Suomela and Jara Uitto [BOR+18]. In both cases, we are interested in the computational complexity of these two problems.

## 4.1 Reconfiguration of spanning trees with many or few leaves

### 4.1.1 Introduction

The MINIMUM SPANNING TREE problem was one of the first reconfiguration problem to be studied with a complexity perspective in the seminal paper by Ito et al. [IDH+08]. In this problem, we are given an edge-weighted graph $G$, an integer $k$ and two spanning trees $T_s$ and $T_t$ of $G$, both of weight at most $k$. The goal is to determine whether there exists a sequence of edge flips that transforms $T_s$ into $T_t$, each intermediate spanning tree in the sequence being of weight at most $k$ as well. Ito et al. [IDH+08] proved that one can always transform $T_s$ into $T_t$; it easily follows from the exchange properties for matroids. One can then ask the following question: is there still a transformation when we add some constraints on the spanning tree? If not, is it possible to decide efficiently if such a transformation exists?

Mizuta et al. [MIZ16] made a first step in that direction by studied the reconfiguration of Steiner trees via edge flips. More precisely, in STEINER TREE RECONFIGURATION, we are given two Steiner trees $T_s$ and $T_t$ (with the minimum number of edges) connecting all the terminal vertices of an unweighted graph $G$. The goal is to determine whether there exists a sequence of edge flips (i.e., we exchange a single edge at each step) transforming $T_s$ into $T_t$. They proved that the problem is PSPACE-complete on split graphs, while linear-time solvable on interval graphs. Note that even if all the Steiner trees have the same number of edges (because we exchange a single edge at each step), the set of vertices may be different. Indeed, if $T_s$ and $T_t$ have the same vertex set, then there is a transformation between $T_s$ and $T_t$ as implied by the result of Ito et al. [IDH+08] discussed above. Mizuta et al. [MHIZ19] later studied the same problem under two new reconfiguration rules consisting in vertex exchanges. In both cases, the problem is PSPACE-hard since it is a generalization of $(s,t)$-shortest paths reconfiguration, which is PSPACE-complete [Bon12]. However, the complexity differs depending on the rule: in one case it is PSPACE-complete on split graphs and planar graphs, while it is polynomial-time solvable on chordal graphs and planar graphs on the other case [MHIZ19].

In the last few years, many reconfiguration problems have been studied through the lens of edge flips such as matchings (see Section 1.5.4 for an extensive discussion on this problem). Hanaka et al. [HIM+18] studied the complexity of the reconfiguration of some specific partial graphs of a given graph. They showed that the reconfiguration of paths (i.e., subgraphs defined by edges of a given graph that are paths) is NP-hard under the TJ rule (i.e., an edge flip with no restriction on the edges that are flipped). On the other hand, they proved that the reconfiguration of cycles or cliques is linear-time solvable. The case of complete bipartite graphs $K_{n,m}$ is in P, for any pair of positive integer $n, m$. Note that all of these results hold for any graph, and for both TJ and TS (one can only flip two adjacent edges). Finally, they proved that the reconfiguration of trees (not necessarily spanning) is linear-time solvable under TJ.

In this section, we study the reconfiguration of spanning trees with some constraints on the number of leaves. The results presented are from [BIK+20]. We first consider in Section 4.1.2 the reconfiguration of spanning trees with at least $k$ leaves. Recall that the spanning trees with at least $k$ leaves of a graph $G$ on $n$ vertices are related to its connected dominating sets of size $n-k$ (see Section 2.2.1); we will use this observation in the proof of Theorem 4.2. We will then consider the reconfiguration of spanning trees with few leaves in Section 4.1.3.

Before moving on to the results we obtained, let us give some definitions and notations that we will need in the remaining of this section. Let $G$ be a graph on $n$ vertices, and let $T$ be a spanning tree of $G$. Every vertex of degree one is a *leaf* and every vertex of degree at least two is an *internal node*. A vertex of degree at least three is called a *branching node*. Recall that the number of leaves of $T$ is equal to $(\sum_{v \in T}(\max\{0, d_T(v)-2\}))+2$. We denote by $in(T)$ the set of internal nodes of $T$. Note that the number of leaves is indeed $n - |in(T)|$.

### 4.1.2 Spanning tree with many leaves

As we said, we first consider the reconfiguration of spanning trees with at least $k$ leaves via edge flips. More precisely, we are interested in the following problem:

SPANNING TREE WITH MANY LEAVES

**Instance:** A graph $G$, an integer $k \geq 2$, two spanning trees $T_s$ and $T_t$ of $G$ with at least $k$ leaves.

**Question:** yes if and only if there exists a transformation via edge flips between $T_s$ and $T_t$ such that all the intermediate trees have at least $k$ leaves.

We will prove that SPANNING TREE WITH MANY LEAVES is PSPACE-complete even when restricted to bipartite graphs, split graphs, or planar graphs. We will then show that if $k = n - 2$, then the problem is polynomial-time solvable. This result will allow us to show that the problem is polynomial-time solvable on cographs as well. Before moving on to the main results of this section, let us prove the following:

**Lemma 4.1.** *Let $G$ be a graph and $T_1, T_2$ be two trees. There exists a transformation from $T_1$ to $T_2$ such that every intermediate tree $T$ satisfies $in(T) \subseteq in(T_1) \cup in(T_2)$.*
*In particular, all the trees with the same set of internal nodes are in the same connected component of the reconfiguration graph.*

*Proof.* Let us prove that we can iteratively add an edge of $E(T_2) \setminus E(T_1)$ to $T_1$ and remove an edge of $E(T_1) \setminus E(T_2)$ without creating any internal node in $V \setminus (in(T_1) \cup in(T_2))$. Let $uv \in E(T_2) \setminus E(T_1)$. We add this edge to $T_1$ and observe that it creates a unique cycle in the resulting tree $T'$. If it does not create an internal node in $V \setminus (in(T_1) \cup in(T_2))$, we remove from the cycle any edge of $E(T_1) \setminus E(T_2)$. Otherwise, assume $u \notin in(T_1) \cup in(T_2)$. In particular $u$ is a leaf of both $T_1$ and $T_2$, and $uv$ is an edge of $T_2$ so $v \in in(T_2)$. Since $u$ is a leaf of $T_2$, the cycle in $T'$ passes through the other edge incident to $u$. We remove it in order to keep a connected graph. □

**PSPACE-completeness for split graphs, bipartite graphs and planar graphs**

**Theorem 4.2.** SPANNING TREE WITH MANY LEAVES *is PSPACE-complete even when restricted to bipartite graphs or split graphs.*

*Proof.* We first prove Theorem 4.2 for bipartite graphs and then explain how we can adapt the proof for split graphs. We give a polynomial-time reduction from $\mathsf{DSR}_{\mathsf{TAR}}$. Recall that Haddadan et al. [HIM+16] showed that this problem is PSPACE-complete. More precisely, they proved that given a graph $G$ and $D_s$, $D_t$ two dominating sets of $G$, deciding whether there is a reconfiguration sequence between $D_s$ and $D_t$ under the $\mathsf{TAR}(\max\{|D_s|, |D_t|\} + 1)$ rule is PSPACE-complete (see Section 3.2.1).

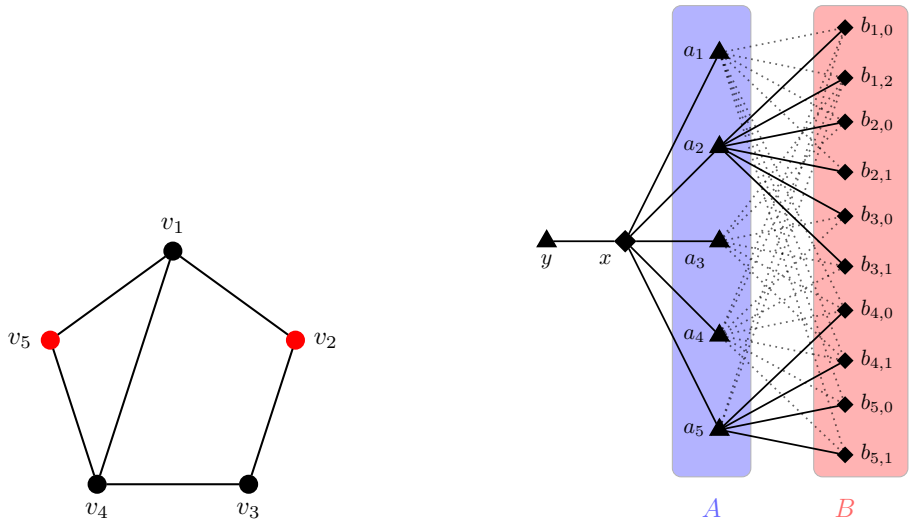
Let $G = (V, E)$ be a graph with vertex set $V(G) = \{v_1, v_2, \ldots, v_n\}$ and let $D_s$, $D_t$ be two dominating sets of $G$. Free to add vertices to the set of smallest size, we can assume without loss of generality that $D_s$ and $D_t$ are both of size $k$. Let $(G, k+1, D_s, D_t)$ be the corresponding instance of $\mathsf{DSR}_{\mathsf{TAR}}$, where $k+1$ is the threshold that we cannot exceed. We construct the bipartite graph $G'$ as follows: we make a first copy $A = \{a_1, a_2, \ldots, a_n\}$ of the vertex set of $G$, and a second copy $B = \{b_{1,0}, b_{1,1}, b_{2,0}, b_{2,1}, \ldots, b_{n,0}, b_{n,1}\}$ where we double each vertex. We add an edge between $a_i \in A$ and $b_{j,k} \in B$ for $k \in \{0, 1\}$ if and only if $v_j \in N_G[v_i]$. Note that $N(b_{i,0}) = N(b_{i,1})$, for every $1 \leq i \leq n$. We finally add a vertex $x$ adjacent to all the vertices in $A$ and we attach it to a degree-one vertex $y$. See Figure 4.1 for an illustration. Note that $G'$ is bipartite since $A \cup \{y\}$ and $B \cup \{x\}$ induce two independent sets.

*Claim* 1. For every spanning tree $T$ of $G'$, $in(T) \cap A$ is a dominating set of $G$.

*Proof.* Let $b_i$ be a vertex of $B$. Since $x$ is an internal node of $T$, there is a path from $b_i$ to $x$. Since $N(b) \subseteq A$, the second vertex of the path is in $A$. So there exists an internal node of $T$ incident to $b_i$. □

*Claim* 2. For every spanning tree $T$ of $G'$, there exists a tree $T_A$ in the connected component of $T$ such that $in(T_A) \subseteq in(T) \cap (A \cup \{x\})$.

*Proof.* If $in(T) \subseteq A \cup \{x\}$, the conclusion holds. So we can assume that there exists $b \in B$ such that $b \in in(T)$. Let us prove that we can transform $T$ into another spanning tree $T'$ such that




(a) Original graph $G$. (b) Corresponding bipartite graph $G'$.

Figure 4.1 – Example for the reduction of Theorem 4.2: the dominating set $D = \{v_2, v_5\}$ of $G$ is depicted by the red vertices and the spanning tree of $G'$ associated with $D$ is the tree induced by the solid edges. For the split case, we add all the possible edges in $G'[A]$ so that $G'[A \cup \{x\}]$ is a clique and $G'[B \cup \{y\}]$ an independent set.

$in(T') \subseteq in(T) \setminus \{b\}$ without creating a new internal node. First, recall that $x \in in(T)$ since $x$ must be an internal node in any spanning tree of $G$. Let $a$ be the unique neighbor of $b$ in the path from $b$ to $x$ in $T$. Now, for every vertex $a' \neq a$ incident to $b$, we remove the edge $ba'$ and create the edge $xa'$. Since $x$ is internal in every tree, it does not increase the number internal nodes. Since $b$ is on the path between $a'$ and $x$ in $T$, it keeps the connectivity of the graph. After all these operations, the resulting tree $T_A$ satisfies $in(T_A) \subseteq in(T) \setminus \{b\}$. We repeat this operation until no vertex of $B$ is internal. □

Let $D$ be a dominating set of $G$ of size $k$. We can associate with $D$ a spanning tree of $G'$ with $k+1$ internal nodes as follows. We attach every vertex in $A \cup \{y\}$ to $x$. Every vertex $b_i \in B$ is a leaf adjacent to a vertex that dominates $v_i$ in $D$. If $v_i$ has more than one neighbor in $D$, we choose the one with the smallest index. This spanning tree is called the *spanning tree associated with* $D$. See Figure 4.1 for an example.

Let $(G, k+1, D_s, D_t)$ be an instance of $\mathsf{DSR}_{\mathsf{TAR}}$. It is clear that $G'$ can be constructed in polynomial-time as well as $T_s$ and $T_t$ the spanning trees associated with $D_s$ and $D_t$. It remains to prove that $(G, k+1, D_s, D_t)$ is yes-instance of $\mathsf{DSR}_{\mathsf{TAR}}$ if and only if $(G', k', T_s, T_t)$ is a yes-instance of Spanning Tree with Many Leaves.

($\Leftarrow$) Suppose first that there exists a reconfiguration sequence of spanning trees $S' = \langle T_0 = T_s, T_1, \ldots, T_{\ell'} = T_t \rangle$, where each spanning tree has at most $k+2$ internal nodes. Since $x$ is an internal node of any spanning tree of $G'$, $D_i = in(T_i) \cap A$ has size at most $k+1$, for every $0 \leq i \leq \ell'$. Moreover, by construction of $T_s$ and $T_t$, $in(T_s) \cap A = D_s$ and $in(T_t) \cap A = D_t$. For every vertex $b$ of $B$ and every $i$, there exists a vertex of $A$ in the path from $b$ to $x$ in $T_i$. It follows that the set $D_i$ is a dominating set of $G$, for every $0 \leq i \leq \ell'$. Hence, $\langle D_s = D_0, \ldots, D_{\ell'} = D_t \rangle$ is a transformation from $D_s$ to $D_t$. It remains to prove that $|D_{i+1} \triangle D_i| \leq 1$ for every $0 \leq i < \ell'$ to guarantee the existence of a $\mathsf{TAR}(k+1)$-reconfiguration sequence between $D_s$ and $D_t$ in $G$. What we will show is actually a bit more subtle. We will show that it is not necessarily the case but that, if it is not the case for some $i$, there exists a dominating set $D'_i$ such that $D_i, D'_i, D_{i+1}$ satisfies $|D_i \triangle D'_i| \leq 1$ and $|D'_i \triangle D_{i+1}| \leq 1$ which is enough to conclude.

We consider an edge flip between two consecutive spanning trees of $S'$, let us say $T_i$ and $T_{i+1}$. Let $e_i$ (respectively $e_{i+1}$) be the edge in $E(T_i) \setminus E(T_{i+1})$ (resp. $e_{i+1} = E(T_{i+1}) \setminus E(T_i)$). We denote by $e_i \rightsquigarrow e_{i+1}$ the edge flip that transforms $T_i$ into $T_{i+1}$. Since $G'$ is bipartite and $y$ has degree one in $G'$, both $e_i$ and $e_{i+1}$ have an endpoint in $A$ and $|\{e_i, e_{i+1}\} \cap A| \leq 2$. Hence, $|D_i \triangle D_{i+1}| \leq 2$. If $e_i$ and $e_{i+1}$ are incident to a same vertex of $A$, the edge flip preserves its degree and thus $|D_i \triangle D_{i+1}| = 0$. Let us denote by $a_i$ (resp. $a_{i+1}$) the vertex in $A$ incident to $e_i$ (resp. $e_{i+1}$) in $A$. Observe that $|D_i \triangle D_{i+1}| \leq 1$ unless $a_i$ has degree two and $a_{i+1}$ is a leaf in $T_i$.

First assume that the other endpoint of $e_i$ is a vertex $b_i$ in $B$. Let $b'_i$ be the vertex $b_{j,1}$ if $b_i$ is $b_{j,0}$ or $b_{j,0}$ if $b_i$ is $b_{j,1}$, i.e., $b'_i$ corresponds to the false twin of $b_i$. We claim that there exists an internal node of $T_i$ distinct from $a_i$ incident to $b'_i$. By contradiction. The neighbor of $b'_i$ on the unique path from $b'_i$ to $x$ has to be $a_i$ (since otherwise the neighbor of $b'_i$ which is not $x$ has to have a path to $x$ which provides the desired internal node). Since $a_i$ has degree two, the two neighbors of $a_i$ are $b_i$ and $b'_i$. But this $P_3$ has to be connected to $x$. So $b_i$ or $b'_i$ are incident to another internal node $a'$ of $A$. But then $D_i \setminus \{a_i\}$ is a dominating set (since $a'$ dominates $b_i$) and then setting $D'_i = D_i \setminus \{a_i\}$, we have $|D'_i \triangle D_i| \leq 1$ and $|D_{i+1} \triangle D'_i| \leq 1$.

Now assume that $e_i = a_i x$. Let $b_i$ be the other neighbor of $a_i$ in $T_i$. Let $b'_i$ be the vertex $b_{j,1}$ is $b_i$ is $b_{j,0}$ or $b_{j,0}$ if $b_i$ is $b_{j,1}$. The neighbor $a$ of $b'_i$ on the path from $b'_i$ to $x$ is neither $a_i$ nor $a_{i+1}$. So, again $D'_i = D_i \setminus \{a_i\}$ is a dominating set and we have $|D'_i \triangle D_i| \leq 1$ and $|D_{i+1} \triangle D'_i| \leq 1$. So the conclusion follows.

($\Rightarrow$) We now prove the other direction. Suppose that there exists a TAR($k+1$)-reconfiguration sequence $S' = \langle D_0 = D_\mathsf{s}, D_1, \ldots, D_{\ell'} = D_\mathsf{t} \rangle$, from $D_\mathsf{s}$ to $D_\mathsf{t}$ in $G$. Let us prove that, for every $i$, there exists an edge flip between a tree with internal node $\{x\} \cup A(D_i)$ and a tree with internal nodes included in $\{x\} \cup A(D_{i+1})$ (for every $j$, $A(D_j)$ is the set of vertices of $A$ corresponding to the set $D_j$). The existence of a transformation from $T_\mathsf{s}$ to $T_\mathsf{t}$ follows since all the trees with the same set of internal nodes of $A$ are in the same connected component of the reconfiguration graph by Lemma 4.1 and Claim 2. Free to permute $D_i$ and $D_{i+1}$, we can assume that $D_i$ contains $D_{i+1}$. Let $u$ be the vertex of $D_i \setminus D_{i+1}$. Now, let $T_i$ be a spanning tree with internal nodes included in $D_i \cup \{x\}$. By Claim 2, we can assume that the set of internal nodes of the spanning tree is included $\{x\} \cup A(D_i)$. Now, we can flip the edges in such a way that all the edges incident to $x$ are in the spanning tree (since $x$ already is an internal node, it cannot increase the number of internal nodes). Now for every edge $a_u b$ in the spanning tree, we can replace it by an edge $a_v b$ with $v \in D_{i+1}$ since $D_{i+1}$ is a dominating set. After this transformation, the resulting tree has its internal nodes in $x \cup A(D_{i+1})$ which completes the proof.

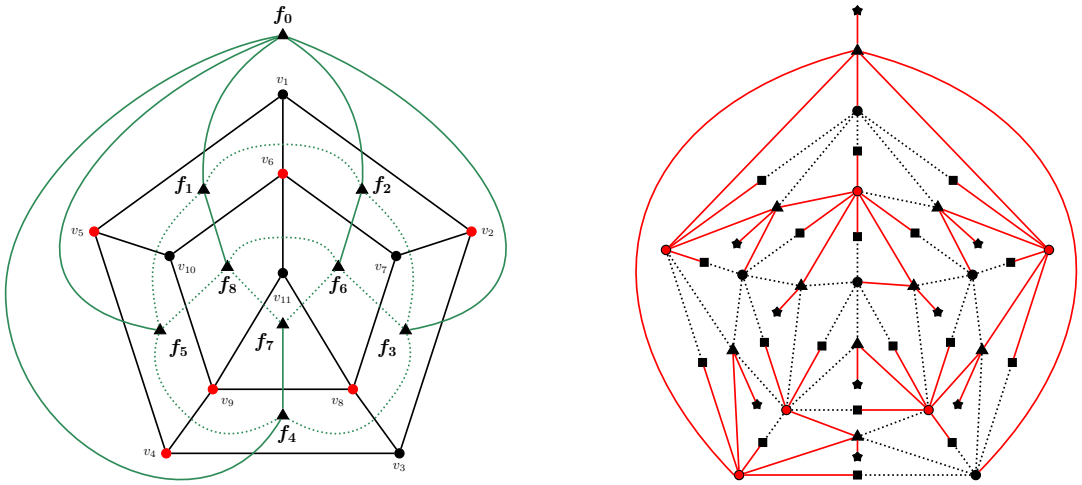
We now discuss how to adapt the proof for split graphs. First, we add an edge between any two vertices in $A$ so that $G'[A]$ is a clique. Then, observe that $G'[A \cup \{x\}]$ is a clique, and $G'[B \cup \{y\}]$ an independent set. Given two dominating sets $D_\mathsf{s}$ and $D_\mathsf{t}$, we associate with $D_\mathsf{s}$ and $D_\mathsf{t}$ the two corresponding spanning trees $T_\mathsf{s}$ and $T_\mathsf{t}$ of $G'$ in the same way as in the proof for bipartite graphs. Now, given a reconfiguration sequence between $D_\mathsf{s}$ and $D_\mathsf{t}$, the same proof as for bipartite graphs also holds here. A transformation for bipartite graphs indeed gives a transformation for split graphs. The converse direction also holds since we can assume that no vertex of $B$ is internal all along the transformation. Suppose now that there is a reconfiguration sequence $S'$ between $T_\mathsf{s}$ and $T_\mathsf{t}$. We can assume that every vertex in $B$ is a leaf in any spanning tree $T_i$ of $S'$ for the same reason as in the proof for bipartite graphs. Since $x$ must be an internal node in any spanning tree, we can suppose that no edge between two vertices in $A$ is added to $S'$. Suppose that an edge $a_i a_j$ is added. This edge must have replaced either the edge $x a_i$ or the edge $x a_j$. In any way, we cannot decrease the number of internal nodes since $x$ is still an internal node. It follows that $S'$ only touches edges between $A$ and $B$. Hence, we can conclude as in the proof for bipartite graphs. The conclusion follows. □

We now focus on planar graphs, and we show the following result:

**Theorem 4.3.** SPANNING TREE WITH MANY LEAVES *is PSPACE-complete on planar graphs.*

We give a polynomial-time reduction from the reconfiguration of minimum vertex covers under TJ, abbreviated in MVCR. Hearn and Demaine [HD05] showed that the reconfiguration of maximum independent set under TJ is PSPACE-complete. Since the complement of a maximum independent set is a minimum vertex cover, we directly get the PSPACE-completeness of MVCR. We use a reduction which is a slight adaptation of the reduction used in [MHIZ19, Theorem 4]. Let $G = (V, E)$ be a planar graph and let $(G, C_s, C_t)$ be an instance of MVCR. We can assume that $G$ is given with a planar embedding of $G$ since such an embedding can be found in polynomial time. Let $F(G)$ be the set of faces of $G$ (including the outer face). We construct the corresponding instance $(G', k, T_s, T_t)$ as follows (see Figure 4.2 for an example).

We define $G'$ from $G$ as follows. We start from $G$ and first subdivide every edge $uv \in E(G)$ by adding a new vertex $w_{uv}$. Then, for every face $f \in F(G)$, we add a new vertex $w_f$ adjacent to all the vertices of the face $f$. Finally, we attach a leaf $u_f$ to every vertex $w_f$. Note that $G'$ is a planar graph and $|V(G')| = |V(G)| + |E(G)| + 2 \cdot |F(G)|$. The vertices $w_{uv}$ for $uv \in E$ (resp. $w_f$ for $f \in F$) are *edge-vertices* (resp. *face-vertices*). The vertices $u_f$ for every $f$ are called the *leaf-vertices*. Note that, for every spanning tree $T$, all the face-vertices are internal nodes of $T$ and all the leaf-vertices are leaves of $T$. The vertices of $V(G')$ which are neither edge, face or leaf vertices are called *original vertices*. Finally, we choose arbitrarily order of $V(G)$ and $F$. It will permit us to define later a canonical spanning tree for every vertex cover.




(a) Original labeled planar graph $G$.

(b) Corresponding planar graph $G'$.

Figure 4.2 – Reduction for Theorem 4.3. The vertex cover $C$ of $G$ is depicted by the red vertices. The dual graph is the graph induced by the green edges. The spanning tree obtained by the BFS is represented by the solid edges. The face-vertices (respectively edge-vertices) of $G'$ are depicted by triangles (resp. squares). The spanning tree $T$ of $G'$ associated with the vertex cover $C$ is the tree induced by the red edges. The number of leaves of $T$ is $2(|E(G)| + 1) - |C| = 32$.

**Lemma 4.4.** *Every spanning tree of $G'$ has at most $2(|E(G)| + 1) - \tau(G)$ leaves.*

*Proof.* Let $k = 2(|E(G)| + 1) - \tau(G)$. Assume by contradiction that $G'$ has a spanning tree $T$ with at least $k + 1$ leaves. First, observe that if we require every edge-vertex to be a leaf in $T$, then $T$ has at most $k$ leaves. Indeed, as we already noticed, every face-vertex is an internal node. Then, minimizing the number of original vertices that have to be internal nodes in $T$ is equivalent to minimize the size of a vertex cover in $G$. Hence, the total number of internal

nodes in $T$ is at least $|F(G)| + \tau(G)$ and thus the number of leaves is at most $|V(G')| - |F(G)| - \tau(G) = |V(G)| + |E(G)| + |F(G)| - \tau(G) = k$ since $|F(G)| = 2 - |V(G)| + |E(G)|$ by Euler's formula.

It follows that since $T$ has at least $k+1$ leaves, then $T$ *must* contain an edge-vertex $w_{uv}$ as an internal node. So both $uw_{uv}$ and $vw_{uv}$ are in $T$. Let $T' = T \setminus \{uw_{uv}\}$. We denote by $C_u$ (respectively $C_v$) the connected component of $T'$ containing $u$ (respectively $v$). By symmetry, we can assume that $w_f \in C_u$. If we add $vw_f$ to $T'$, the resulting set of edges $T''$ induces a spanning tree of $G'$. Besides, the number of leaves in $T''$ is at least the number of leaves in $T$ since $w_{uv}$ has degree one in $T'$ and $w_f$ was already an internal node in $T$. The number of edge-vertices which are internal nodes have decreased without increasing the number of internal nodes. We repeat this process as long as there is at least one internal edge-vertex. We end up with a spanning tree in which every edge-vertex is a leaf and which contains at least $k+1$ leaves, a contradiction. □

**Lemma 4.5.** *For any minimum vertex cover $C$ of $G = (V, E)$, we can define a canonical tree with exactly $k = 2(|E(G)| + 1) - \tau(G)$ leaves which are all the edge-vertices, all the leaf-vertices and all the original vertices but the ones in $C$. Moreover, this spanning can be computed in polynomial time.*

*Proof.* We first explain how to construct $T$ from $C$. For every edge-vertex $w_{uv}$, we select in $T$ an edge between $w_{uv}$ and a vertex of $\{u, v\} \cap C$ (if both $u$ and $v$ are in $C$ we attach it to the one with the minimum label value). Such a vertex exists since $C$ is a vertex cover of $G$. For every face $f$, we select the edge $w_f u_f$.

Let $f_o$ be the outer face and let $w_o$ be the face-vertex of $f_o$. We attach every vertex of $f_o$ to $w_o$. If the resulting graph is already a spanning tree, we are done.

We say that two faces are adjacent if they share a common edge. We now consider the following graph $G''$: we create a vertex for every face of $G$ and two vertices of $G''$ are adjacent if the corresponding faces of $G$ are adjacent. In other words, $G''$ is the dual graph of $G$, without multiple edges. We then run a BFS algorithm from the vertex of $G''$ which corresponds to $f_o$. Here again, we can first label the vertices in order to process the children in the same order. We use the breadth-first search to incrementally increase the size of the connected component of $T$ which contains $w_o$ and is denoted by $S_o$. Observe that every vertex of $f_o$ belongs to $S_o$.

Now, let $f_i$ be the $i$-th face visited by the breadth-first search traversal. We assume that all the vertices that belong to faces whose index is strictly less than $i$ already belong to $S_o$. This includes the edge-vertices and the face-vertices with their respective degree-one neighbor. We now explain how to add the vertices of $f_i$ to $S_o$. Let $f_j$ be the parent of $f_i$ in the BFS traversal, for some $j < i$. By assumption, all the vertices of $f_j$ belong to $S_o$. Since $f_j$ is the parent of $f_i$, these two faces share at least one edge. Among all the edges incident to both $f_i$ and $f_j$, we pick the one which is covered in $C$ by the vertex with the smallest identifier. We denote by $u$ this vertex. We attach every vertex in $f_i \setminus S_o$ to the face-vertex $w_i$. Finally, we attach $w_i$ to $u$.

Therefore, at the end of the BFS traversal, every vertex belongs to $S_o$. Since at every step, we only attach vertices that did not belong to $S_o$ before, we do not create any cycle. It follows that the resulting graph is a spanning tree. Besides, it is clear that it can be computed in polynomial time. It remains to prove that the number of leaves is exactly $2(|E(G)| + 1) - \tau(G)$. First, recall that for every planar graph $G = (V, E)$, the number of faces of $G$ is precisely $2 - |V| + |E|$. Now, let $T$ be the spanning tree obtained by the previous algorithm. We classify the vertices of $G'$ in four different categories: the edge-vertices, the face-vertices, the leaves attached to these face-vertices, and finally the original vertices from $G$. By construction, each edge-vertex and each vertex in $V(G) \setminus C$ is a leaf in $T$. On the other hand, each face-vertex is an internal node in

$T$ since it must be adjacent to its degree-one neighbor and it must be connected to the rest of the spanning tree $T$. Finally, since $C$ is minimum and thus minimal, for every vertex $u \in C$, there is an edge $uv \in E(G)$ which is only covered by $u$. Therefore, it follows from the construction of $T$ that the corresponding edge-vertex $w_{uv}$ is attached to $u$ and thus that $u$ is an internal node.

As a result, the total number of leaves in $T$ is $|F(G)| + |E(G)| + |V(G)| - |C| = 2(|E(G)| + 1) - \tau(G)$, as desired. □

Recall that $(G, C_s, C_t)$ is an instance of MVCR. We already explained how to construct the corresponding graph $G'$ from $G$. By Lemma 4.5, we can compute in polynomial time two spanning trees $T_s$ and $T_t$ from $C_s$ and $C_t$ with $2(|E(G)| + 1) - \tau(G)$ leaves. Finally, we set $k = 2(|E(G)| + 1) - \tau(G)$). Let $(G', k, T_s, T_t)$ be the resulting instance of Spanning Tree with Many Leaves. It remains to prove that $(G, C_s, C_t)$ for MVCR is a yes-instance if and only if $(G', k, T_s, T_t)$ is a yes-instance for Spanning Tree with Many Leaves.

($\Rightarrow$) Suppose first that $(G, C_s, C_t)$ is a yes-instance and let $S = \langle C_1 = C_s, C_2, \ldots, C_\ell = C_t \rangle$ be a reconfiguration sequence between $C_s$ and $C_t$. For every vertex cover $C_i$ in the sequence, there exists a spanning tree $T_i$ of $G'$ associated with $C$ with $k$ leaves by Lemma 4.5. It is sufficient to show that we can transform two spanning trees $T_i$ and $T_{i+1}$ corresponding to two consecutive vertex covers $C_i$ and $C_{i+1}$, without increasing the number of internal nodes during the transformation. Let $u$ be the vertex of $C_i \setminus C_{i+1}$ and let $v$ be the vertex of $C_{i+1} \setminus C_i$. We first claim that $uv \in E(G)$. Suppose that $uv \notin E(G)$. Since $v \notin C_i$, all the neighbors of $v$ belong to $C_i$ by the definition of vertex cover. Therefore, $C_{i+1} \setminus \{v\}$ contains $N(v)$ and thus is a vertex cover. A contradiction with the minimality of $k$.

Since $v \notin C_i$, it follows from the construction of $T_i$ that $v$ is a leaf. Therefore, before attaching any vertex to $v$, we first need to reduce the degree of $u$. Since $C_i \triangle C_{i+1} = \{u, v\}$, we have that $N[u] \setminus \{v\} \subseteq C_i$. Recall that every vertex that belongs to $C_i$ is an internal node in $T_i$. Let $X$ be the set of edge-vertices except $w_{uv}$ attached to $u$ in $T_i$. First, we attach every vertex in $X$ to its other extremity.

Now, we root $T_i$ and $T_{i+1}$ on the leaf attached to the face-vertex of the outer face, denoted by $w_o$. If $u$ belongs to the outer face, its parent in $T_i$ and $T_{i+1}$ is $w_o$. Therefore, for every face $f$ incident to $u$ such that the corresponding face-vertex $w_f$ is attached to $u$ in $T_i$, we attach $w_f$ to the same vertex as in $T_{i+1}$, except if this vertex is $v$. Since we do not want to increase the number of internal nodes, we first need to attach $w_f$ to a vertex in $(f \cap C_i) \setminus \{u\}$. Note that this vertex exists since any vertex cover contains at least two vertices per face. It follows that now $u$ has degree two. Therefore, we can attach the edge-vertex $w_{uv}$ so that $u$ becomes a leaf and $v$ and internal node. Let $T'$ be the resulting tree. Finally, we can now attach to $v$ every face-vertex that is adjacent to it in $T_{i+1}$.

If $u$ does not belong to the outer face, we need to be more careful since we should not isolate $u$ while modifying $T_i$ into $T_{i+1}$. Recall that the parent of $u$ in $T_i$ is the face-vertex corresponding to the first face incident to $u$ visited during the BFS traversal. Since the labeling of the faces is independent of the vertex cover, $u$ has the same parent in $T_{i+1}$ as in $T_i$. The same argument also applies to $v$ and thus the parent of $v$ is the same in $T_i$ and $T_{i+1}$. Therefore, $(G', k, T_s, T_t)$ is a yes-instance, as desired.

($\Leftarrow$) For the other direction, let $S' = \langle T_1 = T_s, T_2, \ldots, T_\ell = T_t \rangle$ be a reconfiguration sequence between $T_s$ and $T_t$ such that the number of leaves is at least $k$ at any time. Recall that the number of leaves in $T_s$ and $T_t$ is maximal. Hence, each spanning tree in $S'$ has exactly $k$ leaves.

We claim that every edge-vertex is a leaf in any spanning tree of $S'$. First, recall that this statement holds for $T_s$ and $T_t$. Let $T_i$ be the first spanning tree in $S'$ which contains an edge-vertex as an internal node. Since every edge-vertex is a leaf in $T_{i-1}$ and $|E(T_{i-1}) \triangle E(T_i)| = 2$,

exactly one edge-vertex in $T_i$ is an internal node. Let $w_{uv}$ be this vertex. We assume without loss of generality that $uw_{uv} \in E(T_{i-1})$ and thus the edge in $T_i \setminus T_{i-1}$ is $vw_{uv}$. We consider the (only) edge in $T_{i-1} \setminus T_i$, denoted by $e$. $T_{i-1}$ contains three kinds of edges: between an original vertex and an edge-vertex, between an original vertex and a face-vertex, or between a leaf-vertex and a face-vertex. Since all the vertices of the form $u_f$ or $w_{xy}$ have degree one in $T_{i-1}$, $e$ is necessarily of the form $xw_f$, i.e., an edge linking a face-vertex and an original vertex. Recall that $w_f$ is an internal node in any spanning tree of $G'$. Since $w_{uv}$ is a leaf in $T_{i-1}$ but not in $T_i$, the degree of $x$ in $T_{i-1}$ *must* be two, otherwise we would increase the total number of internal nodes. Note that $uv \in E(G)$ since $w_{uv}$ is an edge-vertex of $G'$ and thus $G'$ contains a face-vertex $w_{f'}$ adjacent to both $u$ and $v$. Let $T'_i$ be the forest obtained from $T_i$ by removing the edge $vw_{uv}$ and observe that $T'_i = T_{i-1} \setminus \{xw_f\}$. We denote by $C_u$ (respectively $C_v$) the connected component of $u$ (respectively $v$) in $T'_i$. We apply the same argument as in the proof of Lemma 4.5. The node $w_{f'}$ has a neighbor either in $C_u$ or in $C_v$ (which might be $u$ or $v$) but not in both $C_u$ and $C_v$ otherwise $T_i$ would contain a cycle. We assume without loss of generality that $w_{f'} \in C_u$. Then, observe that if we add the edge $vw_{f'}$ to $T'_i$, we get a spanning tree of $G'$ such that $|T_i \triangle T'_{i+1}| = 2$ but with $k+1$ leaves, a contradiction.

It follows that for every $T_i \in S'$, $1 \leq i \leq \ell$, the number of leaves in $T_i$ is exactly $k = 2(|E(G)|+1) - \tau(G)$, and every edge-vertex of $G'$ is a leaf in $T_i$. From $T_i$, we can deduce a vertex cover $C_i$ of $G$: the vertex that covers the edge $uv \in E(G)$ in $C_i$ corresponds to the neighbor of the edge-vertex $w_{uv}$ in $T_i$. In particular, the corresponding vertex covers of $T_\mathsf{s}$ and $T_\mathsf{t}$ are $C_\mathsf{s}$ and $C_\mathsf{t}$, respectively.

Then, from $S'$, we can deduce a sequence $S'' = \langle C_1 = C_\mathsf{s}, C_2, \ldots, C_{\ell'} = C_\mathsf{t} \rangle$ of vertex covers of $G$. Note that the length of $S''$ is not necessarily the same as the length of $S'$, i.e., it is possible that two adjacent spanning trees $T_i$ and $T_{i+1}$ in $S'$ give the same corresponding vertex cover of $G$. It remains to prove that $|C_i| = \tau(G)$ for every $1 \leq i \leq \ell'$, and $|C_i \triangle C_{i+1}| = 2$ for any two adjacent vertex covers of $S''$, i.e., $S''$ is a TJ-sequence of minimum vertex covers of $G$. Since $|C_1| = |C_{\ell'}| = \tau(G)$, it is sufficient to prove that $|C_i \triangle C_{i+1}| = 2$, for every $1 \leq i < \ell'$. Let $C_i$ and $C_{i+1}$ be two consecutive vertex covers in $S''$. Let $i'$ be the maximal index such that the vertex cover induced by the spanning tree $T_{i'} \in S'$ is $C_i$. Due to the maximality of $i'$, the vertex cover induced by $T_{i'+1}$ corresponds to $C_{i+1}$, since it cannot be $C_i$. Therefore, the edge in $T_{i'} \setminus T_{i'+1}$ is between an edge-vertex and an original vertex. We denote by $uw_{uv} \in E(G')$ this edge. Then, since $w_{uv}$ has degree one in $T_{i'}$, the edge in $T_{i'+1} \setminus T_{i'}$ must be $vw_{uv}$. Therefore, $C_{i+1} = (C_i \setminus \{u\}) \cup \{v\}$ and thus $|C_i \triangle C_{i+1}| \leq 2$ holds, for every $1 \leq i < \ell'$ as desired. Hence, $(G', k, T_\mathsf{s}, T_\mathsf{t})$ is a yes-instance. This concludes the proof of Theorem 4.3.

### Polynomial-time results

We now prove that Spanning Tree with Many Leaves is polynomial-time solvable if $k \geq n-2$ for any graph $G$ with $n$ vertices. We will then use this result to show that the problem is polynomial-time solvable as well if the input graph is a cograph.

**Theorem 4.6.** *Let $G$ be a graph and $T_\mathsf{s}$ or $T_\mathsf{t}$ be two spanning trees with at most two internal nodes. Then we can check in polynomial time if one can transform the other via a sequence of spanning trees with at most two internal nodes.*

*Proof.* We first consider the case where either $T_\mathsf{s}$ or $T_\mathsf{t}$ has one internal node, but not both. We assume without loss of generality that $in(T_\mathsf{s}) = \{u\}$. If $u \in in(T_\mathsf{t})$, we just have to attach every leaf in $T_\mathsf{t}$ to $u$, one by one. It follows that $in(T_\mathsf{s}) \cap in(T_\mathsf{t}) = \emptyset$ and thus $u$ has degree one in $T_\mathsf{t}$. Hence, if we want to reconfigure $T_\mathsf{s}$ into $T_\mathsf{t}$, we *must* remove all but one of the edges incident

to $u$ and thus we must create a new internal node. Therefore, it is sufficient to consider the last following case: $|in(T_s)| = |in(T_t)| = 2$.

First, observe that if $in(T_s) = in(T_t)$, then $(G, k, T_s, T_t)$ is a yes-instance. Indeed, we just have to change the parent of a node, and this can be done without increasing the number of internal nodes. Hence, in the remaining of the proof, we only consider the case $in(T_s) \neq in(T_t)$.

A vertex $u$ is a *pivot* vertex of $G$ if $\deg(u) \geq n-2$ in $G$ ($\deg(u)$ being the size of the neighborhood of $u$, $u$ not included). A spanning tree $T$ of $G$ is *frozen* if all the spanning trees in its component of the reconfiguration graph have the same internal nodes.

*Claim* 1. Let $T$ be a spanning tree of $G$. If $in(T)$ does not contain a pivot vertex, then $T$ is frozen.

*Proof.* By contradiction. Assume that $in(T)$ does not contain a pivot vertex and thus each vertex in $in(T)$ has degree at most $n-3$. Then, we want to prove that we cannot modify $in(T)$. Let $in(T) = \{u, v\}$, and note that $uv \in E(T)$. Note also that since $\deg(u) \leq n-3$ and $\deg(v) \leq n-3$, both $u$ and $v$ have degree at least three in $T$. Since $k = 2$ and $|in(T)| = 2$, we first need to lower the degree of $u$ or $v$ to one or two, without creating a new internal node. Suppose without loss of generality that we want to lower the degree of $u$, the other case follows by symmetry. First, observe that we cannot remove the edge $uv \in E(T)$ with an edge flip because it would create a new internal node, as the degree of both $u$ and $v$ is at least three. Recall that $\sum_{u \in V(T)} \deg_T(u) = 2n-2$. Since $T$ has $n-2$ leaves, $\deg_T(u) + \deg_T(v) = n$. Hence, if we want $u$ to have degree two, $v$ must have degree $n-2$, a contradiction. ◊

*Claim* 2. Let $u$ be a pivot vertex. All the trees containing $u$ as an internal node are in the same connected component of the reconfiguration graph.

*Proof.* Let $T$ and $T'$ be two trees such that $u \in in(T) \cap in(T')$. If the other internal nodes (if they exist) are the same, then the conclusion follows from Lemma 4.1. So we can assume that $in(T) = \{u, v\}$ and $in(T') = \{u, w\}$ with $v \neq w$. Since $\deg(u) \geq n-2$, there exists a spanning tree $T_2$ with internal nodes $\{u, v\}$ such that $\deg(u) = n-2$ and $\deg(v) = 2$ and $uv \in T_2$. By Lemma 4.1, this spanning tree is in the component of $T$. Let $z$ be the neighbor of $v$ distinct from $u$. Now remove the edge $vz$ and create $wz$ or $uz$ (one of them must exist since $in(T_2) = \{u, w\}$. The internal nodes of the resulting tree is in $\{u, w\}$ and then the conclusion follows by Lemma 4.1. ◊

We say that a spanning tree $T$ *contains a pivot vertex* if $in(T)$ contains a pivot vertex. By Claim 2, if $T_s$ and $T_t$ contains a common pivot vertex, then the answer is positive. (Note that the existence of a pivot vertex can be checked in polynomial time). If $T_s$ or $T_t$ does not contain any pivot vertex, then the answer is negative by Claim 1 (except if the set of internal nodes are the same by Lemma 4.1). So we restrict our attention to the case where they contain a pivot vertex which is different.

Let $in(T_s) = \{u, v\}$ and $in(T_t) = \{x, y\}$ where $u$ and $x$ are pivot vertices (note that we can possibly have $v = y$). If $u$ (or $x$) is a universal vertex, we can modify $in(T_s)$ (or $in(T_t)$) into a spanning tree $T$ with $in(T) = \{u\}$ (resp. $\{x\}$). Claim 2 ensures that both $T_s$ and a spanning containing $u$ and $x$ as internal nodes are in the same component. And this latter spanning tree is in the component of $T_t$ by Claim 2. So we can assume that none of the four internal nodes is universal.

If $in(T_s)$ or $in(T_t)$ contains two pivot vertices, without loss of generality $in(T_s)$, then $\{u, x\}$ or $\{v, x\}$ dominates $G$. So there exists a spanning tree $T$ with $in(T) = \{u, x\}$ or $\{v, x\}$. Up to symmetry, let us say $\{u, x\}$. By Claim 2, $T$ is both in the connected component of $T_s$ and $T_t$.

So $in(T_s)$ and $in(T_t)$ contain exactly one pivot vertex; respectively $u$ and $x$. Observe that, if we want to reconfigure $T_s$ into $T_t$, we must remove $u$ from the spanning tree at some point

since it does not belong to $in(T_t)$. But then, just before disappearing, the second internal node has to have degree $n-2$ in the spanning tree, and then has to be a pivot vertex. So the previous paragraph ensures that $T_s$ can be transformed into $T_t$ if and only if there exists a spanning tree in the component of $T_s$ with two pivot vertices. It is the case if and only if there exists a second pivot vertex $w$ such that $\{u, w\}$ dominates the graph, which can be checked in polynomial time. □

We now have all the ingredients to prove the following theorem:

**Theorem 4.7.** SPANNING TREE WITH MANY LEAVES *can be decided in polynomial time on cographs.*

*Proof.* Let $G = (V, E)$ be a cograph and let $(G, k, T_s, T_t)$ be an instance of SPANNING TREE WITH MANY LEAVES. We denote by $n$ the number of vertices of $G$. First, since $T_s$ and $T_t$ are two spanning trees of $G$, $G$ must be connected. Hence, $G$ has been obtained from the join of two cographs, let us say $A$ and $B$. Recall that maximizing the number of leaves of a spanning tree is equivalent to minimizing the number of internal nodes. Hence, in the remaining of this proof, we denote by $k$ the threshold on the maximum number of internal nodes.

If $k = 1$, any spanning tree of $G$ is a star and thus contains exactly one internal node. Therefore, two spanning trees of $G$ are reconfigurable if and only if $G$ contains at most three vertices or the same internal node by Lemma 4.1. Hence, we can safely assume that $k \geq 2$. Since G is the join of two cographs, $G$ can be partitioned into two subsets $A$ and $B$ such that $G[A]$ and $G[B]$ are two cographs, and we have all possible edges between $A$ and $B$. Let $T$ be a spanning tree of $G$. We say that $T$ is an *$A$-tree* (resp. *$B$-tree*) if $in(T) \subseteq A$ (resp. $in(T) \subseteq B$. Otherwise, we say that $T$ is an *$(A, B)$-tree*.

If $k = 2$, Theorem 4.6 ensures that the problem can be decided in polynomial time. So from now on, we can assume that $k \geq 3$. In this case, we claim that $(G, k, T_s, T_t)$ is a yes-instance.

Let us first prove by induction on the size of $G$ that there exists a transformation from any tree $T$ with at most $k$ internal nodes to a tree $T'$ with at most $k-1$ of them such that all along the transformation there exists a vertex $x$ which is always an internal node. We moreover prove that this transformation can be found in polynomial time. If $T$ has at most two internal nodes, the conclusion follows. So we can assume that $T$ has exactly $k$ internal nodes.

If $T$ is an $(A, B)$-tree, we can reach $T'$ as follows. Let $a \in in(T) \cap A$ and $b \in in(T) \cap B$ such that $ab \in T$ (such an edge must exist). Using edge flips, we make $a$ adjacent to any vertex in $B$ and $b$ adjacent to every vertex of $A$ (which is possible since $A - B$ is a join). After all these modifications, the resulting tree has exactly two internal nodes.

So we can assume that $T$ is an $A$-tree or a $B$-tree, without loss of generality an $A$-tree. Thus every vertex in $B$ is a leaf and then the restriction $T_A$ of $T$ to $G[A]$ also is a spanning tree of $G[A]$. By induction, since $G[A]$ is a connected cograph, we can find in polynomial time a transformation of $T_A$ into a spanning tree $T'_A$ in such a way that $x$ is an internal node all along the transformation (and this transformation can be found in polynomial time). This transformation can be adapted for $T$ by first connecting all the vertices of $B$ to $x$ using edge flips and then transforming the edges of $G[A] \cap T$ into $T'_A$. All along the transformation $x$ is an internal node and at any step the set of internal nodes are precisely the ones of the tree restricted to $G[A]$.

So we can assume that $T_s$ and $T_t$ have at most $k-1$ internal nodes. Let us define a *canonical* spanning tree $T_c$ with two internal nodes and show that both $T_s$ and $T_t$ can be reconfigured into $T_c$. We define $in(T_c)$ as follows: we pick a vertex $a \in A$, and a vertex $b \in B$ arbitrarily. We only explain without loss of generality how to reconfigure $T_s$ into $T_c$.

Since $|in(T_s)| < k$, we can trivially modify it into $in(T_c)$. We only show the statement for $|in(T_s)| = 2$, and $k = 3$. The proof is similar for other values of $k$. Let $in(T_s) = \{u, v\}$. Suppose first that $T_s$ is an $A$-tree or a $B$-tree. We will consider the case where $T_s$ is an $(A, B)$-tree later. We assume without loss of generality that $T_s$ is an $A$-tree. We first add $b$ to $in(T_c)$, i.e., we attach each vertex in $A$ to $b$. Observe that we can now remove a vertex in $\{u, v\}$ since all the vertices in $A$ are covered by $b$ and only one vertex is needed to cover $B$. It follows that $T_s$ is now an $(A, B)$-tree with two internal nodes. Hence, we can now attach each leaf in $B$ to $a$ (it creates a third internal node but this is allowed since $k \geq 3$). It remains to remove the vertex in $(in(T_s) \cap A) \setminus \{a\}$. This concludes the proof of Theorem 4.7. □

### 4.1.3 Spanning trees with few leaves

In this section, we consider the reconfiguration of spanning trees with at most $k$ leaves. More precisely, we are interested in the following problem:

SPANNING TREE WITH AT MOST $k$ LEAVES

**Instance:** A graph $G$, an integer $k \geq$, two spanning trees $T_s$ and $T_t$ of $G$ with at most $k$ leaves.

**Question:** yes if and only if there exists a transformation via edge flips between $T_s$ and $T_t$ such that all the intermediate trees have at most $k$ leaves.

We will prove the following theorem:

**Theorem 4.8.** SPANNING TREE WITH AT MOST $k$ LEAVES *is PSPACE-complete for every* $k \geq 3$.

In order to prove Theorem 4.8, we first prove it for $k = 3$. We will then explain how we can modify this proof in order to get the hardness for any $k \geq 3$.

**Theorem 4.9.** SPANNING TREE WITH AT MOST 3 LEAVES *is PSPACE-complete.*

In order to prove Theorem 4.9, we will provide a reduction from the reconfiguration of minimum vertex covers under TAR. Wrochna [Wro18] showed that MAXIMUM INDEPENDENT SET RECONFIGURATION under TJ is PSPACE-complete even when restricted to bounded bandwidth graphs. However, recall that the complement of an independent set is a vertex cover. Besides, Kamiński et al. [KMM12] observed that the TJ model and TAR model are equivalent when the threshold is the minimum value of a vertex cover plus one. Hence, MINIMUM VERTEX COVER RECONFIGURATION is PSPACE-complete under the TAR model.

The idea of the proof of Theorem 4.9 is to adapt a reduction from VERTEX COVER to HAMILTONIAN PATH. Let $(G = (V, E), k)$ be an instance of VERTEX COVER. This reduction creates a graph $H(G)$ which contains a Hamiltonian path if and only if $G$ admits a vertex cover of size $k$. In particular, we will show that there is a "canonical way" to define a vertex cover from any Hamiltonian path. The reduction is provided in the next section together with some properties of the spanning trees with at most three leaves in $H(G)$. In order to adapt the proof in the reconfiguration setting, we need to prove that the proof is "robust" with respect to several meanings of the word. First, we need to show that, if we consider a spanning tree with at most three leaves in $H(G)$ then there is a "canonical" vertex cover of size at most $k + 1$ associated with it. Proving that this vertex cover always has size at most $k + 1$ is the first technical part of the proof. Then, for any edge flip between two spanning trees with at most three leaves, there is a corresponding "transformation" between the canonical vertex covers associated with them. We

need to prove that for any two adjacent spanning trees in $H(G)$, their canonical vertex covers are either the same or are incident in the TAR model (in $G$).

Finally, we need to prove that it is possible to transform a Hamiltonian path $P_1$ (associated with a vertex cover $X$) into a Hamiltonian path $P_2$ associated with a vertex cover $Y$ via spanning trees with at most three leaves if and only if $X$ can be transformed into $Y$ in the TAR model.

**The reduction and construction of $H(G)$**

The reduction is a classical reduction (see Theorem 3.4 of [GJ79] for a reference) from the optimization version of VERTEX COVER to the optimization version of HAMILTONIAN PATH. Let $G$ be a graph and $k$ be a non-negative integer. We provide a reduction from VERTEX COVER of size at most $k$ to HAMILTONIAN PATH. Let us construct a graph $H(G)$ (abbreviated into $H$ when no confusion is possible) as follows.

For each edge $e = uv$ of $G$, we create the following *edge-gadget* $\mathcal{G}_e$ represented in Figure 4.3. The edge-gadget $\mathcal{G}_e$ has four *special vertices* denoted by $x_u^e, x_v^e, y_u^e, y_v^e$. The vertices $x_u^e$ and $x_v^e$ are called the *entering vertices* and $y_u^e$ and $y_v^e$ the *exit vertices*. The gadget contains 8 additional vertices denoted by $r_1^e, \ldots, r_8^e$. When $e$ is clear from context, we will omit the superscript. The graph induced by these twelve vertices is represented in Figure 4.3. The vertices $r_1^e, \ldots, r_8^e$ are *local vertices* and their neighborhood will be included in the gadget. The only vertices connected to the rest of the graphs are the special vertices.

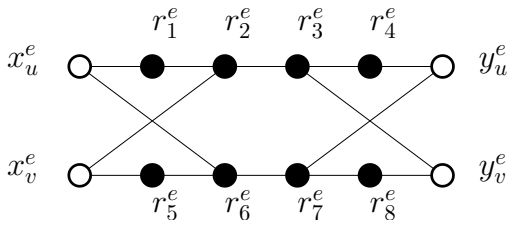



Figure 4.3 – Edge-gadget corresponding to the edge $e = uv$. The white vertices are the only ones connected to the outside.

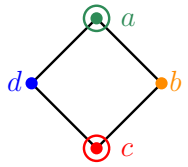
We add an independent set $Z = \{z_1, \ldots, z_{k+1}\}$ of $k+1$ new vertices to $V(H)$. And we finally add to $V(H)$ two more vertices $s_1, s_2$ in such a way that $z_1$ (resp. $z_{k+1}$) is the only neighbor of $s_1$ (resp. $s_2$) in $H(G)$. Since $s_1$ and $s_2$ have degree one in $H(G)$, $s_1$ and $s_2$ are leaves in any spanning tree of $H(G)$. In particular, the two endpoints of any Hamiltonian path of $H(G)$ are necessarily $s_1$ and $s_2$.

Let us now complete the description of $H(G)$ by explaining how the special vertices are connected to the other vertices of $H(G)$. Let $u \in V(G)$. Let $E' = e_1, \ldots, e_\ell$ be the set of edges incident to $u$ in an arbitrary order. We connect $x_u^{e_1}$ and $y_u^{e_\ell}$ to all the vertices of $Z$. For every $1 \leq i \leq \ell - 1$, we connect $y_u^{e_i}$ to $x_u^{e_{i+1}}$. The edges $y_u^{e_i} x_u^{e_{i+1}}$ are called the *special edges of* $u$. The *special edges* of $H(G)$ are the union of the special edges for every $u \in V(G)$ plus the edges incident to $Z$ but $s_1 z_1$ and $s_2 z_{k+1}$. This completes the construction of $H(G)$ (see Figure 4.4).

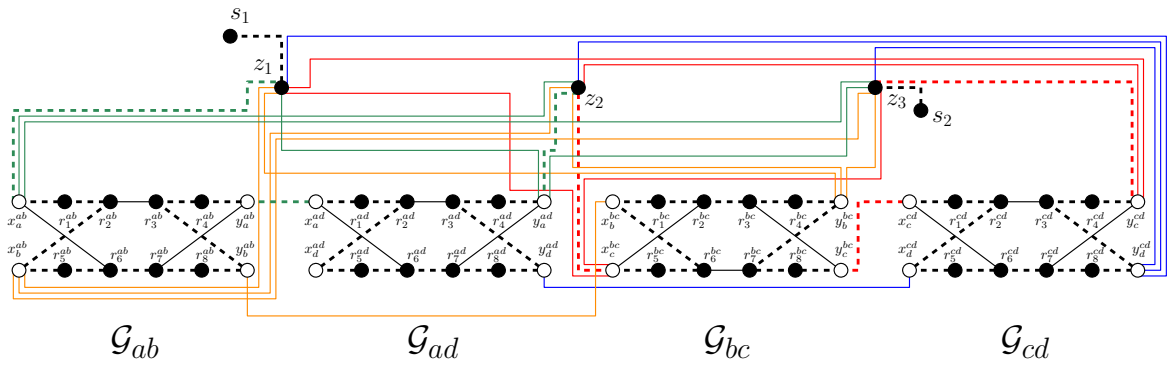
**Basic properties of $H(G)$**

**Remark 4.10.** *If $T$ is a spanning tree of $H(G)$ with at most $\ell$ leaves, then at most $\ell - 2$ of them are in $V(H) \setminus \{s_1, s_2\}$.*

For a spanning tree $T$, we say that an edge-gadget *contains a leaf* if one of the twelve vertices of the edge-gadget is a leaf of $T$. If the spanning tree is a Hamiltonian path, Remark 4.10 ensures that no edge-gadget contains a leaf. Besides, at most one edge-gadget contains a leaf if $T$ is a




(a) Original instance $(G, k)$ of Minimum Vertex Cover with a vertex cover $\{a, c\}$.




(b) Graph $H(G)$ obtained from the reduction. The ordering for the vertices of the vertex cover $\{a, c\}$ of $G$ is the lexicographic order, as well as the ordering of the edges incident to each vertex. The corresponding Hamiltonian path is depicted by the thick dashed edges.

Figure 4.4 – Illustration of the reduction of Theorem 4.9.

spanning tree with at most three leaves. An edge-gadget *contains a branching node* of $T$ if one of the twelve vertices of the gadget is a vertex of degree at least three. Any spanning tree with at most three leaves indeed contains at most one branching node.

Let $T$ be a spanning tree of $H(G)$. An edge-gadget is *irregular* if at least one of its twelve vertices is not of degree two in $T$, i.e., if it contains a branching node or a leaf. An edge-gadget is *regular* if it is not irregular. By abuse of notation we say that $e \in E(G)$ is regular (resp. irregular) if the edge-gadget of $e$ is regular (resp. irregular). A vertex $u$ is *regular* if every edge incident to $u$ is regular. The vertex $u$ is *irregular* otherwise.

Let $S$ be a subset of vertices of $H(G)$. We denote by $\delta_T(S)$ the set of edges with exactly one endpoint in $S$. When there is no ambiguity, we omit the subscript $T$. Moreover, if $S$ is the singleton $\{u\}$, we write $\delta_T(u)$ for $\delta_T(\{u\})$. Given an edge $e$ of $G$ and a spanning tree $T$ of of $H(G)$, $\delta_T(e)$ denotes the set of edges of $T$ with exactly one endpoint in the edge-gadget $\mathcal{G}_e$ of $e$. The restriction $T(\mathcal{G}_e)$ of a spanning tree $T$ *around an edge-gadget* $\mathcal{G}_e$ is the set of edges with both endpoints in $\mathcal{G}_e$ plus the edges of $\delta_T(\mathcal{G}_e)$ (which are considered as "semi edges" with one endpoint in $\mathcal{G}_e$).

**Lemma 4.11.** *Let $T$ be a spanning tree of $H$ and $\mathcal{G}$ be a regular edge-gadget. Then the tree $T$ around the edge-gadget $\mathcal{G}$ is one of the two graphs represented in Figure 4.5. Note that the graph of Figure 4.5(b) has to be considered up to symmetry between $u$ and $v$.*

In order to prove Lemma 4.11, we will need the following lemma that will be useful all along our proof:

**Lemma 4.12.** *Let $R$ be the graph restricted to an edge-gadget. There is no Hamiltonian path from one vertex of $\{x_u^e, y_u^e\}$ to one vertex of $\{x_v^e, y_v^e\}$ in $R$.*

*Proof.* By contradiction. Let us denote by $w_1, w_2$ the two endpoints of a Hamiltonian path $P$. If $w_1, w_2$ are the two entering (resp. exit) vertices, then both exit (resp. entering) vertices must

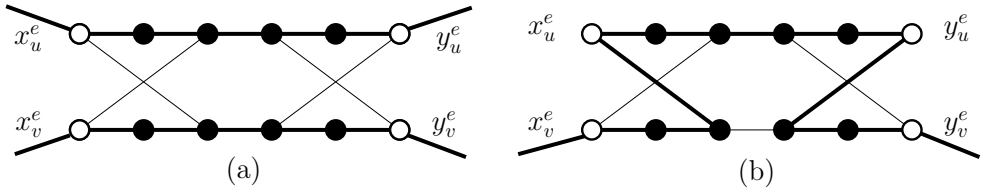



Figure 4.5 – The two possible subgraphs around a regular edge-gadget $\mathcal{G}$. Thick edges are edges in the tree. Edges with one endpoint in the gadget are edges of $\delta(\mathcal{G})$.

have degree two in $P$. If both exit vertices have degree two, then one of $r_3r_4$ or $r_7r_8$ do not exist in $P$ since otherwise $P$ admits a cycle. And then $r_4$ or $r_8$ are leaves of $P$, a contradiction since $P$ is a Hamiltonian path in $R$. Similarly, the same holds if both entering vertices have degree two.

So, by symmetry, we can assume that $w_1 = x_u^e$ and $w_2 = y_v^e$. Since $x_v^e$ and $y_u^e$ have degree two and all the local vertices have degree two in $P$, the subpaths $x_u^e r_1 r_2 x_v^e r_5 r_6$ and $r_3 r_4 y_u^e r_7 r_8 y_v^e$ are in $P$. It is impossible to connect these two paths into a Hamiltonian path in $R$, a contradiction. □

Let us now prove Lemma 4.11:

*Proof.* Remark that since all the vertices of the edge-gadget $\mathcal{G}_e$ have degree two in $T$, the number of edges with one endpoint in the gadget is even (the subgraph of $T$ induced by the vertices of $\mathcal{G}_e$ being a union of paths). Moreover, since $r_1, r_4, r_5, r_8$ are not leaves of $T$ and have degree two in $H(G)$, both edges incident to them are in $T$. So the number of edges of $\delta_T(\mathcal{G}_e)$ incident to each of $x_u^e, y_u^e, x_v^e, y_v^e$ is either zero or one. In particular, $|\delta_T(\mathcal{G}_e)| \leq 4$.

If $|\delta_T(\mathcal{G}_e)| = 2$, then, since the edge-gadget is regular, the restriction of $T$ to the edge-gadget is a Hamiltonian path $P$. By Lemma 4.12, the endpoints of $P$ cannot be one vertex of $\{x_u, y_u\}$ and one vertex of $\{x_v, y_v\}$. So, by symmetry, we can assume that the endpoints of $P$ are $x_u$ are $y_u$. Since, $r_1, x_v, r_5, r_8, y_v, r_4$ have degree two in the subgraph induced by the edge-gadget, it forces all the edges of the gadget but $x_u r_6, y_u r_7, r_6 r_7$ and $r_2 r_3$ to be in $P$. Since $P$ is a Hamiltonian path from $x_u$ to $y_u$, $r_5 r_6 \in E(T)$ which gives the graph of Figure 4.5(b) (up to symmetry.).

So we can now assume that $|\delta_T(\mathcal{G}_e)| = 4$. Since at most one edge of $\delta_T(\mathcal{G}_e)$ is incident to each special vertex, all these vertices have degree one in the subtree induced by the vertices of $\mathcal{G}_e$. So, the subforest induced on the gadget must be a union of two paths. Since $r_1, r_4, r_5$ and $r_8$ have degree two, the only way to complete this set of edges into a Hamiltonian path provides the graph of Figure 4.5(a), which completes the proof. □

If $P$ is a Hamiltonian path of $H$, then Remark 4.10 ensures that all its edge-gadgets are regular. And then, by Lemma 4.11, for every edge-gadget $\mathcal{G}$, the graph around $\mathcal{G}$ is one of the two graphs of Figure 4.5.

#### Vertex Cover and Hamiltonian Path

Suppose that $G$ has a vertex cover $X = \{v_1, \ldots, v_k\}$ of size $k$. We claim that the following set of edges $F$ induces a Hamiltonian path in $H(G)$. We start with $F = \emptyset$. For every $i \leq k$, we add to $F$ the edge between $z_i$ and the entering vertex of the first edge of $v_i$ and the edge between $z_{i+1}$ an the exit vertex of the last edge of $v_i$. For every $v_i \in X$, all the special edges of $v_i$ are added to $F$. The edges $s_1 z_1$ and $s_2 z_{k+1}$ are also in $F$. We claim that, for each edge-gadget $\mathcal{G}$ corresponding to the edge $uv$, either two edges or four edges of $F$ have exactly one endpoint in $F$. Indeed, if none of them are selected, then by construction of $F$, neither $u$ nor $v$ are in $X$, a contradiction since $X$ is a vertex cover of $G$. Moreover, by construction of $F$, $x_v^e$ is an endpoint of an edge of $F$ if and only if $y_v^e$ also is. Note moreover that: (i) no local vertex of the edge-gadget is incident to an edge

of $F$, (ii) special vertices are incident to at most one, and (iii) vertices of $Z$ are incident to two of them. So in order to complete $F$ into a Hamiltonian path, we add the edges of Figure 4.5(a) or (b) depending if two or four edges of the current set $F$ are incident to a vertex of the edge-gadget (two when one endpoint is in $X$, four is both of them are in $X$). The set $F$ induces a Hamiltonian path, as proved in [GJ79]. This Hamiltonian path is called *a Hamiltonian path associated with the vertex cover* $X$. Note that there might be several Hamiltonian paths associated with the same vertex cover since the path depends on the "ordering" of $X$. Indeed we have to choose which entering vertex is attached to $z_1, z_2, \dots, z_k$ which gives a natural ordering of $X$.

Conversely, let us explain why we can associate with every Hamiltonian path $P$ a vertex cover. Let $\mathcal{G}$ be an edge-gadget. The graph $H[\mathcal{G}]$ is the subgraph induced by the twelve vertices of the edge-gadget. Note that the subgraph of $P$ induced by $\mathcal{G}$ is not the graph around $\mathcal{G}$, which contains the semi-edges leaving $\mathcal{G}$.

**Lemma 4.13.** *Let $G$ be a graph, $T$ be a spanning tree of $H(G)$, and $u$ be a regular vertex of $T$. If there exists an edge $e \in E(G)$ with endpoint $u$ such that $x_u^e$ or $y_u^e$ has degree one in the subgraph of $T$ induced by the vertices of $H[\mathcal{G}_e]$, then, for every edge $e'$ with endpoint $u$, $x_u^{e'}$ and $y_u^{e'}$ have degree one in the subgraph of $T$ induced by the vertices of $H[\mathcal{G}_{e'}]$.*
*In particular, there is an edge of $T$ between $Z$ and the first entering vertex of $u$ and an edge between $Z$ and the last exit vertex of $u$.*

*Proof.* By symmetry, $x_u^e$ has degree one in the subgraph of $T$ induced by the vertices of $H[\mathcal{G}_e]$. Since the graph around the gadget is one of the two graphs of Figure 4.5 (which corresponds to the only possible restrictions of $T$ around a regular edge-gadget), for both $x_u^e$ and $y_u^e$, an edge of $T$ is leaving the gadget. If $e$ is the first (resp. last) edge of $u$, then there is an an edge linking $x_u^e$ (resp. $y_u^e$) to $Z$. Otherwise, let us denote by $e'$ (resp. $e''$) the edge before (resp. after) $e$ in the order of $u$. The only edge incident to $x_u^e$ (resp. $y_u^e$) in $\delta_T(\mathcal{G}_{e'})$ is $x_u^e y_u^{e'}$ (resp. $y_u^e x_u^{e''}$). Since $u$ is regular, both $x_u^e y_u^{e'}$ and $y_u^e x_u^{e''}$ are in $T$. And then we can repeat the same argument on $e'$ (resp. $e''$) until we reach the first (resp. last) edge of $u$. □

If, for a regular vertex $u$ and an edge $e = uv$, $x_u^e$ or $y_u^e$ have degree one in $H[\mathcal{G}_e]$, then there is a path between two vertices of $Z$ passing through all the special vertices $x_u^{e'}$ and $y_u^{e'}$ for every $e'$ incident to $u$ and all the vertices on this path have degree two. Note that the union of all such vertices forms a vertex cover of $G$.

### Defining a vertex cover

Let $T$ be a spanning tree with at most three leaves. By Lemma 4.11, for every edge-gadget $\mathcal{G}_e$, if $T(\mathcal{G}_e)$ is not one of the two graphs of Figure 4.5, $\mathcal{G}_e$ contains a branching node or a leaf. So Remark 4.10 implies:

**Remark 4.14.** *There are at most two irregular edge-gadgets. Thus there are at most four irregular vertices.*

Indeed, if $T$ has two leaves, all the edge-gadgets are regular. If $T$ has three leaves, the third leaf must be in an edge-gadget, creating an irregular edge-gadget. And this leaf might create a new branching node which might be in another edge-gadget than the one of the third leaf. So the number of irregular edge-gadget is at most two, and thus the number of irregular vertices is at most four (if the edges corresponding to these two edge-gadgets have pairwise distinct endpoints).

Let $T$ be a spanning tree of $H(G)$ with at most three leaves. A vertex $v$ is *good* if there exists an edge $e = vw$ for $w \in V(G)$ such that $x_v^e$ or $y_v^e$ has degree one in the subtree of $T$ induced

by the twelve vertices of the edge-gadget of $e$. In other words, if we simply look at the edges of $T$ with both endpoints in $\mathcal{G}_e$, $x_v^e$ or $y_v^e$ has degree one (or said again differently, $x_v^e$ or $y_v^e$ are adjacent to exactly one local vertex). Let us denote by $S(T)$ the set of good vertices.

**Lemma 4.15.** *Let $T$ be a spanning tree with at most three leaves of $H(G)$ and $e = uv$ be an edge of $G$. At least one special vertex of the edge-gadget $\mathcal{G}_e$ has degree one in the subgraph of $T$ induced by the vertices of $\mathcal{G}_e$.*
*In particular, $S(T)$ is a vertex cover.*

*Proof.* Let $R$ be the subgraph of $H(G)$ induced by the vertices of $\mathcal{G}_e$. Let $T'$ be the restriction of $T$ to $R$. Assume by contradiction that none of the four special vertices have degree one in $T'$. Since special vertices $Y$ have degree two in $R$, the special vertices have degree zero or degree two in $T'$. We claim that the number of special vertices of degree zero is at most one. Indeed, if $x_u^e$ (resp. $y_u^e, x_v^e, y_v^e$) has degree zero in $T'$, then $r_1$ (resp. $r_4, r_5, r_8$) is a leaf of $T$. Since $T$ has at most three leaves, Remark 4.10 ensures that at most one of them has degree one in $T'$ and thus at least three vertices of $Y$ have degree two in $T'$. So, we can assume without loss of generality that both entering vertices have degree two in $T'$. Then, $x_u r_1, x_u r_6, x_v r_2$ and $x_v r_5$ are edges. Since $T$ is a tree, one of $r_1$ or $r_5$ are leaves. Now if $y_u$ (resp. $y_v$) has degree zero in $T'$ then $r_4$ (resp. $r_8$) is a leaf of $T$. And, if both $y_u, y_v$ have degree two, then $r_4$ or $r_8$ are leaves. In both cases, we have a contradiction with Remark 4.10. □

So, for every tree $T$ with at most three leaves, $S(T)$ is a vertex cover. We say that $S(T)$ is *the vertex cover associated with $T$*.

### Spanning Tree Reconfiguration to Vertex Cover Reconfiguration

The goal of this section is to prove that an edge flip reconfiguration sequence between spanning trees with at most three leaves in $H(G)$ provides a TAR vertex cover reconfiguration sequence in $G$. So we want to prove that (i) for every spanning tree $T$ with at most three leaves, $|S(T)| \leq k+1$; and (ii), for every tree $T'$ obtained via an edge flip from $T$, $|S(T) \setminus S(T')| + |S(T') \setminus S(T)| \leq 1$.

**Lemma 4.16.** *Let $T$ be a spanning tree of $H(G)$ with at most three leaves. Let $u$ be a vertex of $G$ and $e$ be an irregular edge with endpoint $u$. Assume moreover that no edge before $u$ (resp. after $u$) in the ordering of $u$ are irregular. Then if there is an edge of $\delta_T(\mathcal{G}_e)$ incident to $x_u^e$ (resp. $y_u^e$) then there is an edge between $Z$ and the first (resp. last) entering (resp. exit) vertex of $u$.*

*Proof.* Assume that an edge of $\delta_T(\mathcal{G}_e)$ is incident to $x_u^e$. Since $\mathcal{G}_e$ is the unique irregular edge-gadget for $u$, we can conclude using the arguments of Lemma 4.13. □

Let us now prove that $|S(T)| \leq k+1$ for any spanning tree $T$ with at most three leaves. When no confusion is possible, we will write $S$ for $S(T)$.

**Lemma 4.17.** *Every spanning tree $T$ of $H(G)$ with at most three leaves satisfies $|S(T)| \leq k+1$.*

*Proof.* Assume by contradiction that $|S| \geq k+2$. By Remark 4.14, at least $k-2$ vertices of $S$ are regular. By Lemma 4.13, for each regular vertex $w \in S$, there is an edge of $T$ between $Z$ and the first entering vertex of $w$ and $Z$ and the last exit vertex of $w$. So at least $2k-4$ edges of $\delta_T(Z)$ are incident to regular vertices. Moreover two edges of $\delta_T(Z)$ are incident to $s_1$ and $s_2$. So, $T$ already has $2k-2$ edges in $\delta_T(Z)$. Since $|Z| = k+1$ and $T$ has at most three leaves, Remark 4.10 ensures that $\delta_T(Z)$ has size $2k+1, 2k+2$ or $2k+3$. Indeed, if either all the vertices of $Z$ have degree two or if $Z$ contains both the vertex of degree three and the vertex of degree

one, then $|\delta_T(Z)| = 2k+2$. Otherwise, if $Z$ only contains the vertex of degree one (resp. three), and then $|\delta_T(Z)| = 2k+1$ (resp. $2k+3$). Moreover, if there is no irregular edge-gadget then, since $|S| \geq k+2$, Lemma 4.13 ensures that $Z$ is incident to at least $2k+4$ edges, a contradiction. So there is one or two irregular edge-gadgets by Remark 4.14.

**Case 1.** *$T$ has exactly one irregular edge-gadget $\mathcal{G}_e$ for $e = uv$.*

Since $|S| \geq k+2$, $k$ vertices are regular (otherwise the number of edges incident to $Z$ would be at least $2k+4$ using the argument above, a contradiction). So by Lemma 4.13, $2k$ edges of $\delta_T(Z)$ are incident to regular vertices and two are incident to $s_1$ and $s_2$. So it already gives $2k+2$ edges in $\delta_T(Z)$. Moreover, since $T$ is connected, at least one edge is in $\delta_T(\mathcal{G}_e)$. So by Lemma 4.16, exactly one edge of $T$ is in $\delta_T(\mathcal{G}_e)$. Note that it already gives $2k+3$ edges incident to $Z$ so a vertex of $Z$ has degree three. And then, in $T$, all the vertices of $\mathcal{G}_e$ but at most one have degree two and the last one have degree one. Moreover, $|\delta_T(\mathcal{G}_e)| = 1$.

Let $R$ be the graph restricted to $\mathcal{G}_e$ and $T'$ be the subforest of $T$ restricted to $R$. Since both $u$ and $v$ are in $S$, at least one vertex $v_1$ in $\{x_u^e, y_u^e\}$ (resp. $v_2$ in $\{x_v^e, y_v^e\}$) has degree one in $R$. Since all the vertices have degree two in $T$ but at most one and $|\delta_T(R)| = 1$, the graph $T'$ on $V(\mathcal{G}_e)$ is a Hamiltonian path between $v_1$ and $v_2$. In particular, all the local vertices must have degree two in $T'$. By Lemma 4.12, there is no Hamiltonian path between $v_1$ an $v_2$, a contradiction.

**Case 2.** *There are two irregular edge-gadgets $\mathcal{G}_1$ and $\mathcal{G}_2$.*

Since each special edge-gadget of $T$ contains a vertex of degree one or a vertex of degree three by Lemma 4.11, all the vertices of $Z$ have degree two in $T$. So, $|\delta_T(Z)| = 2k+2$. Since we have seen that at least $2k-4$ edges of $\delta_T(Z)$ are incident to regular vertices, there are at most four edges between $Z$ and special vertices of irregular vertices.

*Subcase* 2.1. The two irregular edge-gadgets are not endpoint disjoint.

We denote by $u_1u_2$ and $u_2u_3$ the two edges of the irregular edge-gadgets. We can assume without loss of generality that the edge-gadget of $u_1u_2$ contains a vertex of degree one and the one of $u_2u_3$ contains a vertex of degree three.

Since $u_1u_2$ (resp. $u_2u_3$) is the unique irregular edge incident to $u_1$ (resp. $u_3$), all the edges incident to $u_1$ (resp. $u_3$) before and after $u_1u_2$ (resp $u_2u_3$) in the ordering of $u_1$ (resp. $u_3$) are regular. So if there is an edge of $\delta(\mathcal{G}_{u_1u_2})$ (resp. $\delta(\mathcal{G}_{u_2u_3})$) incident to the entering or exit vertex of $u_1$ (resp. $u_3$), Lemma 4.16 ensures that this edges creates an additional edge incident to $Z$.

Let $a \geq 0$ such that $|S| = k+2+a$. Let us first prove that $a = 0$. Since there are three irregular vertices, there are at least $k-1+a$ regular vertices. So by Lemma 4.13, at least $2k-2+2a$ edges of $\delta_T(Z)$ are incident to regular vertices and two are incident to $s_1$ and $s_2$ by Remark 4.10. So in total, it already gives $2k+2a$ edges incident to $Z$. Since $|\delta_T(Z)| = 2k+2$, if $a > 0$ then there is no edge between $Z$ and an entering or exit vertex of an irregular vertex.

So no edge of $\delta(\mathcal{G}_{u_1u_2})$ is incident to the entering or exit vertex of $u_1$ and the same holds for $u_3$ in $\delta(\mathcal{G}_{u_2u_3})$ by Lemma 4.16 (since $u_1u_2$ are and $u_2u_3$ are the only irregular edges incident to respectively $u_1$ and $u_3$). Up to symmetry, we can assume that $u_1u_2$ is before $u_2u_3$ in the ordering of $u_2$. So Lemma 4.16 ensures that no edge is incident to the entering vertex of $u_2$ in $\delta(\mathcal{G}_{u_1u_2})$ and the exit vertex of $u_2$ in $\delta(\mathcal{G}_{u_2u_3})$ (these edges are the only irregular edge-gadgets containing $u_2$). So if $\delta_T(\mathcal{G}_{u_1u_2})$ (respectively $\delta_T(\mathcal{G}_{u_2u_3})$ is not empty, it can only contain an edge incident to $y_{u_2}^{u_1u_2}$ (respectively $x_{u_2}^{u_2u_3}$).

But since $T$ is connected, at least one edge has to leave from $\mathcal{G}_{u_1u_2}$ and $\mathcal{G}_{u_2u_3}$. So $T$ have to contain the edges leaving $y_{u_2}^{u_1u_2}$ and $x_{u_2}^{u_2u_3}$. Note that it might be the same edge if $u_1u_2$ and $u_2u_3$ are consecutive in the ordering of $u_2$. But since the gadgets between them are regular, all the

vertices between $y_{u_2}^{u_1u_2}$ and $x_{u_2}^{u_2u_3}$ in $T$ have degree two and does not contain any vertex of $Z$. And then the vertices of the two edge-gadgets cannot be in the connected component of $s_1$, a contradiction.

So we must have $|S| = k+2$ and $u_1, u_2$ and $u_3$ are in $S$. Indeed, there are $k-1$ regular vertices in $S$ and at most three irregular vertices candidates to be in $S$.

Let $e_1 = u_1u_2$. Let $R$ be the graph restricted to $\mathcal{G}_{u_1u_2}$ and $T'$ be the subforest of $T$ restricted to $R$. Since $\mathcal{G}_{e_1}$ does not contain any vertex of degree three and contains exactly one leaf, $T'$ is a union of paths (some of them might be reduced to a single vertex). Moreover, since $T$ has at most one leaf distinct from $s_1, s_2$, at most one local vertex (whose neighborhood is completely included in the edge-gadget) is a leaf of a path in $T'$. Since $T'$ contains a leaf and no vertex of degree at least three, $|\delta(\mathcal{G}_{u_1u_2})|$ is odd (since the sum of the degrees of $V(\mathcal{G}_{u_1u_2})$ is even in $T'$ and odd in $T$ and the difference only consists of edges in $\delta(\mathcal{G}_{u_1u_2})$). If an entering or exit vertex contributes for two edges in $\delta(\mathcal{G}_{u_1u_2})$, one of its local neighbors is a leaf (since this vertex has degree at most two by assumption and one of its local neighbors has degree exactly two in $H$). So at most one edge incident to each -but at most one- entering and exit vertices is in $\delta(\mathcal{G}_{u_1u_2})$. Thus we have $|\delta(\mathcal{G}_{u_1u_2})| \in \{1,3,5\}$.

First assume $|\delta(\mathcal{G}_{u_1u_2})| = 5$, then there are two edges of $\delta(\mathcal{G}_{u_1u_2})$ incident to the same special vertex of the gadget. By construction of $H(G)$, a special vertex of $\mathcal{G}_{u_1u_2}$ is either incident to exactly one edge of $\delta(\mathcal{G}_{u_1u_2})$ if it is not the first entering or last exit vertex, or all the edges of $\delta(\mathcal{G}_{u_1u_2})$ incident to it goes to $Z$. So two edges of $\delta(\mathcal{G}_{u_1u_2})$ are between $Z$ and a special vertex of $\mathcal{G}_{u_1u_2}$. So it already creates two new edges incident to $Z$. Moreover, since $|\delta(\mathcal{G}_{u_1u_2})| = 5$, at least one edge leaving the gadget is incident to each entering or exit vertex. So by Lemma 4.16, since $u_1u_2$ is the only irregular gadget for $u_1$, it creates at least one more edge in $\delta_T(Z)$. Since $\delta_T(Z)$ already contains $2k-2$ edges incident to entering or exit vertices of the $k-1$ regular vertices, and two edges incident to $s_1$ and $s_2$, we have $|\delta_T(Z)| \geq 2k+3$, a contradiction. So from now on, we can assume that $|\delta(\mathcal{G}_{u_1u_2})| \in \{1,3\}$.

Since $u_1 \in S$, an entering or exit vertex of $u_1$ has degree one in the restriction of $T$ to some edge-gadget containing $u_1$. If an entering or exit vertex of $u_1$ has degree one in the subtree $T'$ of $T$ restricted to the edge-gadget for an edge distinct from $u_1u_2$, then Lemma 4.16 ensures that there is an edge between $Z$ and the first entering vertex or the last exit vertex of $u_1$. Now assume that at least one vertex of $x_{u_1}^{u_1u_2}, y_{u_1}^{u_1u_2}$ has degree one in $T'$. Either an edge of $T$ incident to $x_{u_1}^{u_1u_2}$ or $y_{u_1}^{u_1u_2}$ leaves the edge-gadget, and then one edge goes to $Z$ by Lemma 4.16. Otherwise, without loss of generality, $x_{u_1}$ has degree one in $T'$ and in $T$. So all the other vertices of the edge-gadget have degree two in $T$. So free to virtually add an edge between $x_{u_1}$ and the rest of the graph, the gadget becomes regular and then by Lemma 4.11, the vertex $y_{u_1}$ has an edge to the rest of the graph (in $T$), which finally goes to $Z$ by Lemma 4.16. So, there is at least one edge of $\delta_T(Z)$ incident to a special vertex of $u_1$.

Recall that $\mathcal{G}_{u_2u_3}$ contains a vertex of degree three and no leaves. Let us prove that because of this edge-gadget, we can add two edges incident to $Z$. If two of the three edges of the degree-three vertex are in $\delta(\mathcal{G}_{u_2u_3})$, we have already seen that, by definition of $H(G)$, the other endpoints of these edges are in $Z$. And then the conclusion follows. The restriction $T''$ of $T$ to the vertices of $\mathcal{G}_{u_2u_3}$ is a forest. Note that the leaves of $T''$ can only be special vertices since all the vertices of $\mathcal{G}_{u_2u_3}$ have degree at least two in $T$. If $T''$ has at least three leaves, then by Lemma 4.16, at least two of them creates an edge incident to $Z$ since the only one which does not create it is $x_{u_2}^{u_2u_3}$. Indeed, by Lemma 4.16, all the edges of $\delta(\mathcal{G}_{u_2u_3})$ incident to a special vertex of $u_3$ immediately creates an edge incident to $Z$. The same holds for $y_{u_2}^{u_2u_3}$ since $u_2u_3$ is the last irregular edge incident to $u_3$. So if $T''$ has three leaves, it creates two edges incident to $Z$ (indeed three edges are leaving the edge-gadget and only the one, if it exists, incident to $x_{u_2}^{u_2u_3}$ does not create an edge incident to $Z$). So we can assume that $T''$ has exactly two leaves and

then the degree-three vertex is an entering or exit vertex. Since this vertex has degree two in $T''$, $T''$ contains two other leaves. And again there are three distinct special vertices incident to an edge of $\delta_T(\mathcal{G}_{u_2u_3})$. And as in the previous case, Lemma 4.16 ensures that at least two of them are creating one new edge incident to $Z$. So in both cases, the number of edges of $\delta_T(Z)$ incident to entering or exit vertices of $u_2, u_3$ is at least two.

So $|\delta_T(Z)| \geq 2k+3$, a contradiction.

*Subcase* 2.2. The two irregular edge-gadgets are endpoint disjoint.

Let $u_1u_2$ and $u_3u_4$ be the two irregular edges. Let $\mathcal{G}_1 = \mathcal{G}_{u_1u_2}$ and and $\mathcal{G}_2 = \mathcal{G}_{u_3u_4}$. Note that since $u_1u_2$ and $u_3u_4$ are the unique irregular edges for respectively $u_1, u_2, u_3, u_4$, all the edges leaving these edge-gadgets create an edge incident to $Z$ by Lemma 4.16. Since there are at most four edges between $Z$ and special vertices of irregular vertices, we have $|\delta_T(\mathcal{G}_1)| + |\delta_T(\mathcal{G}_2)| \leq 4$. Let us prove by contradiction that $|\delta_T(\mathcal{G}_1)| + |\delta_T(\mathcal{G}_2)| > 4$.

Let us first prove that the number of regular vertices is exactly $k-2$. We have already seen that it has to be at least $k-2$. Assume by contradiction that the number of regular vertices is at least $k-1$. Then, by Lemma 4.13, there are $2k-2$ edges between $Z$ and entering or exit vertices of regular vertices. We also have the edges $s_1z_1$ and $s_2z_2$. Moreover, every edge in $\delta_T(\mathcal{G}_1)$ and $\delta_T(\mathcal{G}_2)$ creates an edges in $\delta_T(Z)$ incident to irregular vertices by Lemma 4.16 and the fact that $u_1u_2$ and $u_3u_4$ are the only irregular edges incident to each of these four vertices. Since there are two irregular edges, all the vertices of $Z$ have degree two and so $|\delta_T(Z)| = 2k+2$. So $|\delta_T(\mathcal{G}_1)| + |\delta_T(\mathcal{G}_2)| = 2$. But since one of $\mathcal{G}_1$ or $\mathcal{G}_2$ contains a vertex of degree three and no leaves, three edges have to leave it, a contradiction. So from now on we can assume that the number of regular vertices is $k-2$ and then all of $u_1, u_2, u_3, u_4$ are in $S$ (since $|S| \geq k+2$).

First assume that, $|\delta_T(\mathcal{G}_1)| = 1$ or $|\delta_T(\mathcal{G}_2)| = 1$, let us say w.l.o.g. $\mathcal{G}_1$. Then, one vertex of the edge-gadget $\mathcal{G}_1$ is a leaf and $\mathcal{G}_2$ contains the vertex of degree three. Since there are two irregular edge-gadgets, all the vertices of $\mathcal{G}_1$ but the leaf have degree two in $T$. Moreover, since both $u_1$ and $u_2$ are in $S$, an entering or exit vertex incident to $u_1$ and $u_2$ have to be of degree one in the restriction of $T$ to one of their edge-gadgets.

We claim that it implies that an entering or exit vertex of both $u_1$ and $u_2$ in the edge-gadget of $\mathcal{G}_1$ has degree one in the restriction of $T$ to $\mathcal{G}_1$. Let us first prove that an edge of $\delta_T(\mathcal{G}_1)$ is incident to the entering or exit vertices of $u_1$, and that the same holds for $u_2$. Let us prove the statement for $u_1$ and assume by contradiction that it is not the case. Let $e'_i$ be the closest edge from $u_1u_2$ in the ordering of $u_1$ such that $x_{u_i}^{e'_i}$ or $y_{u_i}^{e'_i}$ has degree one in the graph restricted to $\mathcal{G}_{e'_i}$. Since $e'_i$ is regular, it implies by Lemma 4.11 that an edge of $T$ is incident to the exit vertex of the gadget before $e'_i$ and the entering vertex of the gadget after $e'_i$. So an edge of $T$ leaving the gadget $\mathcal{G}_1$ is incident to entering or exit vertices of $u_1$, denoted by $x$. Now, since $\mathcal{G}_1$ contains one leaf and no vertex of degree three, if $x$ has degree one in $T$, its degree-two incident local neighbor also is a leaf, a contradiction. So it has degree two and then has degree one in the gadget. A similar proof gives the same for $u_2$.

So one of the vertices $\{x_{u_1}^{u_1u2}, y_{u_1}^{u_1u2}\}$ and one of the vertices $\{x_{u_2}^{u_1u2}, y_{u_2}^{u_1u2}\}$ have degree one in the subgraph $T'$ of $T$ induced by the vertices of $\mathcal{G}_1$. Since all the vertices but at most one (which cannot be a local vertex) have degree two in $T$ and $|\delta_T(\mathcal{G}_1)| = 1$ by assumption, $T'$ is a Hamiltonian path on $\mathcal{G}_1$ between one vertex of $\{x_{u_1}^{u_1u2}, y_{u_1}^{u_1u2}\}$ and one vertex of $\{x_{u_2}^{u_1u2}, y_{u_2}^{u_1u2}\}$, a contradiction with Lemma 4.12. So we cannot have $|\delta(\mathcal{G}_1)| = 1$.

So we can assume that $|\delta_T(\mathcal{G}_1)| = 2$ and $|\delta_T(\mathcal{G}_2)| = 2$. Let $\mathcal{G}_2$ be the edge-gadget containing a vertex of degree three and no leaves. Since it contains a branching node and no leaf, at least three edges are in $\delta_T(\mathcal{G}_2)$, a contradiction. □

So the vertex cover $S(T)$ associated with every spanning tree $T$ with at most three leaves has size at most $k+1$. In order to prove that a spanning tree transformation provides a vertex cover transformation for the TAR setting, we have to prove that, for every edge flip, then either $S$ is not modified, or one vertex is added to $S$ or one vertex is removed from $S$.

**Lemma 4.18.** *Let $T_1$ and $T_2$ be two adjacent trees with at most three leaves. Then the symmetric difference between the sets $S(T_1)$ and $S(T_2)$ is at most one.*

*Proof.* We want to prove that $S(T_2) = S(T_1)$ or there exists $x$ such that $S(T_2) = S(T_1) \setminus \{x\}$ or $S(T_2) = S(T_1) \cup \{x\}$. In order to prove it, the rest of the proof is devoted to show that, if after some edge flip, a vertex is added to $S(T_2)$ then no vertex of $S(T_1)$ is removed in $S(T_2)$. We claim that it is enough to conclude. Indeed, since $|S| \leq k+1$ by Lemma 4.17 and $|S| \geq k$ (since $k$ is the minimum size of a vertex cover), if we want the symmetric difference to be at least two, then we must contain at least one vertex in $S(T_1) \setminus S(T_2)$ and conversely. Let us now assume by contradiction that $|S(T_1) \setminus S(T_2)| = 1$ and $|S(T_2) \setminus S(T_1)| = 1$. Let $f$ be the edge of $T_1 \setminus T_2$ and $g$ be the edge of $T_2 \setminus T_1$. Let $u = S(T_2) \setminus S(T_1)$ and $v = S(T_1) \setminus S(T_2)$. Note that in order to modify $S(T)$ (for some tree $T$), we need to modify the degree of a special vertex in an edge-gadget of an edge of $G$ incident to it. So both $f$ and $g$ have to have both endpoints in the same edge-gadget. And the following remark ensures that the addition or deletion of $f$ and $g$ can only modify by one vertex the set $S$. In particular, it implies that $|S(T_1) \triangle S(T_2)| \leq 2$.

**Remark 4.19.** *Let $a, b$ be two special vertices in the same edge-gadget. The distance between $a$ and $b$ is at least three in $H(G)$.*

Remark 4.19 ensures that, if we remove or add an edge of $T$, the degree of exactly one entering or exit vertex is modified. Since $S(T_2) \setminus S(T_1)$ and $S(T_1) \setminus S(T_2)$ are non empty, an entering or exit vertex of $u$ or $v$ has to be incident to $f$ and an entering or exit vertex of the other vertex of $u$ or $v$ has to be incident to $g$. By abuse of notation we will say that $f$ (resp. $g$) *adds $u$ to $S(T_2)$* (resp. *remove $v$ from $S(T_1)$*).

Since the edge $f$ (resp. $g$) adds $u$ or remove $v$, it has to have both endpoints in the same edge-gadget. Indeed, in order to add $u$ to $S(T_2)$ (or remove $v$ from $S(T_1)$) we must modify the degree of $x_v^e$ or $y_v^e$ (resp. $x_u^e$ or $y_u^e$) inside an edge-gadget.

Now let us distinguish cases depending on the degree of the endpoints of $f$. If both endpoints of $f$ are of degree two, then the deletion of $f$ creates two vertices of degree one. By Remark 4.10, at most one of them is a leaf in $T_2$. So $g$ has to be incident to one of them. And by Remark 4.19, the edge $g$ cannot be incident to another special vertex of the edge-gadget. And thus $g$ does not add or remove a good vertex, a contradiction.

If one endpoint of $f$ has degree three and one has degree one, then the deletion of $f$ creates a vertex of degree zero. Thus $g$ must be incident to the degree zero vertex. Again, by Remark 4.19, $g$ cannot add or remove another vertex of $S(T_1)$, a contradiction. Note that we get a similar contradiction if one endpoint of $f$ has degree two and the other has degree one.

So we can assume that one endpoint of $f$ has degree two and the other has degree three. The edge $g$ cannot be added between two vertices of degree at least two in $T_1 \setminus f$ since otherwise $T_2$ would have two branching nodes. So at least one endpoint of $g$ (and even exactly one by Remark 4.10) has degree one in $T_1 \setminus f$. By Remark 4.19, the endpoint of $g$ of degree one was already of degree one in $T_1$ since $g$ has to modify $S$. Moreover, the other endpoint of $g$ has degree exactly two in $T_1 \setminus f$ (otherwise we would create a vertex of degree four in $T_2$), and then by Remark 4.19 has degree two in $T_1$. So in particular, the edge-gadget containing $f$ has one vertex of degree three and all the others have degree two and the edge-gadget containing $g$ has one vertex of degree one and all the others have degree two in $T_1$. Note that the deletion of $f$

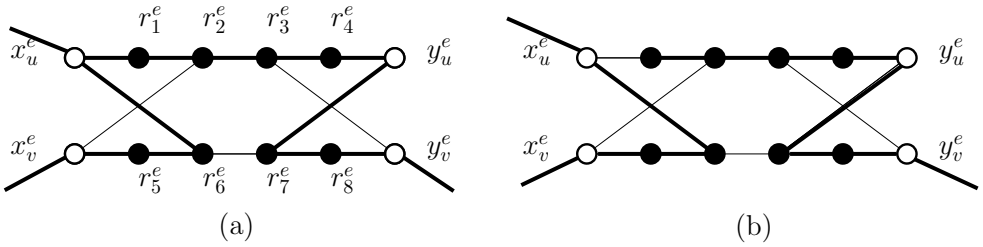



Figure 4.6 – Illustration of the proof of Lemma 4.18.

can have two effects on $S(T_1)$: either a vertex disappears (because the degree of a special vertex drops from one to zero), or a vertex appears (because the degree of a special vertex drops from two to one).

**Case 1.** *$v$ is removed from $S(T_1)$ when $f$ is removed.*

Let $e = wv$ be the edge such that $f$ has both endpoints in $\mathcal{G}_e$. Let $R$ be the subgraph induced by the vertices of $\mathcal{G}_e$ and $T'$ the restriction of $T_1$ on $R$. Since $v$ is removed from $S(T_1)$, it implies that $x_v^e$ or $y_v^e$ has degree one in $T'$ and $f$ is incident to that vertex. Up to symmetry let us assume that it is $x_v^e$. If the edge $f$ is not $x_v^e r_1$, then $r_1$ is a leaf of $T_1$, a contradiction since the degree one vertex has to be in the edge-gadget containing $g$. So the only edge of $T'$ incident to $x_v^e$ is $x_v^e r_1$ and then $f = x_v^e r_1$. Since one the two endpoints of $f$ has degree three in $T_1$ and $r_1$ has degree two in $H(G)$, there are two edges of $\delta(\mathcal{G}_e)$ incident to $x_v^e$.

*Claim* 1. Let $\mathcal{G}_e$ with $e = vw$ be an edge-gadget and $R$ be the subgraph of $H$ induced by the vertices of $\mathcal{G}_e$. There does not exist any tree $T$ such that, in the subgraph of $T$ induced by the vertices of $R$, all the local vertices but $r_1$ have degree two, $x_v^e$ has degree zero and $y_v^e$ has degree two.

*Proof.* Let us denote by $T'$ the subgraph of $T$ induced by the vertices of $R$. Since all the local vertices but $r_1$ have degree two and $y_v^e$ has degree two in $T'$, $T'$ contains the paths $r_1 r_2$, $x_w^e r_5 r_6$ and $r_3 r_4 y_v^e r_7 r_8 y_w^e$. Since $x_v^e r_6$ is not an edge of $T$ (because we assumed that $x_v^e$ has degree zero in $R$) and $r_7$ does not have degree three, $r_6$ is a leaf of $T$, a contradiction. ◊

When $f$ is removed from $T_1$, $v$ is removed from $S$, thus $y_v^e$ has degree zero or two in $T'$. Since all the local vertices of $R$ have degree two in $T_1$ and $r_4$ has degree two in $H(G)$, both edges incident to it are in $T'$. And then $y_v^e$ does not have degree zero. So by Claim 1, the edge-gadget must contain another vertex of degree three or another leaf, a contradiction.

**Case 2.** *$u$ is added to $S(T_1)$ when $f$ is removed.*

Let $e = uv$ be the edge such that $f$ is in $\mathcal{G}_e$. Let $R$ be the subgraph induced by the vertices of $\mathcal{G}_e$. In that case, the vertices $x_u^e$ and $y_u^e$ have degree two in $R \cap T_1$ (if one of them was of degree one, $u$ was already in $S$ and none of them can be of degree zero, otherwise one local vertex should be a leaf, a contradiction since the leaf is in the edge-gadget containing $g$). Since all the local vertices have degree two or three, it implies that $x_v^e r_5 r_6 x_u^e r_1 r_2$ and $y_v^e r_8 r_7 y_u^e r_4 r_3$ are in $T_1$. But then $r_2 r_3$ must be in $T_1$, otherwise they would be leaves of $T_1$ ($x_v^e r_2 \notin E(T_1)$ otherwise $x_v^e r_5 r_6 x_u^e r_1 r_2$ is a cycle (same for $r_3^e y_u^e$)). Moreover $r_6 r_7$ cannot be an edge since otherwise there is a cycle (and both $r_6, r_7$ would have degree three). So the endpoint of $f$ of degree three has to be $x_u^e$ or $y_u^e$, without loss of generality $x_u^e$. So the graph around $\mathcal{G}_e$ is the graph represented in Figure 4.6(a). Note in particular that $|\delta_{T_1}(\mathcal{G}_e)| = 3$ since $x_v^e$ and $y_v^e$ have degree two in $T_1$ and $x_u^e$ has degree three in $T_1$.

Let $e'$ be the edge such that $g$ is in $\mathcal{G}_{e'}$ with $e' = u'v'$. Recall that $u'$ or $v'$ is removed from $S$ when $g$ is added. So we can assume without loss of generality that $g$ is incident to $x^e_{u'}$. Let $R'$ be the subgraph induced by the vertices of $\mathcal{G}_{e'}$. All the vertices in the edge-gadget $\mathcal{G}_{e'}$ have degree two in $T_1$ but one vertex which has degree one. Moreover, $g$ is an edge between a vertex of degree two and a vertex of degree one. Since $u'$ is removed from $S$ when we add $g$, $x^{e'}_{u'}$ has degree exactly one in the restriction of $T_1$ to $R'$.

Let us first assume that $x^{e'}_{u'}r_1$ is in $T_1$ and then $x^{e'}_{u'}r_6$ is not in $T_1$ (i.e., $g = x^{e'}_u r_6$). Since all the local vertices but maybe $r_6$ have degree two and $y^{e'}_{u'}$ has degree two, all the subpaths $r_3r_4y^{e'}_{u'}r_7r_8y^{e'}_{v'}$, $x^{e'}_{v'}r_5r_6$ and $x^{e'}_{u'}r_1r_2$ are in $T_1$. Since $r_3$ must have degree two in $T_1$ and $r_3y^{e'}_{v'}$ closes a cycle, $r_2r_3$ is in $T_1$. Since $x^{e'}_{u'}r_6$ is not in $T_1$ by assumption and $\mathcal{G}_{e'}$ does not contain any vertex of degree three, $r_6$ is a leaf of $T_1$ and then $|\delta_{T_1}(\mathcal{G}_{e'})| = 3$ since $x^{e'}_{u'}$, $x^{e'}_{v'}$ and $y^{e'}_{v'}$ have degree two in $T_1$.

Let us now assume that $x^{e'}_{u'}r_1$ is not in $T_1$ and then $x^{e'}_{u'}r_6$ is (i.e., $g = x^{e'}_{u'}r_1$). Since all the local vertices but $r_1$ have degree two and $y^{e'}_{u'}$ has degree two, all the subpaths $r_3r_4y^{e'}_{u'}r_7r_8y^{e'}_{v'}$, $x^{e'}_{v'}r_5r_6x^{e'}_{u'}$ and $r_1r_2$ are in $T_1$. Since $r_3$ must have degree two in $T_1$ and $r_3y^{e'}_{v'}$ closes a cycle, $r_2r_3$ is in $T_1$. Since $r_1$ is a leaf of $T_1$, $x^{e'}_{u'}$ has degree two in $T_1$. And then $|\delta_{T_1}(\mathcal{G}_{e'})| = 3$ since $x^{e'}_{u'}$, $x^{e'}_{v'}$ and $y^{e'}_{v'}$ have degree two in $T_1$. The graph around $\mathcal{G}_{e'}$ is the graph represented in Figure 4.6(b).

So in both cases ($x^{e'}_{u'}r_1$ or $x^{e'}_{u'}r_6$ in $T_1$), we have $|\delta_{T_1}(\mathcal{G}_{e'})| = 3$. Moreover, we have seen that $|\delta_{T_1}(\mathcal{G}_e)| = 3$. We claim that $S(T_1)$ has size $k+1$. Recall that $f = uv$ and $g = u'v'$. Let us show that $S(T_1) \setminus \{u'\}$ is a vertex cover. For every edge $u'w$ with $w \neq u, v'$, since $S(T_2) = (S(T_1) \cup \{u\}) \setminus \{u'\}$ is a vertex cover, $w$ is in $S(T_1)$. So if an edge is not covered in $S(T_1) \setminus \{u'\}$, it is $uu'$ or $u'v'$. After the edge flip, $x^{u'v'}_{u'}$ and $y^{u'v'}_{u'}$ have even degree in $T_2$ and thus $x^{u'v'}_{v'}$ or $y^{u'v'}_{v'}$ has degree one by Lemma 4.15. Since neither $f$ nor $g$ changes the degree of $x^{u'v'}_{v'}$ nor $y^{u'v'}_{v'}$, $v' \in S(T_1)$. So if an edge is not covered, it is $uu'$. But, since $u \notin S(T_1)$, in the restriction of $T_1$ to $\mathcal{G}_{uu'}$, either $x^{uu'}_{u'}$ or $y^{uu'}_{u'}$ has degree one and this degree does not change after the edge flip, a contradiction since $u' \notin S(T_2)$, so $uu'$ does not exist and then $S(T_1) \setminus \{u'\}$ is a vertex cover. Since a minimum vertex cover has size $k$, $S(T_1)$ has size at least $k+1$ and then exactly $k+1$ by Lemma 4.17.

So $k-2$ vertices of $S(T_1)$ are not incident to any irregular edge-gadgets. Indeed, there are at most four irregular vertices and $u' \notin S(T_1)$ is one of them. By Lemma 4.13, this gives $2k-4$ edges in $\delta(Z)$. Since, for both $\mathcal{G}_e$ and $\mathcal{G}_{e'}$, there are three edges leaving the gadget and since $e$ and $e'$ are endpoint disjoint, this creates 6 more edges incident to $Z$. Since there are moreover the two edges $s_1z_1$ and $s_2z_{k+1}$ in $\delta(Z)$, it gives in total $2k+4$ edges in $\delta(Z)$, a contradiction with the fact that all the vertices of $Z$ must have degree two. □

Lemmas 4.17 and 4.18 immediately imply the following:

**Lemma 4.20.** *If there is an edge flip reconfiguration sequence between two spanning trees $T_1$ and $T_2$, then there is a* TAR*-sequence (with threshold $k+1$) between $S(T_1)$ and $S(T_2)$.*

### Vertex Cover Reconfiguration to Spanning Tree Reconfiguration

We now prove the converse of the previous subsection. Actually, the statement will not be exactly the converse but it will actually be enough to conclude. We will prove that if there is a TJ-transformation sequence between two vertex covers $X$ and $Y$ then we also have an edge flip reconfiguration sequence between Hamiltonian paths corresponding to $X$ and $Y$. Let $X, Y$ be two vertex covers of size $k$. In the TJ-adjacency rule, $X$ and $Y$ are adjacent if there exist two vertices $x$ and $y$ such that $Y = (X \setminus \{x\}) \cup \{y\}$.

We have already remarked that there might be a lot of Hamiltonian paths associated with a vertex cover $X$ in $H(G)$. Note that, in all these paths, for every $u \in X$, the subpath $P_x$ between the first entering vertex of $u$ and the last exit vertex of $u$ is the same. However (i) the order in which these subpaths appear in the path may differ (according to the ordering in which they are attached to $Z$); (ii) when we follow the path from $s_1$ to $s_2$ we might see the path in the ordering of $P_x$ or in the reverse ordering depending on whether the first vertex of $Z$ incident to $P_x$ is incident to the first entering vertex of the last exit vertex. The goal of the proof is to show that, if we have one of them, then we can reach all of them, i.e., change the order of appearance of the paths $P_x$ and reverse their ordering. The first part of this section is devoted to proving that they all are in the same connected component of the reconfiguration graph. Let us first show the following intermediate lemma.

**Lemma 4.21.** *Let $A, B$ be two sets such that $|A| = |B| + 1$ and $G$ be the bipartite graph $\mathcal{B}$ on vertex set $(A, B \cup \{s_1, s_2\})$ where $A$ is complete to $B$ and $s_1, s_2$ be two vertices of $B$, each connected to exactly one (distinct) vertex of $A$. Let $P_1, P_2$ be two Hamiltonian paths with the same endpoints $s_1, s_2$. Then one can transform $P_1$ into $P_2$ via edge flips where all the intermediate spanning trees have at most three leaves.*

*Proof.* We say that two paths $P, P'$ on the same vertex set *agree up to* $i \in \mathbb{N}$ if the first $i$ vertices of $P$ and $P'$ are the same. Note that $P_1, P_2$ agree up to 2 since both start with $s_1$ and $s_1$ only have one neighbor in $\mathcal{B}$. We prove iteratively that if we have two paths that agree up to $i$, then we can transform the second into two paths that agree up to $i + 1$.

Assume that $P_1$ and $P_2$ agree up to $i$. Let $u$ be the $i$-th vertex and $v$ be the $(i+1)$-th in $P_1$. If $v$ also is the $(i+1)$-th vertex in $P_2$, the conclusion holds. So we can assume that the $(i+1)$-th vertex of $P_2$ is $y \neq v$. Let $w$ be the vertex after $v$ in $P_2$. Note that it cannot be $y$ since both $y$ and $v$ are in the same set of $A, B$. We perform the following edge flips in $P_2$: we remove $uy$ to create $uv$. We then remove $vw$ to create $yw$.

After these two operations, all the vertices have degree two. Moreover the intermediate and final graphs are connected. Indeed, since $u, y, v$ appear in $P_2$ in that order, the removal of $uy$ to create $uv$ keeps a connected graph. And one can remark that the two operations just consists in permuting the subpath between $y$ and $v$ in $P_2$. To conclude, we have to prove that the edges we want to create indeed exist in $\mathcal{B}$. Since $B$ is complete to $A$, if $u$ and $w$ are in $B$, the conclusion follows. So we can assume that they are in $A \cup \{s_1, s_2\}$. Since $A$ is complete to $B$, and $u$ is distinct from $s_1$, and $y, v \in B$ (since they are not the last vertices of $P_1$ and $P_2$), the only edge that might not exist is $yw$ if $w = s_2$. But it is impossible since $s_2$ only have one neighbor in $\mathcal{B}$ and then the second to last vertex of $P_1$ and $P_2$ are the same, i.e., $y$ cannot be incident to $u$ in $P_2$. □

Using this lemma, let us prove the following:

**Lemma 4.22.** *Let $G = (V, E)$ be a graph and $X$ be minimum vertex cover of $G$. Then all the Hamiltonian paths associated with $X$ in $H(G)$ are in the same connected component of the reconfiguration graph of spanning trees with at most three leaves.*

*Proof.* Let $k = |X|$. Let us denote by $A$ the set $Z$ of $H(G)$ and by $B$ the set $X$. Note that by construction $|A| = |B| + 1$. We now add two new vertices $s_1, s_2$ one connected to $z_1$ and the other connected to $z_{k+1}$ and create all the edges between $A$ and $B$. We denote by $\mathcal{B}$ the resulting graph that satisfies the condition of Lemma 4.21. Now one can associate with any Hamiltonian path associated with $X$ a path of $\mathcal{B}$ where $x$ in $B$ is connected to $z, z'$ in $A$ if $z$ and $z'$ are the vertices of $Z$ attached to the first and last entering and exit vertices of $x$. By Lemma 4.21, one can transform any path of $\mathcal{B}$ into any other. We claim that such a transformation can be immediately

extended for the Hamiltonian paths of $H(G)$. Indeed, by definition of a Hamiltonian path of $H(G)$ associated with $X$, the subpath $P_u$ between the first entering vertex of $u$ and the last exit vertex of $u$ (for $u \in X$) does not contain any other entering or exit vertex of vertices of $X$ and only contains degree-two vertices. So the connectivity of the graph as well as its non-degree two vertices remain the same if we can contract $P_u$ into a single vertex $u$.

After this operation, we know that in the resulting Hamiltonian path, the subpaths associated with each vertex appear in the same ordering. However, it might be the case that in some path $z_i$ is connected to the first entering vertex $x$ of $u \in X$ and $z_{i+1}$ to the last exit vertex $y$ of $u$ and that we have the converse in the other path. In other words, instead of "reading" the path from the first entering vertex to the last exit vertex we "read" it in the other direction. In that case, for every such $i$, we perform the following edge flips: remove $z_iu$ to create $z_iv$; and then remove $z_{i+1}v$ to create $z_{i+1}u$. □

Let us now prove that any TJ-transformation between two vertex covers $X$ and $Y$ can be adapted into an edge flip transformation between the corresponding Hamiltonian paths via spanning trees with at most three leaves. In order to prove it, we simply have to show that we can do it for each single step transformation.

**Lemma 4.23.** *Let $X$ be a minimum vertex cover of $G$ and $Y = (X \setminus \{u\}) \cup \{v\}$ be another vertex cover, for some vertices $u$ and $v$. Then we can transform any Hamiltonian path associated with $X$ into any Hamiltonian path associated with $Y$ via a sequence of spanning trees with at most three leaves.*

*Proof.* By Lemma 4.22, all the Hamiltonian associated with $X$ are in the same connected component of the reconfiguration graph and the same holds for $Y$. So we simply have to show that there exists a transformation from a Hamiltonian path associated with $X$ into a Hamiltonian path associated with $Y$. First, observe that since $X$ and $Y$ are both minimum vertex covers of $G$ and $Y = (X \setminus \{u\}) \cup \{v\}$, $X \setminus \{u\}$ covers all the edges of $G$, but $uv$. In particular, all the neighbors of $u$ but $v$ are in $X$. Similarly, all the neighbors of $v$ but $u$ are in $Y$. Let $W = X \cap Y$ given with an arbitrary ordering of $W$. The *canonical path associated with* $W, u$ (resp. $W, v$) is the Hamiltonian path of $H(G)$ with the ordering of $u$ (resp. $v$) and then the ordering of $W$. More formally, recall that given a vertex cover $W$, we can define a path $P_w$ for every $w \in W$ between the first entering vertex of $w$ and the last exit vertex of $w$ that does not contain any special vertex of $w' \in W$ with $w' \neq w$. And that any Hamiltonian path associated with $W$ is the concatenation of these paths linked together thanks to the vertices of $Z$. So the ordering of $W$ of a path $P$ is the ordering of appearance of the subpaths $P_w$ for $w \in W$. In particular, in the ordering of $W_u$, the subpath $P_u$ appears at the beginning of the path and thus $P_u$ is connected to $z_1$ and $z_2$.

The *half-path $T_h$ associated with $W, u, v$* is the following. For every edge-gadget $\mathcal{G}_e$ with $e$ distinct from the first edge of $v$, the restriction of $T_h$ around $\mathcal{G}_e$ is one of the graphs of Figure 4.5; If both endpoints of $e$ are in $W \cup \{u, v\} = X \cup Y$, the gadget is the one of Figure 4.5(a), otherwise it is the one of Figure 4.5(b) (the edges of $\delta_{T_h}(\mathcal{G}_e)$ being incident to the entering and exit vertex of $W \cup \{u, v\}$). For the edge-gadget of $e' = vw$ (note that we possibly have $w = u$), the first edge of the ordering of $v$, the restriction of $T_h$ around $\mathcal{G}_{e'}$ is the graph $x_v^{e'} r_1 r_2 r_3 r_4 y_v^{e'}$ and $x_w^{e'} r_5 r_6 r_7 r_8 y_w^{e'}$ plus edges leaving $y_v^{e'}, x_w^{e'}, y_w^{e'}$ but no edge leaving $x_v^{e'}$.

Let us now explain how the vertices of $Z$ are connected to entering and exit vertices. The vertex $z_1$ is incident to $s_1$ and the first entering vertex of $u$. The vertex $z_2$ is incident to the last exit vertex of $u$ and the last exit vertex of $v$. Moreover, the vertex $z_{i+1}$ is incident to the last exit vertex of the $i$-th vertex of $W$ and the first entering vertex of the $(i+1)$-th vertex of $W$. Finally the vertex $z_{k+1}$ is incident to $s_2$. One can easily check that the following holds for $T_h$:

- All the vertices of $T_h$ have degree two but $z_2$ that has degree three and the first entering vertex of $v$ that has degree one. Indeed, $z_2$ is incident to the last exit vertex of $u$, the last exit vertex of $v$ and the first entering vertex of the first vertex of $W$. And the first entering vertex of $v$ is not connected to any vertex of $Z$.
- The subpath of $T_h$ from $s_1$ to $z_2$ is $s_1z_1$ and then the concatenation of the paths (for every edge $e$ incident to $u$) $x_u^e, r_1, r_2, r_3, r_4, y_u^e$ (or $x_u^e, r_5, r_6, r_7, r_8, y_u^e$) connected by the special edges between consecutive edge-gadgets of $u$. Indeed, $W \setminus \{u\}$ covers all the edges but $uv$, all the neighbors of $u$ but $v$ are in $W$. Let $e'' = uv$. The construction of $W, u, v$ also ensures that $\mathcal{G}_{e''}$ contains the subpath $x_u^{e''} r_1 r_2 r_3 r_4 y_u^{e''}$ (or $r_5 r_6 r_7 r_8$), no matter whether $e''$ is the first edge of $v$, or not.
- Similarly the subpath of $T_h$ from the first entering vertex of $v$ to $z_2$ is the concatenation of the paths (for every edge incident to $v$) containing the entering vertex of $v$ for the current edge, $r_1, r_2, r_3, r_4$ (or $r_5$, $r_6$, $r_7$, $r_8$) and the exit vertex of $v$.
- The subpath of $T_h$ from $z_2$ to $s_2$ is the subpath of the canonical path associated with $W, u$ except that for every edge $e = vw$, the graph around $\mathcal{G}_e$ is the graph of Figure 4.5(a) instead of the graph of Figure 4.5(b).

In particular, one can notice that $T_h$ is a tree. Note moreover that, if we denote by respectively $e$ and $e'$ the first edges of $u$ and $v$ respectively, the edge flip $z_1x_u^e$ into $z_1x_v^{e'}$ transforms the half-path of $W, u, v$ into the half-path of $W, v, u$.

So in order to conclude, we simply have to prove that we can transform the canonical path associated with $W, u$ into the half-path associated with $W, v, u$.

The proof is based on local transformations for every edge-gadget iteratively on the gadgets (see Figure 4.7 below).

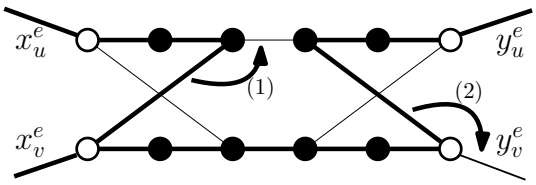



Figure 4.7 – Transformation of an edge-gadget.

Let $P$ be the Hamiltonian path associated with $X, u$. Since $X$ and $Y$ are vertex covers, the path $P$ has the following properties:

- The restriction of $P$ to every edge-gadget with endpoint $v$ is of type Figure 4.5(b). Indeed $v \notin X$ and $X$ is a vertex cover. In particular, the restriction of $P$ to the edge-gadget of $uv$ is of type Figure 4.5(b).
- The restriction of $P$ to every other edge-gadget edge incident to $u$ is of type Figure 4.5(a). Indeed $(X \setminus \{u\}) \cup \{v\}$ is a vertex cover. So all the neighbors of $u$ but $v$ are in $X$.

Let us denote by $x_u$, $x_v$, $y_u$, $y_v$ the first entering vertices of $u$ and $v$ and the last exit vertices of $u$ and $v$, respectively. Since $P$ is associated with $X, u$, $z_1x_u$ and $z_2y_u$ are edges of $P$. Let us delete $z_1x_u$ and add $z_1x_v$. Note that this operation creates a vertex of degree one (namely $x_u$) and a vertex of degree three (namely $x_v$). The resulting graph is indeed connected since only $s_1z_1$ is attached to $z_1$.

Let us prove iteratively on the edges incident to $v$ that we can transform the current graph keeping the degree sequence and the connectivity in such a way that (i) the unique vertex of degree three is the current entering vertex $x$ of $v$, and (ii) there is a subpath attached to $x$ which is $s_1z_1$ and then the concatenation of the paths (for every edge incident to $v$ smaller than the current edge) containing the entering vertex of $v$ for that edge, $r_1, r_2, r_3, r_4$ (or $r_5$, $r_6$, $r_7$, $r_8$) and the exit vertex of $v$.

Note that the property indeed holds at the beginning since $x_v$ the first entering vertex of $v$ has degree three and there is a path $s_1z_1$ attached to it. Since $v$ is not in $X$, the graph around the current edge $e$ is indeed the graph of Figure 4.5(b) in $P$. So in the current tree, we have the graph of Figure 4.7. One can remark that the transformations proposed in Figure 4.7 keeps the degree sequence. Moreover, after these operations, one can note that the property holds up to the next entering vertex of $v$. So we simply have to show that the connectivity is kept to conclude. The first transformation indeed keeps connectivity. The second also keeps connectivity since the next entering vertex of $v$ is not in the subpath between $s_1$ and $x_v^e$.

When we treat the last edge-gadget of $v$, we simply have to connect the last exit vertex to $z_2$ (which now has degree three) in order to obtain the subpath associated with $W, v, u$. Similarly, we can transform the path associated with $W, v$ into the half-path associated with $W, u, v$. And, as we already observed, there is one edge flip that transforms the first into the second. So it is possible to transform the canonical path associated with $W, u$ into the canonical path associated with $W, v$, which completes the proof. □

**Spanning trees with more leaves**

We are now ready to get down to the proof of Theorem 4.8, that is SPANNING TREE WITH AT MOST $k$ LEAVES is PSPACE-complete for every $k \geq 3$:

*Proof.* We perform the same reduction as in the proof of Theorem 4.9 except that in the construction of the graph $H(G)$ we replace the two vertices $s_1, s_2$ by $\ell + 1$ additional vertices $s_1, s_2, \ldots, s_{\ell+1}$ where $s_1$ is connected to $z_1$, $s_2$ is connected to $z_{k+1}$ and $s_3, \ldots, s_{\ell+1}$ are connected to $s_1$. Note that in any spanning tree, $s_2, \ldots, s_{\ell+1}$ are leaves. So Remark 4.10 also holds with this reduction. The rest of the proof is the same. □

### 4.1.4 Concluding remarks

In this section, we studied the reconfiguration of spanning trees with additional constraints on the number of leaves of each spanning tree. We first considered the problem SPANNING TREE WITH MANY LEAVES, where each spanning tree has at least $k$ leaves, for a positive integer $k \geq 2$. We showed that it is PSPACE-complete, even when restricted to bipartite graphs, split graphs or planar graphs. However, one might be interested in the complexity on outerplanar graphs: is it polynomial-time solvable or not? We also proved that SPANNING TREE WITH MANY LEAVES is in P if the input graph is a cograph. As an intermediate result, we showed that if we have two internal nodes and thus $k = n - 2$, then one can solve in polynomial-time SPANNING TREE WITH MANY LEAVES on any graph $G$. We believe that this result can be extended to any $k = n - \ell$ for any positive constant $\ell \geq 2$. However, a very interesting question is to determine whether the problem is FPT when parameterized by $\ell$ or not.

We then studied SPANNING TREE WITH AT MOST $k$ LEAVES, where each spanning tree has at most $k$ leaves, for a positive integer $k \geq 2$. We prove that the problem is PSPACE-complete for every $k \geq 3$. However, the reduction does not preserve any graph parameter. Hence, one might be interested to determine the complexity in restricted graph classes. Finally, note that if $k = 2$, then SPANNING TREE WITH AT MOST $k$ LEAVES is equivalent to HAMILTONIAN PATH

RECONFIGURATION. We were not able to determine the complexity of this problem, and we leave it as an open problem.

## 4.2 Distributed recoloring

In this last section, we again change the host problem since we focus on recoloring. However, unlike the other chapters of this thesis, we study reconfiguration in the so-called LOCAL model used in Distributed Computing. We mainly study distributed recoloring in the case where the input graph is a tree or a subcubic graph. This is joint work with Marthe Bonamy, Mikaël Rabie, Jukka Suomela and Jara Uitto [BOR+18].

### 4.2.1 Introduction

**$k$- recoloring.** Before moving on to the definition of the LOCAL model, let us discuss about $k$-recoloring in a centralized setting. Let $G = (V, E)$ be a graph, and let $k$ be an integer. Recall that a $k$-coloring of $G$ is a function $f : V \mapsto \{1, 2, \dots, k\}$ such that for any two adjacent vertices $u$ and $v$, $f(u) \neq f(v)$. One can define the $k$-recoloring graph $\mathcal{R}_k(G)$ of $G$ in a similar way as discussed in Section 1.5.2. The vertices of $\mathcal{R}_k(G)$ are the $k$-colorings of $G$, and two vertices are adjacent if one can go from one to the other by applying the fixed reconfiguration rule. As for other reconfiguration problems, there is not a unique choice that allows us to modify a $k$-coloring into another one. Actually, two main models have been considered so far. The first one is the *single-vertex recoloring* where one can recolor a vertex with a new color that does not appear in its neighborhood. The second involves Kempe chains, a concept introduced in 1879 by Alfred Kempe in his attempted proof of the Four Color Theorem [Kem79]. A *Kempe chain* is a connected component of the subgraph induced by two color classes of a $k$-coloring of $G$. Then, the second reconfiguration rule is a *Kempe change*; it consists in permuting the colors of a Kempe chain. Though this proof was fallacious, the Kempe change technique has proved useful in, for example, the proof of the Five Color Theorem and a short proof of Brooks' theorem. Note that Kempe change recoloring is a generalization of single-vertex recoloring since the latter corresponds to a Kempe change where the Kempe chain is reduced to a single vertex.

The usual questions considered in $k$-RECOLORING are identical to the ones presented is Section 1.5.2: given a graph $G$ and a positive integer $k$, is $\mathcal{R}_k(G)$ connected? If so, what is its diameter? All of those questions can also be asked for two specific $k$-colorings $s$ and $t$ of $G$. Is there a path between $s$ and $t$ in $\mathcal{R}_k(G)$? If so, what is the length of a shortest transformation between $s$ and $t$? What is the complexity of deciding that?

The reachability question of $k$-RECOLORING is typically a PSPACE-complete problem for both single-vertex recoloring [BC09] and Kempe changes [BHI+19]. The related question of deciding equivalence when a bound $\ell$ on the length of a recoloring sequence is given as part of the input has also been considered under single-vertex coloring by Bonsma et al. [BMNR14]. They proved that the problem is W[1]-hard when parameterized by $k$ (but in XP), but FPT with respect to $k + \ell$. For the special case of trees, we know that the maximum number of operations needed to go from one 3-coloring to another is $\Theta(n)$ [CVDHJ11]. They also proved that the problem is polynomial-time solvable for any graph $G$, and there exist instances that require $\Omega(n^2)$ recoloring steps, where $n$ is the number of vertices of the input graph. While $(\Delta + 1)$-recoloring a graph with no node of degree more than $\Delta$ is not always possible, having $\Delta + 2$ colors always suffices [Jer95], and there are also meaningful results to obtain for the problem of $(\Delta + 1)$-recoloring [FJP16]. Two other settings have received special attention: characterizing fully when 3-recoloring is possible [CVDHJ11, CVdHJ09], and guaranteeing short reconfiguration sequences in the case of sparse graphs for various notions of sparse [BB18, BP16].

Let $G$ be a graph and $k$ be an integer. We say that $G$ is $k$-Kempe mixing if one can always transform any two $k$-colorings of $G$ via Kempe changes. Note that if one can always transform any two $k$-colorings of $G$ via single-vertex recoloring, then $G$ is $k$-Kempe mixing. However, the converse might not be true. Indeed, any bipartite graph is $k$-Kempe mixing for any $k \geq 2$ (see, e.g., [Moh07]) but for any $k \geq 2$, there exist bipartite graphs for which it is not always possible to transform a $k$-coloring into another one via single-vertex recoloring [CvdHJ08]. Mohar conjectured in [Moh07] that for any $k \geq 3$ and for any $k$-regular graph $G$ which is not the complete graph, then $G$ is $k$-Kempe mixing. However, van den Heuvel [vdH13] first observed that for $k = 3$, the 3-prism is not 3-Kempe mixing (see Figure 4.8):

Figure 4.8 – These two 3-colorings of the 3-prism cannot be transformed into each other by Kempe changes.

Feghali et al. [FJP17] showed that for $k = 3$, the 3-prism is the only counterexample to the conjecture of Mohar. In other words, they proved that any 3-regular graph $G$ which is neither $K_4$ nor the 3-prism graph is 3-Kempe mixing. Finally, Bonamy et al. [BBFJ19] proved that the conjecture is true whenever $k \geq 4$. While we will not discuss Kempe recoloring in our work, we point out that recoloring with extra colors is closely connected to Kempe recoloring: Kempe recolorability implies recolorability with one extra color (while the converse is not true). Hence the negative results related to one extra color also hold for Kempe recoloring.

Finally, note that some other variants have also been studied, perhaps most notably the question of how many nodes to recolor at once so that the graph can be recolored [McD15].

**The LOCAL model.** Let us now introduce the LOCAL model used in Distributed Computing we are interested in. For a more complete overview, the reader is referred to the book of David Peleg [Pel00], or the survey by Jukka Suomela [Suo13]. Distributed Computing is rather different than "centralized" graphs problems which are problems we have studied so far. Indeed, in this field, we are given some entities which might be computers, robots, mobile agents and so on that evolve in the same environment. Their goal is to cooperate together in order to achieve a global task, without central control. Because the definition is vague, there are many different models. Some of them work in an asynchronous way, meaning that the different entities may not have the same notion of time or can process tasks at different speeds. Others work synchronously meaning that there exists a central clock used to synchronized all the entities. In the remaining of this section, we assume that each entity corresponds to a processor, which is able to perform some computation.

Research on local algorithm was pioneered by Dana Angluin [Ang80] in her seminal paper "Local and global properties in networks of processors". Angluin explained that this model "attempts to get at some of the fundamental properties of distributed computing by means of the following question: how much does each processor in a network of processors need to know about its own identity, the identities of other processors, and the underlying connection network in order for the network to be able to carry out useful functions?". It is observed that the "underlying connection network" can be modeled as a (connected) graph. More precisely, each processor corresponds to a node and we add an edge between two nodes if the corresponding processors can communicate, i.e., there exists a communication link between them. Moreover, in Angluin's model, it is required that each node initially does not have any knowledge about

the global network. It is also required that all the nodes with the same degree are considered to be identical, i.e., the network is anonymous, without any unique identifier on its nodes. In order to achieve the global task, a basic computation step consists in an exchange of messages by two adjacent nodes. Thanks to the information shared, the two nodes can decide to change their internal state accordingly, without affecting the other nodes. Angluin [Ang80] studied the limitations of her model: she proved that there is no deterministic algorithm that elects a *leader* (i.e., all the nodes have to agree to distinguish a single node) within finite time on a cycle. In a sense, this means that there is now way to break symmetry.

Actually, most functions cannot be carried out by anonymous networks, even for very simple topologies like cycles as discussed above. This impossibility usually results from symmetries that the network may have (for instance, a cycle is a connected 2-regular graph). These symmetries can be broken by means of randomization or by using unique identifiers on the nodes. In the latter, we may assume that each node in the network $G = (V, E)$ receives a unique identifier in the range $[1, |V|]$. Nathan Linial [Lin92] focused on the following question: how can algorithmic graphs problems can be solved in a distributed fashion? In other words, how much knowledge of the network does a node need to have in order to take a decision? By taking a decision, we mean deciding if it has to be in the solution (e.g., a dominating set) or choosing its color if the goal is to find a proper $k$-coloring of the network. For that purpose, Linial [Lin92] introduced the so-called LOCAL model. Let $G = (V, E)$ be a graph on $n$ nodes where each node has a unique identifier between 1 and $n$. Initially, each node only knows its identifier. This model works synchronously: there is global clock that guarantees that all nodes take steps simultaneously in parallel. Each step is called a *round*. At each round, a node can:

- send a message to its neighbors;
- receive messages from its neighbors;
- do a local computation.

Because we are interested in the locality of the problem, i.e., "to what extent a global solution to a computational problem can be obtained from locally available data" in the words of Linial, we assume that (i) we do not have any restriction on the size of the messages (unlike what happens in the CONGEST model); and (ii) each processor has an unbounded computing power. Thus, the complexity of a distributed algorithm (in the LOCAL model) is the maximum number of rounds needed so that each node can output its final state. Note that for any (connected) graph $G$ on $n$ nodes and with diameter $D$, it is possible to solve any decidable problem in $O(D)$ rounds. Indeed, since we do not have any restriction on the size of the messages, in $t$ rounds each node $u$ will have full knowledge of the ball of radius $r$ centered in $u$. In particular, in $D$ rounds, each node will have full knowledge of $G$ (the topology as well as the identifier of each node). Now, one specific node (e.g., the one with the smallest identifier) can compute a solution (in one round) and then broadcast it to each node of the graph, with $D$ additional rounds. Hence, the major question raised by Linial [Lin92] is to determine which (decidable) problems can be solved faster than that.

**Coloring paths.** Let us now give a simple example of a distributed algorithm in the LOCAL model, taken from [HS20]. We are given the path on $n$ vertices $P_n$ where each vertex receives a unique identifier, which might be larger than $n$. The goal is to find a proper 3-coloring of $P_n$. Hence, for each node $u \in P_n$, we want to output a variable $c(u)$ (with $c(u) \in \{1, 2, 3\}$) which corresponds to the final color of node $u$. Note that if we do not have identifiers, the problem cannot be solved deterministically within finite time, even if $n = 2$. Hence, we will make intense use of these identifiers in order to break symmetry. For each node $u \in P_n$, we

denote by $id(u)$ the identifier of $u$. First, observe that the identifiers induce a proper coloring of $P_n$; thus we initially set $c(u) = id(u)$ for any node $u$. Hence, the only thing that we have to do is to reduce the total number of colors so that $c(u) = \{1, 2, 3\}$ holds for any node $u$. The strategy is very simple: at each step, if a node $u$ has the maximum color among all the colors that appear in its neighborhood, $u$ changes its color to the smallest one which is available.

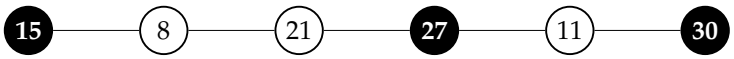



Figure 4.9 – Nodes with identifiers 15, 27 and 30 will change their color at first.

Note that the new color of $u$ belongs to $\{1, 2, 3\}$ since $u$ has at most two neighbors. Moreover, no two adjacent nodes will change their color at the same time. See Figure 4.10 below for an example.

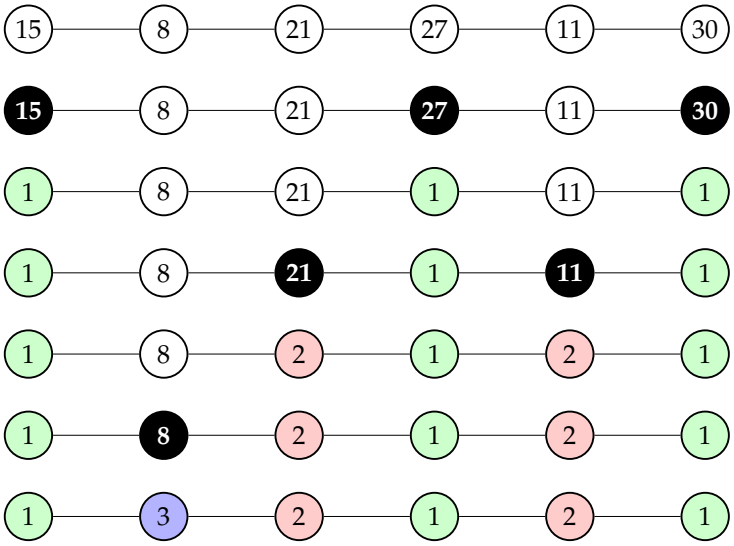



Figure 4.10 – Execution of the algorithm on a $P_6$.

It is observed that once a node has a color in $\{1, 2, 3\}$, it can output it as its final color and inform all of its neighbors. In particular, it will not have to send messages to its neighbors anymore, as they are all aware of the final color. So each node $u$ will run the following algorithm:

**Algorithm 5** 3-coloring of a path (algorithm for node $u$)

1: $F \leftarrow \emptyset$
2: **while** $c(u) \notin \{1, 2, 3\}$ **do**
3: Send "$c(u)$" to all neighbors
4: Receive messages from all neighbors; let $M$ be the set of messages.
5: **if** a node $v$ sent its final color to $u$ **then**
6: $F \leftarrow F \cup \{c(v)\}$
7: **if** $c(u) \notin \{1, 2, 3\}$ and $c(u) > \max\{M \cup F\}$ **then**
8: $c(u) \leftarrow \min\{\{1, 2, 3\} \setminus (M \cup F)\}$
9: Send "final color $c(u)$"
10: Output $c(u)$

More precisely, at each round, the node $u$ will execute once the lines 2 – 8 as long as it has not reached its final color. When this will be the case, it will send a last specific message

to all of its neighbors to inform them about it (line 9) and will output its final color (line 10). For this reason, the set $F$ contains all the final colors of the neighbors of $u$; note that it is not required to know to which nodes they correspond. Finally, we assume that all the nodes send their messages simultaneously. In particular, if a node $u$ satisfies $id(u) \in \{1,2,3\}$, it will execute line 9 during the first round, and thus all of its neighbors will be aware during the first round as well, thanks to line 4.

The problem will be solved once a 3-coloring of the whole path has been found, i.e., once each node has its color from the set $\{1,2,3\}$. The overall complexity of this algorithm corresponds to the maximum number of rounds needed for each node to reach its final color. With this definition, one can observe that Algorithm 5 is not very efficient in the worst case. Indeed, if all the identifiers are ordered in an increasing order, then the complexity of the algorithm is $\Theta(n)$. However, note that one can do it much faster. For instance, one can compute a 3-coloring of a tree with $n$ nodes in time $O(\log n)$ [MR89].

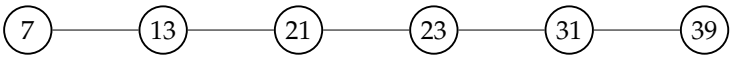



Figure 4.11 – $\Theta(n)$ rounds are needed for this example.

**Distributed coloring.** The literature on the standard distributed coloring is vast; most of the work done so far has focused on coloring graphs of maximum degree $\Delta$ with at most $\Delta+1$ colors. The best overview on the topic is the book by Barenboim and Elkin [BE13]; some recent developments include the following results. Rozhoň et al. [RG20] found recently a randomized algorithm that computes a $(\Delta+1)$-coloring in $\text{poly}(\log\log n)$ rounds. If we restrict to trees, the number of colors can be reduced to $\Delta$ with the cost of increasing the running time to $O(\log_\Delta \log n)$ [CKP16]. On the deterministic side, the best known $(\Delta+1)$-coloring algorithm requires $O(\log^6 n)$ communication rounds [GGR20]. In the case of trees, the *rake-and-compress* method by Miller and Reif gives a 3-coloring in time $O(\log n)$ [MR89] as we said before. However, it is well-known that each graph which is neither an odd cycle nor a complete graph can be colored with at most $\Delta$ colors by Brooks' theorem [Bro41]. Panconesi and Srinivasan [PS95] initiated research on finding fast distributed algorithms to color such graphs with at most $\Delta$ colors. They gave an $O(\log^3 n/\log\Delta)$ deterministic algorithm that uses Kempe changes to find a $\Delta$-coloring. Ghaffari et al. [GHKM18] found an $O(\log\Delta)+2^{O(\sqrt{\log\log n})}$-time randomized algorithm when $\Delta \geq 4$, or in time $O((\log\log n)^2)$ if $\Delta$ is a constant. Aboulker et al. [ABBE18] tackled distributed coloring with a rather different perspective: instead of finding fast algorithm for $\Delta+1$ coloring, they focused on finding coloring with the fewest number of colors in a polylogarithmic number of rounds. A typical example is the case of planar graphs for which a 4-coloring always exists and can be found in quadratic time (in the centralized setting). Goldberg, Plotkin, and Shannon [GPS88] obtained a deterministic distributed algorithm that finds a 7-coloring of a planar graph in $O(\log n)$ rounds. Recently, Aboulker et al. [ABBE18] found an $O(\log^3 n)$-time deterministic algorithm to color a planar graphs with at most six colors. They also showed that no distributed algorithm can color a planar graph with four colors in $o(n)$ rounds. Finally, Linial proved a remarkable theorem in his seminal paper: a 3-coloring of the cycle on $n$ vertices requires $\Omega(\log^* n)$ rounds [Lin92], where $\log^* n$ is the number of times the logarithm function must be iteratively applied before the result is less than or equal to one. This bound is tight by a result of Cole and Vishkin [CV86]. Linial also proved that any algorithm for coloring the $d$-regular tree of radius $r$ which runs in time at most $2r/3$ requires at least $\Omega(\sqrt{d})$ colors [Lin92].

### 4.2.2 Definition of the problem

**Formalization.** Recall that deciding whether a $k$-coloring of a graph $G$ can be transformed into another one is PSPACE-complete for both Kempe changes [BHI+19] and single-node recoloring [BC09]. Thus, this problem is typically inherently *global* and solutions (when they exist) are long, i.e., the length of a transformation between the two $k$-colorings is large in the worst case.

We are now ready to introduce recoloring problems in a *distributed* setting; note that no prior work studied reconfiguration in a distributed way to the best of our knowledge. We will show that there are natural relaxations of the problem that are attractive from the perspective of distributed graph algorithms: they admit solutions that are short and that can be found *locally* (e.g., in sublinear number of rounds). Distributed recoloring problems are closely related to classical symmetry-breaking problems that have been extensively studied in the area of distributed graph algorithms, but as we will see, they also introduce new kinds of challenges.

We will work in the LOCAL model of Distributed Computing presented in Section 4.2.1. In particular, computation proceeds in synchronous rounds: each node sends a message to each of its neighbors, receives a message from each of its neighbors, and updates its local state. Eventually, all nodes have to announce their local outputs and stop; the running time of the algorithm is the number of communication rounds until all nodes stop. We assume that the algorithm is deterministic, and each node is labeled with a unique identifier. Initially, each node is only aware of its identifier and has no knowledge of the graph.

In *distributed recoloring*, each node $v \in V$ is given two colors, an *input color* $s(v)$ and a *target color* $t(v)$. It is guaranteed that both $s$ and $t$ form a proper coloring of $G$, that is, $s(u) \neq s(v)$ and $t(u) \neq t(v)$ for every $uv \in E$. Each node $v \in V$ has to output a finite *recoloring schedule* $x(v) = \langle x_0(v), x_1(v), \ldots, x_\ell(v) \rangle$ for some $\ell = \ell(v)$. For convenience, we define $x_i(v) = x_\ell(v)$ for $i > \ell(v)$. We say that the node *changes its color at time* $i > 0$ if $x_{i-1}(v) \neq x_i(v)$; let $C_i$ be the set of nodes that change their color at time $i$. Define $L = \max_v \ell(v)$; we call $L$ the *length* of the solution. A solution is feasible if the following holds (see Figure 4.12 for a simple example of distributed recoloring steps):

1. $x_0 = s$ and $x_L = t$;
2. $x_i$ is a proper coloring of $G$ for all $i$;
3. $C_i$ is an independent set of $G$ for all $i$.

The key differences between distributed recoloring and classical recoloring are:

1. input and output are given in a distributed manner: no node knows everything about $G$, $s$, and $t$, and no node needs to know everything about $x_i$ or the length of the solution $L$;
2. we do not require that only one node changes its color; it is sufficient that adjacent nodes do not change their colors simultaneously.

Note that a solution to distributed recoloring is locally checkable in the following sense: to check that a solution is feasible, it is enough to check independently for each edge $uv \in E$ that the recoloring sequences $x(u)$ and $x(v)$ are compatible with each other, and for each node $v \in V$ that $x(v)$ agrees with $s(v)$ and $t(v)$. However, distributed recoloring is not necessarily an LCL problem (see [NS95] for a definition) in the formal sense, as the length of the output per node is not a priori bounded. We emphasize that we keep the following aspects well-separated: what is the complexity of *finding* the schedule, and how *long* the schedules are. Hence it makes sense to ask, e.g., if it is possible to find a schedule of length $O(1)$ in $O(\log n)$ rounds (note that the physical reconfiguration of the color of the node may be much slower than communication and computation).

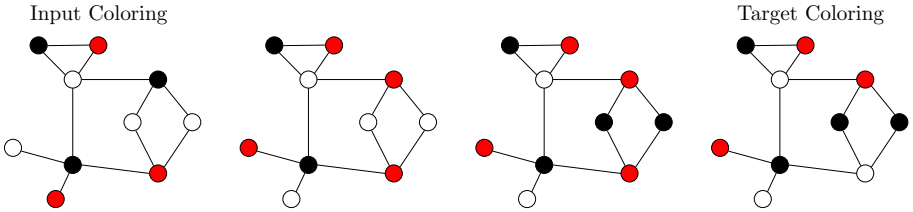



Figure 4.12 – Distributed recoloring: the input coloring $s$ can be seen on the left and the target coloring $t$ on the very right. The illustration shows one possible way to reach the target coloring in three steps by, in each step, changing the colors of an independent set of nodes.

**Recoloring with extra colors.** Recoloring is computationally very hard, as solutions do not always exist, and deciding whether a solution exists is PSPACE-hard. It is in a sense analogous to problems such as finding an *optimal* node coloring of a given graph; such problems are not particularly interesting in the LOCAL model, as the complexity is trivially global. To make the problem much more interesting we slightly relax it. More precisely, we will allow extra colors, i.e., colors that do not appear neither in the source nor in the target $k$-coloring. More precisely, we define a $k + c$ *recoloring problem* (a.k.a. *$k$-recoloring with $c$ extra colors*) as follows:

- We are given colorings with $s(v), t(v) \in [k]$.
- All intermediate solutions must satisfy $x_i(v) \in [k + c]$.

Here we use the notation $[n] = \{1, 2, \ldots, n\}$.

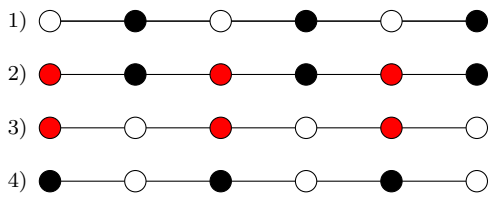



Figure 4.13 – In the input graph, a bipartition is given. Therefore, the target coloring can be reached by using one extra color in three steps.

The problem of $k + c$ recoloring is meaningful also beyond the specific setting of distributed recoloring. For example, here is an example of a very simple observation:

**Lemma 4.24.** *Recoloring with one extra color is always possible in any bipartite graph, with a distributed schedule of length $L = 3$.*

*Proof.* Let the bipartition be $V = V_1 \cup V_2$. First each node $v \in V_1$ switches to $k + 1$, then each $v \in V_2$ switches to color $t(v)$, and finally each $v \in V_1$ switches to color $t(v)$. □

Incidentally, it is easy to extend this result to show that $k$-recoloring with $c = \chi - 1$ extra colors is always possible with a schedule of length $O(c)$ in a graph with chromatic number $\chi$, and in particular $k$-recoloring with $c = k - 1$ extra colors is trivial. Figure 4.13 gives an illustration of recoloring a bipartite graph with one extra color. As a corollary, we can solve distributed $k + 1$ recoloring in trees in $O(n)$ rounds, with a schedule of length $O(1)$: simply find a bipartition of the tree and then apply the above lemma. However, is this optimal? Clearly finding a bipartition in a tree requires $\Omega(n)$ rounds, but can we solve recoloring with one extra color strictly faster?

These are examples of problems that we will study in this section. We initiate the study of distributed complexity of recoloring, with the ultimate objective of finding a complete characterization of graph families and parameters $k$, $c$, and $L$ such that distributed $k + c$ recoloring with schedules of length $L$ can be solved efficiently in a distributed setting. As we will see, the problem turns out to be surprisingly rich already in very restricted settings such as subcubic graphs or 3-regular trees. Many of the standard lower bound techniques fail; in particular, known results on the hardness of graph coloring do not help here, as we are already given two proper colorings of the input graph.

Let us end this introduction on distributed recoloring by an observation related to a variant that we call *weak recoloring*: if two adjacent nodes $u$ and $v$ change their color simultaneously at time $i$, then $\{x_{i-1}(u), x_i(u)\} \cap \{x_{i-1}(v), x_i(v)\} = \emptyset$, that is, we have a proper coloring regardless of whether $u$ or $v$ changes its color first. Let us now contrast weak recoloring with *strong recoloring*, in which adjacent nodes never change colors simultaneously. Trivially, strong recoloring solves weak recoloring. But the converse is also true up to constant factors: if we have $k$ input colors and a solution to weak recoloring of length $L$, then we can also find a solution to strong recoloring of length $kL$. To see this, we can implement one weak recoloring step in $k$ strong recoloring substeps such that in substep $j$ nodes of input color $j$ change their colors. As our focus is on the case of a small number of input colors, we can equally well study strong or weak recoloring here; all of our results hold for either of them. However, we present our results using strong recoloring, as it has a more convenient definition.

### 4.2.3 Warmup – simple results

We start by presenting a number of simpler upper and lower bounds that also serve as an introduction to the topic of distributed recoloring.

#### Upper bounds

**Lemma 4.25.** *In any graph, $k + c$ recoloring for $c = k - 1$ is possible in 0 communication rounds, with a schedule of length $O(k)$.*

*Proof.* Generalize the idea of Lemma 4.24; note that the schedule of node $v$ depends only on $s(v)$ and $t(v)$, and not on the colors of any other node around it. □

**Lemma 4.26.** *In paths and trees, 3-recoloring is possible in $O(n)$ rounds, with a schedule of length $O(n)$.*

*Proof.* Every node has full knowledge of the graph. The statement can be intuited by induction on the size of the tree, but we delay a formal proof to Section 4.2.4 and more precisely Lemma 4.38. □

**Lemma 4.27.** *In cycles and paths, $3 + 1$ recoloring is possible in $O(1)$ rounds, with a schedule of length $O(1)$.*

*Proof.* Use the input coloring to find a maximal stable set $I$. Nodes of $I$ switch to color 4. Nodes of $V \setminus I$ induce paths of length $O(1)$, apply Lemma 4.26 there to recolor each of the paths by brute force, without using the extra color 4. Finally, nodes of $I$ switch to their target colors. □

**Lemma 4.28.** *Let $G$ be a graph of maximum degree at most $\Delta$, and let $k \geq \Delta + 2$. Then $k$-recoloring with $c$ extra colors is at least as easy as $(k - 1)$-recoloring with $c + 1$ extra colors.*

*Proof.* Given a $k$-coloring $x$, we can construct a $(k-1)$-coloring $x'$ as follows: all nodes of color $k$ pick a new color from $\{1, 2, \ldots, k-1\}$ that is not used by any of their neighbors. Note that $x \rightsquigarrow x'$ is a valid step in distributed recoloring (nodes of color $k$ form an independent set), and by reversing the time, also $x' \rightsquigarrow x$ is a valid step. Hence to recolor $s \rightsquigarrow t$ with $c$ extra colors, it is sufficient to recolor $s' \rightsquigarrow t'$ with $c+1$ extra colors (color $k$ no longer appears in the input and target colorings and can be used as an auxiliary color during recoloring). Then we can put everything together to form a recoloring schedule $s \rightsquigarrow s' \rightsquigarrow t' \rightsquigarrow t$, with only constant overhead in the running time and schedule length. □

**Lemma 4.29.** *In subcubic graphs, $4+1$ recoloring is possible in $O(1)$ rounds, with a schedule of length $O(1)$.*

*Proof.* Use the input coloring to find a maximal independent set $I$ in constant time. Nodes of $I$ switch to color 5. Delete $I$; we are left with a graph $G'$ that consists of paths and isolated nodes. Apply Lemmas 4.28 and 4.27 to solve $4+0$ recoloring in each connected component of $G'$. Finally nodes of $I$ can switch to their target colors. □

### Lower bounds

**Lemma 4.30.** *Recoloring without any extra colors is not possible in the following settings for some pairs of input and target colorings:*

*(a)* 2*-recoloring paths or trees.*
*(b)* 2*-recoloring cycles.*
*(c)* 3*-recoloring cycles.*
*(d)* 2*-recoloring cubic graphs.*
*(e)* 3*-recoloring cubic graphs.*
*(f)* 4*-recoloring cubic graphs.*

*Proof.* We can construct a source coloring in which no node can make a move, and a target coloring different from the input coloring. Here we show examples of the source coloring $s$; the target coloring can be constructed by $t(v) \equiv s(v) + 1 \bmod k$:

(a) A path with 2 nodes, $s = \begin{bmatrix} 1 & 2 \end{bmatrix}$.

(b) A 4-cycle, $s = \begin{bmatrix} 1 & 2 & 1 & 2 \end{bmatrix}$.

(c) A 4-cycle, $s = \begin{bmatrix} 1 & 2 & 3 & 2 \end{bmatrix}$.

(d) Complete bipartite graph $K_{3,3}$, with $s$ constructed from the bipartition.

(e) Prism graph: connect the nodes of a 3-cycle colored with $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ to another 3-cycle colored with $\begin{bmatrix} 2 & 3 & 1 \end{bmatrix}$, in this order.

(f) Complete graph $K_4$. □

**Lemma 4.31.** *In paths and trees, 3-recoloring without extra colors requires $\Omega(n)$ rounds and produces schedules of length $\Omega(n)$ in the worst case. This holds also in the case of 3-regular trees.*

*Proof.* Consider a long path with the input coloring $1, 2, 3, 1, 2, 3, \ldots, 1, 2, 3$ and observe that a node of degree 2 can change its color only after at least one neighbor has changed colors. We can embed such a path also in a 3-regular tree. □

**Lemma 4.32.** *In trees, 4-recoloring without extra colors requires $\Omega(\log n)$ time and produces schedules of length $\Omega(\log n)$ in the worst case.*

*Proof.* It is sufficient to construct a 3-regular tree in which each node is surrounded by nodes of all other colors: color the root with color 1 and its neighbors with colors 2, 3, and 4. Then recursively for each leaf node of color $x$ that is already adjacent to a node of color $y$, add two new neighbors with the colors $\{1,2,3,4\} \setminus \{x,y\}$, etc., and continue in a balanced manner such that the distance between the root and the nearest leaf is logarithmic. Now a non-leaf node can change its color only once its neighbor has changed its color. □

### 4.2.4 Recoloring algorithm for trees

In this section, we provide two efficient algorithms for recoloring and list-recoloring trees; the list-coloring problem will be defined later.

***k*-recoloring.** We first consider node recoloring; we will then focus on list recoloring. Our main result regarding node recoloring on trees is the following:

**Theorem 4.33.** *For any $k \in \mathbb{N}$, for every tree $T$ on $n$ nodes, for any two $k$-colorings $\alpha, \beta$ of $T$, we can compute in $O(\log n)$ rounds how to recolor $T$ from $\alpha$ to $\beta$ with one extra color and a schedule of length $O(1)$.*

We first discuss how to compute efficiently an independent set with some desirable properties. For this, we use a simple modification of the *rake and compress* method by Reif and Miller [MR89]. More precisely, we iterate rake and compress operations, and label nodes based on the step at which they are reached. We then use the labels to compute an independent set satisfying given properties. We finally explain how to make use of the special independent set to obtain an efficient recoloring algorithm, in each case.

**Definition 4.34.** *A* light $h$-labeling *is a labeling $V \to [h]$ such that for any $i \in [h]$:*

1. *Any node labeled $i$ has at most two neighbors with label $\geq i$, at most one of which with label $\geq i+1$.*
2. *No two adjacent nodes labeled $i$ both have a neighbor with label $\geq i+1$.*

**Lemma 4.35.** *There is an $O(\log n)$-round algorithm that finds a light $(4\log n)$-labeling of a tree.*

*Proof.* As discussed above, we merely use a small variant of the *rake and compress* method. At step $i$, we remove all nodes of degree one and all nodes of degree two that belong to a chain of at least three nodes of degree two, and assign them label $i$.

One can check that this yields a light labeling. It remains to discuss how many different labels are used, i.e., how many steps it takes to delete the whole tree. Let us argue that no node remains after $4\log n$ rounds. Let $T$ be a tree, let $V_1$ (resp. $V_2$, $V_3$) be the number of nodes of degree one (resp. two, at least three) in the tree, and let $T'$ be the tree obtained from $T$ by replacing any maximal path of nodes of degree two with an edge. Note that $|V(T')| = |V_1| + |V_3|$. Let $W$ be the set of nodes in $T$ that have degree two with both neighbors of degree two. Note that $|V_2 \setminus W| \leq 2|E(T')| = 2(|V_1| + |V_3| - 1)$. Note also that $|V_1| \geq |V_3|$, simply by the fact that there are fewer edges than nodes in a tree. It follows that $|W| \geq |V_2| - 2(|V_1| + |V_3| - 1) = |V(T)| - |V_1| - |V_3| - 2(|V_1| + |V_3| - 1) \geq |V(T)| - 6|V_1|$. Consequently, we obtain

$|W| + |V_1| \geq \frac{|V|}{6}$. In other words, at every step, we remove in particular $W \cup V_1$, hence at least a sixth of the nodes. It follows that at after $k$ steps, the number of remaining nodes is at most $n \cdot \left(\frac{5}{6}\right)^k$. Note that this is less than one once $k \geq 4 \log n$. □

We now discuss how to make use of light $h$-labelings.

**Lemma 4.36.** *For any graph $T$, any 3-coloring $\alpha$ of $T$, and any integer $h$, let $L$ be a light $h$-labeling of $T$. There is an $O(h)$-round algorithm that finds a maximal independent set $S$ such that $T \setminus S$ only has connected components on one or two nodes.*

*Proof.* In brief, we proceed as follows: at step $i = h, h-1, \ldots, 1$, we first add all nodes of label $i$ which have a neighbor of label $\geq i+1$ that is not in $S$ (they form an independent set by definition of a light label), then use the 3-coloring to obtain a fast greedy algorithm to make $S$ maximal on the nodes of label $\geq i$. The algorithm is the following.

**Algorithm 6** DECOMPOSING INTO AN INDEPENDENT SET AND COMPONENTS OF SIZE $\leq 2$

**Require:** A tree $T$, a 3-coloring $\alpha$ and a light $h$-label of $T$.
**Ensure:** A set $S$ of $V(T)$ such that $G[S]$ is an independent set and every connected component of $G[V \setminus S]$ has size at most 2.

```
for i from h down to 1 do
    for u with label i (in parallel) do
        If u has a neighbor of higher label that is not in S, add u to S
    for j from 1 to 3 do
        for u with label i and color j (in parallel) do
            If N(u) ∩ S = ∅, add u to S
```

The fact that the output $S$ is an independent set follows directly from the construction, as does the fact that the running time in $O(h)$ rounds. We note that no connected component of $T \setminus S$ contains nodes of different labels, due to the first operation at step $i$.

It remains to argue that for any $i$, the nodes of label $i$ that do not belong to $S$ only form connected components of size one or two. Assume for a contradiction that there is a node $u$ of label $i$ which has two neighbors $v$ and $w$, also of label $i$, such that none of $\{u, v, w\}$ belongs to $S$. By definition of a light label, the node $u$ has no other neighbor of label $\geq i$, a contradiction to the fact that we build $S$ to be an MIS among the nodes of label $\geq i$. □

Combining Lemmas 4.35 and 4.36, and observing that a 3-coloring of a tree can be obtained in $O(\log n)$ rounds, we immediately obtain the following.

**Lemma 4.37.** *There is an $O(\log n)$-round algorithm that finds an MIS in a tree, such that every component induced by non-MIS nodes is of size one or two.*

We are now ready to prove Theorem 4.33.

*Proof of Theorem 4.33.* First, we use Lemma 4.37 to obtain in $O(\log n)$ rounds a maximal independent set $S$ such that $T \setminus S$ only has connected components of size one or two. We recolor each node in $S$ with the extra color. Remove $S$, and recolor each component from $\alpha$ to $\beta$ without using any extra colors; this can be done in $O(1)$ recoloring rounds. Each node in $S$ can then go directly to its color in $\beta$. □

**List-recoloring.** We know focus on list-recoloring. Before moving on to our results, we first need to define what is a list coloring. Given a graph $G = (V, E)$, a *list-assignment* is a function which assigns to each node $v \in V$ a list of colors $L(v)$. An *L-coloring* of $G$ is a function $c$ that assigns to each node $v \in V$ a color $c(v) \in L(v)$ such that for any two adjacent nodes $u, v \in V$, we have $c(u) \neq c(v)$. A graph $G$ is *k-list-colorable* if it admits an *L-coloring* for every list-assignment where each node has a list of size at least $k$. It is observed that list-coloring generalizes node-coloring if we consider the special case where each node receives the same input list. The notion of *L-recoloring* is a natural generalization of $k$-recoloring: the same elementary steps are considered, and every intermediate coloring must be an $L$-coloring of $G$. However, we have to use a more convoluted approach since there is no global extra color that we can use. Before discussing 4-list-recoloring, we discuss 3-list-recoloring. For the sake of intuition, we start by presenting an algorithm for 3-recoloring trees, and explain afterwards how to adapt it for the list setting.

**Lemma 4.38.** *For every tree $T$ with radius at most $p$ and for any two 3-colorings $\alpha, \beta$ of $T$, we can compute in $O(p)$ rounds how to 3-recolor $T$ from $\alpha$ to $\beta$ with a schedule of length $O(p)$.*

*Proof.* Let $c\colon V \to [3]$ be a 3-coloring of $T$. We introduce an identification operation: Given a leaf $u$ and a node $v$ such that $u$ and $v$ have a common neighbor $w$, we recolor $u$ with $c(v)$, and from then on we pretend that $u$ and $v$ are a single node. In other words, we delete $u$ from the tree we are considering, and reflect any recoloring of $v$ to the node $u$. Note that these operations can stack up: the recoloring of a single node might be reflected on an arbitrarily large independent set in the initial tree.

We now briefly describe an algorithm to recolor a 3-coloring into a 2-coloring $c'$ in $O(p)$ rounds, with schedule $O(p)$. First, root $T$ on a node $r$ which is at distance at most $p$ of any node of $T$. Any node of $T$ which is not adjacent to the root has a *grandparent*, which is defined as its parent's parent. Then, at each step, we consider the set $A$ of leaves of $T$ which have a grandparent, if any. We identify each leaf in $A$ with its grandparent (note that the notion of grandparent guarantees that this operation is well-defined, and that the operation results in $A$ being deleted).

This process stops when $T$ consists only of the root $r$ and its children. We select one of the children arbitrarily and identify the others with it. This results in $T$ being a single edge. Note that the color partition of $c'$ is compatible with the identification operations, as we only ever identify nodes at even distance of each other. We then recolor $T$ into $c'$: this is straightforward in the realm of 3-recoloring.

We can now choose a 2-coloring of $T$ (this can be done in $O(p)$ rounds), and apply the above algorithm to 3-recolor both $\alpha$ and $\beta$ to that 2-coloring. This results in a 3-recoloring between $\alpha$ and $\beta$ with schedule $O(p)$. □

The same idea can be adapted to list coloring:

**Lemma 4.39.** *For every tree $T$ with radius at most $p$, for any list assignment $L$ of at least 3 colors to each node, for any two $L$-colorings $\alpha, \beta$ of $T$, we can compute in $O(p)$ rounds how to $L$-recolor $T$ from $\alpha$ to $\beta$ with schedule $O(p)$.*

*Proof.* We adapt the identification operation introduced in the proof of Lemma 4.38, merely by adapting the notion of having the same color. Let $u$ and $v$ be two nodes with a common neighbor $w$. We say $u$ has the same color as $v$ *with respect to* $w$ in the following cases:

- if $L(u) \neq L(w)$, then $u$ is colored with the smallest element of $L(u) \setminus L(w)$;

- if $L(u) = L(w)$ and the color of $v$ belongs to $L(u)$, then $u$ is colored the same as $v$;
- if $L(u) = L(w)$ and the color of $v$ does not belong to $L(u)$, then $u$ is colored with the smallest element of $L(w)$ that differs from the color of $w$.

Therefore, when we identify a leaf $u$ with a node $v$ that has a common neighbor $w$ with $u$, we first assign to $u$ the same color as $v$ with respect to $w$, and from then on we pretend that $u$ and $v$ are a single node. In other words, any recoloring of $v$ is mirrored on $u$ so that at each step, the node $u$ has the same color as $v$ with respect to $w$. Note that in some cases it may be that the color of $u$ does not actually change when the color of $v$ does.

When the operations stack up, i.e., a node $u$ is identified with a node $v$ which is identified with a node $x$, we do not claim transitivity of the relation. In particular, $u$ and $x$ have no common neighbor, hence them having the same color is not well-defined. We merely enforce that $u$ has the same color as $v$ with respect to their common neighbor, and that $v$ has the same color as $x$ with respect to their common neighbor.

We insist on the fact that the definition of having the same color only depends on the list assignment. In particular, let us consider the situation once no more identification operation can be operated, i.e., the tree has been identified into an edge (see the proof of Lemma 4.38). The coloring of the edge characterizes entirely the coloring of the whole tree, regardless of the initial coloring. Therefore, we can pick an arbitrary $L$-coloring of the edge, and recolor both $\alpha$ and $\beta$ into the corresponding $L$-coloring of the tree in $O(p)$ rounds with schedule $O(p)$.

This results in computing in $O(p)$ rounds an $L$-recoloring between $\alpha$ and $\beta$ with a schedule of length $O(p)$. □

Finally, we need a last lemma to split the tree in small components. We slightly adapt the proof of Lemma 4.36 and obtain the following:

**Lemma 4.40.** *For any tree $T$, any 3-coloring $\alpha$ of $T$, and any integer $h$, let $L$ be a light $h$-label of $T$. There is a $O(h)$-round algorithm that finds a maximal independent set $S$ such that no node has two neighbors in $S$ and $T \setminus S$ only has connected components of radius $O(h)$.*

*Proof.* The algorithm is far simpler than Algorithm 6. We compute the set $R$ of nodes with no neighbor of higher label. We note that $T[R]$ is a collection of paths. We compute an independent set $S \subseteq R$ that is maximal subject to the property that no node in $R$ has two neighbors in $S$. Note that by definition of light label, no node outside of $R$ may have two neighbors in $R$ (hence in $S$). It remains to argue that $T \setminus S$ only has connected components of radius $O(h)$. We point out that every connected component of $T[R]$ contains an element of $S$. Therefore, any connected subset of nodes of $T[R]$ has at most one neighbor of higher label, since $T$ is a tree. Together with the fact that any connected component of $T[R \setminus S]$ has at most two nodes, we derive the conclusion. □

**Theorem 4.41.** *For every tree $T$ on $n$ nodes and any list assignment $L$ of at least four colors to every node of $T$, for any two $L$-colorings $\alpha, \beta$ of $T$, we can compute in $O(\log n)$ rounds how to $L$-recolor $T$ from $\alpha$ to $\beta$ with schedule of length $O(\log n)$.*

*Proof.* Compute in $O(\log n)$ rounds an independent set $S$ such any two elements of $S$ are at distance at least three of each other and every connected component of $T \setminus S$ has radius $O(\log n)$. By Lemmas 4.35 and 4.40 and the fact that a 3-coloring of a tree can be computed in $O(\log n)$ rounds [MR89], we compute in $O(\log n)$ rounds an $L$-coloring $\gamma$ of $T \setminus S$ such that every node adjacent to an element $u \in S$ has a color different from $\alpha(u)$ and $\beta(u)$. Note that this coloring exists since any tree is 2-list-colorable. Use Lemma 4.39 to recolor each connected component

of $T \setminus S$ from $\alpha$ to $\gamma$. Recolor every element of $S$ with its color in $\beta$. Use Lemma 4.39 to recolor each connected component $T \setminus S$ from $\gamma$ to $\beta$. Note that this yields an $L$-recoloring of $T$ from $\alpha$ to $\beta$ with schedule $O(\log n)$. □

Note that a direct corollary of Theorem 4.41 is that for any two $k-$colorings $\alpha$, $\beta$ of a tree with $k \geq 4$, a schedule of length $\Theta(\log n)$ can be found in $\Theta(\log n)$ rounds.

### 4.2.5 Recoloring algorithm for subcubic graphs

In this section we study recoloring in subcubic graphs (graphs of maximum degree at most three); our main result is summarized in the following theorem:

**Theorem 4.42.** *For every subcubic graph $G$ on $n$ nodes, for any two 3-colorings $\alpha, \beta$ of $G$, we can compute in $O(\log^2 n)$ rounds how to recolor $G$ from $\alpha$ to $\beta$ with one extra color and a schedule of length $O(\log n)$.*

A *theta* is formed of three node-disjoint paths between two nodes. Note that in particular if a graph contains two cycles sharing at least one edge, then it contains a theta. We note $B^k(u)$ the set of nodes at distance at most $k$ to $u$, that is the ball of radius $k$ centering in $u$.

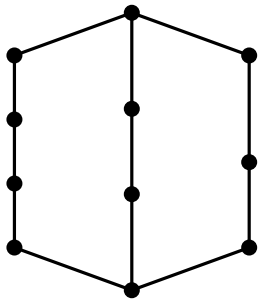


Figure 4.14 – A theta graph.

We show here, roughly speaking, that there is around every node a nice structure that we can use to design a valid greedy algorithm for the whole graph. This proof is loosely inspired by one in [ABBE18].

**Lemma 4.43.** *For every subcubic graph $G$ on $n$ nodes, for every node $u \in V(G)$, there is a node $v$ with degree at most two or a theta that is contained in $B^{2\log n}(u)$.*

*Proof.* Assume for a contradiction that there is a subcubic graph $G$ on $n$ nodes with a node $u$ such that $B^{2\log n}(u)$ contains no node of degree two nor any theta. Let $B$ be the set of nodes at distance at most $2\log n$ from $u$, and $B^-$ the set of nodes at distance at most $2\log n - 1$ from $u$. Let $\mathcal{C}$ be the set of cycles of $G$ contained in $B$. Note that cycles in $\mathcal{C}$ are edge-disjoint by assumption on $u$ and thus node-disjoint since $G$ is cubic. We select a set $\mathcal{E}$ by picking for every $C \in \mathcal{C}$ an arbitrary edge in $E(C)$ among those with both endpoints farthest from $u$. Note that $|\mathcal{E}| = |\mathcal{C}|$, and that by choice of $\mathcal{E}$, every edge in $B$ with both endpoints at the same distance of $u$ is selected in $\mathcal{E}$. Therefore, the distance to $u$ yields a natural orientation of the edges in $B \setminus \mathcal{E}$, orientation from closer node to $u$ toward further node. We also note that by choice of $\mathcal{E}$, for any edge $wx$ in $\mathcal{E}$ such that $x$ is farther away from $u$ than $w$, the node $x$ has another neighbor $y$ at the same distance of $u$ as $w$. In that case, note that the edge $xy$ does not belong to $\mathcal{E}$. We claim as a consequence that the distance from $u$ is the same in $B$ as in $B \setminus \mathcal{E}$.

For any node $w \in B$, we say an outgoing edge is *useful* if it does not belong to $\mathcal{E}$. In addition to the above remarks, we make two observations:

1. every node in $B^-$ has at least one useful edge;
2. if a node $w$ in $B^-$ has only one useful edge $wx$, then $x$ has two outgoing useful edges.

Let us consider the graph $H$ obtained from $G[B]$ by removing all edges in $\mathcal{E}$. We claim that every node in $B$ has degree at least two in $H$, and that no two adjacent nodes in $H$ have degree two: this is immediate from the observations and remarks above. We also observe that $H$ is a tree. Let $H'$ be the graph obtained from $H$ by replacing every node of degree two with an edge. We note that $H'$ is a 3-regular tree of root $u$ and with no leaf at distance less than $\log n$ of $u$. It follows that $H'$ contains at least $1 + 3 \cdot 2^{\log n} > n$ nodes, a contradiction. □

**Lemma 4.44.** *Let $G$ be a subcubic graph, let $p$ be an integer, and let $\mathcal{A}$ be a collection of thetas and nodes of degree $\leq 2$ in $G$ each at distance at least two of each other. Let $r \geq 1$ be such that no element of $\mathcal{A}$ has diameter more than $\frac{r}{2}$. If the nodes of $G \setminus (\bigcup_{A \in \mathcal{A}} A)$ can be partitioned into $S$ and $F$ such that $G[S]$ is an independent set and $G[F]$ is a forest of radius at most $p$, then there is a partition $(S', F')$ of $\bigcup_{A \in \mathcal{A}} A$ such that $G[S \cup S']$ is an independent set and $G[F \cup F']$ is a forest of radius at most $p + r$.*

*Proof.* Our construction ensures that any pair of nodes that are not connected in $G[F]$ are not connected in $G[F \cup F']$ neither. Hence, it suffices to prove that the statement holds for a single element of $\mathcal{A}$, since the elements of $\mathcal{A}$ are by hypothesis non-adjacent. Let $A$ be an element of $\mathcal{A}$. We consider two cases depending on whether $A$ is a node of degree at most two or is a theta.

- If $A$ consists of a node $v$ of degree one, or two, we set $v$ to be in $F'$ if it has a neighbor in $S$, in $S'$ otherwise. Note that since $v$ has at most one neighbor in $F$, the radius of $F \cup F'$ is at most one more than that of $F$.
- If $A$ consists of a theta with endpoints $u$ and $v$ and three node-disjoint paths $P_1, P_2, P_3$, we prove independently that each $P_i$ admits a partition that is compatible with $u$ being set to $S'$ and $v$ to $F'$, in such a way that the connected component of $F \cup F'$ that contains $v$ is contained in $F'$. We do this by induction on the number of nodes in $P_i$. If $P_i$ has no internal node, the conclusion immediately follows. If all the neighbors of $P_i$ at distance at least three of $u$ through $P_i$ are in $S$, we set all of $P_i$ to $F'$. Otherwise, let $w$ be the neighbor of $P_i$ in $F$ that with smallest distance ($\geq 3$) to $u$ through $P_i$. Let $x$ be the neighbor of $w$ in $P_i$. We apply induction on $P_i \setminus \{\text{nodes closer to } u \text{ than } x\}$, with $x$ in the role of $u$. Note that $x$ is distinct from $v$ and not adjacent to $u$, by construction. The nodes between $u$ and $x$ on $P_i$ are added to $F'$. Note that these nodes are connected to at most one component of $G[F]$, on the first node of $P_i$. We extend the resulting decomposition to the rest of $P_i$ by setting all corresponding nodes to $F'$. □

Given a graph $G = (V, E)$, an $(\alpha, \beta)$-ruling set is a set $S \subseteq V$ such that (i) the distance between any two nodes of $S$ is at least $\alpha$, and (ii) each node in $V \setminus S$ is at distance at most $\beta$ from a node in $S$.

**Lemma 4.45.** *Let $G$ be a subcubic graph on $n$ nodes. We can compute in $O(\log^2 n)$ rounds a partition $(S, F)$ of the nodes of $G$ that $G[S]$ is an independent set and $G[F]$ is a forest of radius $O(\log n)$.*

*Proof.* To that purpose, we combine the previous lemmas in Algorithm 7. The algorithm computes a decomposition as desired and runs in $O(\log n) + RS(n)$ rounds, where $RS(n)$ is the number of rounds necessary to compute a $(4 \log n, 8 \log n)$-ruling set in a subcubic graph. We derive from [PS92] that $RS(n) = O(\log^2(n))$, hence the conclusion. □

**Algorithm 7** DECOMPOSING INTO A SMALL FOREST AND AN INDEPENDENT SET

**Require:** A subcubic graph $G$.
**Ensure:** A decomposition $(F, S)$ of $V(G)$ such that $G[S]$ is an independent set and every connected component of $G[F]$ has radius at most $\log n$.

**for** $u$ in $V(G)$ (in parallel) **do**
    Acquire knowledge on $B^{2\log n}(u)$
    Select in the node set of $B^{2\log n}(u)$ a configuration $C(u)$ that is a minimal theta or a node of degree 1 or 2
Compute a $(4\log n, 8\log n)$-ruling set $X$ in $G$
Define $\mathcal{A} = \cup_{u\in X}\{C(u)\}$
Compute the distance of every node in $G$ to an element of $\mathcal{A}$
Let $F = S = \emptyset$
**for** $i = 8\log n$ downto 1 **do**
    Extend the partition $(F, S)$ to the nodes at distance $i$ from $\mathcal{A}$, more precisely:
    Each connected component is a path or cycle where no internal node has an already assigned neighbor, let $U_i$ be the set of the internal nodes
    Assuming a pre-computed MIS on each layer for the sets $U_i$, assign that MIS to $S$
    Extend greedily on the remaining nodes (which form bounded-size components), assigning nodes to $S$ when possible, to $F$ when not
Extend the partition $(F, S)$ to the nodes belonging to an element of $\mathcal{A}$ using Lemma 4.44

We are now ready to prove Theorem 4.42, which we do in a similar fashion as Theorem 4.33.

*Proof of Theorem 4.42.* Use Lemma 4.45, and obtain a decomposition $(S, F)$ as stated. Recolor all of $S$ to the extra color, then use Lemma 4.39 on each connected component of $G[F]$ so that all nodes of $F$ reach their target color (remember that each connected component of $G[F]$ has radius $O(\log n)$). Finally recolor each node of $S$ with its target color. □

### 4.2.6 Concluding remarks

In this section, we introduced $k$-recoloring in the LOCAL model of distributed computing. Because the problem of reconfiguring a $k$-coloring into another one is inherently global as it is PSPACE-complete to determine if such a transformation exists, we decided to allow extra colors. These extra colors are colors that appear neither in the source nor in the target $k$-coloring, but that we can use throughout the transformation. We moreover focus on positive instances, that is we only consider instances where a reconfiguration schedule exists. Hence, our goal is to minimize two different parameters: the number of communication rounds and the length of the reconfiguration schedule (i.e., the length of the transformation).

We first considered 3-recoloring on trees and we studied the impact of the addition of extra colors. If we do not allow any extra color, the problem can be solved in $O(n)$ rounds with a schedule of length $O(n)$ by reconfiguring the two 3-colorings into a 2-coloring. Note that this is tight by Lemma 4.31. On the other hand, if we allow two extra colors, then the problem becomes trivial: it can be solved without any communication round and a schedule of length $O(1)$. Hence, we could ask what happens in-between? Therefore, we studied 3+1 recoloring and showed that the problem can be solved in $O(\log n)$ rounds with a schedule of constant length. It remains open to determine whether the number of communication rounds is optimal or not, and we leave it as an open problem. By using a small adaptation of the proofs, one can show that one can always transform a 4-list coloring into another one with a schedule of length $O(\log n)$ in $O(\log n)$ communication rounds. Here again, note that this result is tight.

Moreover, the idea used in the proofs of Theorems 4.33 and 4.41 of finding a maximal independent set whose removal leaves a forest where each connected component has small diameter can be generalized in the following sense:

**Lemma 4.46.** *Assume that we are given a graph $G$ and input and target colorings with $k \geq 3$ colors. Assume that in $O(f(n))$ rounds we can find an independent set $I$ of $G$ such that $V \setminus I$ induces a forest of trees of depth at most $O(d(n))$. Then in $O(f(n) + d(n))$ rounds we can solve $k+1$ recoloring, with a schedule of length $O(d(n))$.*

*Proof.* Each node in $I$ switches to color $k+1$. We then use the algorithm described in the proof of Lemma 4.38 to find a recoloring with schedule of length $O(d(n))$ for each connected component after the removal of $I$. After that, each node of $I$ can switch to its final color. □

Our results regarding 3-recoloring of trees are summarized in the following tables:

Table 4.1 – Results: distributed recoloring in 3-regular trees.

| input colors | extra colors | schedule length | communication rounds | reference |
|---|---|---|---|---|
| 2 | 0 | $\infty$ | | Lemma 4.30 |
| 2 | 1 | $O(1)$ | 0 | Lemma 4.25 |
| 3 | 0 | $\Theta(n)$ | $\Theta(n)$ | Lemmas 4.26 and 4.31 |
| 3 | 1 | $O(1)$ | $O(\log n)$ | Theorem 4.33 |
| 3 | 2 | $O(1)$ | 0 | Lemma 4.25 |
| 4 | 0 | $\Theta(\log n)$ | $\Theta(\log n)$ | Theorem 4.41 and Lemma 4.32 |
| 4 | 1 | $O(1)$ | $O(1)$ | Lemma 4.29 |
| 4 | 3 | $O(1)$ | 0 | Lemma 4.25 |
| 5 | 0 | $O(1)$ | $O(1)$ | Lemmas 4.29 and 4.28 |

Table 4.2 – Results: distributed recoloring in trees.

| input colors | extra colors | schedule length | communication rounds | reference |
|---|---|---|---|---|
| 2 | 0 | $\infty$ | | Lemma 4.30 |
| 2 | 1 | $O(1)$ | 0 | Lemma 4.25 |
| 3 | 0 | $\Theta(n)$ | $\Theta(n)$ | Lemmas 4.26 and 4.31 |
| 3 | 1 | $O(1)$ | $O(\log n)$ | Theorem 4.33 |
| 3 | 2 | $O(1)$ | 0 | Lemma 4.25 |
| 4 | 0 | $\Theta(\log n)$ | $\Theta(\log n)$ | Theorem 4.41 and Lemma 4.32 |
| 4 | 1 | $O(1)$ | $O(\log n)$ | Theorem 4.33 |
| 4 | 3 | $O(1)$ | 0 | Lemma 4.25 |

Finally, we studied 3-recoloring of subcubic graphs in Section 4.2.5. More precisely, we showed that these graphs have a particular structure that can be used to greedily solve the problem on the whole graph. Our main result is that one can transform any two 3-colorings of a subcubic graph with one extra color in $O(\log^2 n)$ rounds with a schedule of length $O(\log n)$. Here again, we leave as an open problem to determine whether this result is optimal or not. Our results regarding subcubic graphs are presented in Table 4.3.

Table 4.3 – Results: distributed recoloring in subcubic graphs.

| input colors | extra colors | schedule length | communication rounds | reference |
|---|---|---|---|---|
| 2 | 0 | $\infty$ | | Lemma 4.30 |
| 2 | 1 | $O(1)$ | 0 | Lemma 4.25 |
| 3 | 0 | $\infty$ | | Lemma 4.30 |
| 3 | 1 | $O(\log n)$ | $O(\log^2 n)$ | Theorem 4.42 |
| 3 | 2 | $O(1)$ | 0 | Lemma 4.25 |
| 4 | 0 | $\infty$ | | Lemma 4.30 |
| 4 | 1 | $O(1)$ | $O(1)$ | Lemma 4.29 |
| 4 | 3 | $O(1)$ | 0 | Lemma 4.25 |
| 5 | 0 | $O(1)$ | $O(1)$ | Lemma 4.28 |

In addition to the open questions mentioned above, it would be obviously interesting to design algorithm for other graph classes. We could also image to introduce distributed reconfiguration for other graph problems. In that context, we would like to mention the work by Censor-Hillel and Rabie [CHR20] where they study the reconfiguration of maximal independent sets in the LOCAL model as well. At each step, the set of nodes that change their membership status must induce an independent set. Their main result is that for any graph $G$ of diameter at least three, one can always reconfigure two MIS of $G$ with a schedule of length 28, and where each intermediate solution is an independent 4-dominating set (i.e., each node is at distance at most four from a vertex in the set). This schedule can be compute in $O(MIS + R32)$ rounds, where $MIS$ is the complexity of finding an MIS on a worst-case graph and $R32$ is the complexity of finding a (3,2)-ruling set on a worst-case graph. An immediate corollary of this result is that for graphs of bounded maximum degree one can compute the constant length schedule within $O(\log^* n)$ rounds. Censor-Hillel and Rabie proved that this result is tight, but it remains open to determine whether their algorithm is optimal for general graphs.

# Conclusion

In this thesis, we studied some reconfiguration problems in graphs, often with a domination-related problems common base. To conclude this work, let us mention our main results, and recall some open questions that are of particular interest to us raised by this thesis.

In Chapter 2, we considered a structural question regarding the relation between dominating sets and connected dominating sets of a graph. More precisely, we proved the following conjecture by Camby and Schaudt [CS14]: for any graph $G$, it holds that $\gamma_c(H) \leq 2 \times \gamma(H)$ for any induced subgraph $H$ of $G$ if and only if $G$ does not contain $P_9$, $C_9$ or the following graph as an induced subgraph:

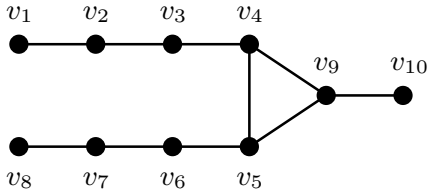



Figure 1 – The third forbidden induced subgraph.

A natural but ambitious question is the following: can we give a (finite) collection of forbidden induced subgraphs to characterize *PoC*-Near-Perfect graphs with threshold $t$, for any $t \in [1,3)$? For instance, we proved that for $t = 2$, the collection contains only $P_9$, $C_9$ and $H$.

In Chapter 3, we focused our attention on the reconfiguration of dominating sets, mainly under the TAR and TS rules. In Section 3.1, we studied some specific values on the threshold $k$ (that is the maximum size of each intermediate solution) that ensures the reconfiguration graph $\mathcal{R}_{k'}(G)$ to be connected, for any $k' \geq k$. We first improved a result by Haas and Seyffarth [HS17] by showing that the $\mathcal{R}_k(G)$ is connected and has linear diameter whenever $k \geq \Gamma(G) + \alpha(G) - 1$. We then showed results for several graph classes: we proved that $\mathcal{R}_k(G)$ is connected and has linear diameter for planar graphs whenever $k \geq \Gamma(G) + 3$ and for $K_\ell$-free minor graphs as long as $k \geq \Gamma(G) + C\ell\sqrt{\log_2 \ell}$, for some constant $C$. Note that these two results are obtained from a more general one on $d$-minor sparse graphs, that is graphs whose all bipartite minors have average degree less than $d$ (e.g., planar graphs are 4-minor sparse graphs). Moreover, our proofs are algorithmic and a reconfiguration sequence can be computed in polynomial time. However, the best lower bound for planar graphs is $\Gamma(G) + 1$; hence it remains open for $k = \Gamma(G) + 2$ and we conjectured that $\mathcal{R}_{\Gamma(G)+2}(G)$ is connected. Finally, we proved that if $k \geq \Gamma(G) + tw(G) + 1$, then $\mathcal{R}_k(G)$ is connected and has diameter at most $4(n+1) \cdot (tw(G)+1)$. Here again, the proof is constructive. Since a tree decomposition of width $r$ can be computed in time $2^{O(r^3)} \cdot n$ [Bod96], this yields an FPT algorithm parameterized by the treewidth that outputs a linear transformation between any pair of dominating sets of size at most $k$. It remains open to determine whether $\mathcal{R}_k(G)$ is connected or not for $k \in \{\Gamma(G) + tw(G) - 1, \Gamma(G) + tw(G)\}$. In Section 3.2, we investigated the computational complexity of DOMINATING SET

RECONFIGURATION under token sliding. We showed that the problem is PSPACE-complete for various graph classes including split graphs or bipartite graphs. On the other hand, we gave polynomial-time algorithm for cographs and dually chordal graphs, a superclass of both trees and interval graphs. However, for all the classes for which $\text{DSR}_{\text{TS}}$ is PSPACE-complete (respectively in P), DOMINATING SET is NP-complete (resp. in P). Hence, the most challenging problems for $\text{DSR}_{\text{TS}}$ are to find graph classes for which the host problem is "easy" (in P) but $\text{DSR}_{\text{TS}}$ is NP-hard, or vice-versa. In Section 3.3, we again considered the token addition and removal rule, but with a rather different perspective. We considered an optimization variant recently introduced by Ito et al. [IMNS19] for INDEPENDENT SET RECONFIGURATION which attempts to answer the following question: given a dominating set $D$ of a graph $G$ and an integer $k$, what is the smallest dominating set of $G$ that can be obtained from $D$ under the TAR($k$) rule? It is observed that this new problem generalizes DOMINATING SET by setting $D = V$ and $k = |V|$. It immediately follows that it is NP-hard; we actually proved its PSPACE-completeness in several graph classes. However, we were not able to determine the complexity of OPT-DSR on planar graphs. We know that it NP-hard, but we believe that it is PSPACE-complete. Nevertheless, our main contribution is related to its parameterized complexity. We first proved that it is W[2]-hard with respect to $k$ while it is FPT when parameterized by $d + s$ (where $d$ is the degeneracy and $s$ the size of the target dominating set) or by $\tau$, the minimum size of a vertex cover of $G$.

Finally, in Chapter 4, we focused on two different reconfiguration problems. We first investigated the complexity of the reconfiguration of spanning trees with additional constraints on the number of leaves. We proved that it is PSPACE-complete to decide whether there exists an edge flip transformation between two spanning trees where each spanning tree must have at least $k$ leaves. This result holds in particular for bipartite graphs, split graphs or planar graphs. Note that because spanning trees with many leaves is highly-related to minimum connected dominating sets, it is not surprising that the PSPACE-completeness for split graphs or bipartite graphs follows from polynomial-time reductions from DOMINATING SET RECONFIGURATION. However, we proved that if we allow at most two internal nodes, then the problem becomes polynomial-time solvable. This result is of special interest, as it can be used to prove that the problem is in P as well for cographs. More generally, can we decide if there exists a transformation between two spanning trees with at most $n - \ell$ leaves in time FPT with respect to $\ell$? On the other hand, we showed that it is also PSPACE-complete to determine whether one can reconfigure two spanning trees where each solution has at most $k$ leaves. The result holds for any $k \geq 3$, and the proof is an adaptation of a well-known reduction from VERTEX COVER to HAMILTONIAN PATH. Moreover, note that if $k = 2$, then SPANNING TREE WITH AT MOST $k$ LEAVES is equivalent to HAMILTONIAN PATH RECONFIGURATION whose complexity remains open. In Section 4.2, we introduced reconfiguration in a distributed setting. More precisely, we studied $k$-recoloring in the LOCAL model in Distributed Computing, where no two adjacent nodes can change their color simultaneously. In particular we focused on 3-recoloring of trees and subcubic graphs. We proved that if we allow one extra color, then 3-recoloring of an $n$-vertex tree can be solved in $O(\log n)$ rounds with a schedule of length $O(1)$. This result can be adapted to obtain a similar result for list-recoloring: 4-list-recoloring can be solved in $O(\log n)$ rounds with a schedule of length $O(\log n)$. Finally, we proved that 3-recoloring with one extra color can be solved in $O(\log^2 n)$ rounds and with a schedule of length $O(\log n)$ on subcubic graphs. One may ask whether our results for $3 + 1$ recoloring on trees and subcubic graphs are optimal (mainly with regard to the number of rounds) or not. Finally, it would be interesting to study the reconfiguration of other graph problems in a distributed setting.

# Bibliography


[ABBE18] P. Aboulker, M. Bonamy, N. Bousquet, and L. Esperet. Distributed Coloring in Sparse Graphs with Fewer Colors. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, PODC '18, page 419–425, New York, NY, USA, 2018. Association for Computing Machinery.

[ACP87] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of Finding Embeddings in a k-Tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, April 1987.

[AFF+05] J. Alber, H. Fan, M. R. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, and U. Stege. A refined search tree technique for Dominating Set on planar graphs. *Journal of Computer and System Sciences*, 71(4):385–405, November 2005.

[AFK17] S. Alikhani, D. Fatehi, and S. Klavžar. On the Structure of Dominating Graphs. *Graphs and Combinatorics*, 33(4):665–672, May 2017.

[AFN04] J. Alber, M. R. Fellows, and R. Niedermeier. Polynomial-time data reduction for dominating set. *Journal of the ACM*, 51(3):363–384, May 2004.

[AG08] N. Alon and S. Gutner. Linear Time Algorithms for Finding a Dominating Set of Fixed Size in Degenerated Graphs. *Algorithmica*, 54(4):544–556, July 2008.

[AH77] K. Appel and W. Haken. Every planar map is four colorable. Part I: Discharging. *Illinois J. Math.*, 21(3):429–490, 09 1977.

[AHK77] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. Part II: Reducibility. *Illinois J. Math.*, 21(3):491–567, 09 1977.

[AHN04] O. Aichholzer, F. Hurtado, and M. Noy. A lower bound on the number of triangulations of planar point sets. *Computational Geometry*, 29(2):135 – 145, 2004.

[Ami01] E. Amir. Efficient Approximation for Triangulation of Minimum Treewidth. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 7–15, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[AMP15] O. Aichholzer, W. Mulzer, and A. Pilz. Flip Distance Between Triangulations of a Simple Polygon is NP-Complete. *Discrete & Computational Geometry*, 54(2):368–389, June 2015.

[Ang80] D. Angluin. Local and global properties in networks of processors (Extended Abstract). In *Proceedings of the twelfth annual ACM symposium on Theory of computing - STOC '80*. ACM Press, 1980.

[AP89] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, April 1989.

[APT79] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121 – 123, 1979.

[BB72] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, Inc., USA, 1972.

[BB14] M. Bonamy and N. Bousquet. Reconfiguring Independent Sets in Cographs. *CoRR*, abs/1406.1433, 2014.

[BB17] M. Bonamy and N. Bousquet. Token sliding on chordal graphs. In *Proceedings of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science* (*WG 2017*), pages 127–139, 2017.

[BB18] M. Bonamy and N. Bousquet. Recoloring graphs via tree decompositions. *European Journal of Combinatorics*, 69:200–213, 2018.

[BBFJ19] M. Bonamy, N. Bousquet, C. Feghali, and M. Johnson. On a conjecture of Mohar concerning Kempe equivalence of regular graphs. *Journal of Combinatorial Theory, Series B*, 135:179 – 199, 2019.

[BBH+19] M. Bonamy, N. Bousquet, M. Heinrich, T. Ito, Y. Kobayashi, A. Mary, M. Mühlenthaler, and K. Wasa. The Perfect Matching Reconfiguration Problem. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, pages 80:1–80:14, 2019.

[BC79] B. Bollobás and E. J. Cockayne. Graph-theoretic parameters concerning domination, independence, and irredundance. *Journal of Graph Theory*, 3(3):241–249, 1979.

[BC09] P. Bonsma and L. Cereceda. Finding Paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theoretical Computer Science*, 410(50):5215 – 5226, 2009. Mathematical Foundations of Computer Science (MFCS 2007).

[BCD98] A. Brandstädt, V. D. Chepoi, and F. F. Dragan. The algorithmic use of hypertree structure and maximum neighbourhood orderings. *Discrete Applied Mathematics*, 82(1):43 – 77, 1998.

[BCM97] A. Burger, E. Cockayne, and C. Mynhardt. Domination and irredundance in the queens' graph. *Discrete Mathematics*, 163(1-3):47–66, January 1997.

[BDO21] M. Bonamy, P. Dorbec, and P. Ouvrard. Dominating sets reconfiguration under token sliding. *Discrete Applied Mathematics*, 301:6–18, 2021.

[BE13] L. Barenboim and M. Elkin. Distributed graph coloring: Fundamentals and recent developments. *Synthesis Lectures on Distributed Computing Theory*, 4(1):1–171, 2013.

[Ber58] C. Berge. *Théorie des graphes et ses applications*. Collection universitaire de mathématiques. Dunod, 1958.

[Ber73] C. Berge. *Graphs and Hypergraphs*. Graphs and Hypergraphs. North-Holland Publishing Company, 1973.

[Ber84] A. Bertossi. Dominating sets for split and bipartite graphs. *Information Processing Letters*, 19(1):37–40, 1984.

[BFH94] H. L. Bodlaender, M. R. Fellows, and M. T. Hallett. Beyond NP-Completeness for Problems of Bounded Width (Extended Abstract): Hardness for the W Hierarchy. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94, page 449–458, New York, NY, USA, 1994. Association for Computing Machinery.

[BHI+19] M. Bonamy, M. Heinrich, T. Ito, Y. Kobayashi, H. Mizuta, M. Mühlenthaler, A. Suzuki, and K. Wasa. Diameter of Colorings Under Kempe Changes. In *Lecture Notes in Computer Science*, pages 52–64. Springer International Publishing, 2019.

[BHIM19] N. Bousquet, T. Hatanaka, T. Ito, and M. Mühlenthaler. Shortest Reconfiguration of Matchings. In *Graph-Theoretic Concepts in Computer Science*, pages 162–174. Springer International Publishing, 2019.

[BIK+20] N. Bousquet, T. Ito, Y. Kobayashi, H. Mizuta, P. Ouvrard, A. Suzuki, and K. Wasa. Reconfiguration of Spanning Trees with Many or Few Leaves. In F. Grandoni, G. Herman, and P. Sanders, editors, *28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[BJO20] N. Bousquet, A. Joffard, and P. Ouvrard. Linear Transformations Between Dominating Sets in the TAR-Model. In Y. Cao, S.-W. Cheng, and M. Li, editors, *31st International Symposium on Algorithms and Computation (ISAAC 2020)*, volume 181 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[BKW14] P. Bonsma, M. Kamiński, and M. Wrochna. Reconfiguring Independent Sets in Claw-Free Graphs. In R. Ravi and I. L. Gortz, editors, *Algorithm Theory – SWAT 2014*, pages 86–97, Cham, 2014. Springer International Publishing.

[BM86] H.-J. Bandelt and H. M. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182 – 208, 1986.

[BMNR14] P. Bonsma, A. E. Mouawad, N. Nishimura, and V. Raman. The complexity of bounded length graph recoloring and CSP reconfiguration. In *International Symposium on Parameterized and Exact Computation*, pages 110–121. Springer, 2014.

[BMOS20] A. Blanché, H. Mizuta, P. Ouvrard, and A. Suzuki. Decremental Optimization of Dominating Sets Under the Reconfiguration Framework. In L. Gąsieniec, R. Klasing, and T. Radzik, editors, *Combinatorial Algorithms*, pages 69–82, Cham, 2020. Springer International Publishing.

[BMP17] N. Bousquet, A. Mary, and A. Parreau. Token Jumping in Minor-Closed Classes. In R. Klasing and M. Zeitoun, editors, *Fundamentals of Computation Theory*, pages 136–149, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[Bod96] H. L. Bodlaender. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, December 1996.

[Bon12] P. Bonsma. The Complexity of Rerouting Shortest Paths. In *Mathematical Foundations of Computer Science 2012*, pages 222–233. Springer Berlin Heidelberg, 2012.

[Bon16] P. S. Bonsma. Independent Set Reconfiguration in Cographs and their Generalizations. *Journal of Graph Theory*, 83(2):164–195, 2016.

[BOR+18] M. Bonamy, P. Ouvrard, M. Rabie, J. Suomela, and J. Uitto. Distributed Recoloring. In U. Schmid and J. Widder, editors, *Proceedings of DISC 2018*, volume 121 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[BP16] N. Bousquet and G. Perarnau. Fast recoloring of sparse graphs. *European Journal of Combinatorics*, 52:1–11, 2016.

[BPTW10] J. Böttcher, K. P. Pruessmann, A. Taraz, and A. Würfl. Bandwidth, expansion, treewidth, separators and universality for bounded-degree graphs. *European Journal of Combinatorics*, 31(5):1217 – 1227, 2010.

[Bro41] R. L. Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2):194–197, April 1941.

[BRW85] E. A. Bender, L. B. Richmond, and N. C. Wormald. Almost all chordal graphs split. *Journal of the Australian Mathematical Society. Series A. Pure Mathematics and Statistics*, 38(2):214–221, April 1985.

[Bv'HKP17] R. Belmonte, P. van 't Hof, M. Kamiński, and D. Paulusma. The price of connectivity for feedback vertex set. *Discrete Applied Mathematics*, 217:132–143, January 2017.

[Bü60] J. R. Büchi. Weak Second-Order Arithmetic and Finite Automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[CFK+15] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[CGH75] E. Cockayne, S. Goodman, and S. Hedetniemi. A linear algorithm for the domination number of a tree. *Information Processing Letters*, 4(2):41–44, November 1975.

[CH77] E. J. Cockayne and S. T. Hedetniemi. Towards a theory of domination in graphs. *Networks*, 7(3):247–261, 1977.

[Cha98] M.-S. Chang. Efficient Algorithms for the Domination Problems on Interval and Circular-Arc Graphs. *SIAM Journal on Computing*, 27(6):1671–1694, December 1998.

[CHHH11] E. Connelly, K. R. Hutson, S. T. Hedetniemi, and T. Haynes. A Note on $\gamma$-Graphs. *AKCE International Journal of Graphs and Combinatorics*, 8(1):23–31, 2011.

[CHM78] E. J. Cockayne, S. T. Hedetniemi, and D. J. Miller. Properties of Hereditary Hypergraphs and Middle Graphs. *Canadian Mathematical Bulletin*, 21(4):461–468, December 1978.

[CHR20] K. Censor-Hillel and M. Rabie. Distributed reconfiguration of maximal independent sets. *Journal of Computer and System Sciences*, 112:85–96, September 2020.

[CKP16] Y.-J. Chang, T. Kopelowitz, and S. Pettie. An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model. In *Foundations of Computer Science (FOCS)*, pages 615–624, 2016.

[CKX10] J. Chen, I. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40):3736 – 3756, 2010.

[CL10] J. Cardinal and E. Levy. Connected vertex covers in dense graphs. *Theoretical Computer Science*, 411(26-28):2581–2590, June 2010.

[CLB81] D. Corneil, H. Lerchs, and L. Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163 – 174, 1981.

[CM93] E. Cockayne and C. Mynhardt. The sequence of upper and lower domination, independence and irredundance numbers of a graph. *Discrete Mathematics*, 122(1-3):89–102, November 1993.

[CN84] G. J. Chang and G. L. Nemhauser. The k-Domination and k-Stability Problems on Sun-Free Chordal Graphs. *SIAM Journal on Algebraic Discrete Methods*, 5(3):332–345, September 1984.

[Coo71] S. A. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.

[Cou90] B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12 – 75, 1990.

[CP84] D. Corneil and Y. Perl. Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9(1):27–39, September 1984.

[CS14] E. Camby and O. Schaudt. The price of connectivity for dominating set: Upper bounds and complexity. *Discrete Applied Mathematics*, 177:53–59, November 2014.

[CV86] R. Cole and U. Vishkin. Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing - STOC '86*. ACM Press, 1986.

[CvdHJ08] L. Cereceda, J. van den Heuvel, and M. Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(5):913 – 919, 2008. Selected Papers from 20th British Combinatorial Conference.

[CVdHJ09] L. Cereceda, J. Van den Heuvel, and M. Johnson. Mixing 3-colourings in bipartite graphs. *European Journal of Combinatorics*, 30(7):1593–1606, 2009.

[CVDHJ11] L. Cereceda, J. Van Den Heuvel, and M. Johnson. Finding paths between 3-colorings. *Journal of graph theory*, 67(1):69–82, 2011.

[DDF+16] P. Drange, M. Dregi, F. Fomin, S. Kreutzer, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, F., F. Villaamil, S. Saurabh, S. Siebertz, and S.Sikdar. Kernelization and sparseness: the case of dominating set. In *33rd Symposium on Theoretical Aspects of Computer Science* (*STACS 2016*), pages 31:1–31:14, 2016.

[DDJS20] M. DeVos, A. Dyck, J. Jedwab, and S. Simon. Which Graphs Occur as $\gamma$-Graphs? *Graphs Comb.*, 36(4):1219–1246, 2020.

[DF95] R. G. Downey and M. R. Fellows. Fixed-Parameter Tractability and Completeness I: Basic Results. *SIAM Journal on Computing*, 24(4):873–921, August 1995.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[DF13] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer London, 2013.

[DFHT05] E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and $H$-minor-free graphs. *Journal of the ACM*, 52(6):866–893, November 2005.

[DK09] A. Dawar and S. Kreutzer. Domination Problems in Nowhere-Dense Classes. In R. Kannan and K. N. Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 157–168, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[DKR15] P. Dorbec, G. Košmrlj, and G. Renault. The domination game played on unions of graphs. *Discrete Math.*, 338(1):71–79, 2015.

[DM82] P. Duchet and H. Meyniel. On Hadwiger's number and the stability number. *Ann. Discr. Math.*, 13:71–74, 1982.

[Dou92] R. J. Douglas. NP-completeness and degree restricted spanning trees. *Discrete Mathematics*, 105(1-3):41–47, August 1992.

[Edm65] J. Edmonds. Paths, Trees, and Flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[EFF04] J. Ellis, H. Fan, and M. Fellows. The dominating set problem is fixed parameter tractable for graphs of bounded genus. *Journal of Algorithms*, 52(2):152–168, August 2004.

[Elg61] C. C. Elgot. Decision Problems of Finite Automata Design and Related Arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961.

[EMN18] M. Edwards, G. MacGillivray, and S. Nasserasr. Reconfiguring minimum dominating sets: the $\gamma$-graph of a tree. *Discussiones Mathematicae Graph Theory*, 38(3):703, 2018.

[FB02] G. W. Flake and E. B. Baum. Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1):895 – 911, 2002.

[FG65] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15(3):835–855, 1965.

[FH77] S. Foldes and P. L. Hammer. Split graphs. In *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing (Louisiana State Univ., Baton Rouge, La., 1977)*, pages 311–315, 1977.

[FHHH11] G. Fricke, S. M. Hedetniemi, S. T. Hedetniemi, and K. R. Hutson. $\gamma$-graphs of graphs. *Discussiones Mathematicae Graph Theory*, 31(3):517–531, 2011.

[FHL05] U. Feige, M. Hajiaghayi, and J. R. Lee. Improved Approximation Algorithms for Minimum-Weight Vertex Separators. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 563–572, New York, NY, USA, 2005. Association for Computing Machinery.

[FJP16] C. Feghali, M. Johnson, and D. Paulusma. A Reconfigurations Analogue of Brooks' Theorem and Its Consequences. *Journal of Graph Theory*, 83(4):340–358, 2016.

[FJP17] C. Feghali, M. Johnson, and D. Paulusma. Kempe equivalence of colourings of cubic graphs. *European Journal of Combinatorics*, 59:1–10, 2017.

[FT87] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, July 1987.

[FT06] F. V. Fomin and D. M. Thilikos. Dominating Sets in Planar Graphs: Branch-Width and Exponential Speed-Up. *SIAM Journal on Computing*, 36(2):281–309, January 2006.

[Gav74] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47 – 56, 1974.

[Gav20] C. Gavoille. Analyse d'algorithme. Lecture notes, Université de Bordeaux, 2020.

[GGR20] M. Ghaffari, C. Grunau, and V. Rozhoň. Improved Deterministic Network Decomposition. *CoRR*, abs/2007.08253, 2020.

[GH64] P. C. Gilmore and A. J. Hoffman. A Characterization of Comparability Graphs and of Interval Graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964.

[GHKM18] M. Ghaffari, J. Hirvonen, F. Kuhn, and Y. Maus. Improved Distributed Delta-Coloring. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. ACM, July 2018.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability*: *A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[GKM19] M. Gupta, H. Kumar, and N. Misra. On the Complexity of Optimal Matching Reconfiguration. In *SOFSEM 2019: Theory and Practice of Computer Science*, pages 221–233. Springer International Publishing, 2019.

[GKMP09] P. Gopalan, P. G. Kolaitis, E. Maneva, and C. H. Papadimitriou. The Connectivity of Boolean Satisfiability: Computational and Structural Dichotomies. *SIAM J. Comput.*, 38(6):2330–2355, March 2009.

[GMAV62] E. M. L. G. M. Adelson-Velskii. An algorithm for organization of information. *Dokl. Akad. Nauk SSSR*, 146(2), 1962.

[GPS88] A. V. Goldberg, S. A. Plotkin, and G. E. Shannon. Parallel Symmetry-Breaking in Sparse Graphs. *SIAM Journal on Discrete Mathematics*, 1(4):434–446, November 1988.

[HD05] R. Hearn and E. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1–2):72–96, 2005.

[HHS97] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Domination in graphs : advanced topics*. Marcel Dekker, New York, 1997.

[HHS98] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of domination in graphs*. Marcel Dekker, 1998.

[HIM+16] A. Haddadan, T. Ito, A. E. Mouawad, N. Nishimura, H. Ono, A. Suzuki, and Y. Tebbal. The Complexity of Dominating Set Reconfiguration. *Theor. Comput. Sci.*, 651(C):37–49, October 2016.

[HIM+18] T. Hanaka, T. Ito, H. Mizuta, B. Moore, N. Nishimura, V. Subramanya, A. Suzuki, and K. Vaidyanathan. Reconfiguring Spanning and Induced Subgraphs. In *Lecture Notes in Computer Science*, pages 428–440. Springer International Publishing, 2018.

[HK20] T. Hulcová and T. Klimošvá. Personal communication, 2020.

[HMPV00] M. Habib, R. McConnell, C. Paul, and L. Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1):59 – 84, 2000.

[HNU99] F. Hurtado, M. Noy, and J. Urrutia. Flipping Edges in Triangulations. *Discrete & Computational Geometry*, 22(3):333–346, October 1999.

[HP05] M. Habib and C. Paul. A simple linear time algorithm for cograph recognition. *Discrete Applied Mathematics*, 145(2):183 – 197, 2005. Structural Decompositions, Width Parameters, and Graph Labelings.

[HS14] R. Haas and K. Seyffarth. The k-Dominating Graph. *Graphs and Combinatorics*, 30(3):609–617, May 2014.

[HS17] R. Haas and K. Seyffarth. Reconfiguring dominating sets in some well-covered and other classes of graphs. *Discrete Mathematics*, 340(8):1802 – 1817, 2017.

[HS20] J. Hirvonen and J. Suomela. Distributed Algorithms. Textbook, Aalto University, 2020.

[IDH+08] T. Ito, E. D. Demaine, N. J. A. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara, and Y. Uno. On the Complexity of Reconfiguration Problems. In *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, pages 28–39, 2008.

[IKK+19] T. Ito, N. Kakimura, N. Kamiyama, Y. Kobayashi, and Y. Okamoto. Shortest Reconfiguration of Perfect Matchings via Alternating Cycles. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, pages 61:1–61:15, 2019.

[IKO14] T. Ito, M. Kamiński, and H. Ono. Fixed-Parameter Tractability of Token Jumping on Planar Graphs. In *Algorithms and Computation*, pages 208–219. Springer International Publishing, 2014.

[IKO+20] T. Ito, M. Kamiński, H. Ono, A. Suzuki, R. Uehara, and K. Yamanaka. Parameterized complexity of independent set reconfiguration problems. *Discrete Applied Mathematics*, 283:336–345, September 2020.

[IMNS19] T. Ito, H. Mizuta, N. Nishimura, and A. Suzuki. Incremental Optimization of Independent Sets Under the Reconfiguration Framework. In *Computing and Combinatorics - 25th International Conference, COCOON 2019, Xi'an, China, July 29-31, 2019, Proceedings*, pages 313–324, 2019.

[INZ15] T. Ito, H. Nooka, and X. Zhou. Reconfiguration of Vertex Covers in a Graph. In K. Jan, M. Miller, and D. Froncek, editors, *Combinatorial Algorithms*, pages 164–175, Cham, 2015. Springer International Publishing.

[Jer95] M. Jerrum. A very simple algorithm for estimating the number of $k$-colorings of a low-degree graph. *Random Structures & Algorithms*, 7(2):157–165, 1995.

[Jof20] A. Joffard. *Graph domination and reconfiguration problems*. PhD thesis, Université Claude Bernard Lyon 1, 2020.

[Kar72] R. M. Karp. Reducibility among Combinatorial Problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations: 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, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, pages 85–103. Springer US, Boston, MA, 1972.

[KBMK93] T. Kloks, H. Bodlaender, H. Müller, and D. Kratsch. Computing treewidth and minimum fill-in: All you need are the minimal separators. In *Algorithms—ESA '93*, pages 260–271. Springer Berlin Heidelberg, 1993.

[Kei93] J. Keil. The complexity of domination problems in circle graphs. *Discrete Applied Mathematics*, 42(1):51 – 63, 1993.

[Kem79] A. B. Kempe. On the geographical problem of the four colours. *American Journal of Mathematics*, 2(3):193–200, 1879.

[KMM12] M. Kamiński, P. Medvedev, and M. Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9 – 15, 2012.

[KS93] E. Korach and N. Solel. Tree-width, path-width, and cutwidth. *Discrete Applied Mathematics*, 43(1):97 – 101, 1993.

[Kur30] C. Kuratowski. Sur le problème des courbes gauches en Topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930.

[Law72] C. L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365 – 372, 1972.

[Lev73] L. A. Levin. Universal Sequential Search Problems. *Problems of Information Transmission*, 9(3), 1973.

[Lev09] E. Levy. *Approximation Algorithms for Covering Problems in Dense Graphs*. PhD thesis, Université Libre de Bruxelles, 2009.

[LFMW17] S. Li, Q. Feng, X. Meng, and J. Wang. An Improved FPT Algorithm for the Flip Distance Problem. In K. G. Larsen, H. L. Bodlaender, and J.-F. Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*, volume 83 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 65:1–65:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Lin92] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, February 1992.

[LM18] D. Lokshtanov and A. E. Mouawad. The complexity of independent set reconfiguration on bipartite graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 185–195, 2018.

[LMP+18] D. Lokshtanov, A. Mouawad, F. Panolan, M. Ramanujan, and S. Saurabh. Reconfiguration on sparse graphs. *J. Comput. Syst. Sci.*, 95:122–131, 2018.

[LMPS19] D. Lokshtanov, A. E. Mouawad, F. Panolan, and S. Siebertz. On the Parameterized Complexity of Reconfiguration of Connected Dominating Sets. *CoRR*, abs/1910.00581, 2019.

[LP15] A. Lubiw and V. Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, November 2015.

[LVA10] S. A. Lakshmanan, A. Vijayakumar, and S. Arumugam. The gamma graph of a graph. *AKCE International Journal of Graphs and Combinatorics*, 7(1):53–59, 2010.

[LŻ20] M. Lemańska and P. Żyliński. Reconfiguring Minimum Dominating Sets in Trees. *Journal of Graph Algorithms and Applications*, 24(1):47–61, 2020.

[Mad68] W. Mader. Homomorphiesätze für Graphen. *Math. Ann.*, 178:154–168, 1968.

[McD15] D. C. McDonald. Connectedness and Hamiltonicity of graphs on vertex colorings. *arXiv preprint arXiv:1507.05344*, 2015.

[MCH79] S. Mitchell, E. Cockayne, and S. Hedetniemi. Linear algorithms on recursive representations of trees. *Journal of Computer and System Sciences*, 18(1):76 – 85, 1979.

[MHIZ19] H. Mizuta, T. Hatanaka, T. Ito, and X. Zhou. Reconfiguration of Minimum Steiner Trees via Vertex Exchanges. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany.*, pages 79:1–79:11, 2019.

[MIZ16] H. Mizuta, T. Ito, and X. Zhou. Reconfiguration of Steiner Trees in an Unweighted Graph. In V. Mäkinen, S. J. Puglisi, and L. Salmela, editors, *Combinatorial Algorithms*, pages 163–175, Cham, 2016. Springer International Publishing.

[MNPR15] A. E. Mouawad, N. Nishimura, V. Pathak, and V. Raman. Shortest Reconfiguration Paths in the Solution Space of Boolean Formulas. In *Automata, Languages, and Programming*, pages 985–996. Springer Berlin Heidelberg, 2015.

[MNR14] A. Mouawad, N. Nishimura, and V. Raman. Vertex cover reconfiguration and beyond. In *Proceedings of the 25th International Symposium on Algorithms and Computation* (*ISAAC 2014*), pages 452–463, 2014.

[MNR+17] A. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki. On the Parameterized Complexity of Reconfiguration Problems. *Algorithmica*, 78(1):274–297, 2017.

[MNRW14] A. E. Mouawad, N. Nishimura, V. Raman, and M. Wrochna. Reconfiguration over Tree Decompositions. In *Parameterized and Exact Computation*, pages 246–257. Springer International Publishing, 2014.

[Moh07] B. Mohar. Kempe Equivalence of Colorings. In A. Bondy, J. Fonlupt, J.-L. Fouquet, J.-C. Fournier, and J. L. Ramírez Alfonsín, editors, *Graph Theory in Paris: Proceedings of a Conference in Memory of Claude Berge*, pages 287–297. Birkhäuser Basel, Basel, 2007.

[Mou15] A. Mouawad. *On Reconfiguration Problems: Structure and Tractability*. PhD thesis, University of Waterloo, 2015.

[MR89] G. L. Miller and J. H. Reif. Parallel Tree Contraction Part 1: Fundamentals. *Advances in Computing Research*, 5:47–72, 1989.

[MS88] B. Monien and I. Sudborough. Min cut is NP-complete for edge weighted trees. *Theoretical Computer Science*, 58(1-3):209–229, June 1988.

[MS99] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1-3):189–241, April 1999.

[MTR19] C. Mynhardt, L. Teshima, and A. Roux. Connected $k$-dominating graphs. *Discrete Mathematics*, 342(1):145–151, January 2019.

[MW15] E. A. Marshall and D. R. Wood. Circumference and Pathwidth of Highly Connected Graphs. *Journal of Graph Theory*, 79(3):222–232, 2015.

[NDM08] J. Nešetřil and P. O. De Mendez. Structural Properties of Sparse Graphs. In M. Grötschel, G. O. H. Katona, and G. Sági, editors, *Building Bridges: Between Mathematics and Computer Science*, pages 369–426. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[Nis18] N. Nishimura. Introduction to Reconfiguration. *Algorithms*, 11(4):52, 2018.

[NS95] M. Naor and L. Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[NS01] F. Nicolai and T. Szymczak. Homogeneous sets and domination: A linear time algorithm for distance—hereditary graphs. *Networks*, 37(3):117–128, 2001.

[Ore62] Ø. Ore. *Theory of Graphs*. Colloquium Publications 2473-3946. American Mathematical Society, 1962.

[Pap76] C. H. Papadimitriou. The NP-Completeness of the bandwidth minimization problem. *Computing*, 16(3):263–270, September 1976.

[Pap94] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[Pel00] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, January 2000.

[Pil14] A. Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, July 2014.

[PS92] A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 581–592. ACM, 1992.

[PS95] A. Panconesi and A. Srinivasan. The local nature of $\Delta$-coloring and its algorithmic applications. *Combinatorica*, 15(2):255–280, 1995.

[Ree92] B. A. Reed. Finding approximate separators and computing tree width quickly. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing - STOC '92*. ACM Press, 1992.

[RG20] V. Rozhoň and M. Ghaffari. Polylogarithmic-Time Deterministic Network Decomposition and Distributed Derandomization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 350–363, New York, NY, USA, 2020. Association for Computing Machinery.

[Rou73] N. D. Roussopoulos. A max {m, n} algorithm for determining the graph H from its line graph G. *Information Processing Letters*, 2(4):108–112, October 1973.

[RR20] D. Rautenbach and J. Redl. Reconfiguring dominating sets in minor-closed graph classes. *CoRR*, abs/2005.13844, 2020.

[RS84] N. Robertson and P. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49 – 64, 1984.

[RT75] D. J. Rose and R. E. Tarjan. Algorithmic Aspects of Vertex Elimination on Directed Graphs. Technical report, Stanford, CA, USA, 1975.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177 – 192, 1970.

[Sbi80] N. Sbihi. Algorithme de recherche d'un stable de cardinalite maximum dans un graphe sans etoile. *Discrete Mathematics*, 29(1):53–76, 1980.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing - STOC'78*. ACM Press, 1978.

[Sch14] K. W. Schwerdtfeger. A Computational Trichotomy for Connectivity of Boolean Satisfiability. *Journal on Satisfiability, Boolean Modeling and Computation*, 8(3–4):173–195, June 2014.

[SM88] D. Sumner and J. Moore. *Domination perfect graphs*. Notice Amer.Math.Soc.26, 1988.

[SMN14] A. Suzuki, A. E. Mouawad, and N. Nishimura. Reconfiguration of Dominating Sets. In *Computing and Combinatorics - 20th International Conference, COCOON 2014, Atlanta, GA, USA, August 4-6, 2014. Proceedings*, pages 405–416, 2014.

[SS08] N. S. Sridharan and K. Subramanian. $\gamma$-Graph of a graph. *Bull. Kerala Math. Assoc*, 5:17–34, 2008.

[STT86] D. D. Sleator, R. E. Tarjan, and W. P. Thurston. Rotation Distance, Triangulations, and Hyperbolic Geometry. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, page 122–135, New York, NY, USA, 1986. Association for Computing Machinery.

[Sum90] D. P. Sumner. Critical concepts in domination. *Discrete Mathematics*, 86(1-3):33–46, December 1990.

[Suo13] J. Suomela. Survey of local algorithms. *ACM Computing Surveys*, 45(2):1–40, February 2013.

[SW79] E. Sampathkumar and H. B. Walikar. The connected domination number of a graph. *J. Math. Phys*, 13(6):607–613, 1979.

[Tho84] A. Thomason. An extremal function for contractions of graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 95(2):261–265, 1984.

[Tho99] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, 46(3):362–394, May 1999.

[Tra61] B. A. Trakhtenbrot. Finite automata and the logic of single-place predicates. *Dokl. Akad. Nauk SSSR*, 140(1):326–329, 1961.

[Tse83] G. S. Tseitin. On the Complexity of Derivation in Propositional Calculus. In *Automation of Reasoning*, pages 466–483. Springer Berlin Heidelberg, 1983.

[Tur36] A. M. Turing. On Computable Numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[TV12] J. A. Telle and Y. Villanger. FPT Algorithms for Domination in Biclique-Free Graphs. In *Algorithms – ESA 2012*, pages 802–812. Springer Berlin Heidelberg, 2012.

[vdH13] J. van den Heuvel. The complexity of change. In S. R. Blackburn, S. Gerke, and M. Wildon, editors, *Surveys in Combinatorics*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013.

[vdZ15] T. C. van der Zanden. Parameterized Complexity of Graph Constraint Logic. In T. Husfeldt and I. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*, volume 43 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 282–293, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[WL99] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications - DIALM '99*. ACM Press, 1999.

[Wro18] M. Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *Journal of Computer and System Sciences*, 93:1 – 10, 2018.

[YG80] M. Yannakakis and F. Gavril. Edge Dominating Sets in Graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, June 1980.

[Zve03] I. E. Zverovich. Perfect connected-dominant graphs. *Discussiones Mathematicae Graph Theory*, 23(1):159, 2003.

[ZZ95] I. E. Zvervich and V. E. Zverovich. An induced subgraph characterization of domination perfect graphs. *Journal of Graph Theory*, 20(3):375–395, November 1995.

**Titre :** Problèmes de reconfiguration dans les graphes

Dans cette thèse, nous nous intéressons à la théorie des graphes, et plus particulièrement à des problèmes de reconfiguration. Pour un problème d'optimisation donné, l'objectif est alors d'étudier les relations existant entre les différentes solutions. Typiquement, est-il possible de transformer étape par étape une solution en une autre à l'aide d'opérations élémentaires, de telle sorte que chaque étape intermédiaire soit également une solution ?

Le problème au départ de cette thèse est celui d'ENSEMBLE DOMINANT, qui consiste à trouver un sous-ensemble $D$ de sommets tel que chaque sommet est dans $D$ ou adjacent à un sommet de $D$. Nous étudions la reconfiguration d'ensembles dominants sous deux opérations élémentaires différentes, principalement d'un point de vue algorithmique. Nous donnons également des conditions nécessaires et suffisantes garantissant qu'une transformation est toujours possible entre deux solutions données. Enfin, nous nous intéressons à la complexité paramétrée d'une variante d'optimisation : étant donné un ensemble dominant $D$, quel est le plus petit ensemble dominant que l'on peut atteindre depuis $D$ sous certaines contraintes ?

Nous nous intéressons également à deux autres questions de reconfiguration. Nous étudions d'une part la complexité de la reconfiguration d'arbres couvrants avec une contrainte sur le nombre de feuilles ; d'autre part la recoloration dans le modèle LOCAL, un modèle de calcul distribué. Pour cette dernière question, nous cherchons à optimiser à la fois le nombre de communications et d'étapes permettant de transformer une coloration en une autre.

**Mots-clés :** graphes, reconfiguration, ensembles dominants

---

**Title:** Reconfiguration problems in graphs

In this thesis, we are interested in graph theory, and more specifically in reconfiguration problems. The goal of this area is to study the relationship between the feasible solutions of a given combinatorial optimization problem. Typically, is it possible to find a step-by-step transformation between two solutions thanks to an elementary operation, in such a way that every intermediate step is also a solution to the problem?

The original problem of this thesis is the so-called DOMINATING SET problem, which consists in finding a subset $D$ of vertices such that each vertex either belongs to $D$ or is adjacent to a vertex in $D$. We study the reconfiguration of dominating sets under two different elementary operations, mainly from an algorithmic point of view. We also provide necessary and sufficient conditions to ensure that a transformation always exists between two given solutions. Finally, we are interested in the parameterized complexity of an optimization variant: given a dominating set $D$, what is the smallest dominating set that is reachable from $D$ under certain constraints?

We are also interested in two other reconfiguration problems. First, we study the complexity of spanning trees reconfiguration with some constraints with respect to the number of leaves. Finally, we introduce recoloring in the LOCAL model in Distributed Computing. In this last problem, we seek to optimize both the number of communication rounds and the number of steps between the two colorings.

**Keywords:** graphs, reconfiguration, dominating sets

---

Laboratoire Bordelais de Recherche en Informatique (LaBRI)
Unité Mixte de Recherche CNRS (UMR 5800)
351 cours de la Libération, 33400 Talence, France

LaBRI